Introduction to struts and tiles with a working example

Step by Step example using struts tiles. Example application shows how to list, create, edit and save data. Source code is provide.

As development environment we used eclipse with the plugin myeclipse.

Generals

Author:
Sascha Wolski

http://www.laliluna.de/tutorial.html – Tutorials for Struts, EJB, xdoclet and eclipse.

Date:
November, 8st 2004
Software:
MyEclipse 3.8
Jboss 3.2.5
Source code:
Source code for this tutorial can be downloaded here.

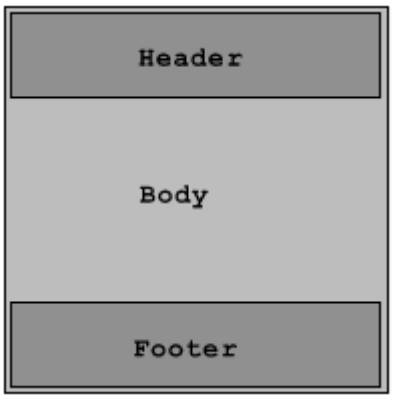
The PDF version is located here.

Introduction

This Tutorial explain how to use struts tiles with a simple example.

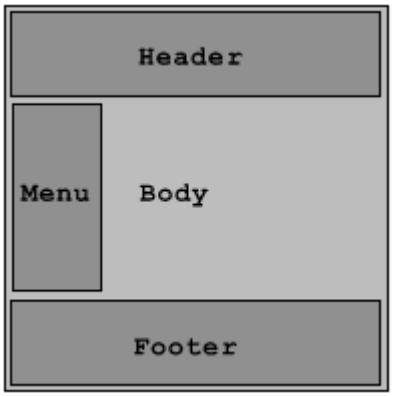
What is struts

Consider a example library application whose web page layout has a header, body and footer. On this example i will explain what is tiles and when you will be use it.



The developer have two alternativ ways to create the layout. He add the header and footer to each page of the site or he use the command `<jsp:include>`, to include the header and footer into the pages. In the first way all pages contains the header and footer source code. When the header or footer will changed, the developer have to change all pages. In the second way the header and footer are placed in seperated files, so the files can be included into the pages where they will be needed.

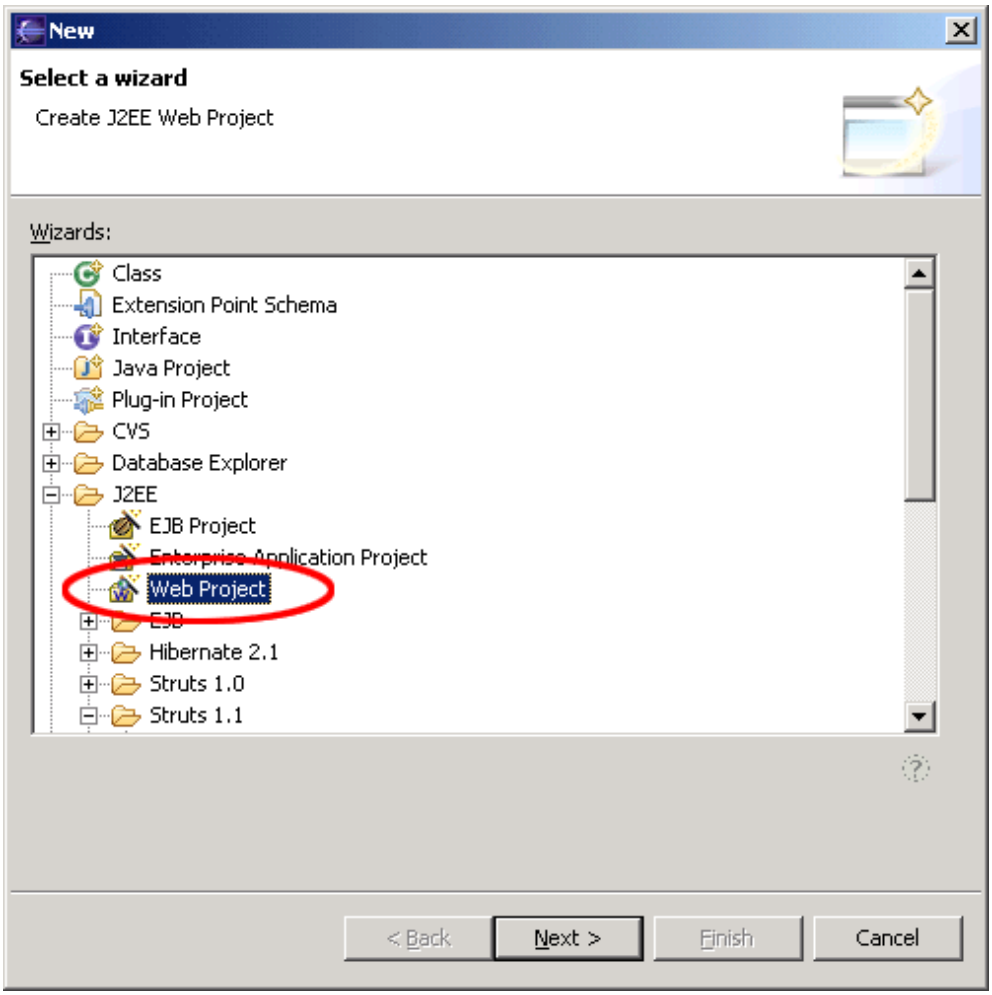
When he change the layout and add a menu to the library application, shown by the picture below, the developer have to change all pages, because he have to add the inlude command of the menu in each page. On this situation tiles is the best way to develope the page layout, thus the developer don`t have to change each page.



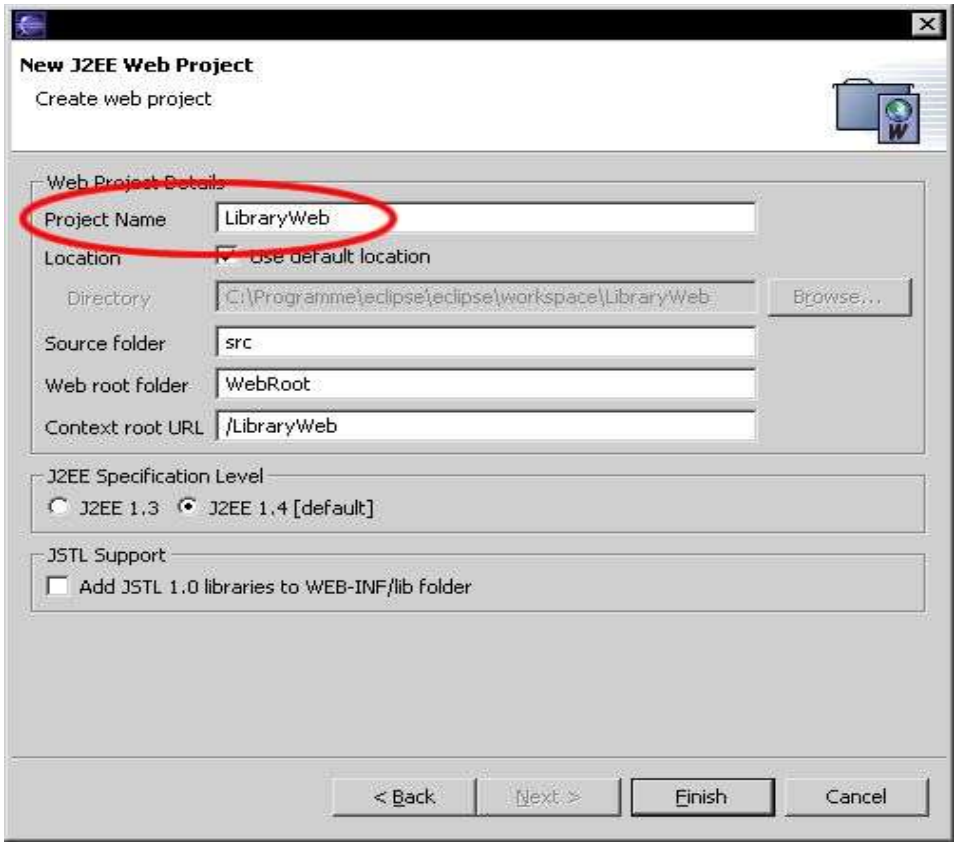
Tiles use a seperate layout file, that contains the container of the layout. When the layout will be changed only the layout file and the tiles configuration files have to change by the developer. That will save many time on large applications.

Your first tiles application

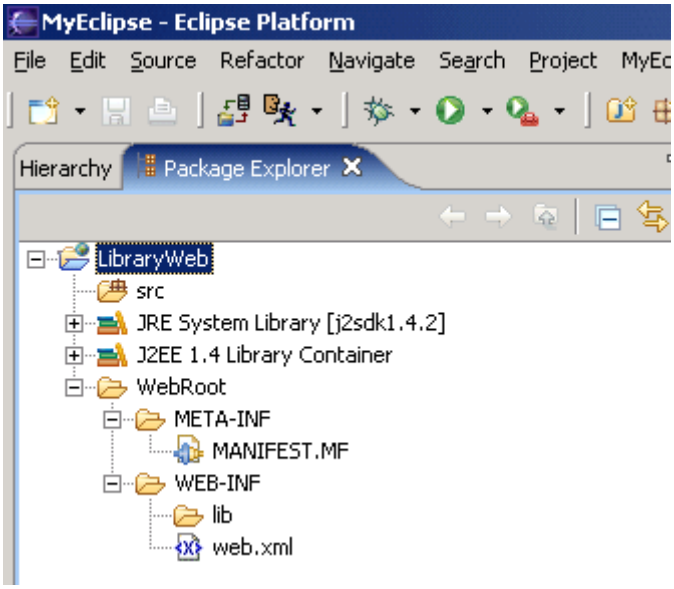
Create a new struts project with `File > New > Project` or use the shortcut `Strg + n`. Select the Wizard in J2EE `Web Project`.



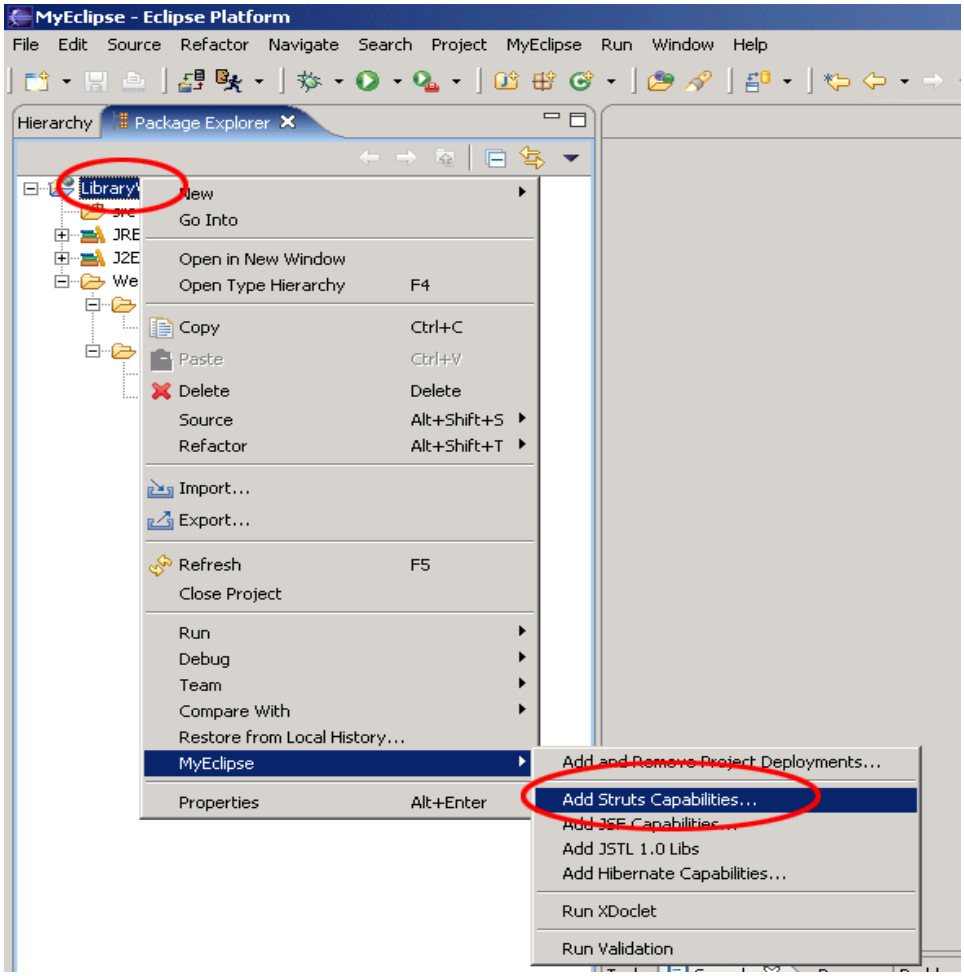
Create a nice name for your project



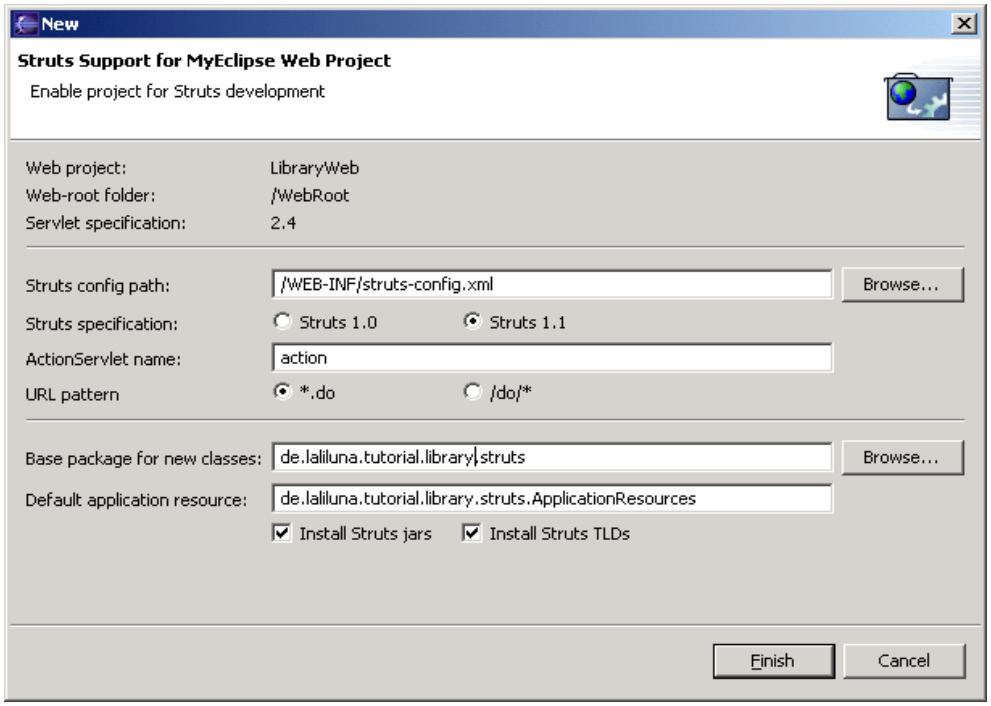
After creating the project, your Package Explorer looks like the picture below.



For now your project is a normal J2EE project, so we need to add the struts capabilityies. Right click on the project and add the capabilityies for struts with `Add Struts Capabilityies.`

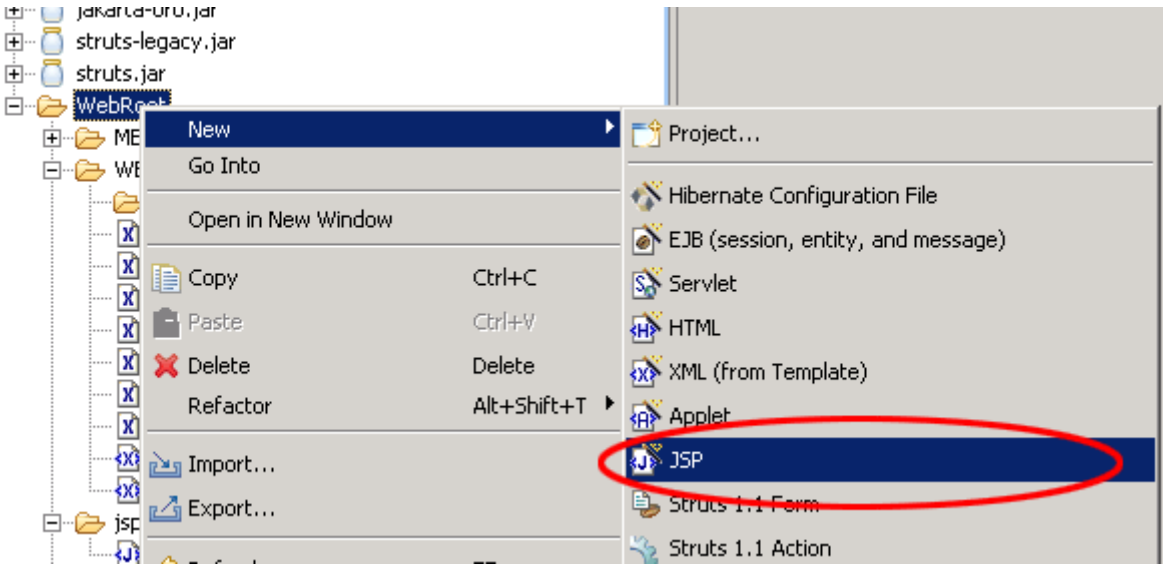


`Change the properties Base package for new classes and Default application resource`

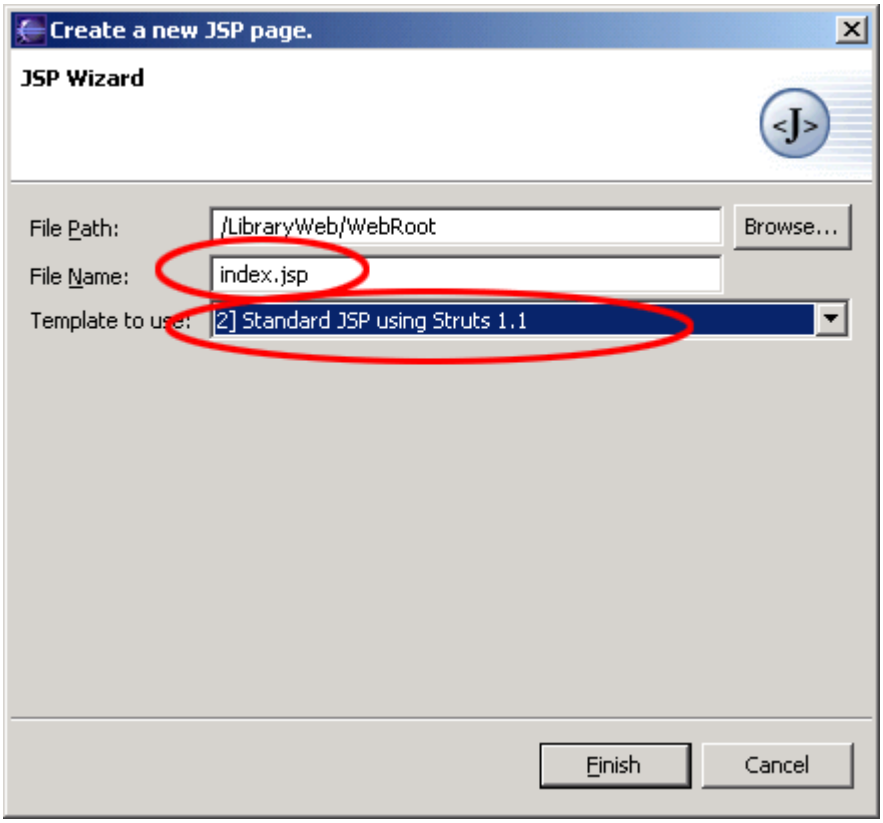


Create a default page

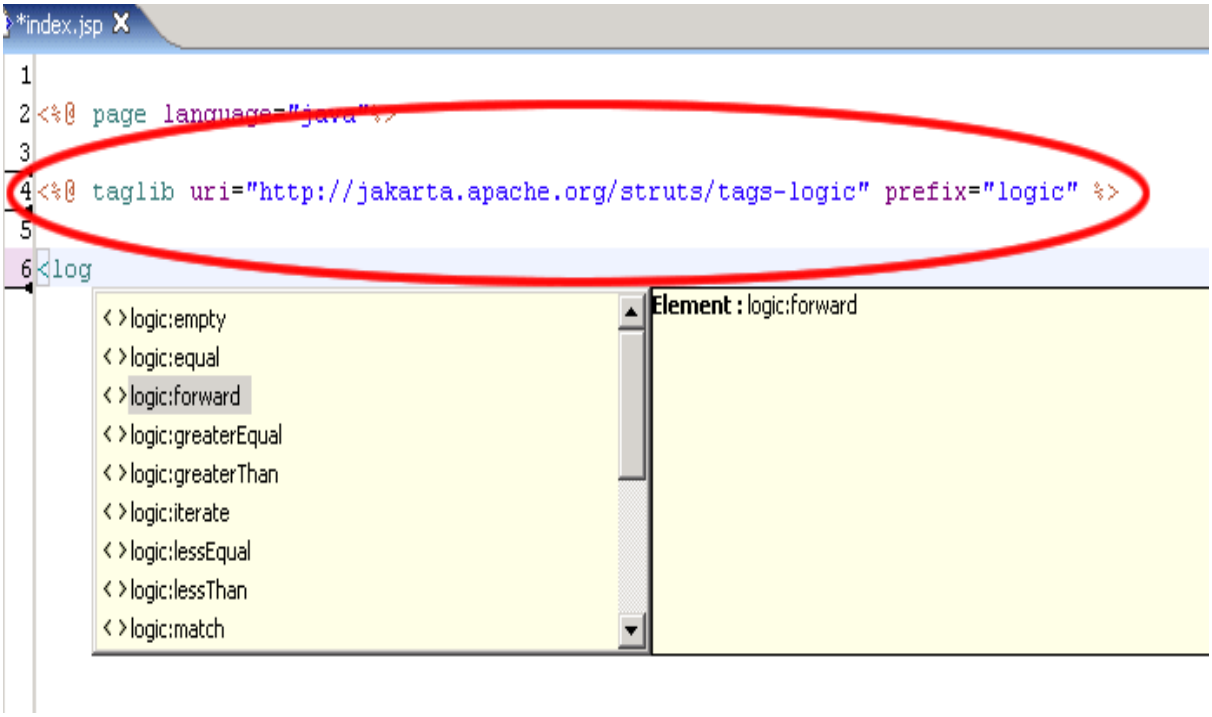
Ok, now we want to create a default page. Right click (yes again) on the Folder `WebRoot` in the Project and choose `New > JSP`.



Set the name to `index.jsp` and choose on `template to use > Standard JSP using Struts 1.1` MyEcplise will use the template to create the JSP File.



You will find the file `index.jsp` in the folder `WebRoot` of the project. On the top of the file you will find the struts tag libraries. These includes will be use to access the tags of struts. In your case we only need the logic tag library.



Insert the following line below the included logic tag.

```
<logic:forward name="welcome" />
```

This line will arranges struts to find a forward with the name `welcome`. If the application don´t find this forward it will leads an error.

Add the tiles functionality

Open the file `struts-config.xml` in the folder `WEB-INF` and replace the following line

```
<controller bufferSize="4096" debug="0" />
```

with

```
<controller processorClass="org.apache.struts.tiles.TilesRequestProcessor"
bufferSize="4096" debug="0" />
```

Below the line

```
<message-resources
parameter="de.laliluna.tutorials.library.struts.ApplicationResources" />
```

add

```
<plug-in className="org.apache.struts.tiles.TilesPlugin">
  <set-property property="definitions-config" value="/WEB-INF/tiles-defs.xml" />
  <set-property property="moduleAware" value="true" />
  <set-property property="definitions-parser-validate" value="true" />
</plug-in>
```

Struts will load the tiles - controller and the tiles - plugin

Create the layout file

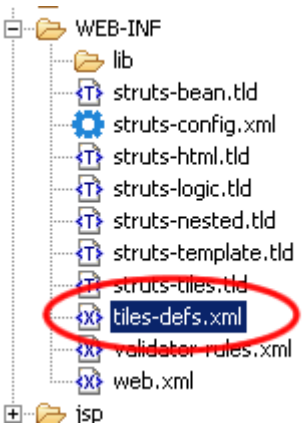
Create a new jsp file in `WebRoot/jsp/` and set the name to `siteLayout.jsp`
Add the following source code.

```
<%@ page language="java"%>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-html" prefix="html" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-tiles" prefix="tiles" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html:html locale="true">
  <head>
    <html:base />
    <title><tiles:getAsString name="title" /></title>
  </head>
  <body>
  <table border="1" width="600" cellspacing="5">
      <tbody>
            <tr><td colspan="2"><tiles:insert attribute="header" /></td></tr>
            <tr>
                  <td width="200"><tiles:insert attribute="navigation" /></td>
                  <td width="400"><tiles:insert attribute="body" /></td>
            </tr>
            <tr><td colspan="2"><tiles:insert attribute="footer" /></td></tr>
      </tbody>
  </table>
  </body>
</html:html>
```

The black marked tags are the placeholder for title, header, navigation, body and footer. The first tag in the listing above is `<tiles:getAsString ...>`. This tag retrieves the title as string and insert it there. The following tags `<tiles:insert ..>` will be used to insert different jsp files, strings and action mappings into the `siteLayout.jsp`.

Create the tiles definitions

Create the file `tiles-defs.xml` in the folder `WEB-INF`, if it not exists.



Open the file and add the following source code.

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

 <!DOCTYPE tiles-definitions PUBLIC
       "-//Apache Software Foundation//DTD Tiles Configuration 1.1//EN"
       "http://jakarta.apache.org/struts/dtds/tiles-config_1_1.dtd">

<tiles-definitions>
      <!-- Base Tiles Definition -->
      <definition name="base.definition" path="/jsp/siteLayout.jsp">
            <put name="header" value="/jsp/header.jsp" />
            <put name="navigation" value="/navigation.do" />
            <put name="footer" value="/jsp/footer.jsp" />
      </definition>

      <!-- Tiles Definition of welcome page -->
      <definition name="page.welcome" extends="base.definition">
            <put name="title" value="Welcome page" />
            <put name="body" value="/jsp/index.jsp" />
      </definition>

      <!-- Tiles Definition of BookList -->
      <definition name="page.booklist" extends="base.definition">
            <put name="title" value="Book list page" />
            <put name="body" value="/jsp/bookList.jsp" />
      </definition>
</tiles-definitions>
```

Within the tag block `<tiles-definitions>` you add the different tiles definitions. With the parameter `path`, of the base definition, you can set a jsp file, that contains the layout of the web site. Within the tag of the base definition, with `put` you set jsp files, strings and action mapping, so you can call them in the layout file. Each of the next defintions extends from the base.definition with the parameter `extends="base.definition"`. They will however reuse the header, navigation and footer from the `base.definition`. Within the extends tiles definitions you only add the jsp files, strings and action mappings that are different from the `base.definition`.

Create the jsp files

Now create the needed jsp files in the folder `WebRoot/jsp/`. Right click on the project > New > JSP.

header.jsp

navigation.jsp

footer.jsp

index.jsp

bookList.jsp

We only need the struts tag libraries on the top of the file, because the container of the web site layout is placed in the `siteLayout.jsp`.

```
<%@ page language="java"%>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean" prefix="bean" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html" prefix="html" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic" prefix="logic" %>
```

Open the file Datei `header.jsp` in the folder `WebRoot/jsp/`.
Change the content of the file to.

```
<%@ page language="java"%>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean" prefix="bean" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html" prefix="html" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic" prefix="logic" %>

That is the header page.
```

Open the file `footer.jsp` and change the content to.

```
<%@ page language="java"%>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean" prefix="bean" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html" prefix="html" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic" prefix="logic" %>

That is the footer page.
```

Repeat it with the file `index.jsp`.

```
<%@ page language="java"%>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean" prefix="bean" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html" prefix="html" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic" prefix="logic" %>

Welcome page.
```

and the file `bookList.jsp`

```
<%@ page language="java"%>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean" prefix="bean" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html" prefix="html" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic" prefix="logic" %>

That is the book list page.
```

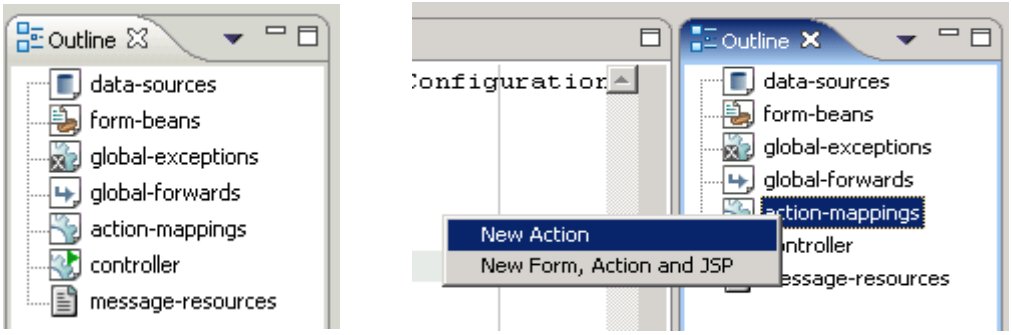
Open the last file `navigation.jsp` and add two links refers to the action mappings, we will create in the next step. `action="welcome"` call the action mapping with the name welcome.

```
<%@ page language="java"%>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean" prefix="bean" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html" prefix="html" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic" prefix="logic" %>

Navigation page
<br />
<html:link action="welcome"> Goto welcome page </html:link>
<br />
<html:link action="bookList"> Goto book list page </html:link>
<br />
<br />
```

The action mapping

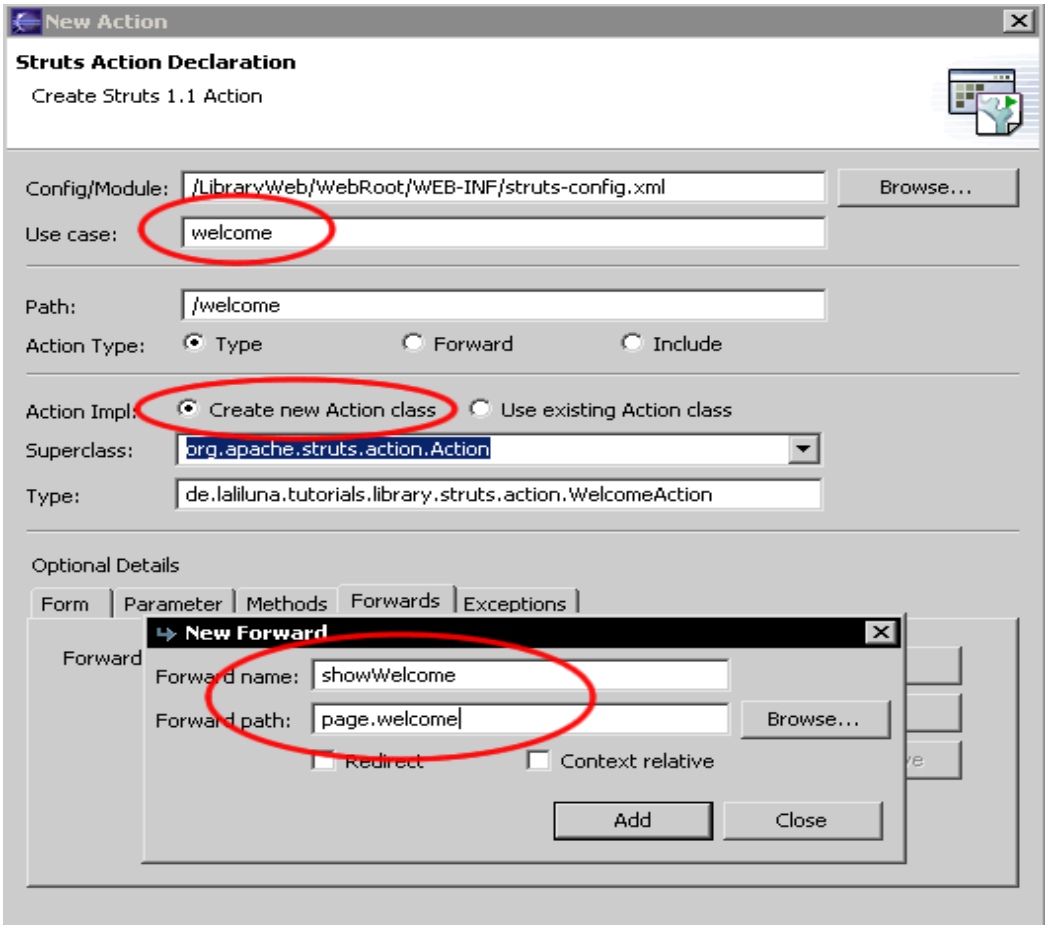
What is a action mapping?
The action mapping is the heart of struts. It managed all actions between the application and the user. You can define which action will be executed by creating a action mapping.

Open the file `struts-config.xml`. Right click on `Action-Mapping > New Action` in the outline window.

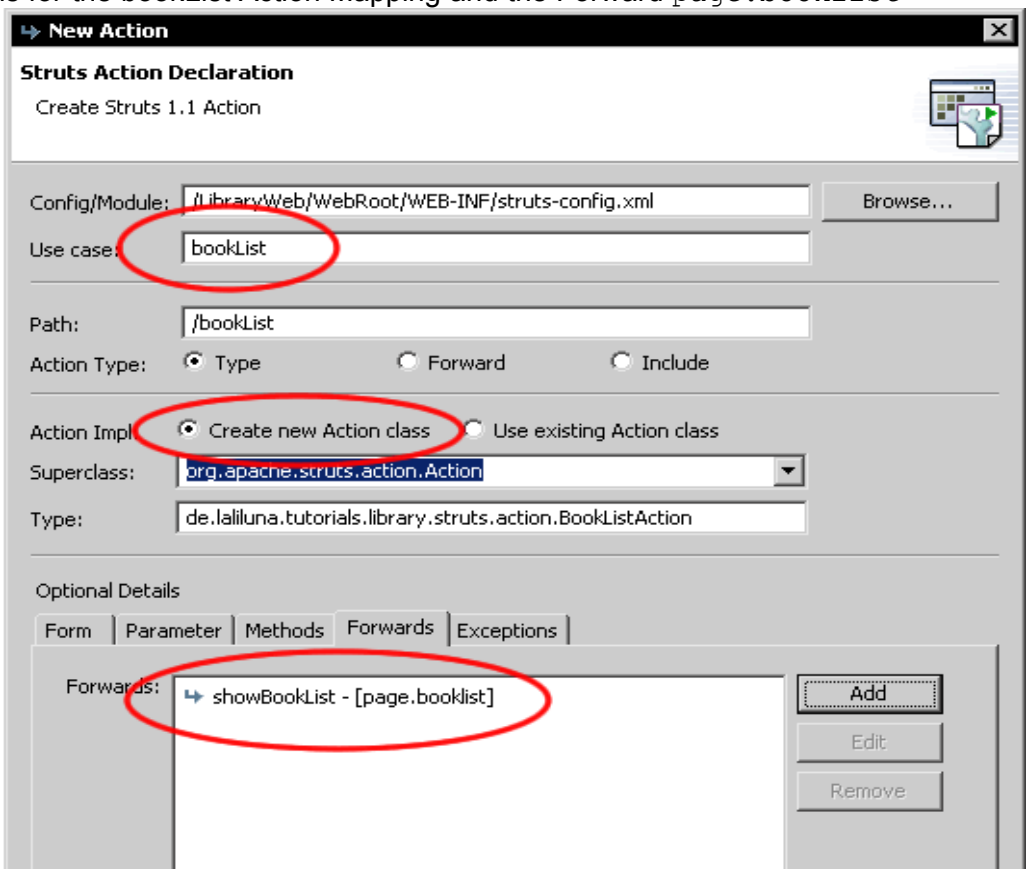


Create three action mappings. The first two mappings refers to the tiles definitions `page.welcome` and `page.booklist`. The last one refers to the navigation.jsp that will display the navigation. We use this mapping in the tiles definition `base.definition`. (see Create the tiles definitions)

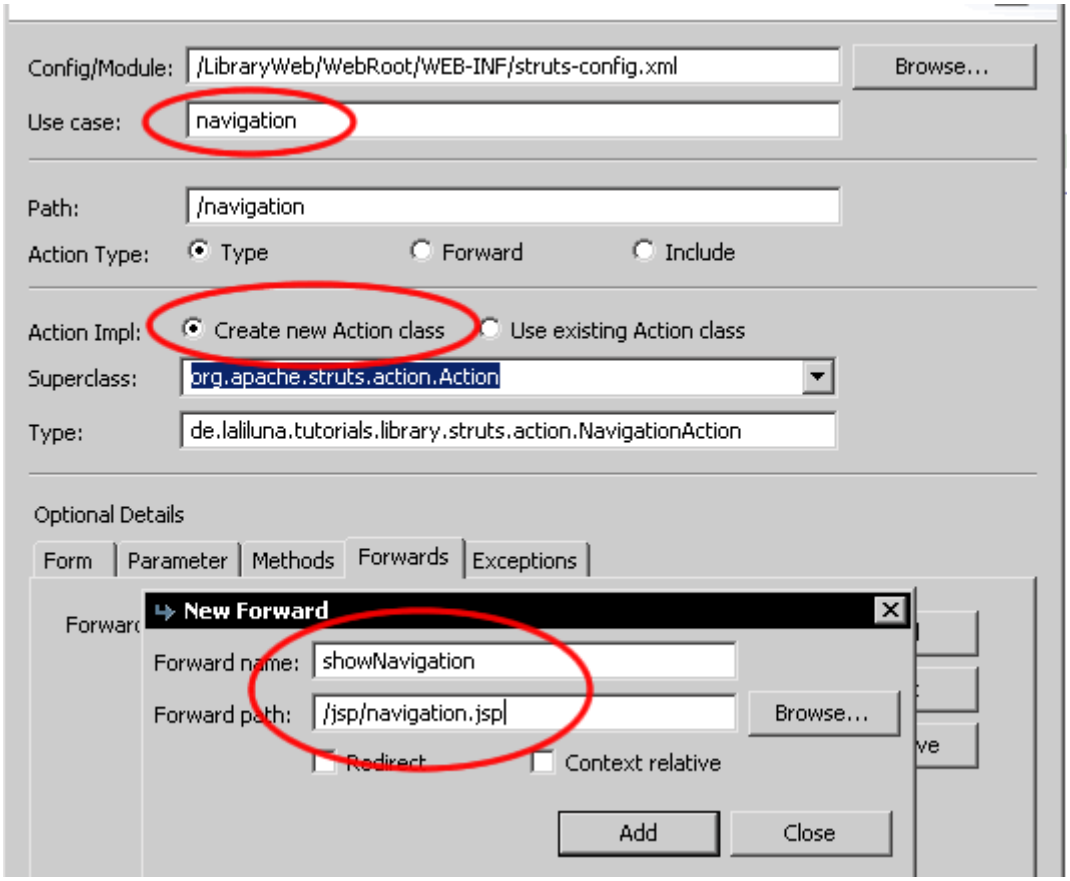
Set the `Use case: welcome` of the action mapping, that will display the welcome page. Choose `Create new Action class`, so MyEclipse will create a action class. On `Forward` add a new Forward to the tiles definition (`page.welcome`) that wil display the welcome page with the header, navigation and footer.



Repeat this for the bookList Action Mapping and the Forward `page.booklist`



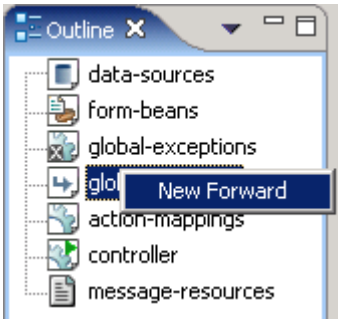
Create the action mapping of the navigation.
On `Forward` we not refer to a tiles definition, we refer directy to the file `navigation.jsp`.



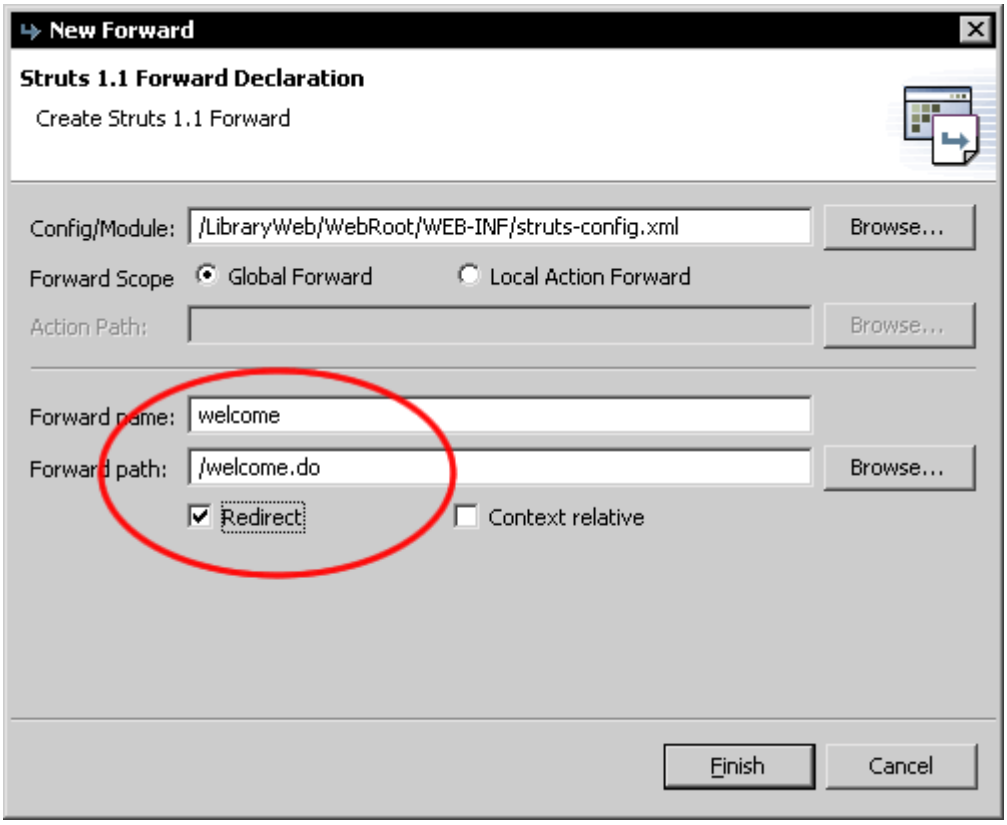
The action forward

What is an action forward?
A action forward can be used to forward to a jsp or action mapping. There are two different action forwards. The global action forward and the local action forward. You can access a global action forward on each jsp or action class. A local action forward can only be accessed by the assigned action class.

Remain in the `struts-config.xml` and the outline window. Create a new forward on global forward with right click.



Choose the `Forward Scope` "Global Forward" and use the name, we have set in the default page. The `Forward path` refers to the action mapping, that will diplay the welcome page.



You find the following in your `struts-config.xml`

```
<global-forwards >
  <forward name="welcome"
           path="/welcome.do"
           redirect="true" />
</global-forwards>

<action-mappings >
  <action path="/welcome"
          type="de.laliluna.tutorials.library.struts.action.WelcomeAction">
          <forward name="showWelcome" path="page.welcome" />
  </action>
  <action path="/bookList"
          type="de.laliluna.tutorials.library.struts.action.BookListAction">
          <forward name="showBookList" path="page.booklist" />
  </action>
  <action path="/navigation"
          type="de.laliluna.tutorials.library.struts.action.NavigationAction">
          <forward name="showNavigation" path="/jsp/navigation.jsp" />
  </action>
</action-mappings>
```

Edit the action classes

Open the action class `NavigationAction.java` in the package `de.laliluna.tutorial.library.action.` Change the method execute.

```
/**
 * Method execute
 * @param mapping
 * @param form
 * @param request
 * @param response
 * @return ActionForward
 */
public ActionForward execute(
 ActionMapping mapping,
 ActionForm form,
 HttpServletRequest request,
 HttpServletResponse response) {

 return mapping.findForward("showNavigation");
}
```

In the next step open the class `WelcomeAction.java` and change the method `execute`.

```
/**
 * Method execute
 * @param mapping
 * @param form
 * @param request
 * @param response
 * @return ActionForward
 */
public ActionForward execute(
      ActionMapping mapping,
      ActionForm form,
      HttpServletRequest request,
      HttpServletResponse response) {

      return mapping.findForward("showWelcome");
}
```

Repeat this on the last class `BookListAction.java`.

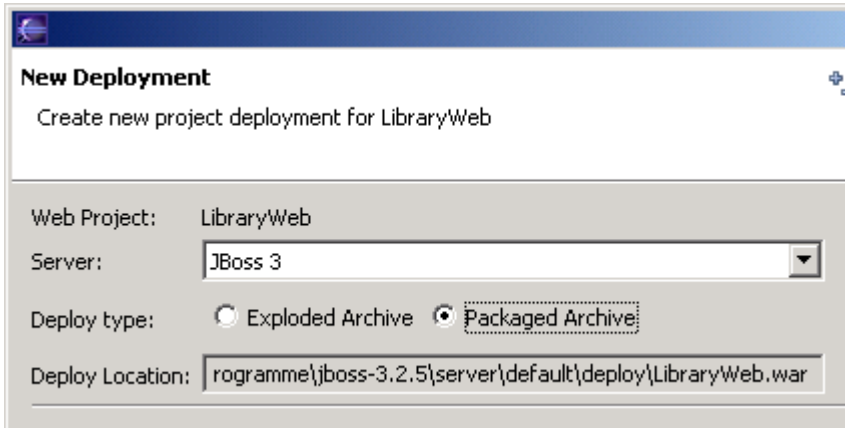
```
/**
 * Method execute
 * @param mapping
 * @param form
 * @param request
 * @param response
 * @return ActionForward
 */
public ActionForward execute(
      ActionMapping mapping,
      ActionForm form,
      HttpServletRequest request,
      HttpServletResponse response) {

      return mapping.findForward("showBookList");
}
```

The command `return mapping.findForward(<forward>);` search for a forward with namen within the clasp and refres to it. We dont add any other functionality to the class,In den Action Klassen werden wir keine weitere Funktionalität hinzufügen, that should not be a component of this tutorials.

Congratulation, thats all.

Test your application

Start the jboss and deploy the project as `Package Archive`



Call the project in your browser http://localhost:8080/LibraryWeb/