A Simple Alternative to the MyFaces ExtensionFilter

By Roger Keays, 23 March 2007

It's great that JSF allows us to build, package and distribute components in such a reusable way. However, one of the shortcomings of the spec (IMHO) is the lack of a common facility to specify external resources that a component may require, such as javascript or CSS files. MyFaces uses a custom `ExtensionFilter` for it's Tomahawk component library to solve this problem. Unfortunately, this solution isn't without its drawbacks. Here are some of the problems I had with this filter, and my ultimate (very simple) solution.

The MyFaces ExtensionFilter has two purposes:

1. To retrieve static resources such as javascript and CSS files.
2. To inject them into the <head> of a page transparently.

For the first purpose, the filter works just fine. For the second purpose, however, the filter must cache the entire response and parse it to find the <head> component. The Tomahawk web page warns that this uses extra memory, but I think the side effects are slightly underexaggerated. The problems I encountered were:

1. Some of my components stream large binary files. The filter was caching all this output in memory. Oh no! This was causing serious memory wastage and performance degradation.
2. It bauks on AJAX requests since there is generally no <head> section in the response and logs the entire response of the request. If you do a lot of AJAX, that's going to fill up your logs pretty fast and also slow down responses.
3. It prevents Shale remoting from sending a 403 NOT MODIFIED response and caused an exception to be logged (bug in my setup?).

Basically it comes down to performance. I can see that the MyFaces team have delegated the job of adding resources via the AddResource interface so I could theoretically implement my own version to work around these issues. That looked way too hard, so I came up with a simpler solution.

I noticed that, actually, you can just hard-code the MyFaces resources in the head section and the components work just fine. The filter is then only used for retrieving these resource since there is no need to inject them into the <head>. With Facelets, you can pass a <ui:param> to your master template to indicate whether or not you need certain resources on a page. Here is an example.

Filter configuration (web.xml):

```
<!-- MyFaces extension filter (not mapped to FacesServlet)-->
```

```
<filter>
  <filter-name>MyFacesExtensionsFilter</filter-name>
  <filter-class>org.apache.myfaces.webapp.filter.ExtensionsFilter</filter-class
</filter>
<filter-mapping>
  <filter-name>MyFacesExtensionsFilter</filter-name>
  <url-pattern>/faces/myFacesExtensionResource/*</url-pattern>
</filter-mapping>
```

Master template head section:

```
<head>

  <!-- myfaces tree2 resources -->
  <ui:fragment rendered="${useMyFacesTree eq 'true'}">
    <script src="faces/myFacesExtensionResource/org.apache.myfaces.renderkit.ht
            type="text/javascript"/>
    <script src="faces/myFacesExtensionResource/org.apache.myfaces.renderkit.ht
            type="text/javascript"/>
  </ui:fragment>

  <!-- myfaces tabPanel resources -->
  <ui:fragment rendered="${useMyFacesTabs eq 'true'}">
    <link href="faces/myFacesExtensionResource/org.apache.myfaces.renderkit.htm
          rel="stylesheet" type="text/css"/>
    <script src="faces/myFacesExtensionResource/org.apache.myfaces.renderkit.ht
            type="text/javascript"/>
  </ui:fragment>
  ...
</head>
```

Client template:

```
<ui:decorate template="master.xhtml">
  <ui:param name="useMyFacesTabs" value="true"/>

  <t:panelTabbedPane>
    <t:panelTab>Tab 1</t:panelTab>
    <t:panelTab>Tab 2</t:panelTab>
  </t:panelTabbedPane>
</ui:decorate>
```

Pretty simple really, but you'd be suprised how long it took me to come up with this idea (hint: it's in the order of days - possibly weeks). I got buried way too deep in interfaces, components and filters.

If you're using JSP you could do something similar, except your conditions might be more complicated since you don't have the luxury of master/client templates (unless you are using Tiles).

This is a problem I believe can be tracked back to the condition that the component tree be built and rendered during the same phase (mostly to accomodate JSP). With separate build and render phases, you could have a <h:head> component which other components could add resources to as they are created during the tree build.

About Roger Keays



Roger Keays is an artist, an engineer, and a student of life. He has no fixed addressand has leftfootprints on 40-something different countries around the world. Roger is addicted to surfing. His other interests are music, psychology, languages, the proper use of semicolons, and finding good food.