ClayAndTiles

Shale Clay and Tiles are similar in that they provide a layer of metadata that is used to compose a page. They are different in the problem space that they are uses in and the granularity of reuse.

Clay is designed around providing information to create a JSF page where Tiles provides information about creating JSP pages.

Tiles allows you to create an abstraction of a JSP page. The entry point of the page is an abstracted metadata definition defined in an XML configuration file. The tiles definition is a collection of symbols that are used by a generic layout to pull in various JSP fragments.

Clay also allows you to create an abstract page entry point similar to Tiles. The entry point is a resource identifier that is not a physical file under the context root. We call these full XML views.

The two look very similar if you would compare the definitions.

Tiles:

```
<definition name=”.mainLayout” path=”/mainLayout.jsp”>
   <put name=”header” value=”/header.jsp”/>
   <put name=”body” value=”/body.jsp”/>
</definition>


<definition name=”.index” extends=".mainLayout">
   <put name=”body” value=”/pages/index.jsp”/>
</definition>
```

Clay:

```
<component jsfid="basePage" extends="clay">
   <attributes>
      <set name="clayJsfid" value="/pages/layout_nsjsp.html" />
   </attributes>
   <symbols>
      <set name="@title" value="commonTitle" />
      <set name="@bodycontent" value="space" />
   </symbols>
</component>

<component jsfid="/index.cxv" extends="basePage">
   <symbols>
      <set name="@bodycontent" value="/pages/index_nsjsp.html"/>
   </symbols>
</component>
```

Clay's focus is meta-data at a component level. At the top level, the component is a page. You could also argue that Tiles could be used the same way. The reuse granularity that I've stated is different between Tiles and Clay (JSP versus JSF) might be more conceptual than what is possible in terms of meta-data.

However, the limitations and strengths are defined by how JSF uses JSP. Lets talk about Clay first. When you use Clay full xml views, the page is completely defined by the JSF component tree. The Clay component is specifically build for creating a subtree that adds to the the JSF component tree. All markup that is not a JSF component is added to the tree as an outputText component.

When you use JSP with JSF in version 1.1, the JSP, non JSF/JSP tags and JSF components all write markup to the same writer. The JSF/JSP tags don't actually render markup. They build the component tree and tell the component to render itself as the tree is build.

Clay builds the entire component tree first and then invokes rendering. All of the page content is represented by a JSF component. Rendering is in two steps. Build the tree and render the markup.

This is one of the challenges of trying to combine Clay full views that include JSP fragments. Clay wants to build the full tree first and then invoke rendering but the fragment would want to invoke rendering while the component tree was being built.

Another challenge is how the JSF/JSP tags keep track of the last JSF/JSP tag. If we wanted to mix view technologies for creating a page, there would have to be a standard for how the JSP/JSF tags work. Right now this is implementation specific. Myfaces handles it different than the 1.1 RI.

Now, JSF 1.2 has changed how the JSF/JSP works together. JSF 1.2 has two steps to rendering, like Clay. The entire component tree is created first and then rendering invoked. The content of the non JSF/JSP tags and markup in the JSP page is robbed from the writer and made into a JSF outputText component.

If we can figure out a common exchange between non-JSP/JSF page composition (Clay) and JSP/JSF page composition, we would be closer to accomplishing the niche that Tiles provides.

I have an example in the sandbox of using clay full xml views (http://svn.apache.org/viewvc/shale/sandbox/shale-clay-mailreader/)

References

- 20060915 user@shale post