Struts Tiles vs. JSP Include

Author: Pankaj Wadhwa, Mindfire Solutions
www.mindfiresolutions.com

Table of Contents

1. Introduction

Large websites often need a common Look and Feel (L&F). If the L&F is hard coded in all pages, changing it becomes a nightmare: you would have to modify nearly all pages.

In this document we will try resolve this issue using Struts Tiles Framework, which have been added into the Struts 1.1 release.

Firstly we will go through a simple struts application that does the work the traditional way and try to figure out the issues that may come after a period of time.

After that we would modify that example using the JSP include technology and again will try to find the problems with this technology.

At last we would take a look at the Struts Tiles framework and modify the example using this technology. We will also list down the differences between the JSP include and Tiles at the end.

So let's start…☺

2. Target Audience

Java developers who are responsible for front end development of JSPs / Html pages. It is expected that the reader of this document is already familiar with how struts application works and the JSP technology.

3. Technology used

JDK 1.4 or above, JSP, Tomcat 5.0, Struts

4. Struts application, the traditional way

Firstly download the struts latest framework from apache website. Extract this and retrieve the struts-blank application. Create a new tomcat application by name StrutsExampleApp and import the struts blank application to it. Now create and add two JSP pages to this application for our example as given below:

First.jsp

```
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html"%>

<html:html>
<body>
        <table>
                <!—Header →
                <tr>
                        <td>
                                <P align=”center”>
                                        <B>Welcome</B>
                                </P>
                        </td>
                </tr>
                <tr>
                        <td>
                                <B>Body for First JSP</B>
                        </td>
                </tr>
                <!—Footer →
                <tr>
                        <td>
                                <P align=”center”>
                                        Copyright © 2005-06
                                </P>
                        </td>
                </tr>
        </table>
<body>
</html:html>
```

Second.jsp

```
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html"%>

<html:html>
<body>
        <table>
                <!—Header →
                <tr>
                        <td>
                                <P align=”center”>
                                        <B>Welcome</B>
                                </P>
                        </td>
                </tr>
                <tr>
                        <td>
                                <B>Body for Second JSP</B>
                        </td>
                </tr>
                <!—Footer →
                <tr>
                        <td>
                                <P align=”center”>
                                        Copyright © 2005-06
                                </P>
                        </td>
                </tr>
        </table>
<body>
</html:html>
```

For simplicity we have not added any form or action class to this project.

Create actions for these JSPs by adding the definition for this in struts-config.xml as shown below:

struts-config.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<!DOCTYPE struts-config PUBLIC
        "-//Apache Software Foundation//DTD Struts Configuration 1.2//EN"
        "http://jakarta.apache.org/struts/dtds/struts-config_1_2.dtd">

<struts-config>
```

```
    <action-mappings>

      <action
          path="/first"
          forward="/First.jsp" />

      <action
          path="/second"
          forward="/Second.jsp" />

    </action-mappings>
</struts-config>
```

Add the project to Tomcat. Start the tomcat server. You can now see the output in browser at http://localhost:8080/first.do and http://localhost:8080/second.do

By seeing at the above JSPs we can easily verify that same code is being repeated in all the JSPs. So there is a code repetition. A small change in Header or Footer would need a lot of work as we have to change it in all the JSPs.

5. Struts application using JSP include

JSP resolves this issue by providing include tag. Syntax for JSP include is shown below:

<jsp:include page="/Header.jsp" />

We can also use simple include tag as below:

<%@ include file="Header.jsp"%>

Note: jsp:include tag place the code to the particular location every time a page is viewed, hence it's a dynamic inclusion. A simple include place the code to the particular location only at first time, hence a static inclusion.

The above-mentioned code shows that the Header.jsp is being included at the particular location.

In our example we need two different JSPs Header.jsp and Footer.jsp. Code for Header.jsp and Footer.jsp would be as shown below:

Header.jsp

```
<P align="center">
        Welcome
</P>
```

Footer.jsp

```
<P align="center">
        Copyright © 2005-06
</P>
```

Our new First.jsp and Second.jsp would be as show below:

First.jsp

```
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html"%>

<html:html>
<body>
        <table>
                <tr>
                        <td>
                                <jsp:include page="/Header.jsp" />
                        </td>
                </tr>
                <tr>
```

```
                        <td>
                                <B>Body for First JSP</B>
                        </td>
                </tr>
                <tr>
                        <td>
                                <jsp:include page="/Footer.jsp" />
                        </td>
                </tr>
        </table>
<body>
</html:html>
```

Second.jsp

```
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html"%>

<html:html>
<body>
        <table>
                <tr>
                        <td>
                                <jsp:include page="/Header.jsp" />
                        </td>
                </tr>
                <tr>
                        <td>
                                <B>Body for Second JSP</B>
                        </td>
                </tr>
                <tr>
                        <td>
                                <jsp:include page="/Footer.jsp" />
                        </td>
                </tr>
        </table>
<body>
</html:html>
```

Our struts-config.xml would be same as before.

As you can now easily see that it is now very easy to modify our header or footer, as we have placed them in new files which are used commonly and can be modified when required and save a lot of time.

6. Introduction to Tiles

When you are using JSP include, you create the layout of the website within the page and then place the actual view component (JSP). Hence you need to repeat the same layout logic to every page causing repetition in the web page. This also discourages you to change the view of the website at a later time as you may have a lot of pages to modify.

“Tiles” is the solution for this problem. Tile is an area or region on web page. Tiles is the technology which is used to create view of a website. Using Tiles you can define the layout as a template, can create complicated layouts and use them through out the application for consistent layout. The purpose of layout is to assemble a group of tiles to specify the format of page. As it is a layout, you insert placeholders (using Tiles insert tag) instead of actual view components (JSPs). The values for the placeholders are defined by an XML. If you want to change the view of the application you can do so by changing the layout only and can save a lot of time.

With Tiles you can do things such as:

- Screen definitions that include inheritance
- Templating: you can create templates and can use them again and again
- Layouts for common pages, menus, and portals
- Dynamic page building
- Reuse tiles
- I18N support for locale-specific loading

7. Struts application using Tiles

Let us try to understand how tiles are better than simple jsp include by modifying our example.

In tiles firstly we have to create Layout.jsp for the layout.

Layout.jsp

```
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html"%>
<%@ taglib uri="/WEB-INF/tiles.tld" prefix="tiles"%>

<html:html>
<body>
        <table>
                <tr>
                        <td>
                                <tiles:insert attribute="header"/>
                        </td>
                </tr>
                <tr>
                        <td>
                                <tiles:insert attribute="body"/>
                        </td>
                </tr>
                <tr>
                        <td>
                                <tiles:insert attribute="footer"/>
                        </td>
                </tr>
        </table>
<body>
</html:html>
```

Tiles:insert tag defines a particular location where we can place any other component. Value for this is retrieved from the XML file which we will create later on.

The only difference is that instead of explicitly including the JSP files, the template file includes a body-content section. This allows us to reuse this template for any page that has this generic format. Once we figure out how to supply the page-specific body content, we can reuse this template over and over again. This one file can then control the layout of multiple pages. If we need to modify the layout of the site, this is the only file we need to change--that's the real power of using a Tiles framework.

Now we will create two JSPs for the body. firstBody.jsp and secondBody.jsp as shown below.

firstBody.jsp

```
<B>
        Body for First JSP
</B>
```

secondBody.jsp

```
<B>
        Body for Second JSP
</B>
```

Create the xml file and specify the definition of all the components. Let's name that file tileDef.xml and specify its pages as shown below:

tileDef.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<component-definitions>
   <definition name="FirstPage">
         <put name="header" value="/Header.jsp" />
         <put name="body" value="/firstBody.jsp"/>
         <put name="footer" value="/Footer.jsp" />
   </definition>
   <definition name="SecondPage">
         <put name="header" value="/Header.jsp" />
         <put name="body" value="/secondBody.jsp"/>
         <put name="footer" value="/Footer.jsp" />
   </definition>
</component-definitions>
```

We will use these definitions (FirstPage and SecondPage) into our struts-config.xml for forwarding purpose. But before doing that we need to include the Tiles plug-in to our application. Add the code given below to our struts-config.xml.

```
<controller
  processorClass="org.apache.struts.tiles.TilesRequestProcessor"/>

<plug-in className="org.apache.struts.tiles.TilesPlugin" >

  <!-- Path to XML definition file -->
  <set-property property="definitions-config"
            value="/WEB-INF/tiles-defs.xml" />
</plug-in>
```

This code should be in between the struts-config tag.

Now use our defined **definitions** for forwarding purpose. The final struts-config.xml is shown below:

struts-config.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<!DOCTYPE struts-config PUBLIC
        "-//Apache Software Foundation//DTD Struts Configuration 1.2//EN"
        "http://jakarta.apache.org/struts/dtds/struts-config_1_2.dtd">

<struts-config>
   <action-mappings>

      <action
         path="/first"
         forward="/FirstPage" />

      <action
         path="/second"
         forward="/SecondPage" />

   </action-mappings>
   <controller
     processorClass =
     "org.apache.struts.tiles.TilesRequestProcessor"/>

   <plug-in className="org.apache.struts.tiles.TilesPlugin" >

     <!-- Path to XML definition file -->
     <set-property property="definitions-config"
                   value="/WEB-INF/tiles-defs.xml" />
   </plug-in>

</struts-config>
```

Restart the tomcat and you can see the output at same location.

8. Tiles Inheritance

We can inherit the Tiles definitions to reduce the code repetition in tileDef.xml as shown below:

tileDef.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<component-definitions>
   <definition name="BasicLayout" page="/jsp/Layout.jsp">
         <put name="header" value="/jsp/header.jsp" />
         <put name="footer" value="/jsp/footer.jsp" />
   </definition>
   <definition name="FirstPage" extends="BasicLayout">
         <put name="body" value="/firstBody.jsp"/>
   </definition>
   <definition name="SecondPage" extends="BasicLayout">
         <put name="body" value="/secondBody.jsp"/>
   </definition>
</component-definitions>
```

As you can see, we have created a BasicLayout here, which defines the values for Header and footer. BasicLayout is being extended by other tags. The value defined in the newly created tag will be given priority than the extended tag.

Struts-config.xml will not have any change here.

9. Tiles vs. JSP Include

9.1. Advantage of Tiles over JSP include

1. Code Repetition is reduced
   Use of Tiles reduce the code repetition to a great extent. Code repetition is bad but repetition of layout logic could be worst. Tiles also handle this issue. As you have layout templates based on which all the pages are combined, you don't need to repeat the code for layout. Other view components are also reusable and can be used in the same application at other places reducing the code repetition.
2. Low coupling between pages
   Coupling is the degree of interactivity between two entities. It is always suggested to minimize coupling between unrelated classes, packages, and so on. Same principle is applied to view components. Tiles reduce the coupling between unrelated view components.
3. High layout control
   Tiles provide great layout control by providing layout templates.
4. I18N support for locale-specific loading
5. Dynamic Page building
   Pages are built dynamically in tiles. You can control the page view by configuring it through xml.
6. Elimination of duplicate and redundant information
   Tiles eliminate the duplicate and redundant information in the configuration file by providing Tiles inheritance.
7. Central location for view components
   Tiles save definition of all the components at one place (in tilesDef.xml) and hence can be modified easily when required.

9.2. Disadvantages of Tiles

1. Increases the number of pages
2. Increases complexity
   Tiles increase complexity by introducing another layout page. Understanding and implementing templating can also be difficult at initial stages.
3. Big API
   Tiles have a bigger API set which may be difficult to understand for getting full benefit of tiles.
4. You have to specify a name to any component you create.