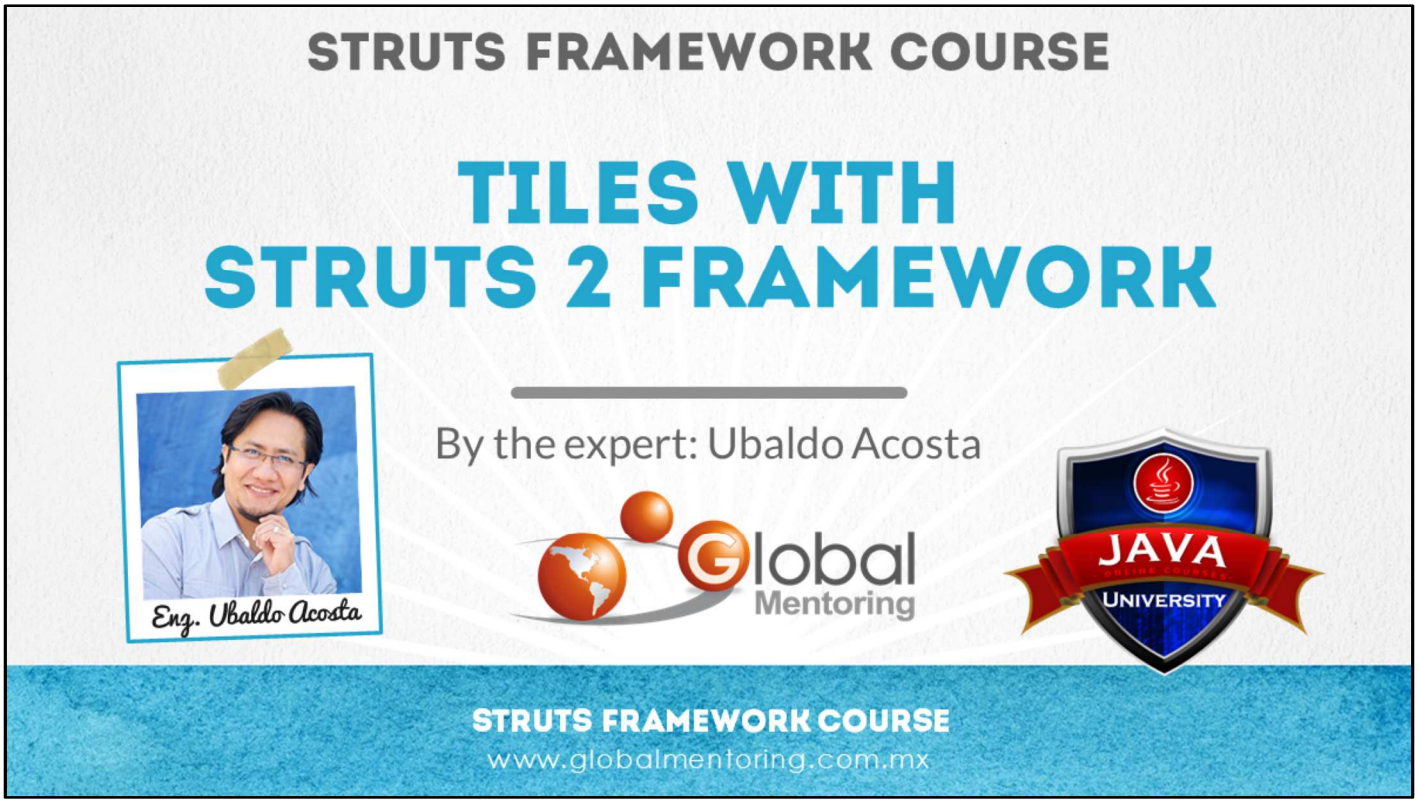
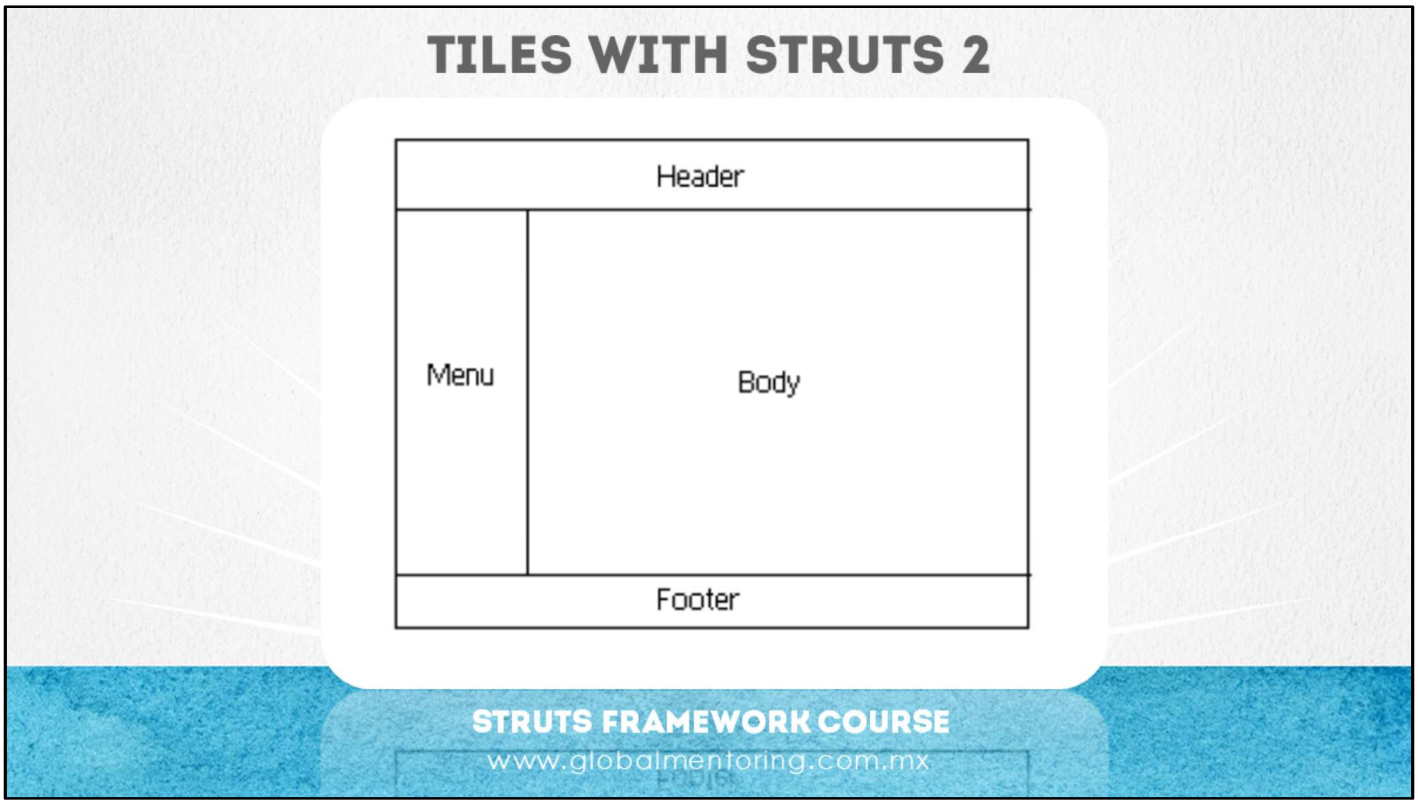


Hello, Ubaldo Acosta greets you again.

In this lesson we will study the topic of Tiles in Struts 2.

Are you ready? Come on!

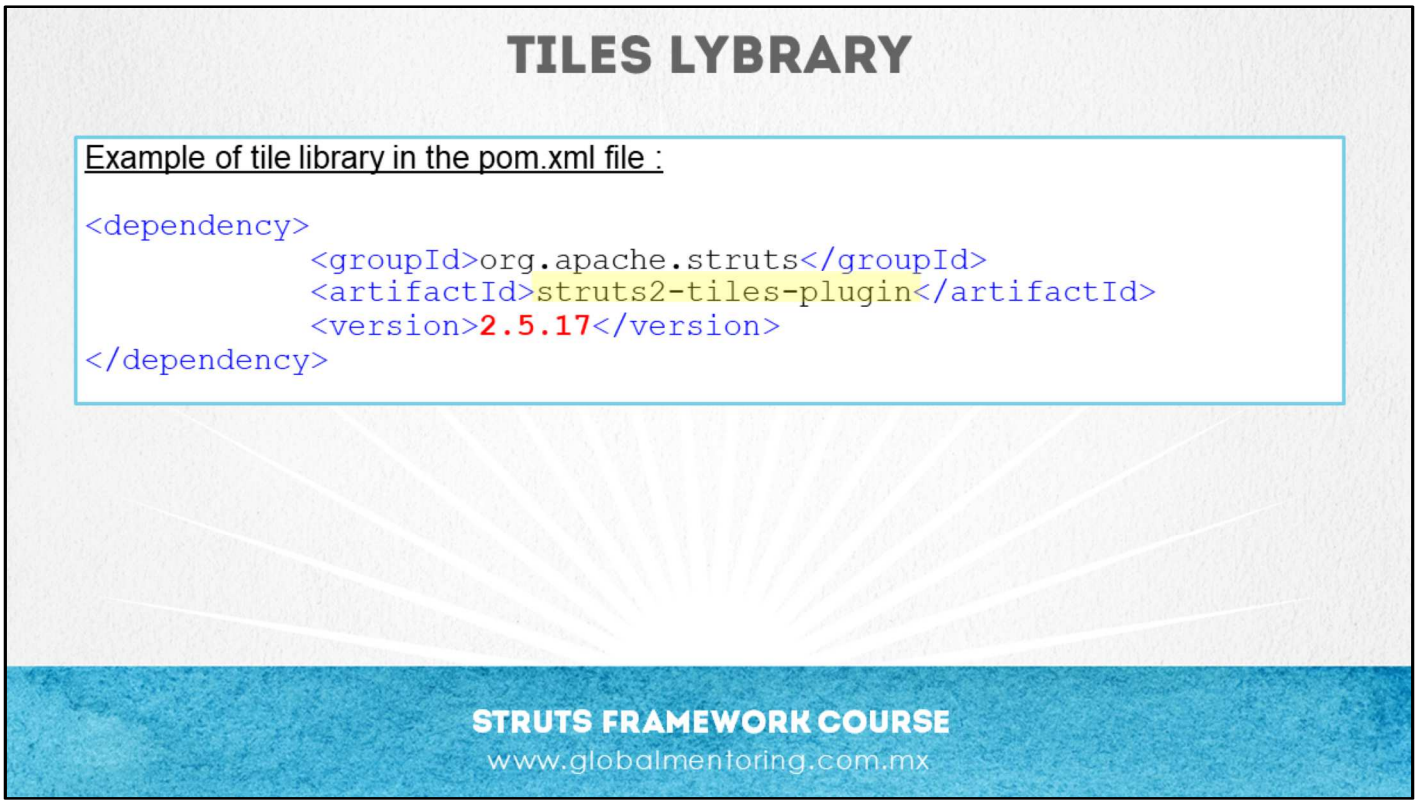


In this lesson we will see the topic of managing Tiles in Struts 2.

In all the previous exercises we have not seen how to reuse the views that we have created. In this section we will see the theme of Tiles, which basically allows us to reuse the views we create and give them an order as shown in the figure.

Of course, the order we give is completely free to choose, we can add as many sections and configurations as necessary. However, the tile theme allows us to reuse, for example, the Header, Menu and Footer of a page and only change what we need, for example, the content of the Body of a page, without having to change the other elements that make up our page.

So in this lesson we will precisely study how we can create these templates and what is necessary to configure them.



The first thing we will need to work with tiles is to add a library (.jar file) to our project. In our case, as we are handling maven, we will add the following dependency to our pom.xml file:

```
<dependency>
            <groupId>org.apache.struts</groupId>
            <artifactId>struts2-tiles-plugin</artifactId>
            <version>2.5.17</version>
</dependency>
```

This is the library or plug-in that allows us to handle the concept of tiles in our project. The version may vary depending on the one we are driving.

Let's see what else we need to add or modify in our project to handle the concept of tiles.

WEB.XML FILE

Ejemplo de modificación del archivo web.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app ...>

    <context-param>
        <param-name>
                    org.apache.tiles.definition.DefinitionsFactory.DEFINITIONS_CONFIG
        </param-name>
        <param-value>/WEB-INF/tiles.xml</param-value>
    </context-param>

    <listener>
        <listener-class>org.apache.struts2.tiles.StrutsTilesListener</listener-class>
    </listener>
    ...

</web-app>
```

We will also need to add the configuration of a parameter and a Struts tile listener to our web.xml file, which must be created in the webapp/WEB-INF folder of our Web project.

This file contains the configuration of the Struts integration, but now it will also have the definition and association with the Struts Tiles plug-in.

We can see an example of the code that we must add to our web.xml file.

ARCHIVO TILES.XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE ...>
<tiles-definitions>
    <definition name="layout" template="/WEB-INF/content/layout.jsp">
        <put-attribute name="title" value="Plantilla"/>
        <put-attribute name="header" value="/WEB-INF/content/header.jsp"/>
        <put-attribute name="menu" value="/WEB-INF/content/menu.jsp"/>
        <put-attribute name="body" value="/WEB-INF/content/body.jsp"/>
        <put-attribute name="footer" value="/WEB-INF/content/footer.jsp"/>
    </definition>
    <definition name="welcomeTile" extends="layout">
        <put-attribute name="title" value="Welcome"/>
        <put-attribute name="body" value="/WEB-INF/content/welcome.jsp"/>
    </definition>
    <definition name="peopleTile"  extends="layout">
        <put-attribute name="title" value="People"/>
        <put-attribute name="body" value="/WEB-INF/content/people.jsp"/>
    </definition>
</tiles-definitions>
```

We will also need to create the tiles.xml file, which will be deposited in the webapp / WEB-INF folder of our Web project.

The tiles.xml file is where we will define the elements of our layout or template. And this is also where we will define each of the pages that we will use in our system and for each page we will indicate which elements will be reused (all the elements of the template are inherited by default) and which elements will be different.

Subsequently, each of the JSPs that make up the layout must be created.

We have several definitions, for example we can observe the definition of "layout",

Actually the definitions that we will use to show a tile or template to our user will be: "welcomeTile" and "peopleTile", and these pages will be called as a new type of result, but now this result will be tile type. This new type of result can not be indicated by means of annotations, so we will have to create the struts.xml file and indicate this new type of return that our methods of the respective Action class will return.

STRUTS.XML FILE

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ...>
<struts>

    <constant name="struts.convention.default.parent.package"
              value="web.actions"/>


    <package name="web.actions" extends="struts-default">
        <result-types>
            <result-type name="tiles"
                         class="org.apache.struts2.views.tiles.TilesResult" />
        </result-types>
    </package>

</struts>
```

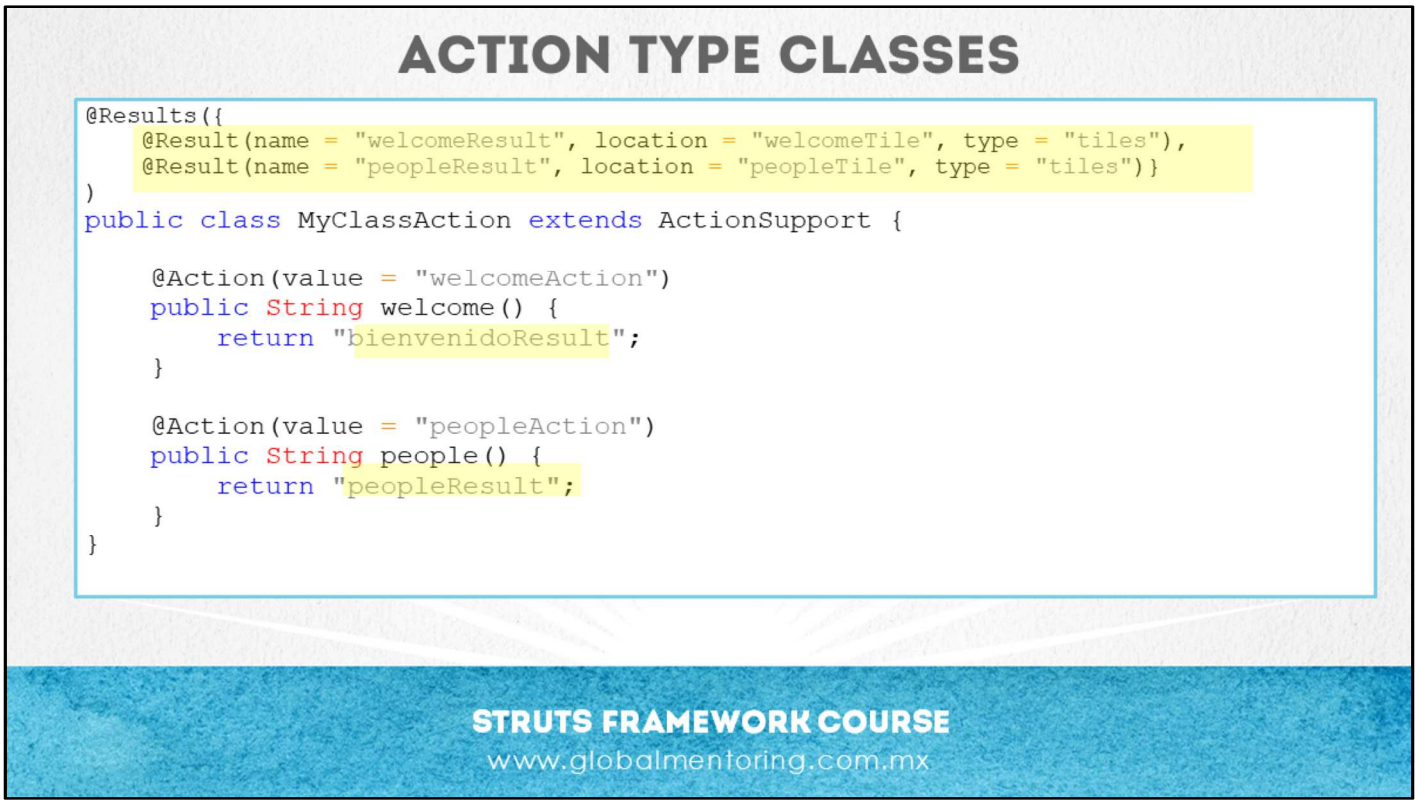
As we mentioned, to indicate the new type of return we need to use our struts.xml file since it is not possible to indicate it by means of annotations.

We can see that we are defining a constant first, which is the name of the package where are the Action type classes that will return the tile type names that we have defined in the previous file.

Later with this previously defined package constant, we create a package to indicate the new type of return called tiles, it can be any name, but for a better reference it is the name that we will use from our class of type Action. Finally we indicate the type of class that we will use, which is of type:

`org.apache.struts2.views.tiles.TilesResult`

And with this we are ready for our actions to return the "tile" type and be recognized by Struts 2.



Finally, our Action classes are ready to return a tile result.

This class called MyClassAction has two methods of type action, the first processes the url "/welcomeAction" and the second processes the urls that match "/peopleAction".

Once this action has been processed, each of these methods returns a tile type that has been configured with a result type annotation at the beginning of the class.

We can see that the @Result type annotation has 3 elements.

1. name = "welcomeResult" is the string that matches the string returned by the action method.
2. location = "welcomeTile" is the name of the tile definition that was added in the tiles.xml file
3. type = "tiles" is the new type that was specified in the struts.xml file

Let's see now how to configure the layout.jsp template that is the one that contains the Struts template definition.

JSP LAYOUT

```
<%@ taglib uri="http://tiles.apache.org/tags-tiles" prefix="tiles" %>
<html>
    <head>...</head>
    <body>
        <table border="1">
            <tr>
                <td>
                    <tiles:insertAttribute name="header" />
                </td>
            </tr>
            <tr>
                <td>
                    <tiles:insertAttribute name="menu" />
                </td>
                <td>
                    <tiles:insertAttribute name="body" />
                </td>
            </tr>
            <tr>
                <td>
                    <tiles:insertAttribute name="footer" />
                </td>
            </tr>
        </table>
    </body>
</html>
```

Now we create the file: layout.jsp. This file is the template or base tile, from this JSP is that will extend the other pages that we have defined in the tiles.xml file.

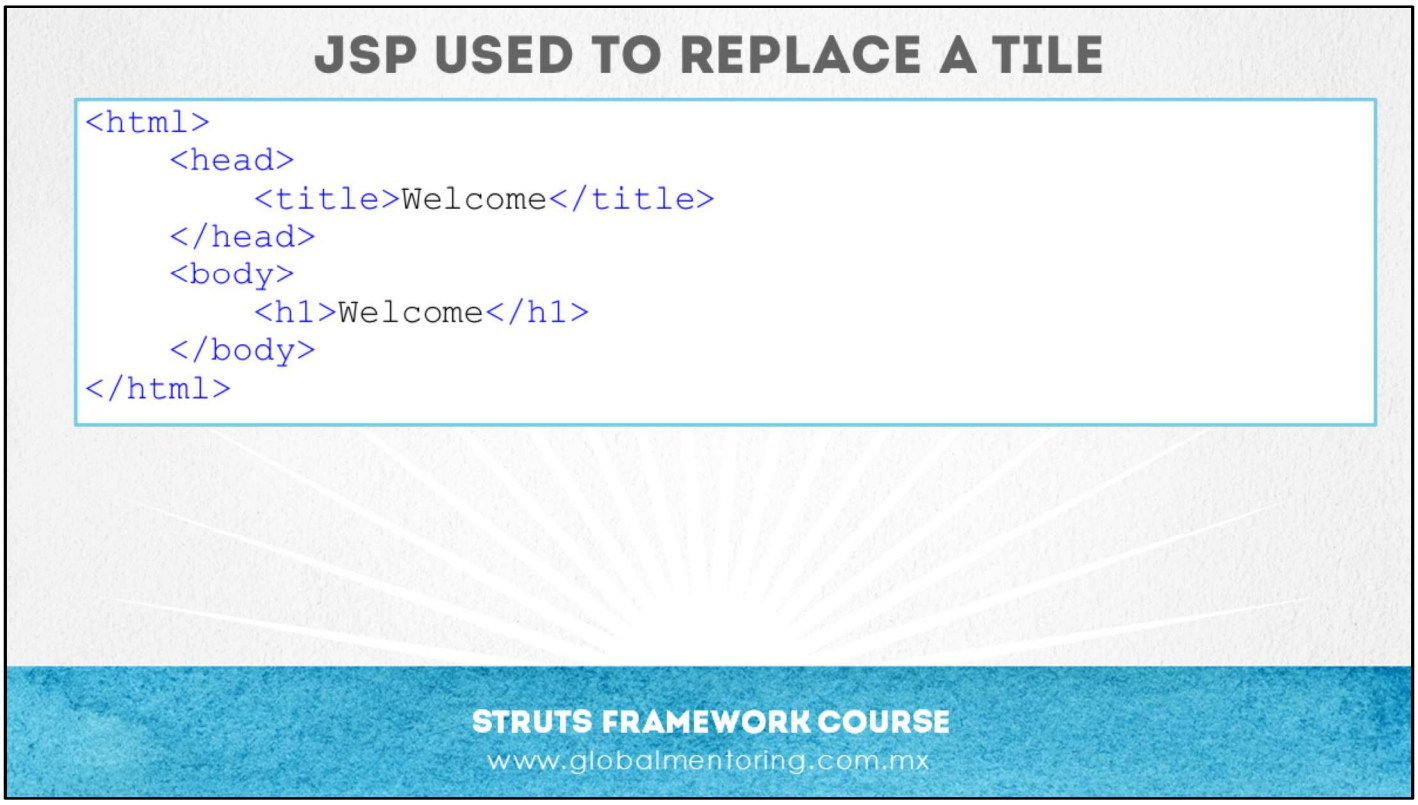
This file layout.jsp is the one that defines the elements that will be inherited by the other pages that use the concept of tiles in the tiles.xml file

Although this JSP is not required to deposit it in the /WEB-INF/content folder since it will not be accessed directly, it could go in another route. But for convenience and to maintain the standard we have handled, we recommend depositing it in the route indicated for the handling of Struts conventions.

Each of the elements defined in the template as <tiles:insertAttribute> will be replaced by the definition that was added in the tiles.xml file. And that's where it was specified which JSP is the one that will take the place of each of the attributes according to the indicated name.

The design of this JSP is totally free, in our case we are adding an HTML table to clearly define each of the elements, but can use any HTML code to adjust and accommodate each of the elements of the template, add more elements, or less, everything depends on the requirements of your web system.

To remember which are the elements that will be replaced by the respective JSPs we can remember the contents of the tiles.xml file. In this file is where we can see how each of the header, menu, body and footer elements will be replaced by each of the JSPs indicated in the tiles.xml file



All the JSPs that the Tiles template carries are normal JSPs, which can carry any kind of valid code in a JSP, for example the JSP of header.jsp, footer.jsp, menu.jsp, etc. These are the JSPs that we will reuse in our template.

But the JSPs that will be unique will be the ones that we will replace in the body of the template. Although this is not a standard or mandatory is the most normal that we only replace the body of our template and all other elements remain the same, although of course, we can replace all elements of our template with new JSPs if we wish.

The code that we observe is from the file: welcome.jsp. This file contains the code for the welcome page.

This is one of the JSP that will use the definition of tiles, so as we saw in the tiles.xml file, welcome Tiles extends the layout tile that was initially defined, called "layout".

This file is required to be deposited in the folder /WEB-INF/content since it will be accessed as a result of executing the respective action, if we are going to use the Struts 2 conventions.

We will note that this file will be used to replace the body section of our template. All other elements will remain the same, except the title and body of the tile defined for this result. Let's see now an example of the use of Tiles in Struts 2.