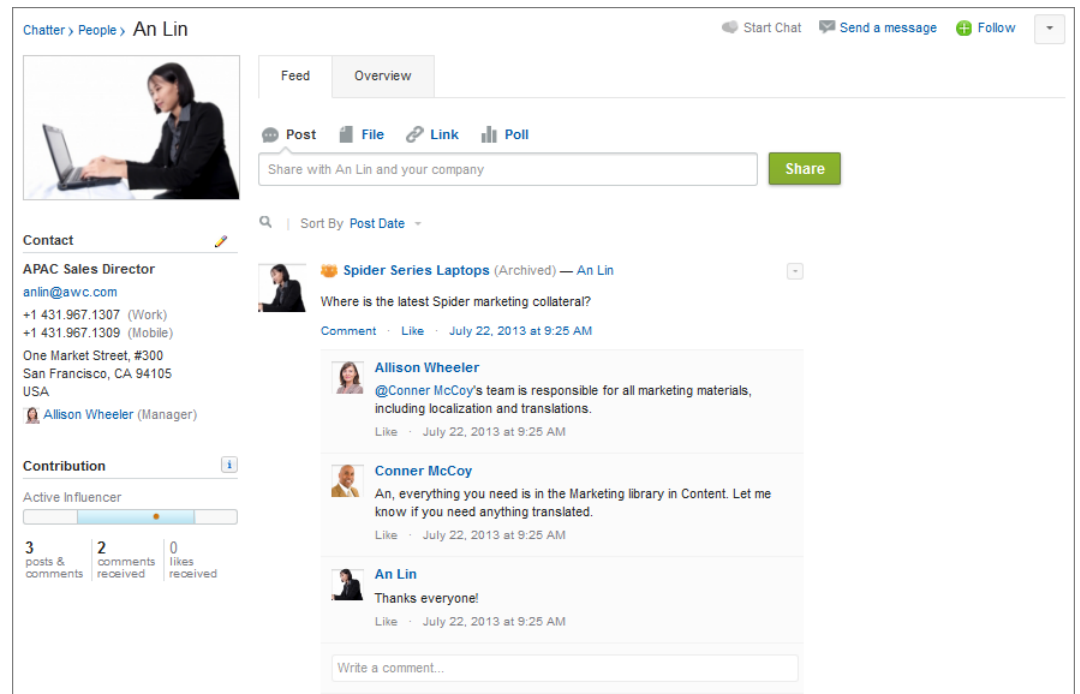# CUSTOMIZING CHATTER PROFILE PAGES

Summary

Chatter profile pages can deploy custom content with the use of subtab apps and custom tabs.

Overview of the Chatter Profile Page

The Chatter profile page now includes profile tabs. With profile tabs, administrators can easily deploy custom content and apps on profiles.

User information is organized in these default tabs.

- Feed tab—Displays your Chatter feed.
- Overview tab—Displays information about you, such as your groups, followers, recent topics, About Me section, and your activity in Communities.

Administrators may configure the tabs on the profile page to include custom tabs or remove default tabs.

Enabling Customization of Chatter Profile Pages

The `Enable Customization of Chatter User Profile Pages` user interface setting enables administrators to customize the tabs on the Chatter user profile page. This includes adding custom tabs or removing default tabs. This feature is automatically enabled in the Winter '14 release. If disabled, users see the Feed and Overview tabs only.

1. From Setup, enter `User Interface` in the `Quick Find` box, then select **User Interface**.

Last updated: December 8, 2015

2. Select `Enable Customization of Chatter User Profile Pages`.

3. Click **Save**.

What is a Subtab App?

An *app* is a group of tabs that work as a unit to provide application functionality. Similarly, a *subtab app* is a collection of tabs that appears on the Chatter profile page. A subtab app can include both default and custom tabs.

Users can see different sets of tabs on the profile page depending on their context. Subtab apps are the various sets of tabs available on specific pages, such as users' profile pages.

These default subtab apps determine which tabs display, depending on the user's context.

Subtab App	Displayed to the user when viewing...
Profile (Others)	Another user inside their internal organization
Profile (Self)	Their own profile inside their internal organization
Profile in Communities (Others)	Another user while inside a community. It's shown only if Communities is enabled.
Profile in Communities (Self)	Their own profile inside a community. It's shown only if Communities is enabled.

Note: End users can't customize the display of subtab apps. Administrators can hide tabs within subtab apps using the `Tab Hidden` option in Tab Settings. Users can see tabs set to `Default Off` and `Default On`.

Managing Subtab Apps

You can view and customize the subtab apps on users' profile pages.

From Setup, enter `Apps` in the `Quick Find` box, then select **Apps** to display your organization's subtab apps.

You can do the following:

- To view details for a subtab app, click the name in the **Subtab Apps** section. This displays its properties, such as which tabs are part of the app, including any tabs that are not yet deployed. Click custom tabs in the Included Tabs list to view details.

- To change the properties of a subtab app, click **Edit** to choose the tabs to include in the subtab app, change their display order, and set the `Default Landing Tab`.

  Note: Administrators can change permission sets or profile settings to limit users' access to each tab. This allows administrators to make specific tabs available to some users, but not to others.

Controlling Subtab App Visibility

Once you have configured subtab apps, you can specify which users can see specific tabs on the profile page.

To control the visibility of tabs within a subtab app:

1. From Setup, enter `Profiles` in the `Quick Find` box, then select **Profiles**.

2. Do one of the following:

- Original profile user interface—Click **Edit** next to the profile you want to modify and scroll to the Tab Settings section.

- Enhanced profile user interface—Click the profile you want to modify and click **Object Settings**. Click the object you want to modify and click **Edit**.

    📝 Note: Some profiles, including Chatter External and Chatter Free users, don't have the permissions needed to view subtab apps.

3. Change the tab settings.

End users can't customize the display of subtab apps. Administrators can hide tabs within subtab apps using the `Tab Hidden` option in Tab Settings. Users can see tabs set to `Default Off` and `Default On`.

4. (Original profile user interface only) To reset users' tab customizations to the tab visibility settings that you specify, select **Overwrite users' personal tab customizations**.

5. Click **Save**.

What is a Profile Tab?

A profile tab is a custom tab you create to display custom object data or other web content on the profile page.

The following types of custom tabs are available for the profile page:

- Custom Web tabs display any external Web-based application or Web page in a Salesforce tab.

- Visualforce tabs display data from a Visualforce page. Visualforce tabs look and function just like standard tabs.

    📝 Note: Custom Object Tabs and Standard Object Tabs are not available as profile tabs.

Profile tabs work just like other custom tabs. Additionally, they can effectively function outside the profile page. However, keep in mind that you lose the ability to toggle tabs without reloading the parent page.

By adding custom tabs on the profile page, you can build and display different types of content including:

- Custom fields

- Internal apps and content

- Org charts

- Wikis or blogs

- Dashboards and other sales, marketing, or service-related information

- External content from sites like LinkedIn® or Twitter®

What Should a Profile Tab Include?

Consider who the viewer is and who is being viewed when creating profile tabs.

When creating content for the profile page, consider the following:

- Is the content specific to individual users?

  Content is displayed on individual profiles and should be focused on a single person, providing interactions and calls to action related to that person and their activity.

- Is it possible to contextualize the content based on the viewer?

  The app knows whether you are viewing your own profile or someone else's, so your design should take that into account:

  - Viewing my own profile: Can I see or do more on my own profile than I can on someone else's?
  - Viewing someone else's profile: Can I see unique information or take action based on the relationship or history between myself and the user I'm viewing?

- Does the content help drive engagement?

  - Consider ways to drive repeat visits to profiles to see new info or respond to calls to action.
  - Your app can easily hyperlink to other people's profiles, or link directly to your profile app from another profile.

Options for Building a Profile Tab

Profile tabs can be built with Web tabs, Visualforce tabs, and Visualforce tabs with a Canvas tag.

- Use a Web tab—Choose whether to send additional context parameters to the webpage, as with any other web tab.
- Use a Visualforce tab—Optionally, extend the standard User controller to properly handle the Salesforce `userId` parameter. This isn't required if you simply want to display related data about the user from other objects.
- Use a Visualforce tab with a Canvas tag—You will also need to use a controller extension and pass the Salesforce `userId` parameter (along with any other relevant parameters) into the Canvas app.

Sample Visualforce Page and User Controller Extension

Extend the standard User controller and create a sample Visualforce page to get started on building custom Visualforce tabs.

User Controller Extension Example

To create your controller extension class, from Setup, enter `Apex Classes` in the `Quick Find` box, then select **Apex Classes**.

```
public with sharing class ProfileTabUserController {
    // Purpose: Custom Chatter profile page
    private ApexPages.StandardController c;

    // Getter methods you can call from your Visualforce page, e.g.
```

```
{! viewingMyProfile }
    public User subjectUser { get; set; }
    public boolean viewingMyProfile { get; set; } // Whether or not
I'm viewing my profile
   public String viewerID { get; set; } // UID string for the viewing
 user
   public String subjectID { get; set; } // UID string for the subject
 user (being viewed)

    // Constructor method for the controller
    public ProfileTabUserController(ApexPages.StandardController
stdController) {
        c = stdController;
        subjectID = getTargetSFDCUID();

        // If we're operating inside a tab running inside of a
profile...
        if (subjectID != null) {
            // Inject the sfdc.userId URL parameter value into the id
 param
            // so the std User controller loads the right User record

            ApexPages.currentPage().getParameters().put('id',
subjectID);
        }

        // Load the User record for the user whose profile we're
viewing
        this.subjectUser = (User)stdController.getRecord();
        Id viewer = Id.valueOf(UserInfo.getUserId());
        Id subject = Id.valueOf(subjectID);
        viewingMyProfile = (viewer == subject);
        viewerID = UserInfo.getUserId();
    }

   // Fetches URL parameter passed into a profile tab indicating which
 user is being viewed
    private String getTargetSFDCUID() {
        return
ApexPages.currentPage().getParameters().get('sfdc.userId');
    }
    // Overrides StandardController save method to force reload of
current page afterwards
    public PageReference save() {
        c.save();
        return ApexPages.currentPage();
    }

   // Overrides StandardController cancel method to force page reload

    public PageReference cancel() {
        c.cancel();
        return ApexPages.currentPage();
```

```
      }
}
```

Visualforce Page Example

To create your page, from Setup, enter `Visualforce Pages` in the `Quick Find` box, then select **Visualforce Pages**.

```
<apex:page showHeader="false" standardController="User"
    extensions="ProfileTabUserController" >

    <apex:outputPanel >

        <p>Your name is {!$User.FirstName} {!$User.LastName}, and
        you're the viewer. Your UID is {!viewerID}.</p>
        <br/>

        <p>You are viewing the profile of {!user.name},
        whose UID is {!subjectID}.</p>
        <br/>

        <p>Are you viewing your own profile? {!viewingMyProfile}</p>
        <br/>

    </apex:outputPanel>

</apex:page>
```

- `{!$User}` refers to the context user's User record.
- `{!user}` refers to the User record of the person whose profile is being viewed. All fields are loaded with this record, including custom fields.

> **Note:** Developers should update their profile apps to use the `showHeader=false` preference in the main Visualforce page tag. This will suppress page headers and footers. If the profile app you develop calls another Visualforce page, the target page must also contain the `showHeader=false` preference, so that Salesforce page headers do not suddenly appear. Alternatively, if you don't manage the code for that target page, you should pass the `isdtp=nv` parameter (headerless parameter) into the page when you invoke it to suppress headers and footers.

Profile Tab Size Limitations

Visualforce tabs on the profile page are limited to a width of 750 pixels.

The profile subtab apps can be configured with Web tabs and Visualforce tabs (including those embedding Canvas applications). Profile tabs function similarly to regular tabs, however some Visualforce tabs won't work well on the profile page because profile tabs are limited to a width of 750 pixels. Additionally, tabs over 900 pixels high include a vertical scrolling bar on the side of the tab content area.

Automatically Resize Profile Tabs

The following code example dynamically resizes the height of a profile tab. This allows varying heights for a tab without the use of an extra scroll bar on the side of the page.

To create your page, from Setup, enter *Visualforce Pages* in the Quick Find box, then select **Visualforce Pages**.

```
<apex:page showHeader="false">
<!-- change background so we can see the resizing -->
<style type="text/css">body {background-color:yellow;}</style>
<script type='text/javascript' src='/canvas/sdk/js/publisher.js'/>
<textarea name="js-console" id="js-console" rows="1" cols="90"
readOnly="true"></textarea>
<script>
     function resize() {
         console.log('SEND RESIZE FROM VF PAGE');
         Sfdc.canvas.publisher.resize( {width : "501px", height :
"301px"});
     }

     function updateDisplay(msg) {
         console.log('GOT EVENT IN VF PAGE: ',  msg);
         var jsConsole = document.getElementById('js-console');
         if(jsConsole) {
           jsConsole.value += msg  + '\n';
           //auto resize the text area so we don't lose it on screen
shot.'
           jsConsole.rows = jsConsole.value.match(/\n/g).length + 1 ;

       }
     }

   Sfdc.canvas.onReady(function() {
       // Optionally listen for the resize callback. Gets called upon
 successful resize
       // and contains actual width and height as it may not be the
requested based off on
       // max values or 'final'
       Sfdc.canvas.publisher.subscribe(
           {name : "sfdc.resize",  onData : function(e)
{console.log(e); updateDisplay('sfdc.resize');}}
       );
   });
   console.log('VFpage rendered');
</script>
<p/>

<input id='refresh' type='button' value='Resize' onclick='resize();'/>
Test page for VF resizing
</apex:page>
```

Profile Page Parameters

Parameters are namespaced to the vendors and tabs themselves to allow you to pass parameters to your pages through the profile and link to your tabs from other locations.

When your profile tab is loaded within the Chatter profile page, a request parameter, `sfdc.userId`, is passed. This parameter contains the User ID of the person whose profile is being viewed. This information is needed so that your app can load the appropriate content.

> Note: The parameter isn't passed if your profile tab is a Web tab.

Namespaces

The profile can host many different vendors and individual tabs, so these parameters are namespaced to the vendors and tabs themselves. In order to support tab switching and changes to tab parameters without reloading, this functionality has been extended to work with the URL fragment—the hash or # portion of the URL—as an alternate method of input.

The namespace will differ depending on the tab type:

- If the tab is part of a managed package, the `tabName` is prefixed by the developer namespace. For example, the tab `ExampleTab` created by `ExampleDeveloper` would be called `ExampleDeveloper.ExampleTab`.

- Tabs created as part of unmanaged developer packages don't have a namespace. In those cases, the `tabName` is simply the tab name specified during tab creation, or the object ID for custom objects. For example, the tab `ExampleTab` created by `ExampleDeveloper` would be called `ExampleTab`.

- Core Salesforce tabs carry the namespace `sfdc`. For example, the tab name for a core tab would be `sfdc.CoreTabName`.

In the following sections, `tabName` refers to any of these cases, depending on context. If a developer namespace is set, `tabName` refers to the fully qualified name `DeveloperName.TabName`.

Parameters

Tab Name Parameter—`tab=[tabName]`
The currently viewed tab may be switched by altering the tab parameter in either the URL query string or the URL fragment. By using the fragment to set this parameter, a tab switch may be effected without requiring a whole-page reload—and perhaps no load at all if the switched-to tab has already been loaded. For example:

- Current URL—`https://salesforce.com/_ui/core/userprofile/ UserProfilePage?tab=Dev.Example`

- From within the Dev.Example tab the developer changes the parent URL to—`https://salesforce.com/_ui/core/userprofile/ UserProfilePage?tab=Dev.Example#tab=Dev.OtherExample`

- Which is then rewritten by the Profile dynamically to—`https://salesforce.com/_ui/core/userprofile/ UserProfilePage?tab=Dev.OtherExample`

In this way, the client tab was able to change the tab parameter without changing the parent URL. The advantage is that a full page reload is not required, and cross-domain restrictions don't apply since changing the URL is allowed (though not reading it).

Arbitrary Tab Parameters—`tab.[tabName].[param]=[value]`
Tabs may pass themselves or other tabs arbitrary parameters through the parent profile URL via either the query string or the URL fragment, treated as a query string. For example:

- Current parent URL—`https://salesforce.com/_ui/core/userprofile/`
  `UserProfilePage?tab=Dev.Example`
- Current child URL—`https://salesforce.com/apex/Example`
- From within the child Dev.Example tab the developer changes the parent URL
  to—`https://salesforce.com/_ui/core/userprofile/`
  `UserProfilePage?tab=Dev.Example#tab.dev.Example.param=value`
- Which is then rewritten by the Profile dynamically
  to—`https://salesforce.com/_ui/core/userprofile/`
  `UserProfilePage?tab=Dev.Example&tab.dev.Example.param=value`
- Which then reloads the child tab
  with—`https://salesforce.com/apex/Example?param=value`

User ID Parameter—`sfdc.userId=[UserID]`

The profile will always pass `sfdc.userId=[UserId]` to all child tab pages excluding Web tabs, which require configuration by the administrator due to security constraints. This is the ID of the profile being viewed, not the ID of the person viewing the profile.