Customizing Alfresco Share JavaScript Controllers

8 Replies

Introduction

In the last blog I introduced extension module "customizations" and demonstrated how they could be used to override an i18n properties file to change the label on the User Dashboard. In this post I'm going to be demonstrating how to change the behaviour of a Web Script by augmenting its JavaScript controller. The extensibility features used are currently available in the latest Alfresco Community source and will be in Alfresco Enterprise 4.0.

Tutorial

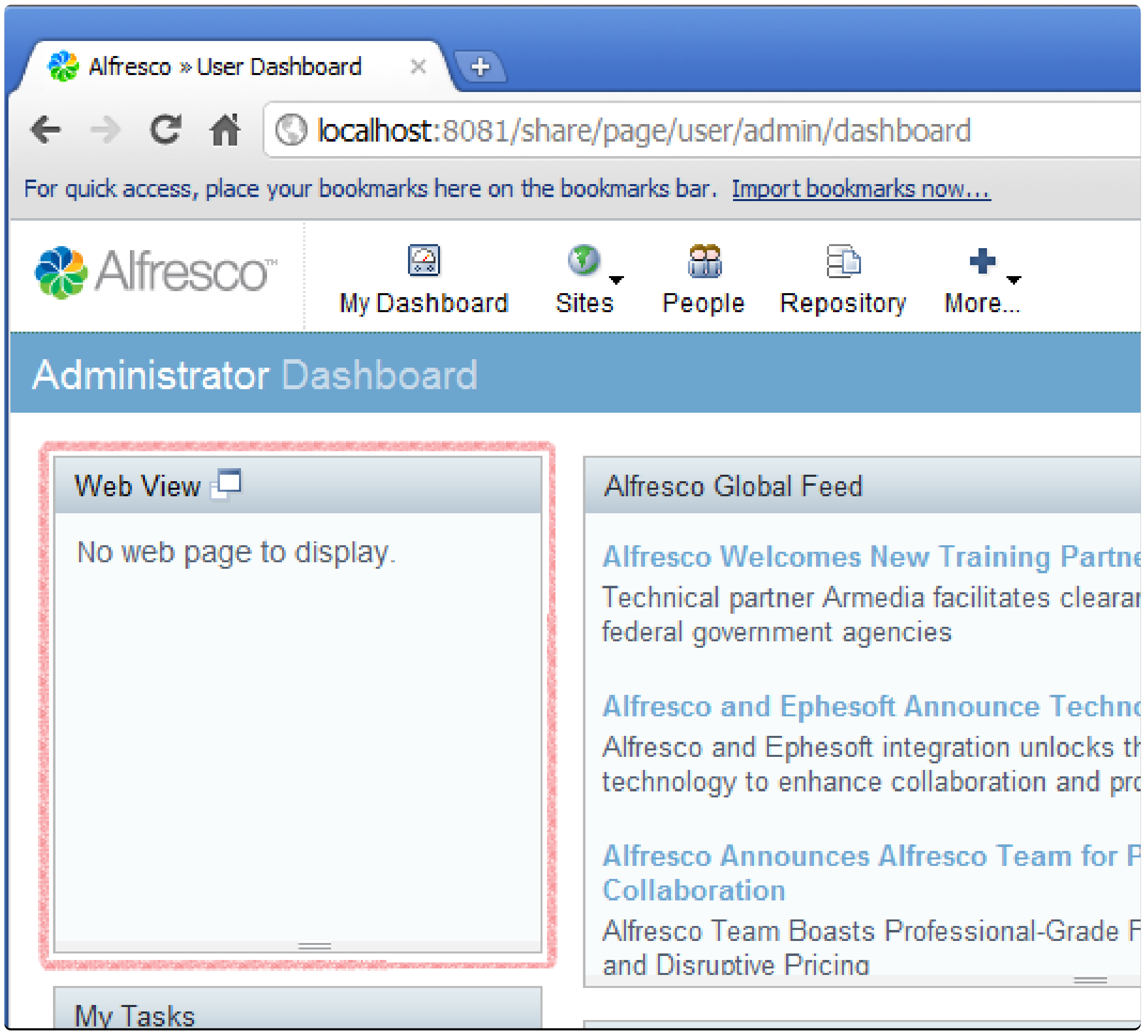
In this tutorial I'm going to demonstrate how to provide an extension to the JavaScript controller file for the Web Script behind the "Web View" dashlet so that the Alfresco.com web site will be displayed if the user hasn't specified another.

Making this kind of change does require some insider knowledge of how Alfresco Share is coded. Fortunately we can use a combination of SurfBug and the Web Scripts service UI to find out everything you need to know. There's no guarantee that you will be able to do everything you want using this method – but it would definitely be worth exploring as a first port of call.

First of all we need to add the "Web View" dashlet to the User Dashboard:

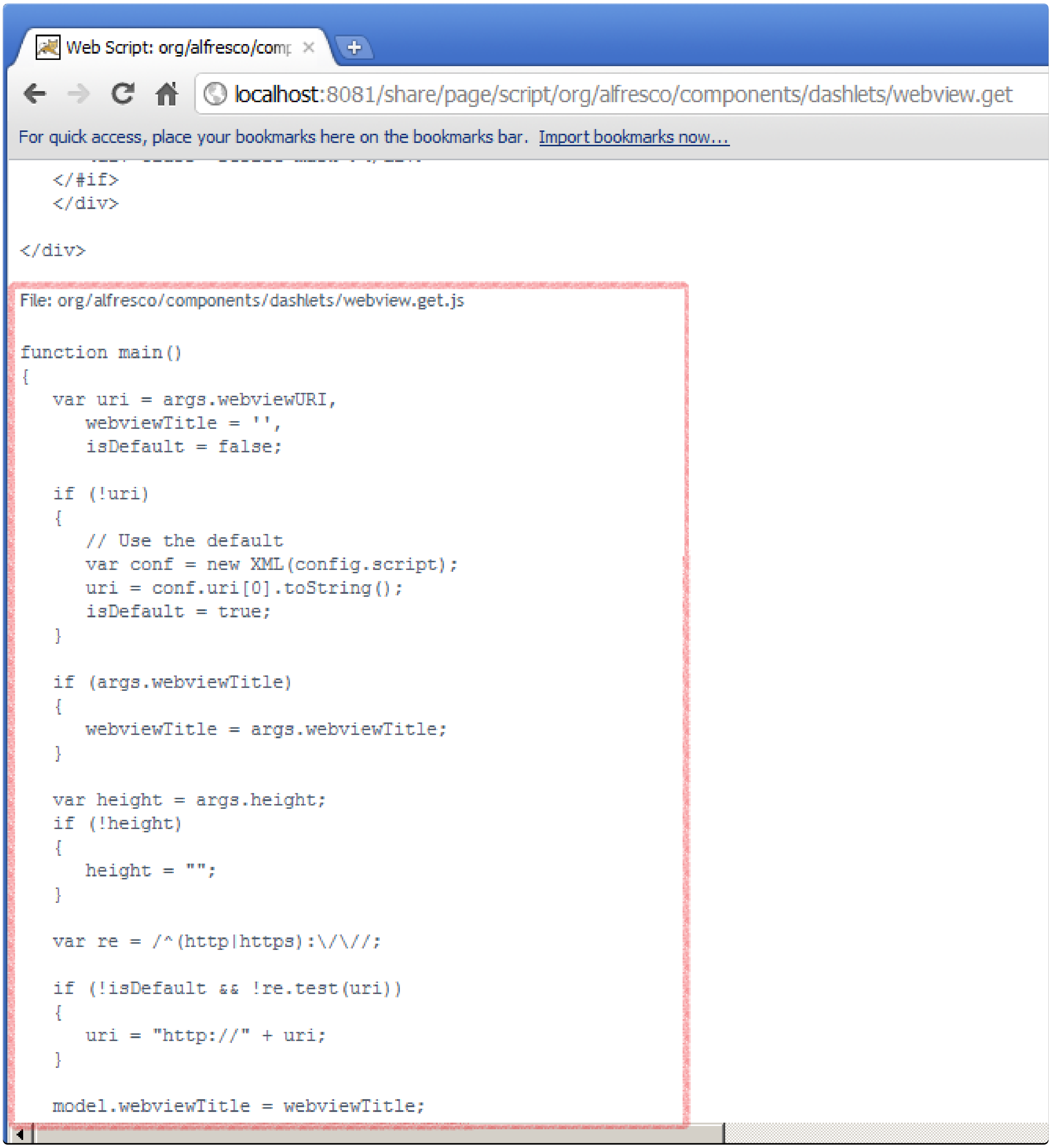
1. Log on to Alfresco Share
2. Click the "Customize Dashboard" button
3. Click on "Add Dashlets"
4. Drag the "Web View" dashlet from the available list and drop it into one of the dashboard columns
5. Click "OK"

Initially the dashlet will display the message "No web page to display" as it has not been configured.


Alfresco » User Dashboard
localhost:8081/share/page/user/admin/dashboard
For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...
Alfresco
My Dashboard
Sites
People
Repository
More...
Administrator Dashboard
Web View
No web page to display.
Alfresco Global Feed
Alfresco Welcomes New Training Partne
Technical partner Armedia facilitates clearar
federal government agencies
Alfresco and Ephesoft Announce Techno
Alfresco and Ephesoft integration unlocks th
technology to enhance collaboration and pro
Alfresco Announces Alfresco Team for P
Collaboration
Alfresco Team Boasts Professional-Grade F
and Disruptive Pricing
My Tasks

Working in the Alfresco Engineering team meant that I already knew that I'd be able to make the customization I wanted – but by enabling SurfBug (http://localhost:8080/share/page/surfBugStatus – if you're using the default port) and clicking on the Web View dashlet you can find the Web Script URI (**"/components/dashlets/webview"**) that renders it and by browsing for Web Scripts by URI (http://localhost:8080/share/page/index/uri/) you can access all the information (including the source of the JavaScript controller). This was explored in greater detail in the last blog if you need a refresher on how to find this information.

By inspecting the source of both the controller and the template you can work out what model properties the template is using and whether or not you can provide an extension to update the model after the base controller but before the template in order to create the desired result.


Web Script: org/alfresco/comp
localhost:8081/share/page/script/org/alfresco/components/dashlets/webview.get
For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...

```
   </#if>
   </div>

</div>
```

File: org/alfresco/components/dashlets/webview.get.js

```
function main()
{
   var uri = args.webviewURI,
      webviewTitle = '',
      isDefault = false;

   if (!uri)
   {
      // Use the default
      var conf = new XML(config.script);
      uri = conf.uri[0].toString();
      isDefault = true;
   }

   if (args.webviewTitle)
   {
      webviewTitle = args.webviewTitle;
   }

   var height = args.height;
   if (!height)
   {
      height = "";
   }

   var re = /^(http|https):\/\//;

   if (!isDefault && !re.test(uri))
   {
      uri = "http://" + uri;
   }

   model.webviewTitle = webviewTitle;
```

Having identified that the dashlet is rendered by a Web Script using the controller " **webview.get.js**" in the "**org.alfresco.components.dashlets**" package we can define a new module with a customization to apply to it.

Edit the "**blog-demo.xml**" file and add the following module configuration:

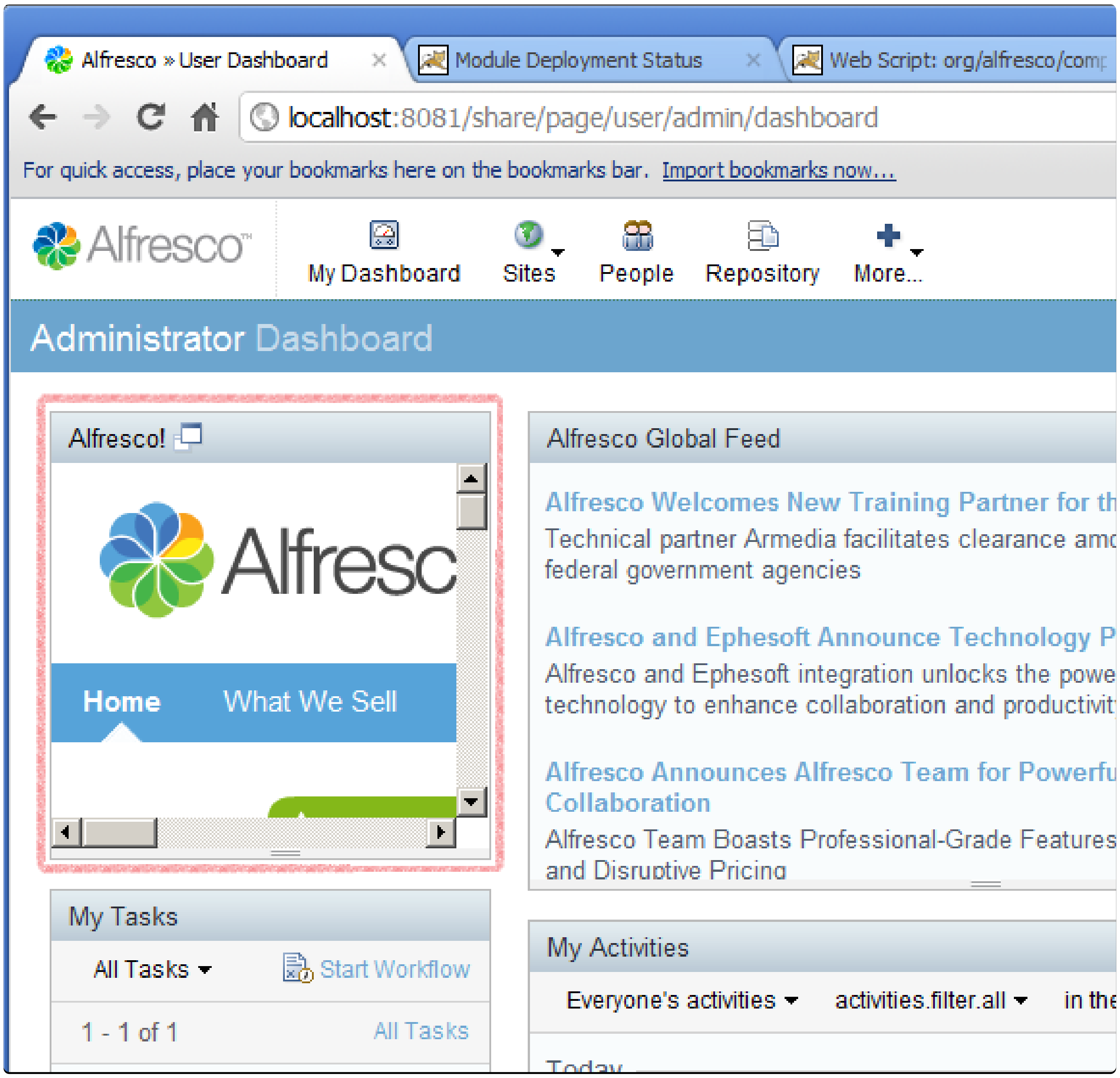
```
<module>
    <id>Blog Module (Web View JavaScript controller change)</id>
    <customizations>
        <customization>
            <targetPackageRoot>org.alfresco.components.dashlets</targetPackageRoot>
            <sourcePackageRoot>blog.demo.customization</sourcePackageRoot>
        </customization>
    </customizations>
</module>
```

Although we’re targeting a different package, we’re going to map it to the same package as we used for the properties customization in the last blog post.

Create a file called “**webview.get.js**” containing the following:

```
if (model.isDefault == true)
{
    model.webviewTitle = "Alfresco!";
    model.uri = "http://www.alfresco.com";
    model.isDefault = false;
}
```

Place the new file in the “**webscripts.blog.demo.customization**” package, rebuild and deploy your JAR, restart the web server and deploy your new module. When you login to Alfresco Share you should see that the Web View dashlet now displays the Alfreco.com website.



Alfresco Welcomes New Training Partner for th
Technical partner Armedia facilitates clearance amo
federal government agencies

Alfresco and Ephesoft Announce Technology P
Alfresco and Ephesoft integration unlocks the powe
technology to enhance collaboration and productivit

Alfresco Announces Alfresco Team for Powerfu
Collaboration
Alfresco Team Boasts Professional-Grade Features
and Disruptive Pricing

Your ability to customize Alfresco Share pages in this way will be somewhat limited to how they have been coded – but it’s worth exploring as an option. In the future we will endeavor to create Web Script controllers with respect to potential extensions, but as the existing controllers weren’t created with this in mind it’s really pot luck as to whether you can achieve what you want.

However, there’s another customization file type that you can also use… in the next blog post I’ll be demonstrating how to customize Web Script FreeMarker templates.