STRUTS FRAMEWORK COURSE

TILES WITH STRUTS 2 FRAMEWORK

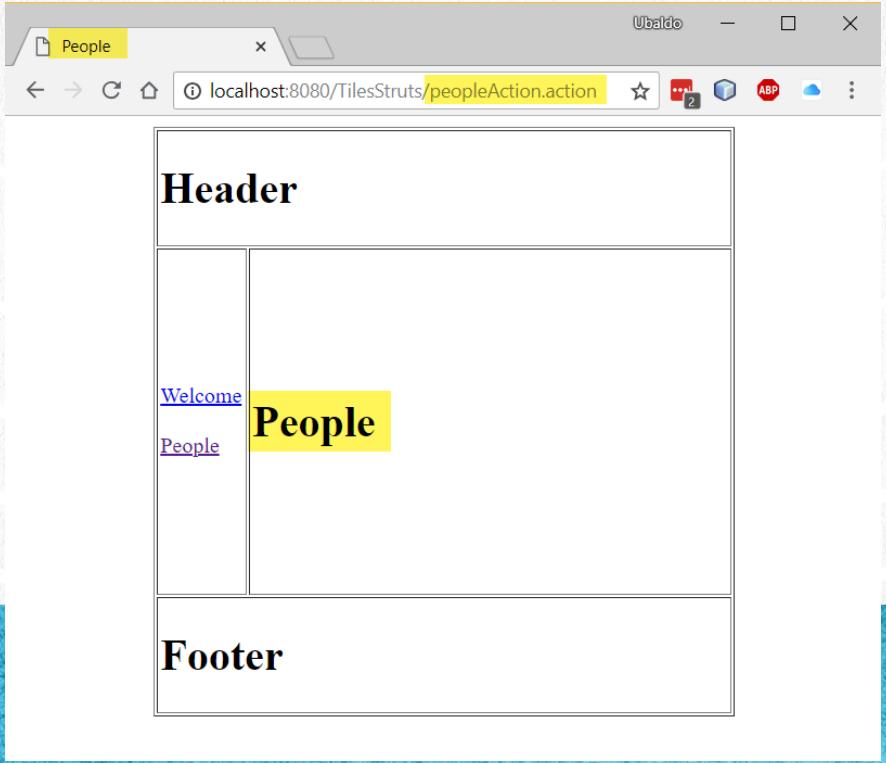
By the expert: Ubaldo Acosta

Global Mentoring

JAVA ONLINE COURSES UNIVERSITY

EXERCISE OBJECTIVE

Create an application to implement the use of Tiles with Struts 2. At the end we should observe the following:



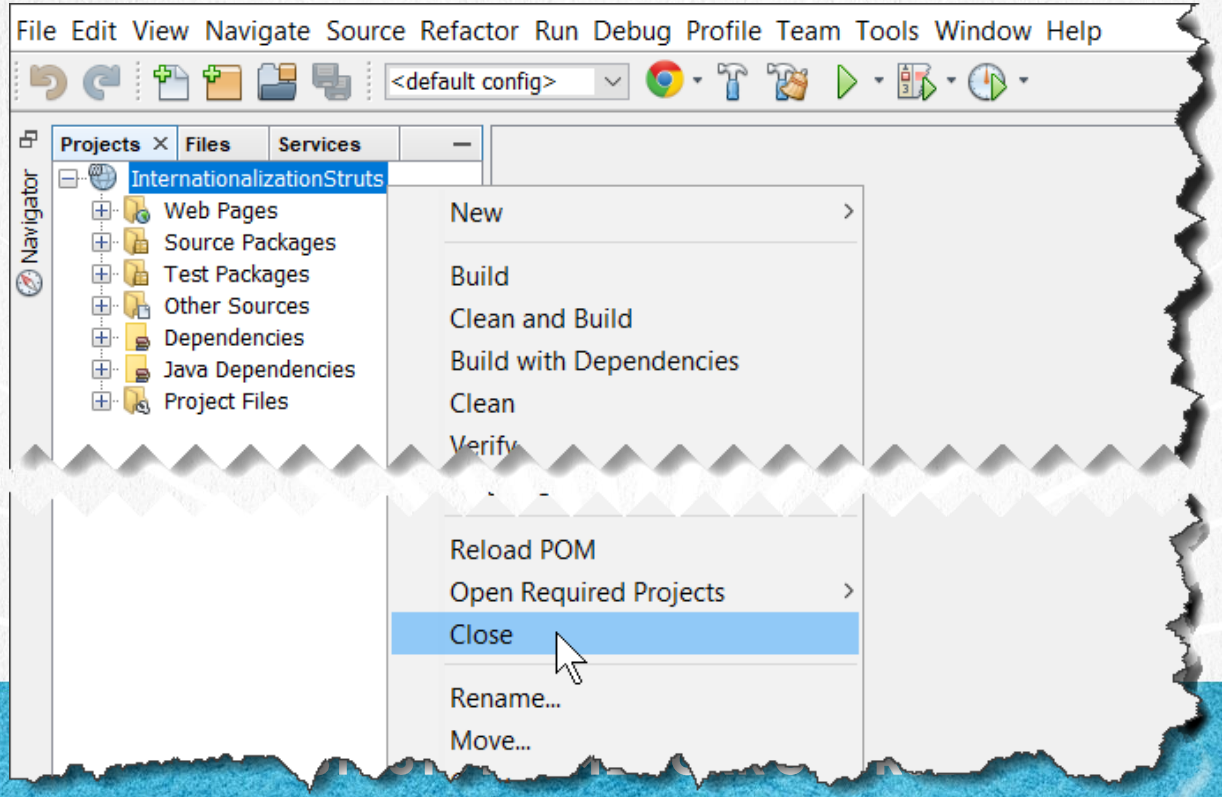
EXERCISE REQUIREMENT

In this project we are going to put into practice the concept of Tiles de Struts.

We will define a main layout (template), and later we will define each of its elements to implement this important concept in Struts 2.

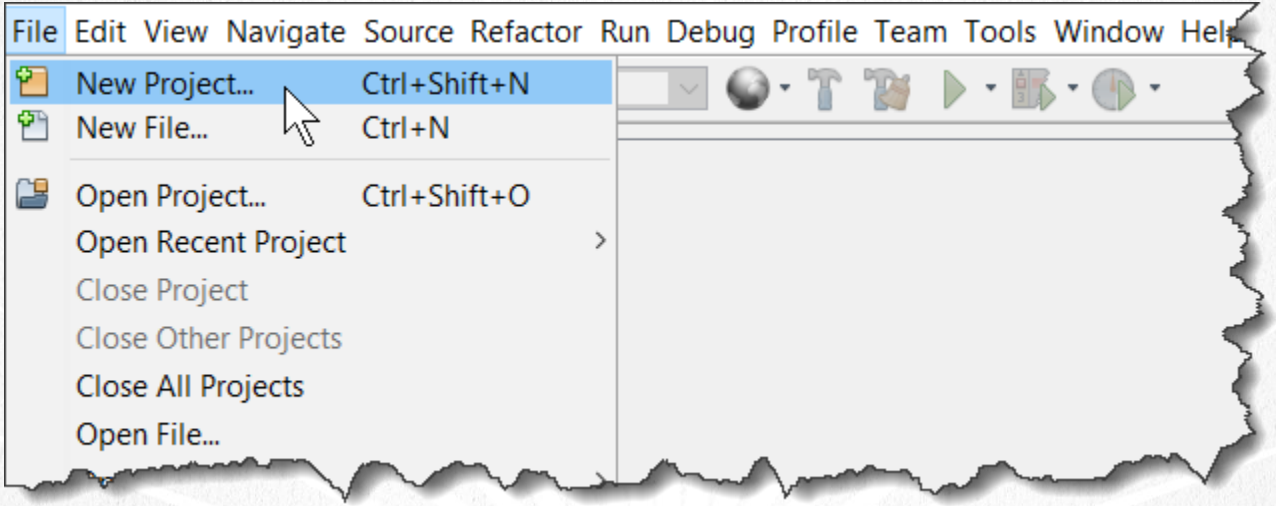
CLOSE PROJECTS THAT WE NO LONGER USE

- We close any other project that we no longer use, if we wish:



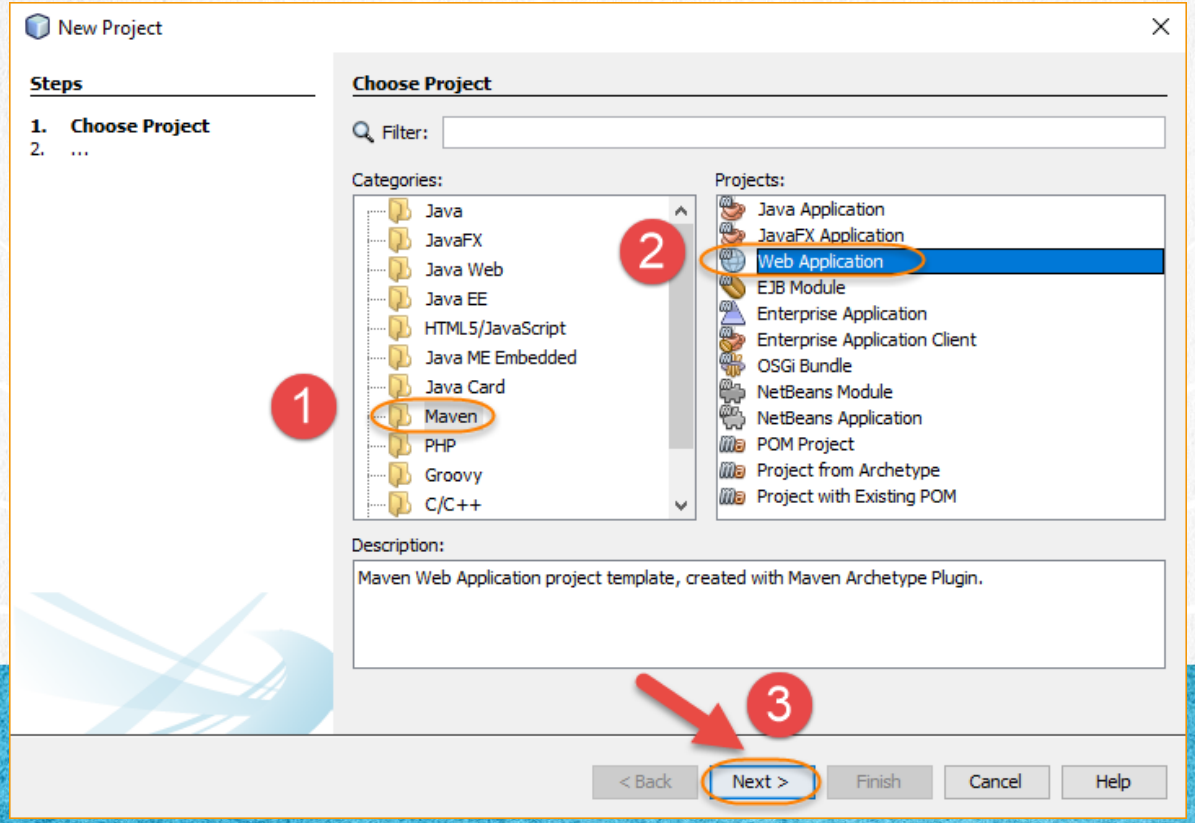
1. CREATE A NEW PROJECT

- We create the new project as shown below:



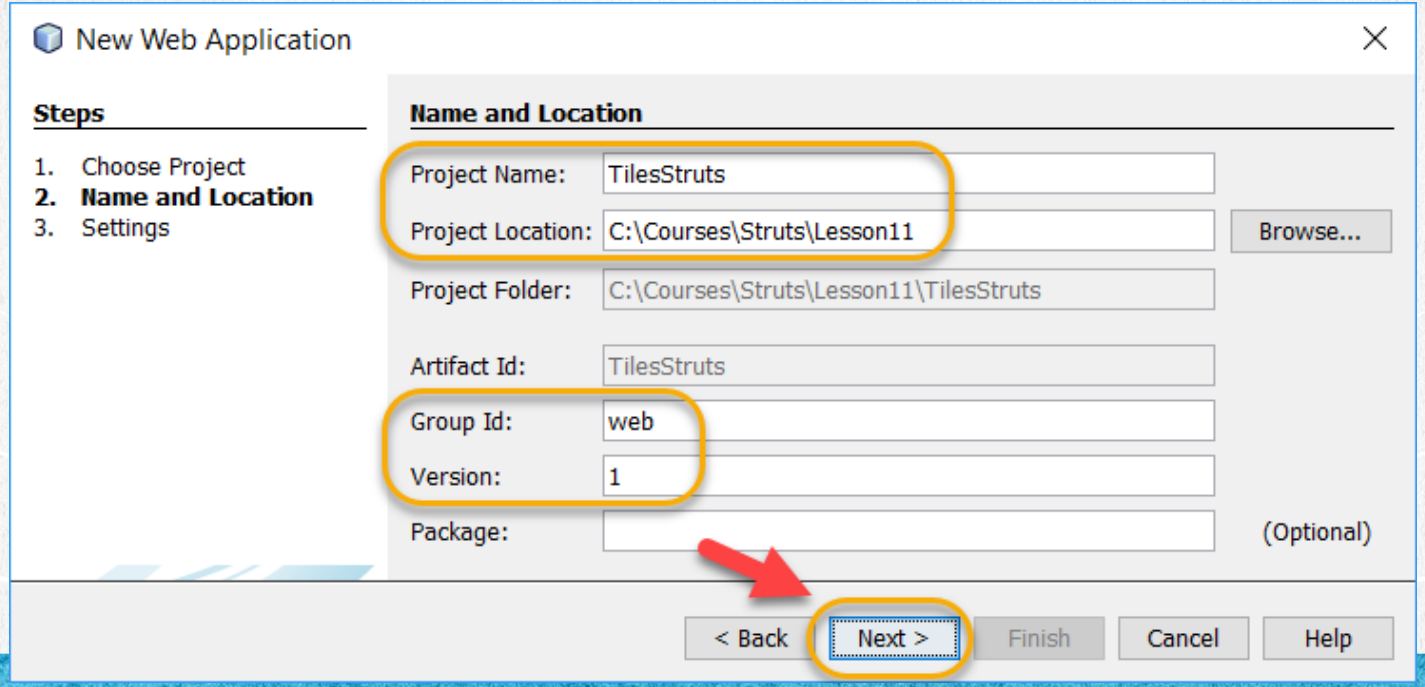
1. CREATE A NEW PROJECT

- We create the new project as shown below:



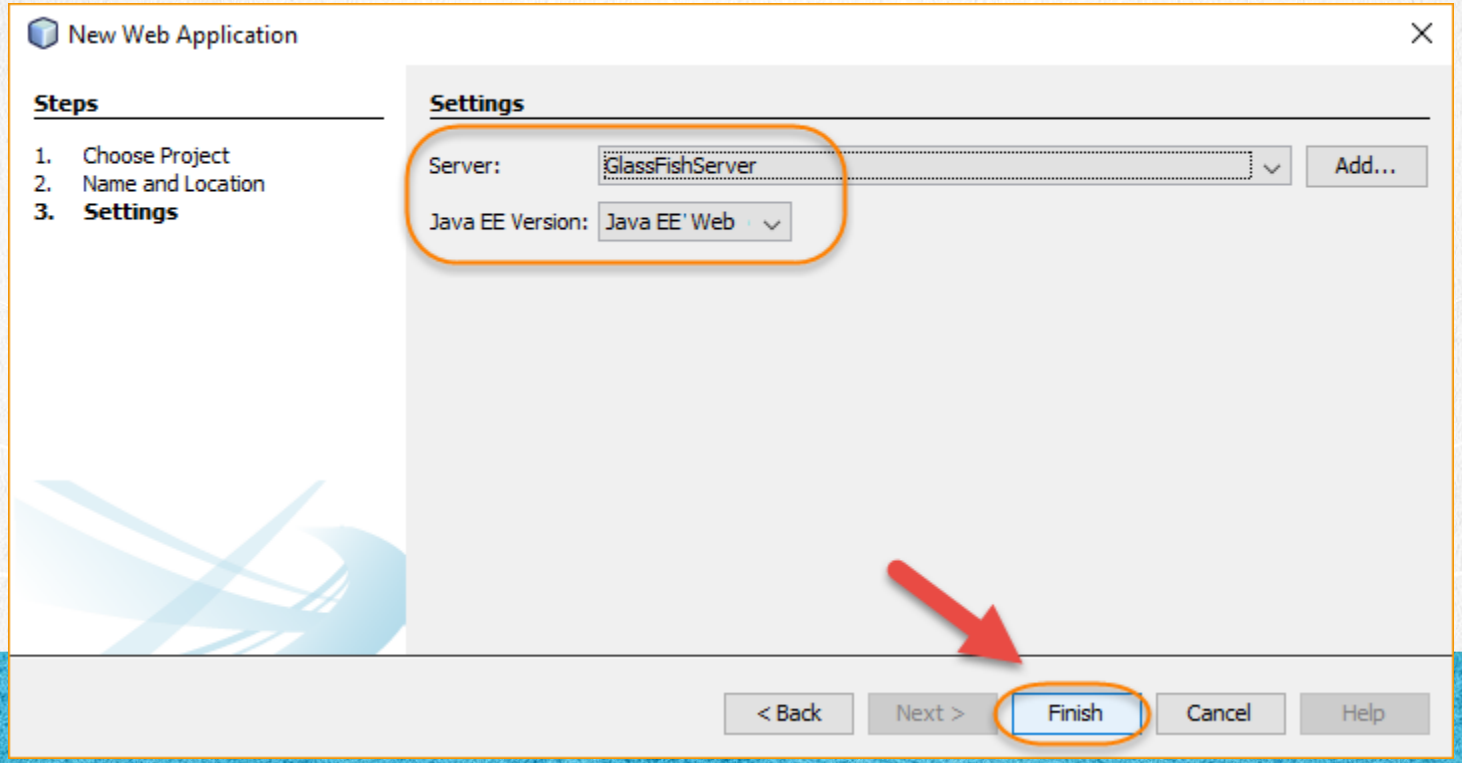
1. CREATE A NEW PROJECT

- We create the new project as shown below:



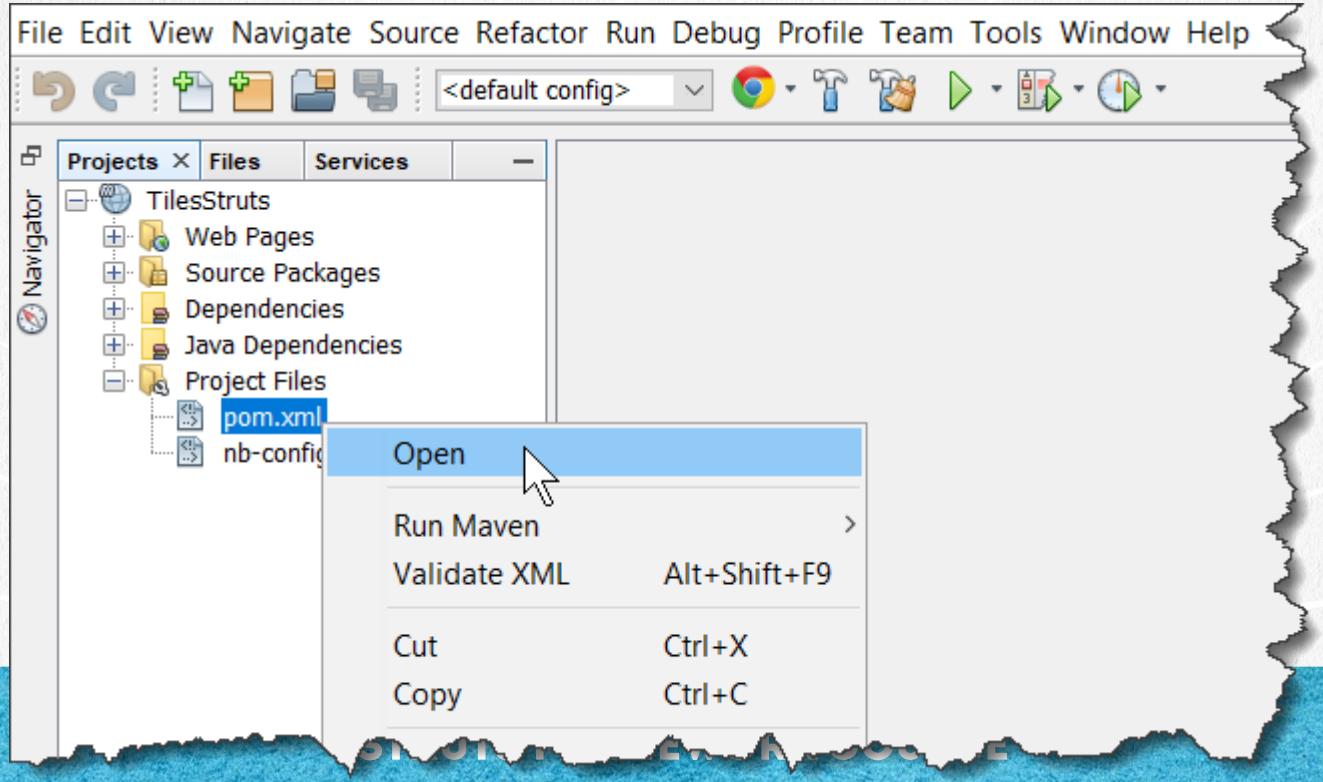
1. CREATE A NEW PROJECT

- We select the values shown:



2. OPEN MAVEN'S POM.XML FILE

- The maven pom.xml file manages the Java libraries we will use:



2. OPEN MAVEN'S POM.XML FILE

- Once opened, we will modify the information completely of this file, with the information provided below:

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

<default config>

Projects | Files | Services

- TilesStruts
  - Web Pages
  - Source Packages
  - Dependencies
  - Java Dependencies
  - Project Files
    - pom.xml
    - nb-configuration.xml

pom.xml [TilesStruts]

Source | Graph | Effective | History

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org
3      <modelVersion>4.0.0</modelVersion>
4
5      <groupId>web</groupId>
6      <artifactId>TilesStruts</artifactId>
7      <version>1</version>
8      <packaging>war</packaging>
9
10     <name>TilesStruts</name>
11
12     <properties>
13         <endorsed.dir>${project.build.directory}/endorsed</endorsed.dir>
14         <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15     </properties>
16
17     <dependencies>
```

3. MODIFY THE FILE

pom.xml:

Click to download

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>web</groupId>
    <artifactId>TilesStruts</artifactId>
    <version>1</version>
    <packaging>war</packaging>

    <name>TilesStruts</name>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <dependencies>
        <dependency>
            <groupId>javax</groupId>
            <artifactId>javaee-web-api</artifactId>
            <version>8.0</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>org.apache.struts</groupId>
            <artifactId>struts2-core</artifactId>
            <version>2.5.17</version>
        </dependency>
```

3. MODIFY THE FILE

pom.xml:

Click to download

```
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-api</artifactId>
            <version>2.11.1</version>
        </dependency>
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-core</artifactId>
            <version>2.11.1</version>
        </dependency>
        <dependency>
            <groupId>org.apache.struts</groupId>
            <artifactId>struts2-convention-plugin</artifactId>
            <version>2.5.17</version>
        </dependency>
         <dependency>
            <groupId>org.apache.struts</groupId>
            <artifactId>struts2-tiles-plugin</artifactId>
            <version>2.5.17</version>
        </dependency>
    </dependencies>
```

3. MODIFY THE FILE

pom.xml:

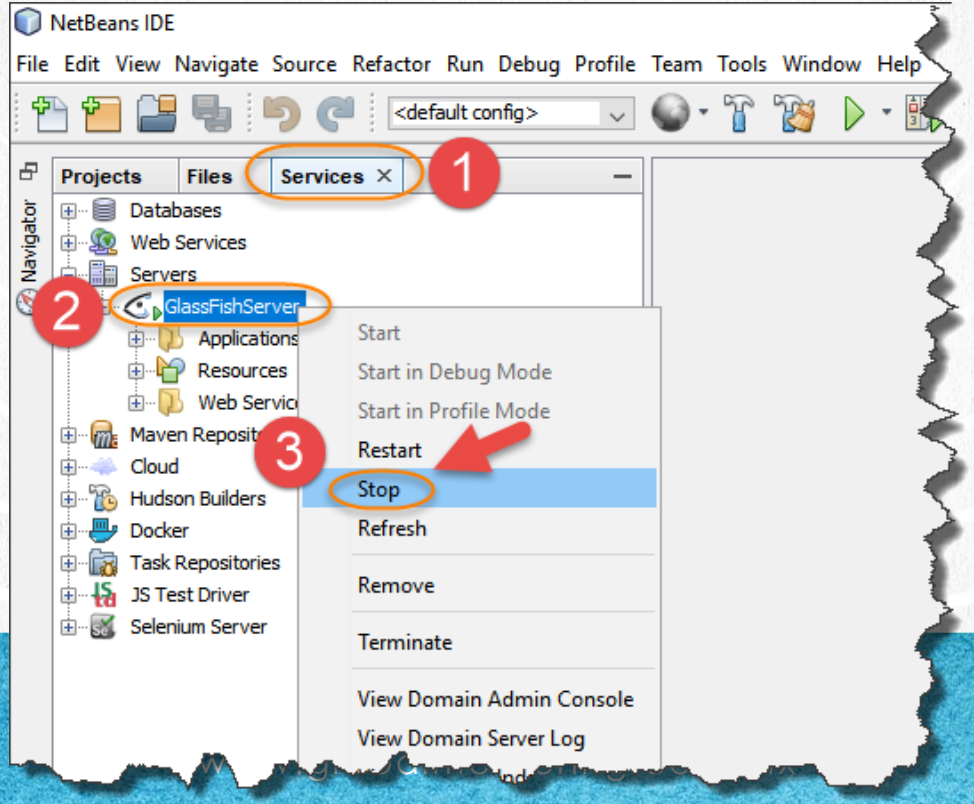
Click to download

```
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-war-plugin</artifactId>
                <version>2.3</version>
                <configuration>
                    <failOnMissingWebXml>false</failOnMissingWebXml>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.7.0</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

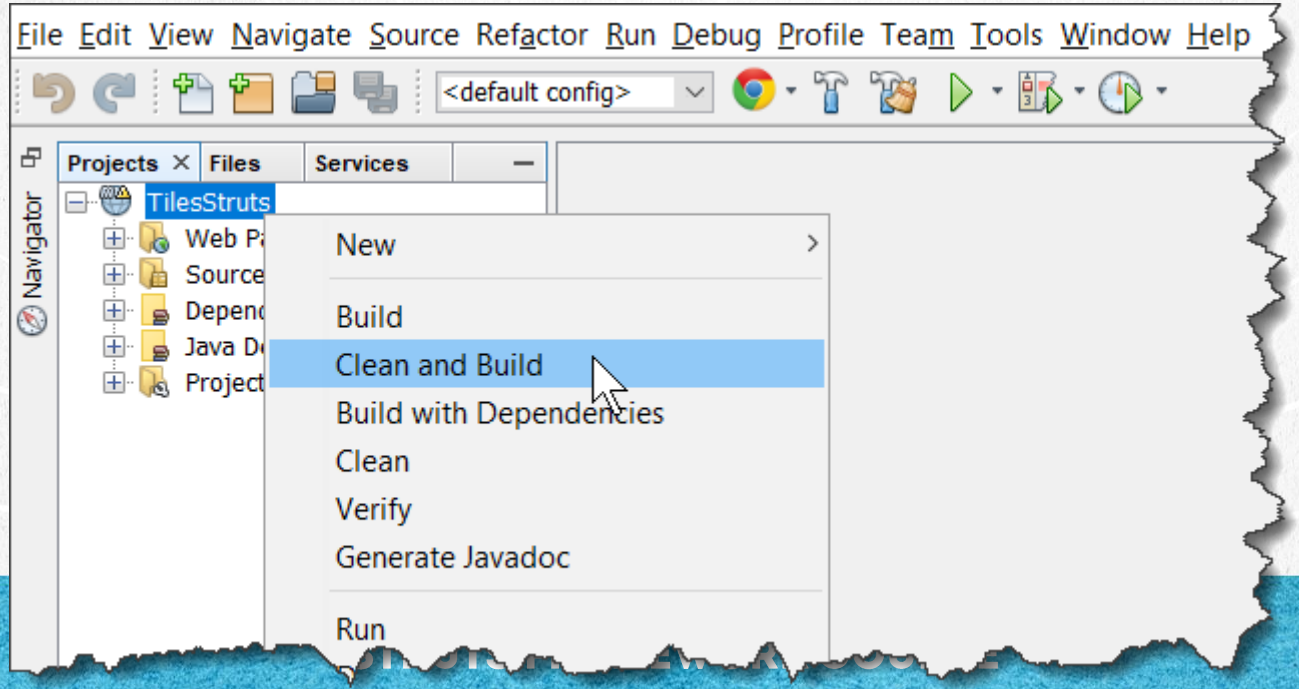
4. STOP GLASSFISH IF IT WAS STARTED

•Before doing Clean & Build of the project to download the new libraries, we verify that the Glassfish server is not started as there may be problems to do the Clean & build process if the server is started. This step is only verification:



5. EXECUTE CLEAN & BUILD

- In order to download the new libraries, we make Clean & Build the project. If for some reason this process fails, you must disable any software such as antivirus, Windows defender or firewall during this process so that the download of Java .jar files is not prevented. Once finished, these services can be activated again. This process may take several minutes depending on your internet speed:



5. EXECUTE CLEAN & BUILD

- If you no longer had to download any library because you could already have all downloaded, the process is faster. In the end we should observe the following:

Output - Build (TilesStruts)

```
--- maven-war-plugin:2.3:war (default-war) @ TilesStruts ---
Packaging webapp
Assembling webapp [TilesStruts] in [C:\Courses\Struts\Lesson11\TilesStruts\target\TilesStruts-1]
Processing war project
Copying webapp resources [C:\Courses\Struts\Lesson11\TilesStruts\src\main\webapp]
Webapp assembled in [1171 msecs]
Building war: C:\Courses\Struts\Lesson11\TilesStruts\target\TilesStruts-1.war

--- maven-install-plugin:2.3.1:install (default-install) @ TilesStruts ---
Installing C:\Courses\Struts\Lesson11\TilesStruts\target\TilesStruts-1.war to C:\Users\user\.m2\repository\web\TilesStruts\1\TilesStruts-1.war
Installing C:\Courses\Struts\Lesson11\TilesStruts\pom.xml to C:\Users\user\.m2\repository\web\TilesStruts\1\TilesStruts-1.pom
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 3.627s
Finished at: Fri Aug 31 18:32:39
Final Memory: 15M/489M
------------------------------------------------------------------------
```

6. CREATE AN XML FILE

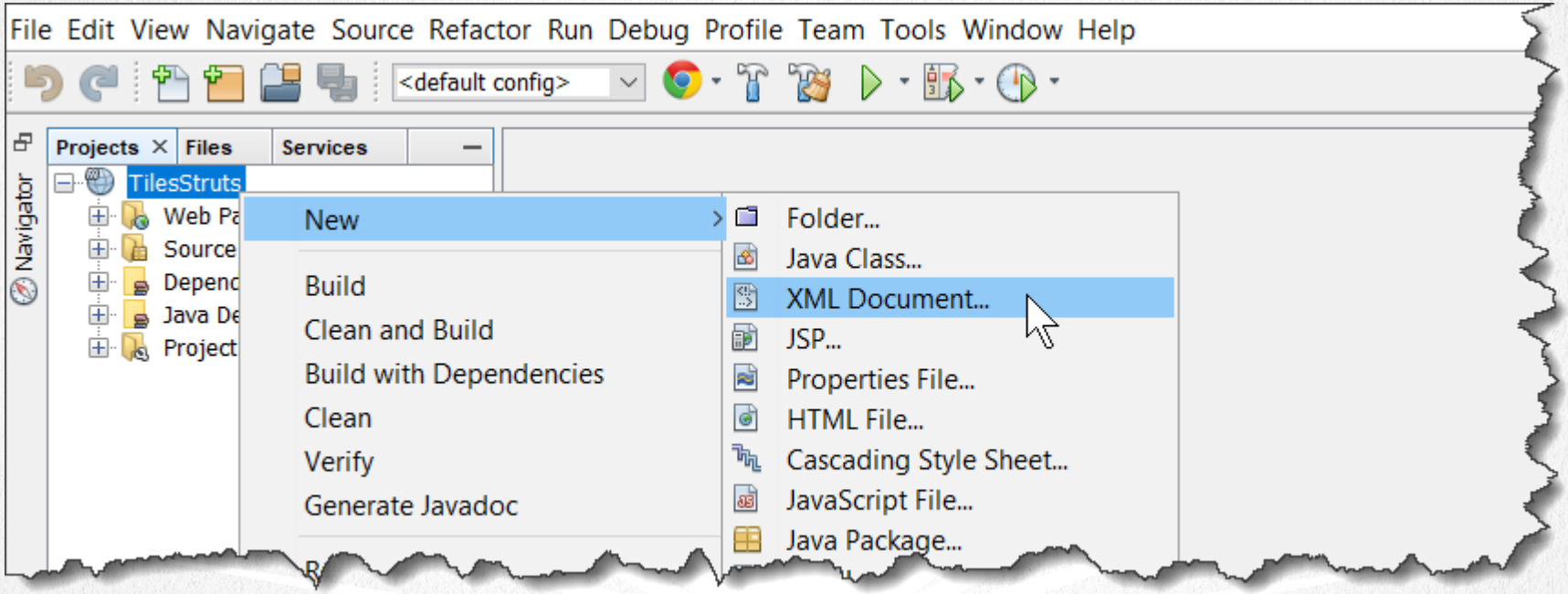
We are going to create the web.xml file below

This file is what allows us to join a Java Web application with the Struts framework, configuring the Struts filter in the web.xml file.

When we add the concept of Tiles, we will also add the tile listener configuration and the tiles.xml file location, this file contains the configuration of the layout (template), as well as the configuration of each page and the reuse of each element of the layout on each page.

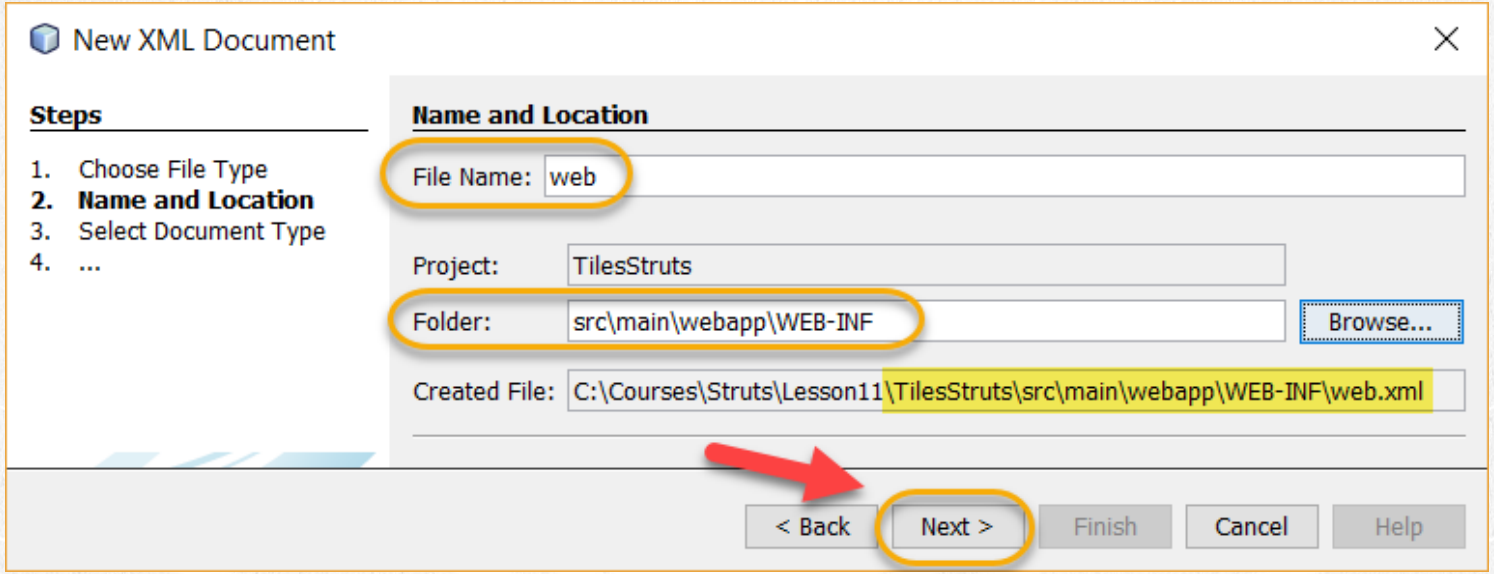
6. CREATE AN XML FILE

- We create the web.xml file and add it to the WEB-INF folder as shown:



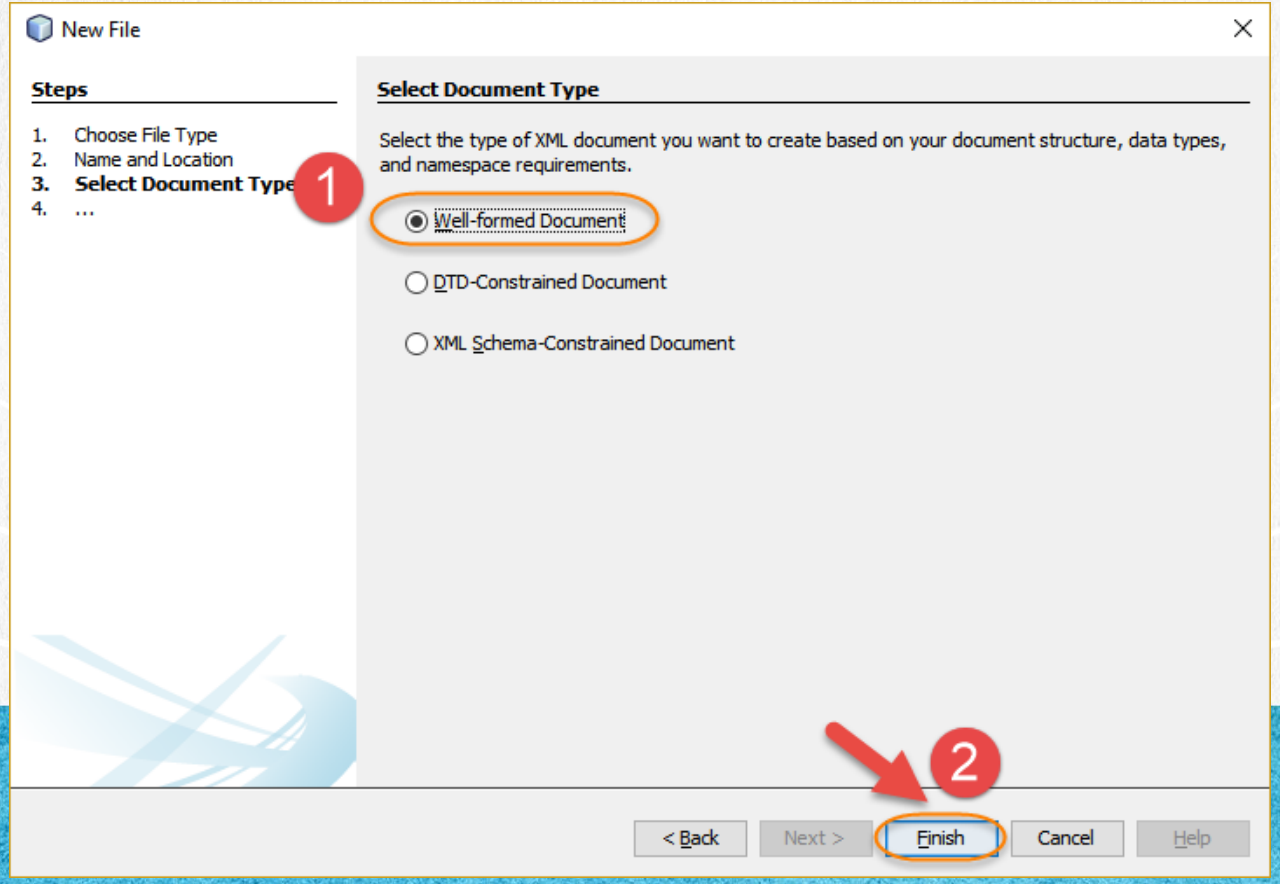
6. CREATE AN XML FILE

- The name of the file is web, it is not necessary to add the extension, it adds it in automatic the IDE since it is an XML type document. Finally we provide the path shown:



6. CREATE AN XML FILE

- We select the indicated type and click on finish.



7. MODIFY THE CODE

web.xml:

Click to download

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="4.0"
         xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
         http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd">

    <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>

    <filter>
        <filter-name>struts2</filter-name>
        <filter-class>org.apache.struts2.dispatcher.filter.StrutsPrepareAndExecuteFilter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>struts2</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
</web-app>
```

8. CREATE AN XML FILE

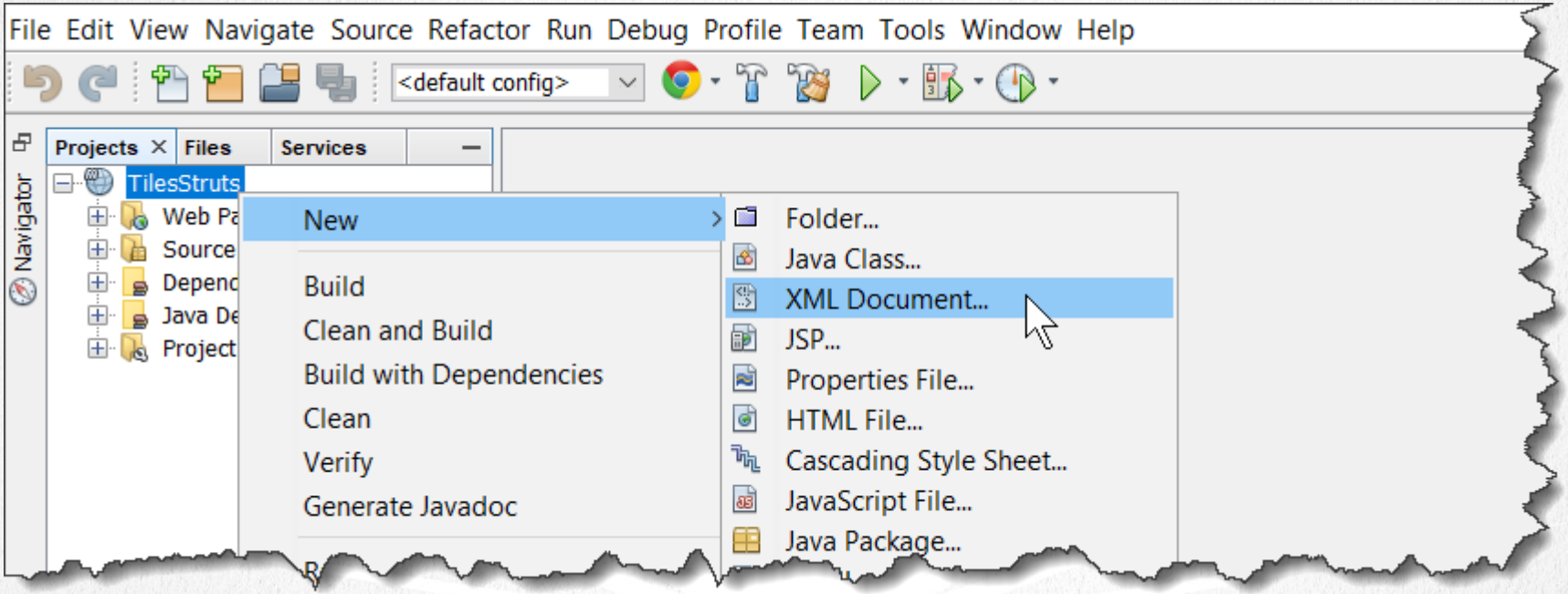
Let's create the tiles.xml file below.

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.

Later we will create each of the JSPs that make up our layout.

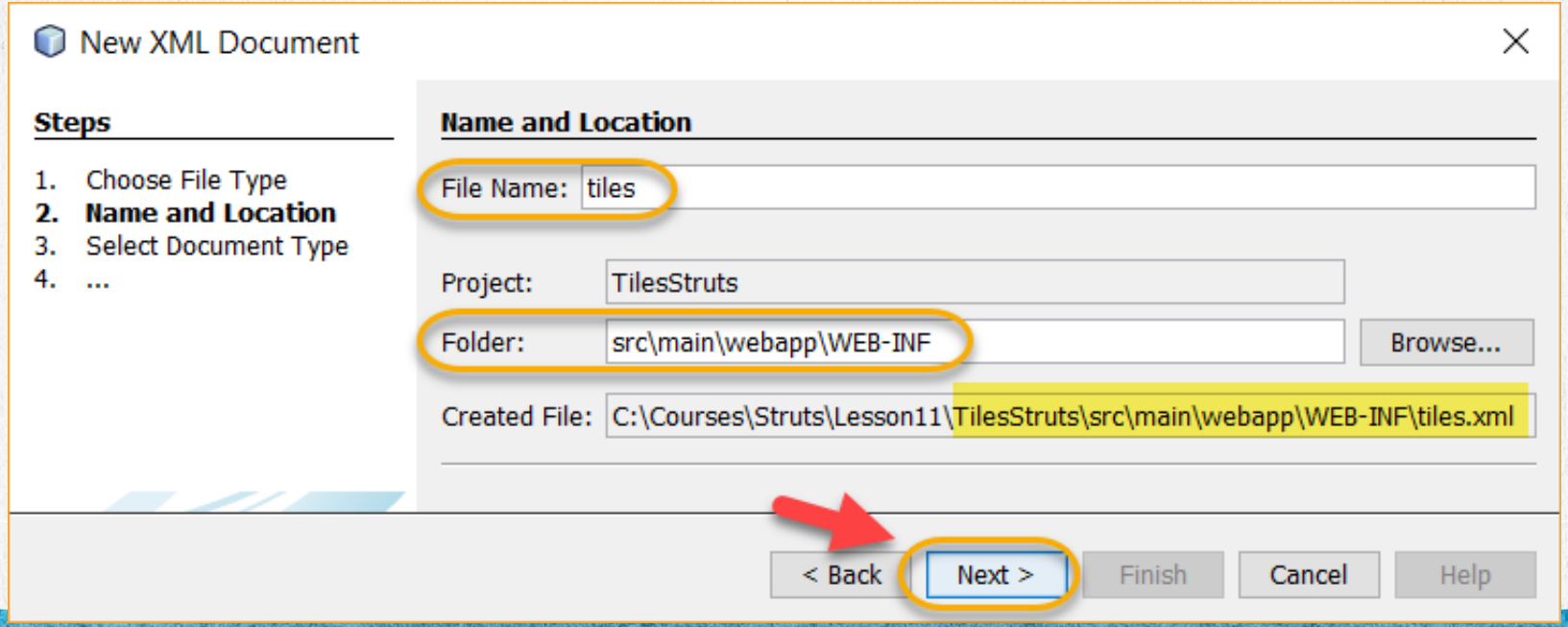
8. CREATE AN XML FILE

- We create the tiles.xml file and add it to the WEB-INF folder as shown:



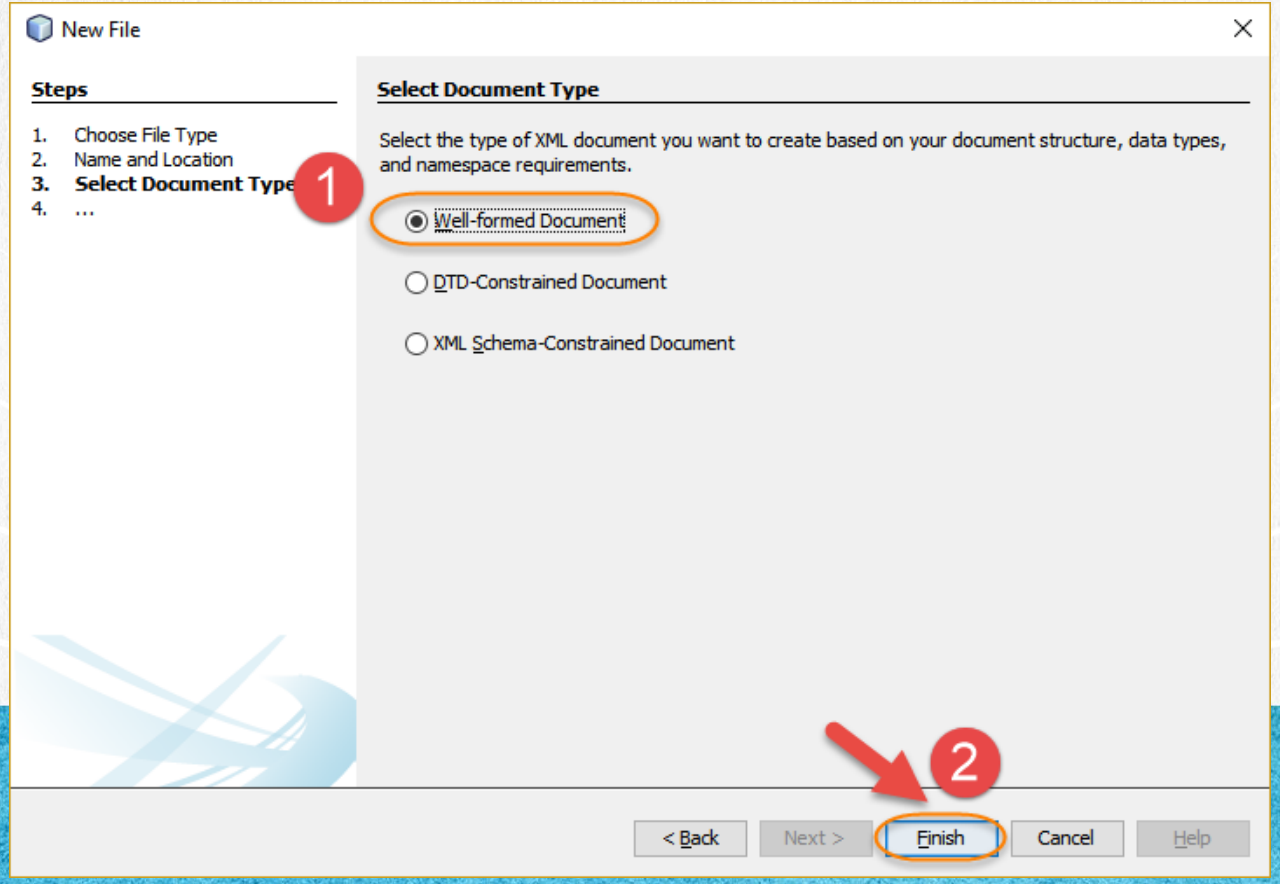
8. CREATE AN XML FILE

- The name of the file is tiles, it is not necessary to add the extension, it automatically adds the IDE since it is an XML type document. Finally we provide the path shown:



8. CREATE AN XML FILE

- We select the indicated type and click on finish.



9. MODIFY THE CODE

tiles.xml:

Click to download

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE tiles-definitions PUBLIC
        "-//Apache Software Foundation//DTD Tiles Configuration 3.0//EN"
        "http://tiles.apache.org/dtds/tiles-config_3_0.dtd">

<tiles-definitions>

    <definition name="layout" template="/WEB-INF/content/layout.jsp">
        <put-attribute name="title"   value="Template"/>
        <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>
```

10. MODIFY THE INDEX.HTML FILE

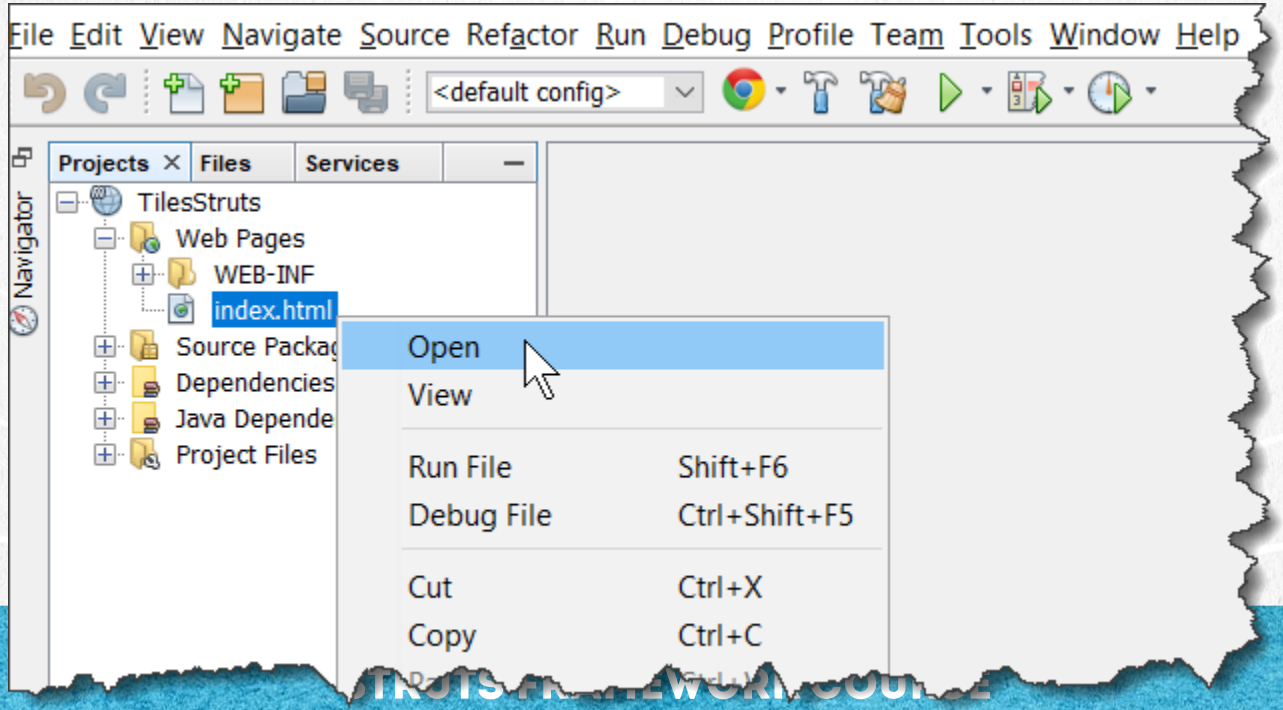
In automatic the IDE adds a file called index.html. However, if this file is not created we must add it to the project at the root level of Web Pages.

The index.html file really is not yet part of the Struts framework, however it will be the entry point for the Struts framework to be executed, since from this file we will indicate which action we want to execute.

In this exercise the path that we will use will be: welcomeAction

10. MODIFY THE INDEX.HTML FILE

- Modify the index.html file. In case this file does not exist at the root level of the Web Pages folder, we created it, as shown:



10. MODIFY THE FILE

index.html:

Click to download

```
<!DOCTYPE html>
<html>
    <head>
        <title>Go to the system</title>
    </head>
    <body>
        <a href="welcomeAction">Go to the system</a>
    </body>
</html>
```

11. CREATE A JAVA CLASS

The class LinkAction.java that we are going to create next will act as Controller (Action) and Model (Bean).

We will extend the ActionSupport class and add two methods that will act as the execute method, called welcome and people. Each of these methods will be annotated using @Action to simplify the configuration of the struts.xml file.

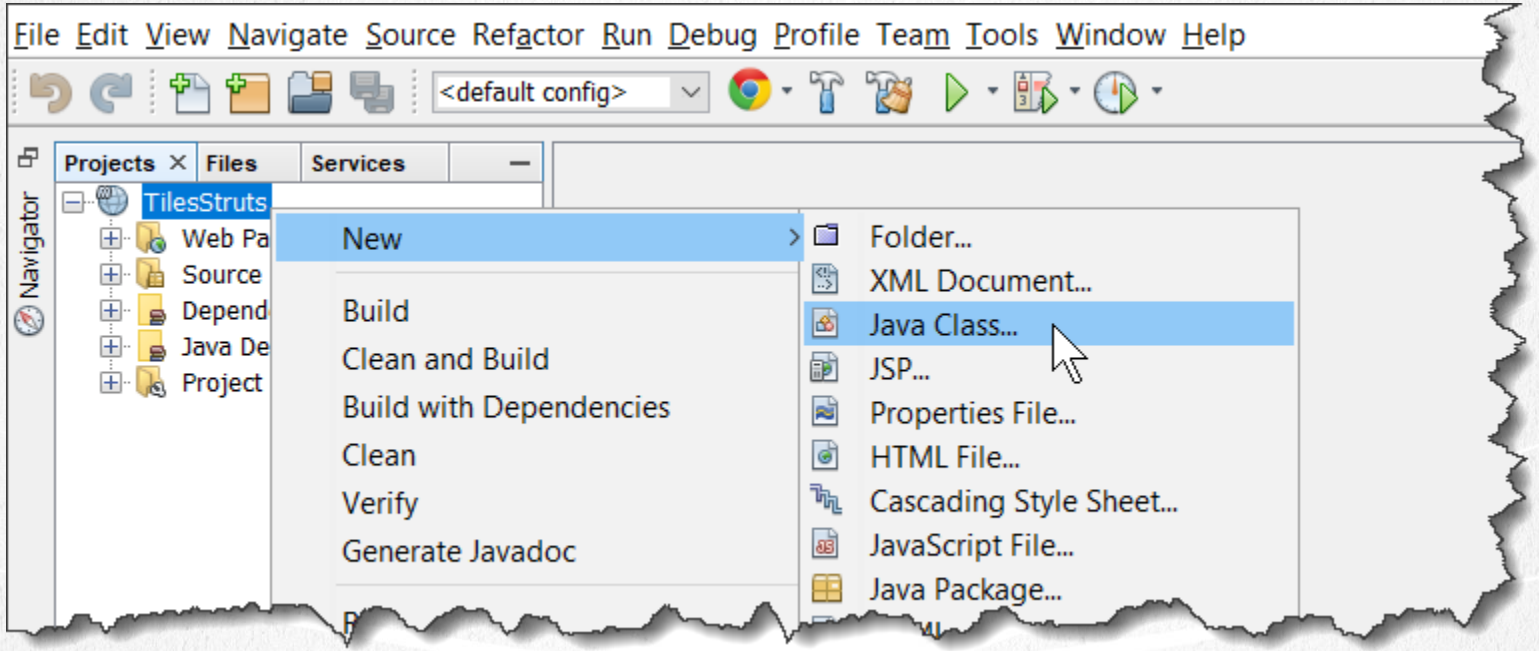
We are going to apply the topic of tiles of Struts, so the "result" of the action methods will return the tile that will be used instead of directly returning the jsp to be displayed. For this we will have to indicate the type of return in the struts.xml file since there is no other way to indicate a return type of type tile.

The name of the result as well as being a type tile, we will also specify the name of tile type that we are going to use to display our view, remember that the tile, layout or template is a set of views or jsp's, that make it be displayed as a single view but reusing several elements, such as header, menu, footer, etc.

Let's see how our LinkAction.java class is:

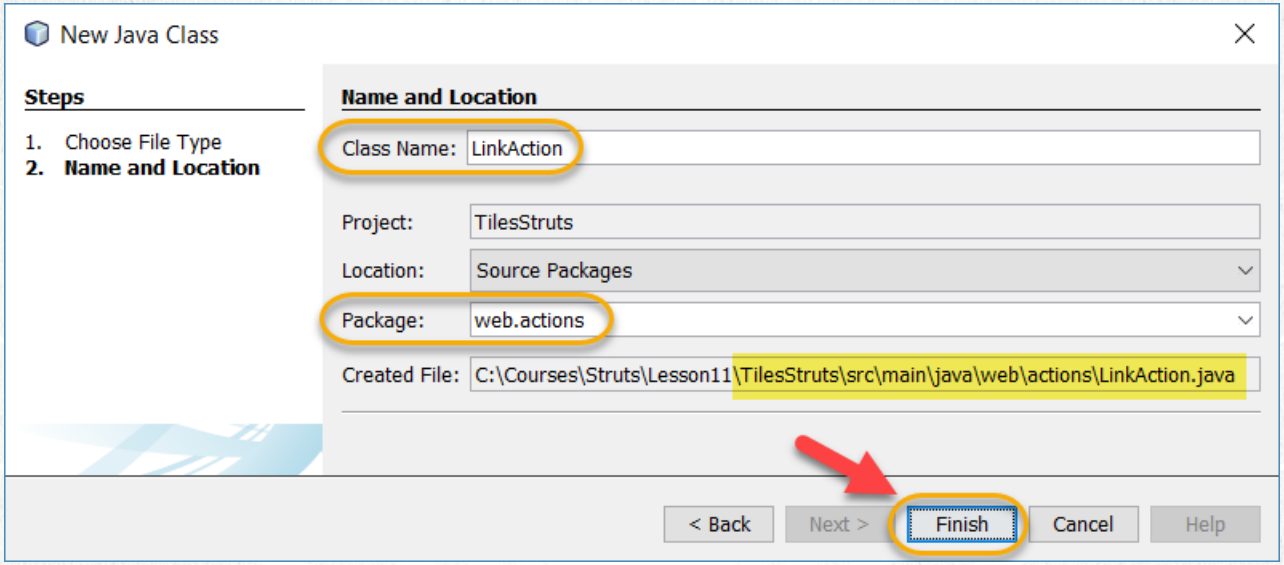
11. CREATE A JAVA CLASS

- We create the LinkAction.java class:



11. CREATE A JAVA CLASS

- We create the LinkAction.java class:



12. MODIFY THE FILE

LinkAction.java:

Click to download

```java
package web.actions;

import com.opensymphony.xwork2.ActionSupport;
import org.apache.struts2.convention.annotation.*;

@Results({
    @Result(name = "welcomeResult", location = "welcomeTile", type = "tiles"),
    @Result(name = "peopleResult", location = "peopleTile", type = "tiles")}
)
public class LinkAction extends ActionSupport {

    @Action(value = "welcomeAction")
    public String welcome() {
        return "welcomeResult";
    }

    @Action(value = "peopleAction")
    public String people() {
        return "peopleResult";
    }
}
```

13. CREATE AN XML FILE

We are going to create the struts.xml file

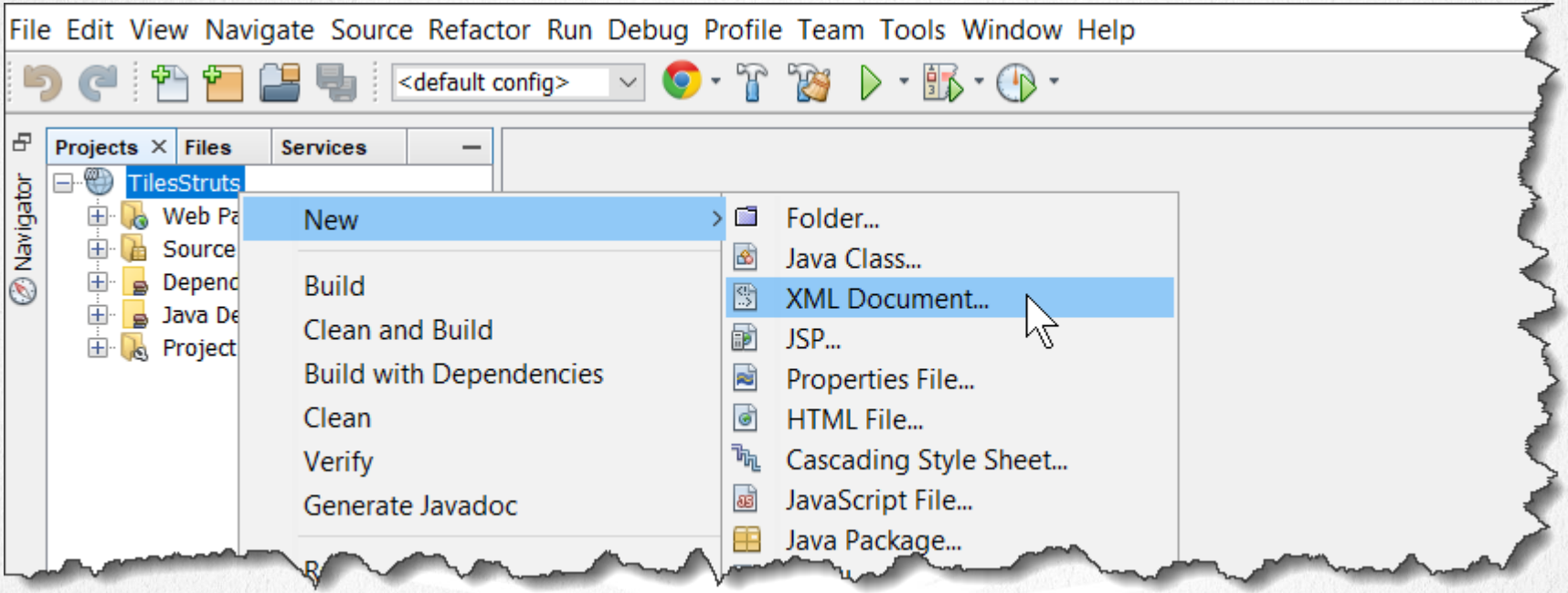
The struts file so far had not been used again, however in the case of tiles, there is no annotation that we can use in our Action class to indicate that we are going to return a tile type, and therefore this can only be specified within the struts.xml file, let's see how our file is.

Recall that our newly defined LinkAction class is in the package: web.actions

Experiencia y Conocimiento para tu vida

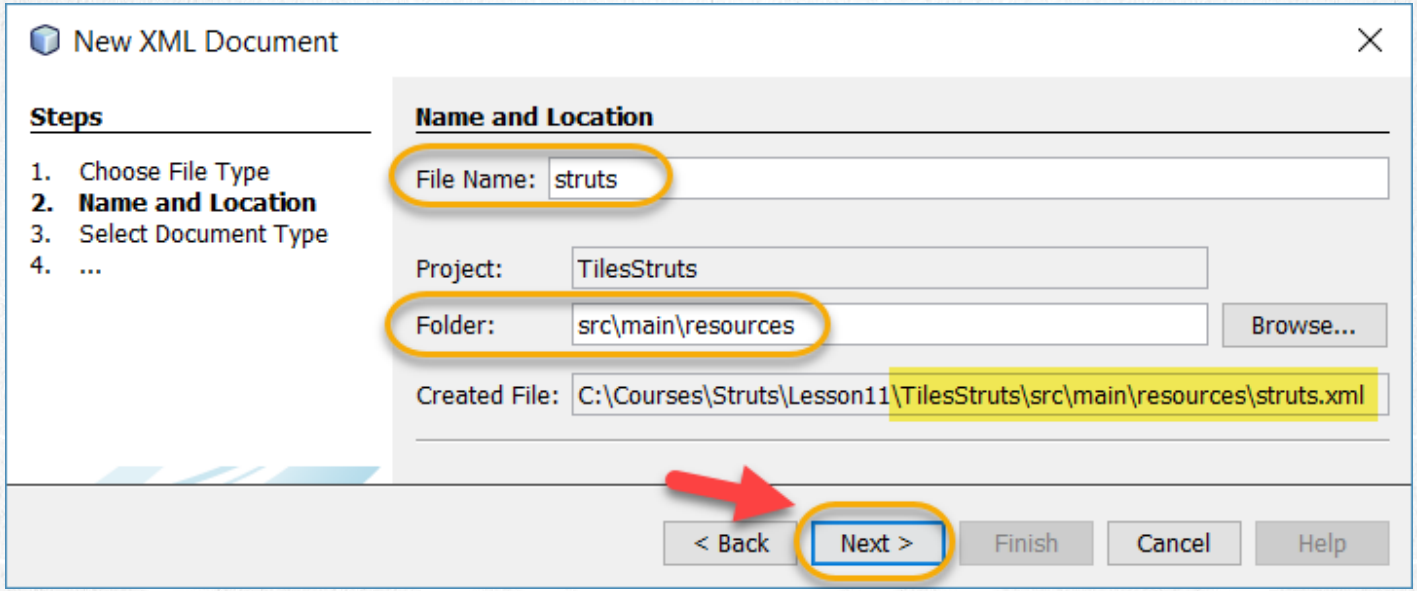
13. CREATE AN XML FILE

- We create the struts.xml file and add it to the resources folder as shown:



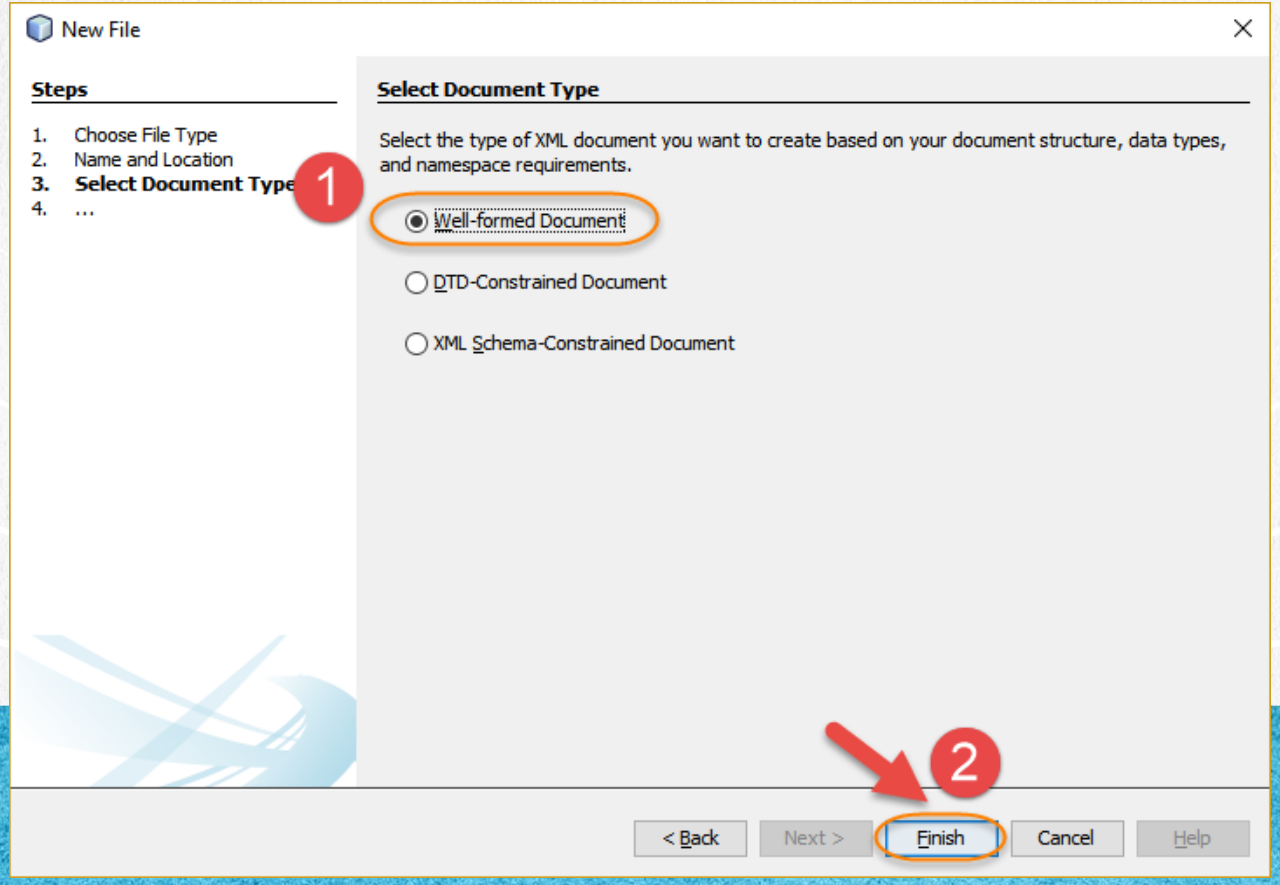
13. CREATE AN XML FILE

- The name of the file is struts.xml. We provide the path shown :



13. CREATE AN XML FILE

- We select the indicated type and click on finish.



14. MODIFY THE FILE

struts.xml:

Click to download

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.5//EN"
    "http://struts.apache.org/dtds/struts-2.5.dtd">
<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>
```

15. CREATE A JSP FILE

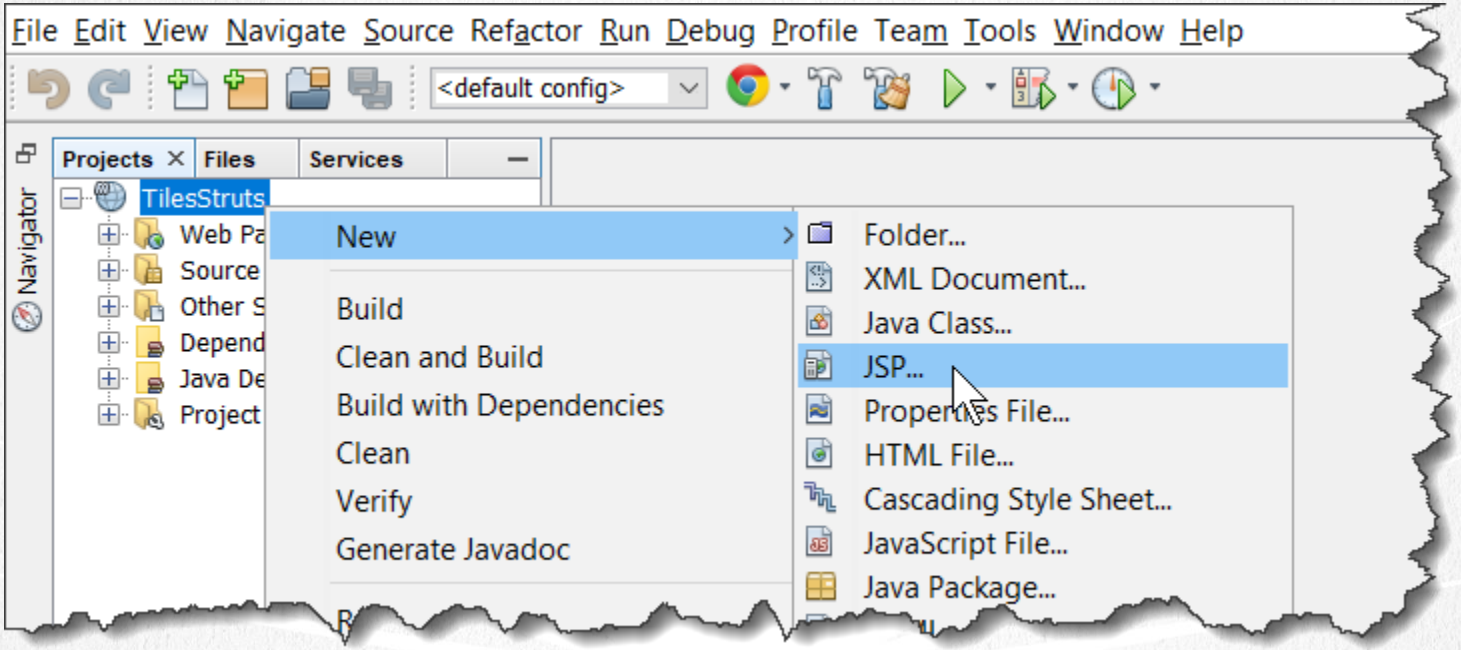
Now we create the file: layout.jsp. This file is the template or base tile, from this JSP is that will extend the other pages that we have defined in the tiles.xml file.

This layout.jsp file 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 path. But for convenience and to maintain the standard we have handled so far, we will deposit it in the indicated path.

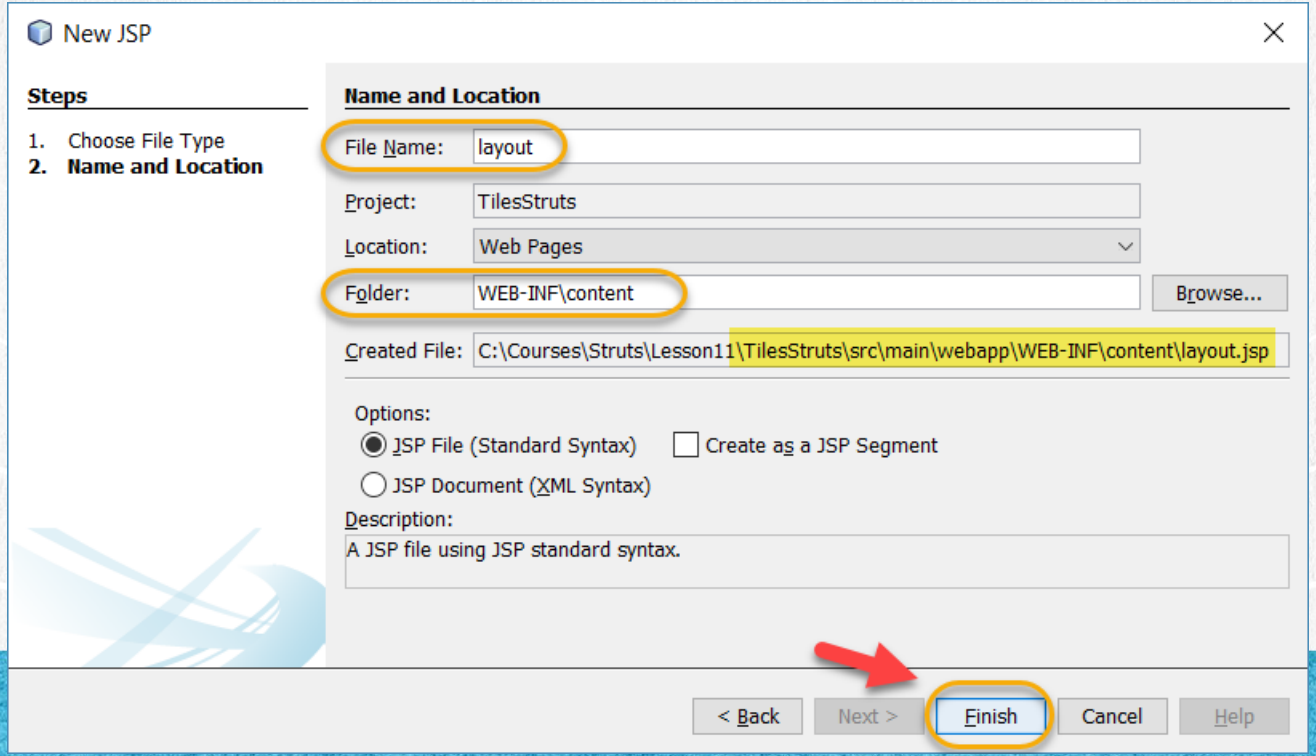
15. CREATE A JSP FILE

- We create the file layout.jsp:



15. CREATE A JSP FILE

- We create the file layout.jsp in the path shown :



16. MODIFY THE CODE

layout.jsp:

Click to download

```
<%@ taglib uri="http://tiles.apache.org/tags-tiles" prefix="tiles" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <title>
            <tiles:insertAttribute name="title" ignore="true" />
        </title>
    </head>
```

16. MODIFY THE CODE

layout.jsp:

Click to download

```
    <body>
        <table border="1" cellpadding="2" cellspacing="2" align="center">
            <tr>
                <td height="30" colspan="2">
                    <tiles:insertAttribute name="header" />
                </td>
            </tr>
            <tr>
                <td height="250">
                    <tiles:insertAttribute name="menu" />
                </td>
                <td width="350">
                    <tiles:insertAttribute name="body" />
                </td>
            </tr>
            <tr>
                <td height="30" colspan="2">
                    <tiles:insertAttribute name="footer" />
                </td>
            </tr>
        </table>
    </body>
</html>
```

17. CREATE A JSP FILE

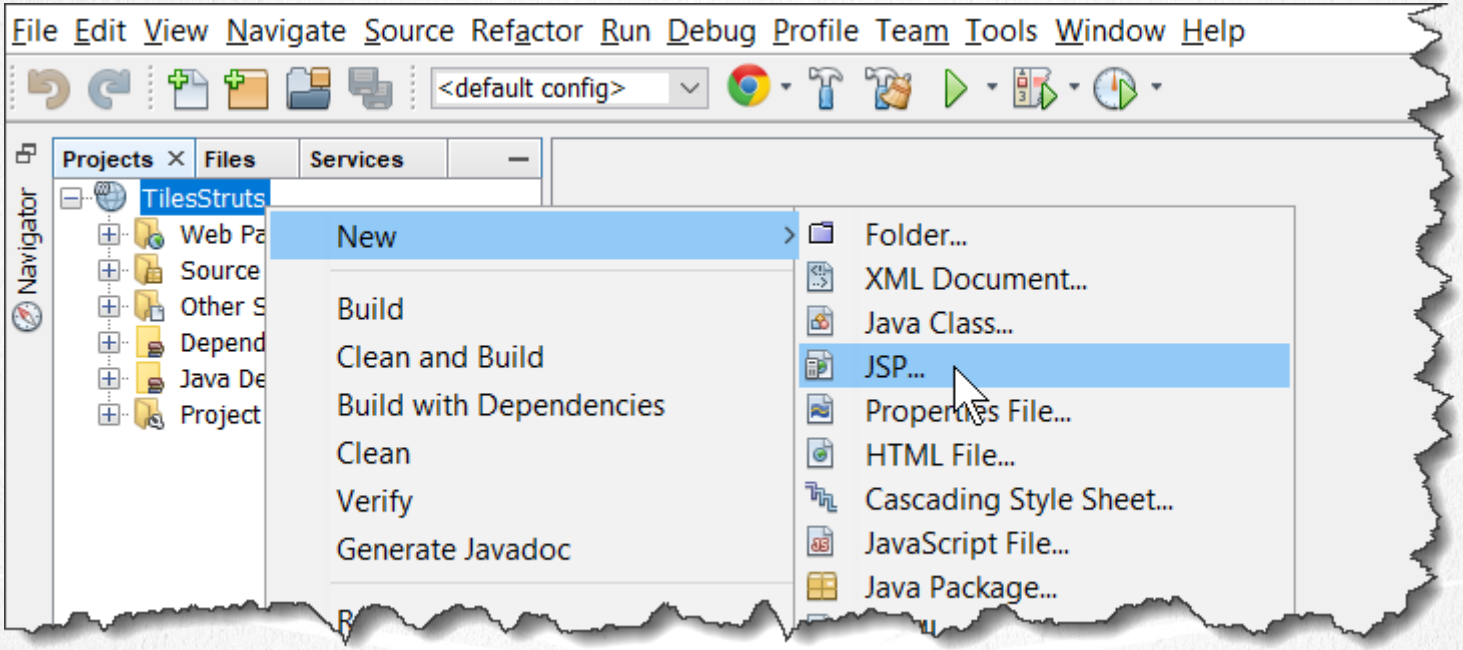
Now we create the file: header.jsp. This file contains the code for the layout header.

Each of the JSPs that we will see below can be quite complex as we wish, however to make the use of Tiles clear, we will limit ourselves to adding only texts for differences in the sections of the tiles.xml template.

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 path. But for convenience and to maintain the standard we have handled so far, we will deposit it in the indicated path.

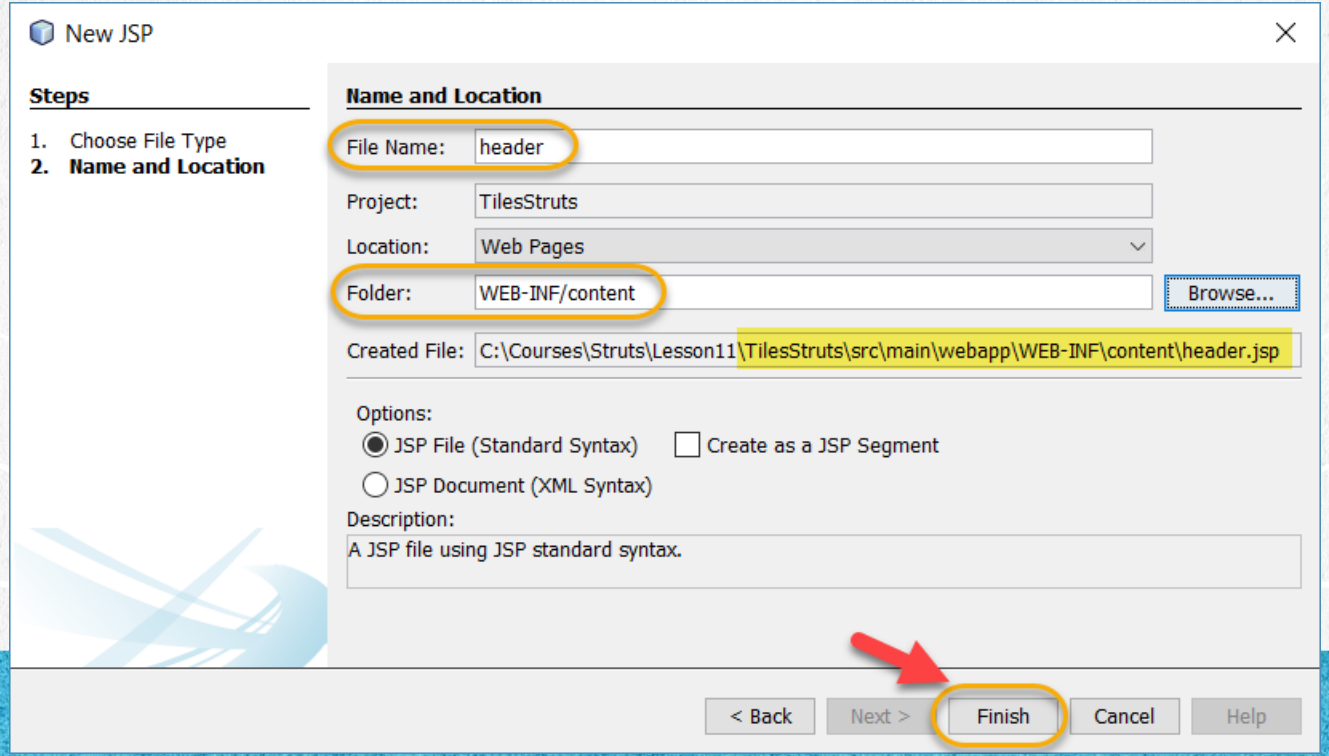
17. CREATE A JSP FILE

- We created the header.jsp file:



17. CREATE A JSP FILE

- We create the file header.jsp in the path shown :



18. MODIFY THE FILE

header.jsp:

Click to download

```
<!DOCTYPE html>
<html>
    <head>
        <title>Header</title>
    </head>
    <body>
        <h1>Header</h1>
    </body>
</html>
```

19. CREATE A JSP FILE

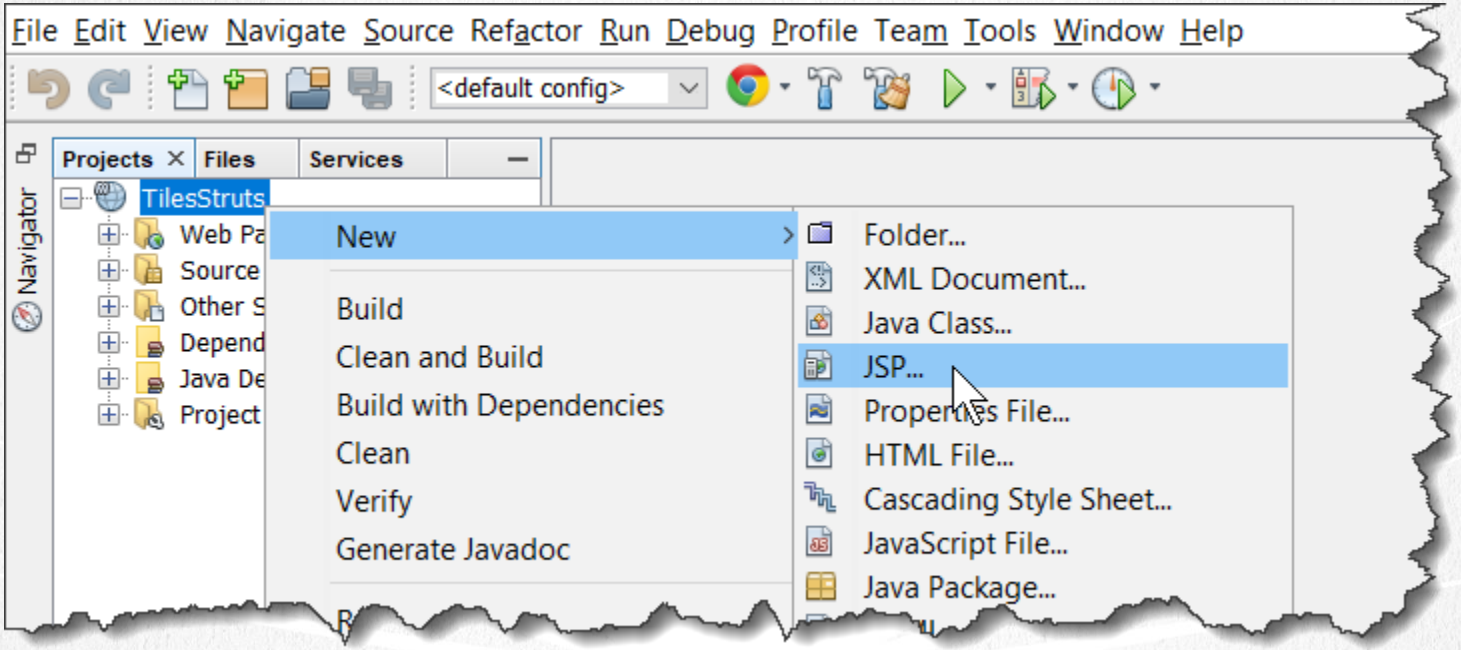
Now we create the file: menu.jsp. This file contains the code for the layout's menu section.

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 path. But for convenience and to maintain the standard we have handled so far, we will deposit it in the indicated path.

Experiencia y Conocimiento para tu vida

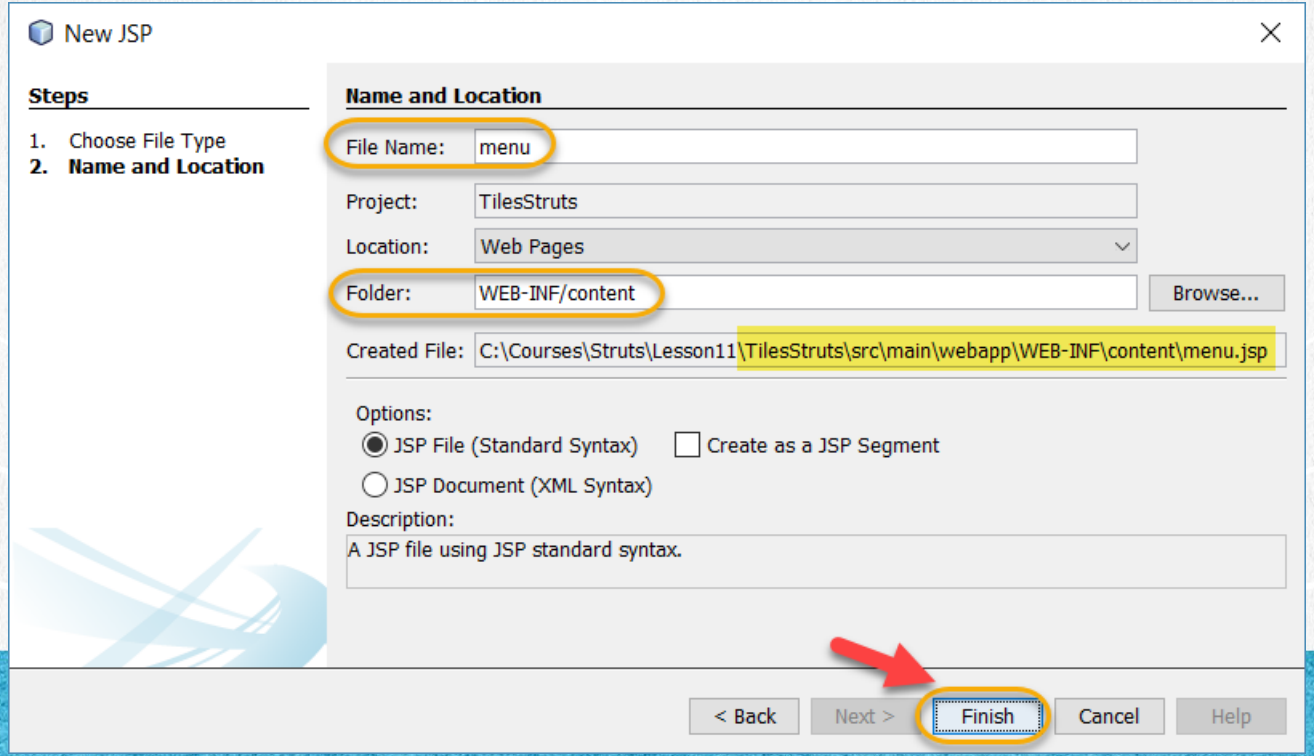
19. CREATE A JSP FILE

- We created the file menu.jsp :



19. CREATE A JSP FILE

- We created the menu.jsp file in the path shown :



20. MODIFY THE CODE

menu.jsp:

Click to download

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib uri="/struts-tags" prefix="s"%>
<!DOCTYPE html>
<a href="<s:url action="welcomeAction"/>" >Welcome</a><br>
<br/>
<a href="<s:url action="peopleAction"/>" >People</a><br>
```

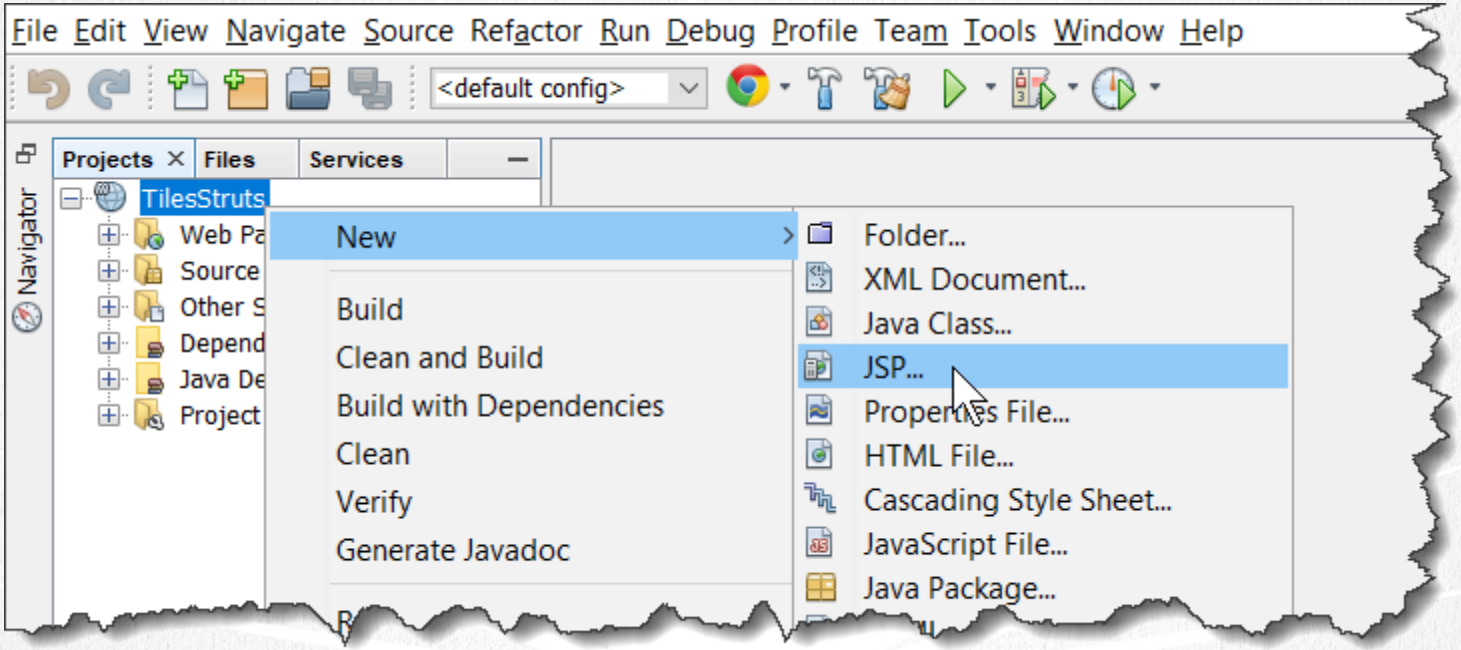
21. CREATE A JSP FILE

Now we create the file: body.jsp. This file contains the code for the body section of the layout.

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 so far, we will deposit it in the indicated path.

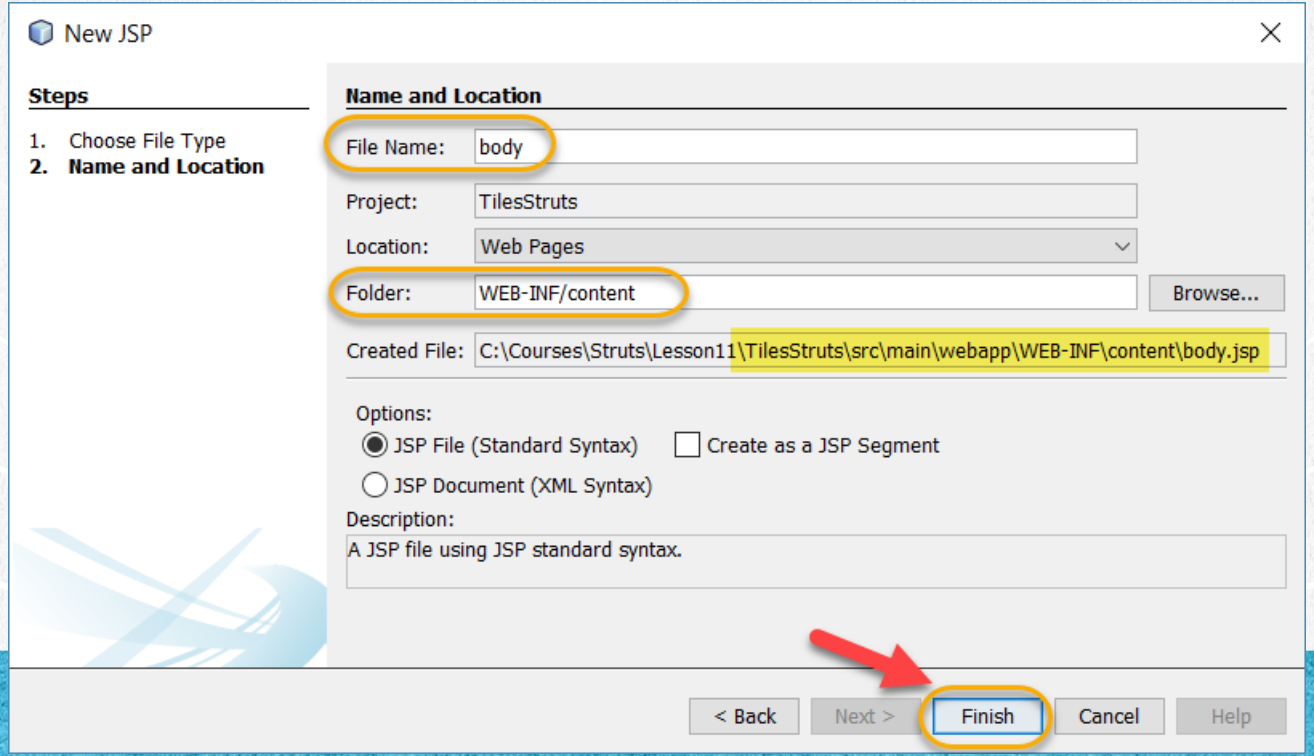
21. CREATE A JSP FILE

- We created the body.jsp file:



21. CREATE A JSP FILE

- We created the body.jsp file in the path shown:



22. MODIFY THE CODE

body.jsp:

Click to download

```
<!DOCTYPE html>
<html>
    <head>
        <title>Body</title>
    </head>
    <body>
        <h1>Body</h1>
    </body>
</html>
```

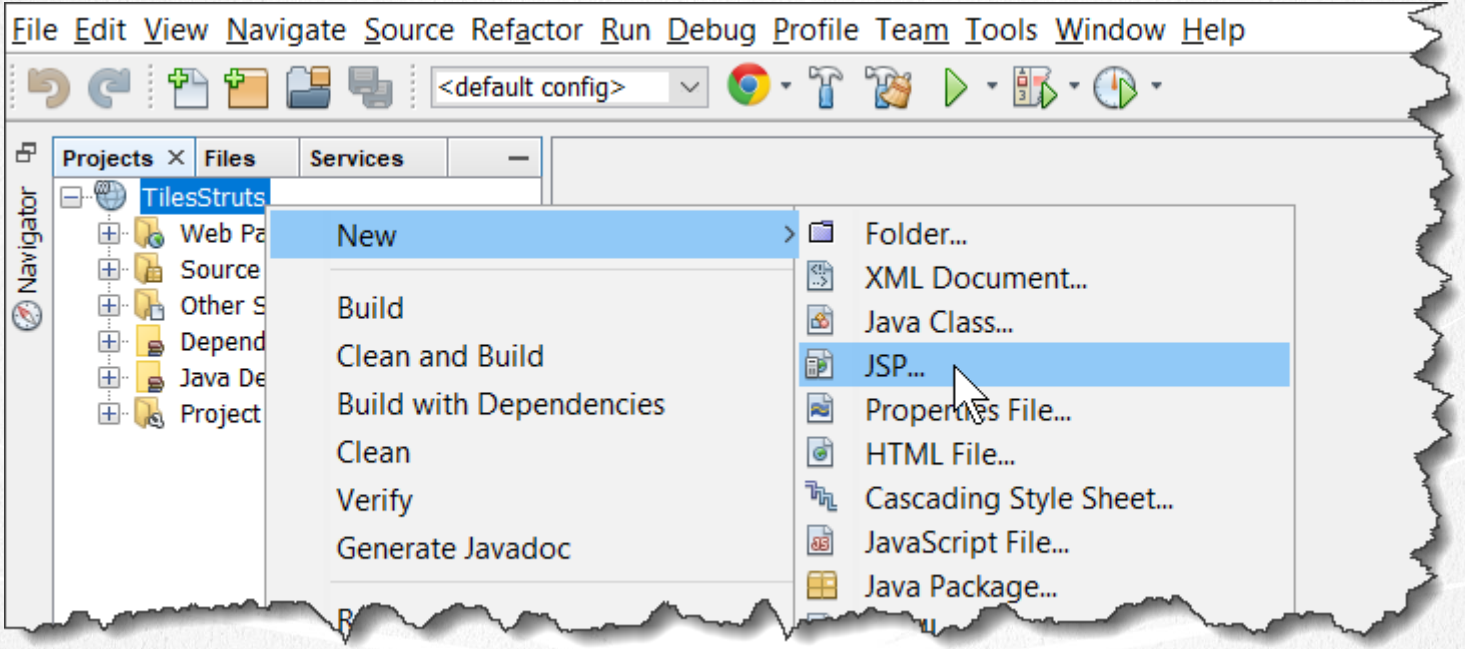
23. CREATE A JSP FILE

Now we create the file: footer.jsp. This file contains the code for the footer section of the layout.

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 path. But for convenience and to maintain the standard we have handled so far, we will deposit it in the indicated path.

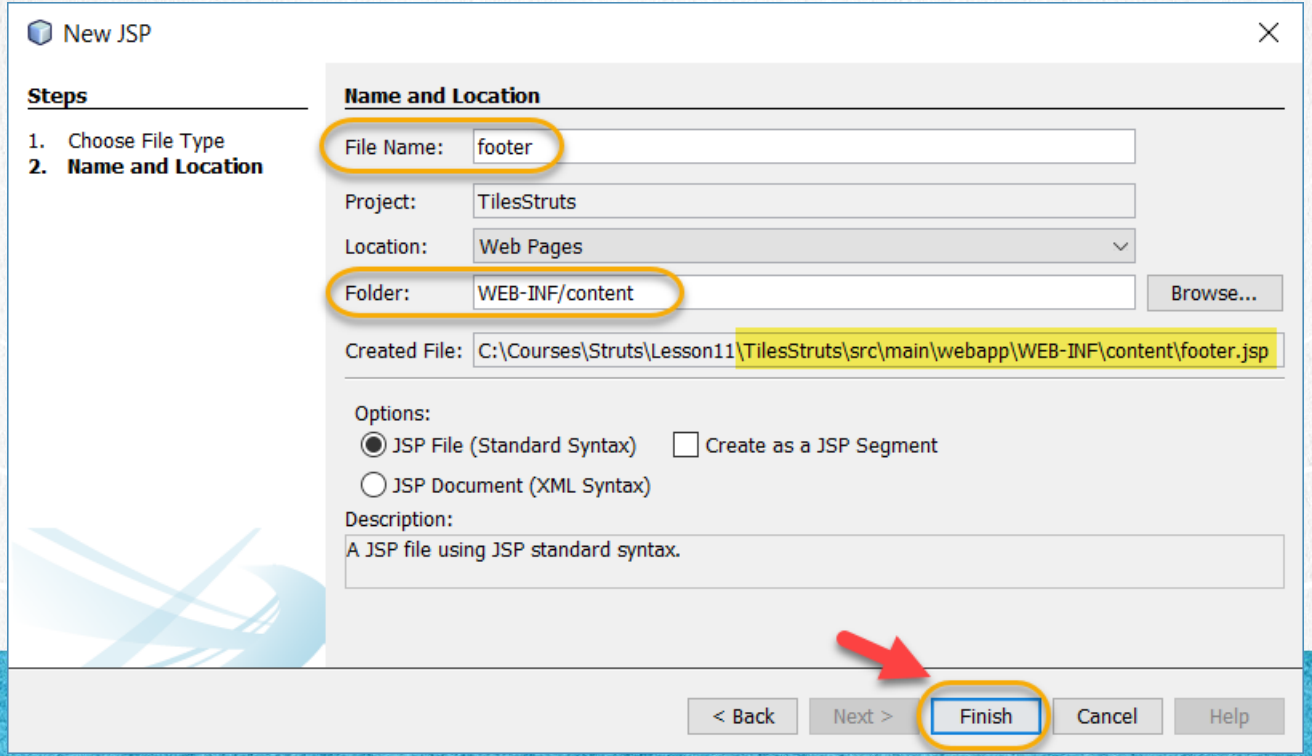
23. CREATE A JSP FILE

- We created the file footer.jsp:



23. CREATE A JSP FILE

- Creamos el archivo footer.jsp en la ruta mostrada:



24. MODIFY THE CODE

footer.jsp:

Click to download

```
<!DOCTYPE html>
<html>
    <head>
        <title>Footer</title>
    </head>
    <body>
        <h1>Footer</h1>
    </body>
</html>
```

25. CREATE A JSP FILE

Now we create the file: welcome.jsp. This file contains the code for the welcome page.

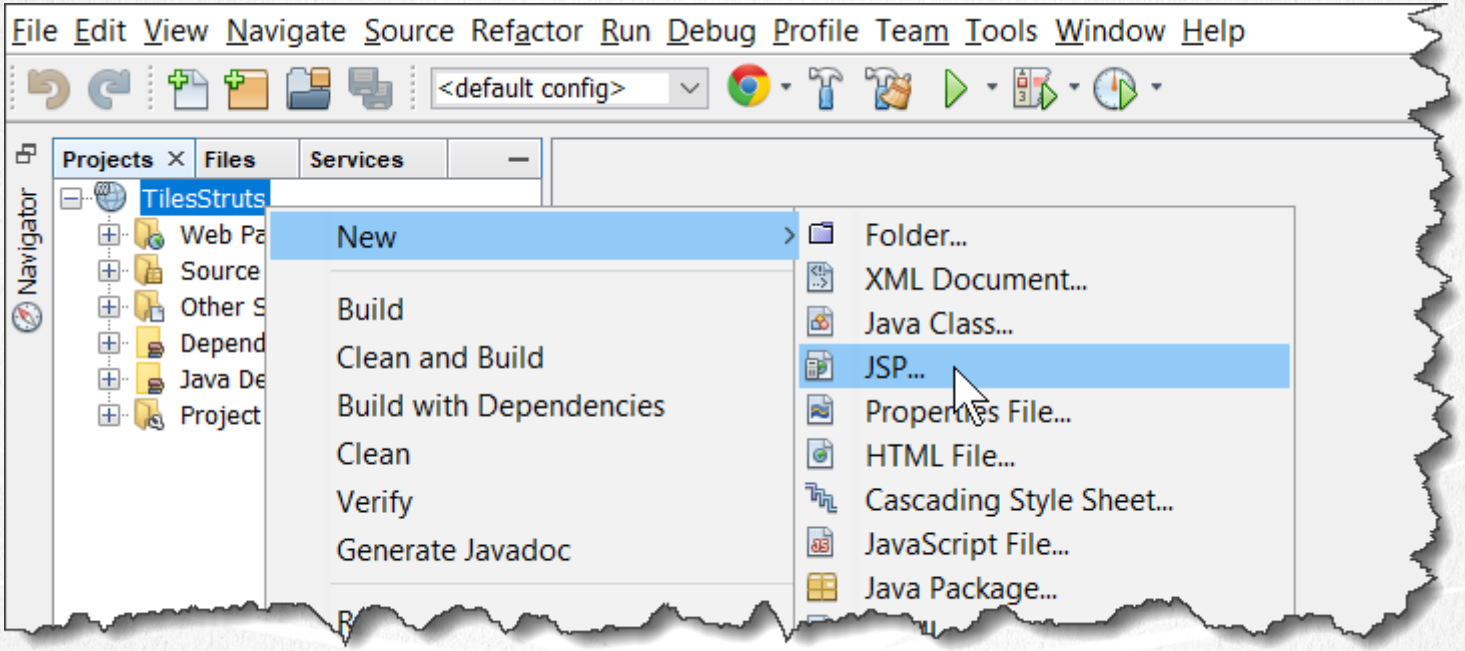
This is the first JSP that will use the definition of tiles, so as we saw in the tiles.xml file, welcome Tiles extends the layout tile that was initially defined.

This file is required to be deposited in the /WEB-INF/content folder since it will be accessed as a result of executing the respective action.

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.

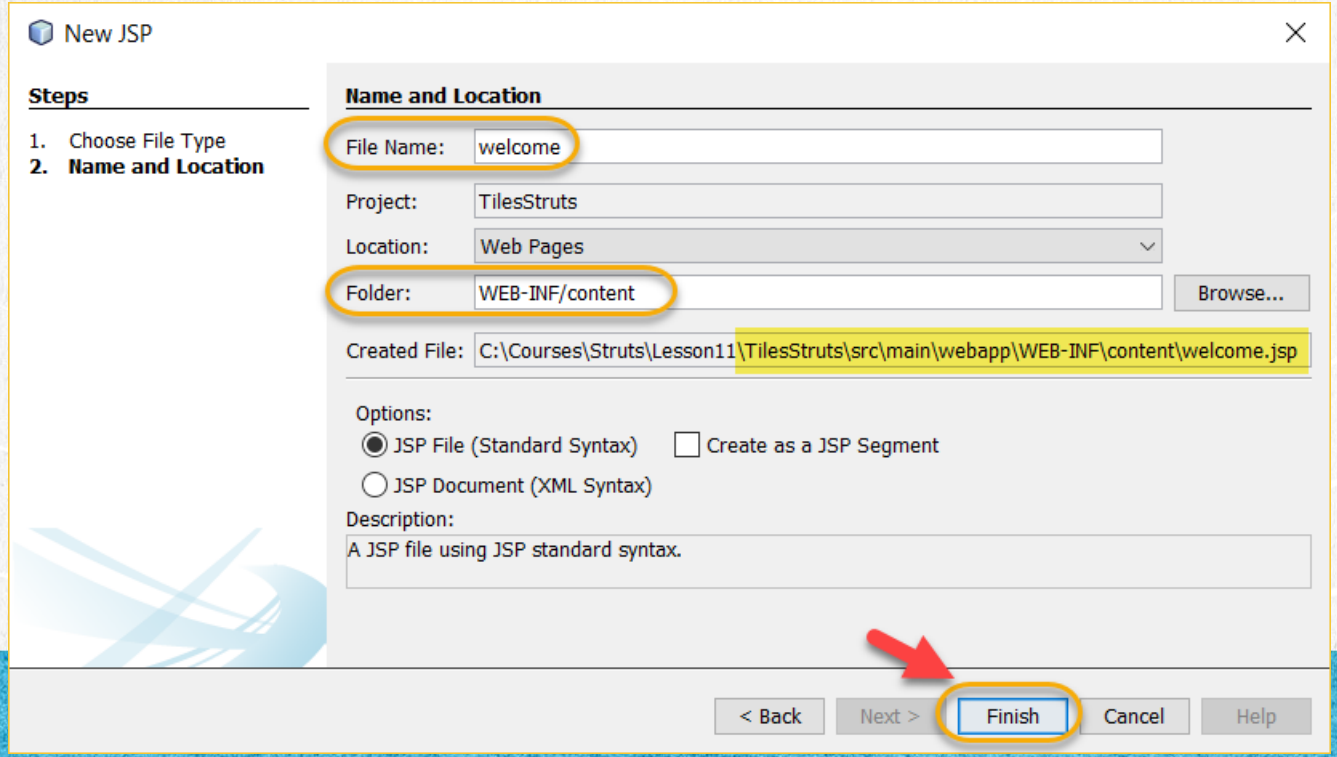
25. CREATE A JSP FILE

- We created the file welcome.jsp :



25. CREATE A JSP FILE

- We created the file welcome.jsp in the path shown:



26. MODIFY THE FILE

welcome.jsp:

Click to download

```
<!DOCTYPE html>
<html>
    <head>
        <title>Welcome</title>
    </head>
    <body>
        <h1>Welcome</h1>
    </body>
</html>
```

27. CREATE A JSP FILE

Now we create the file: people.jsp. This file contains the code for the people page.

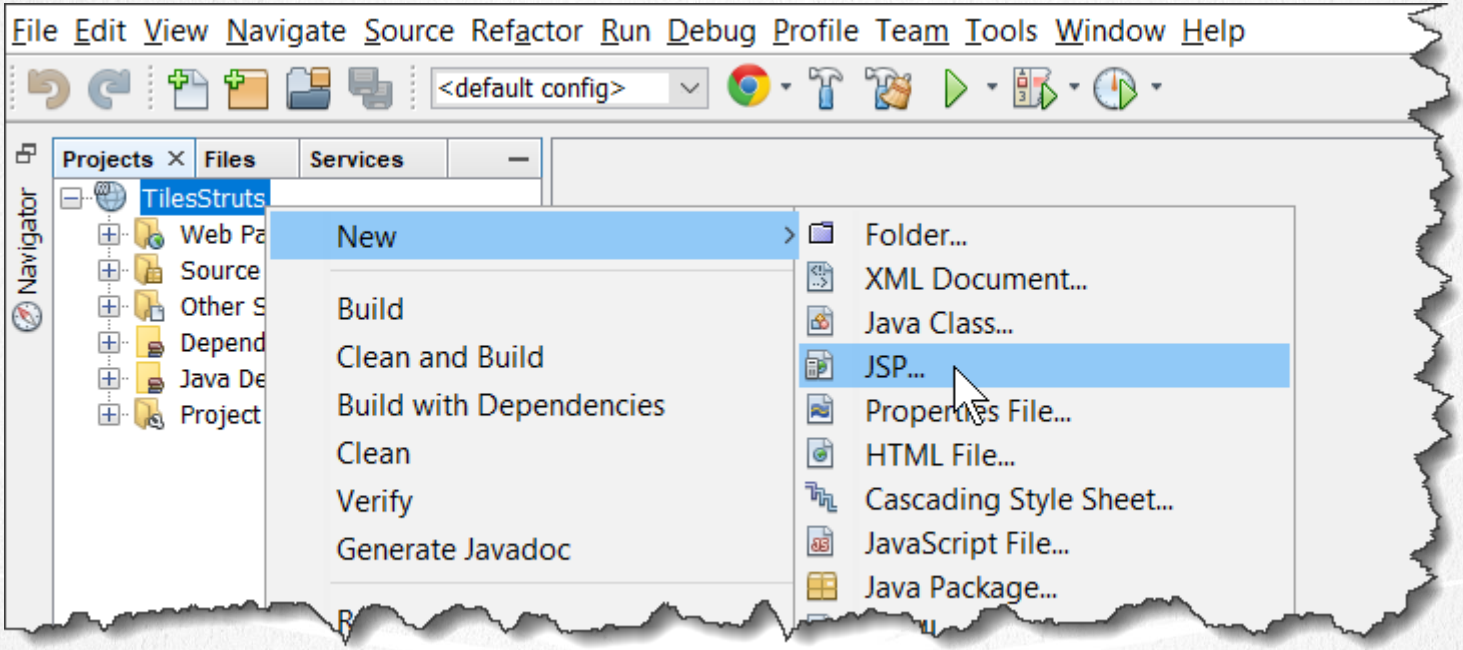
This is the second JSP that will use the definition of tiles, so as we saw in the tiles.xml file, peopleTiles extends from the layout tile that was initially defined.

This file is required to be deposited in the /WEB-INF/content folder since it will be accessed as a result of executing the respective action.

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.

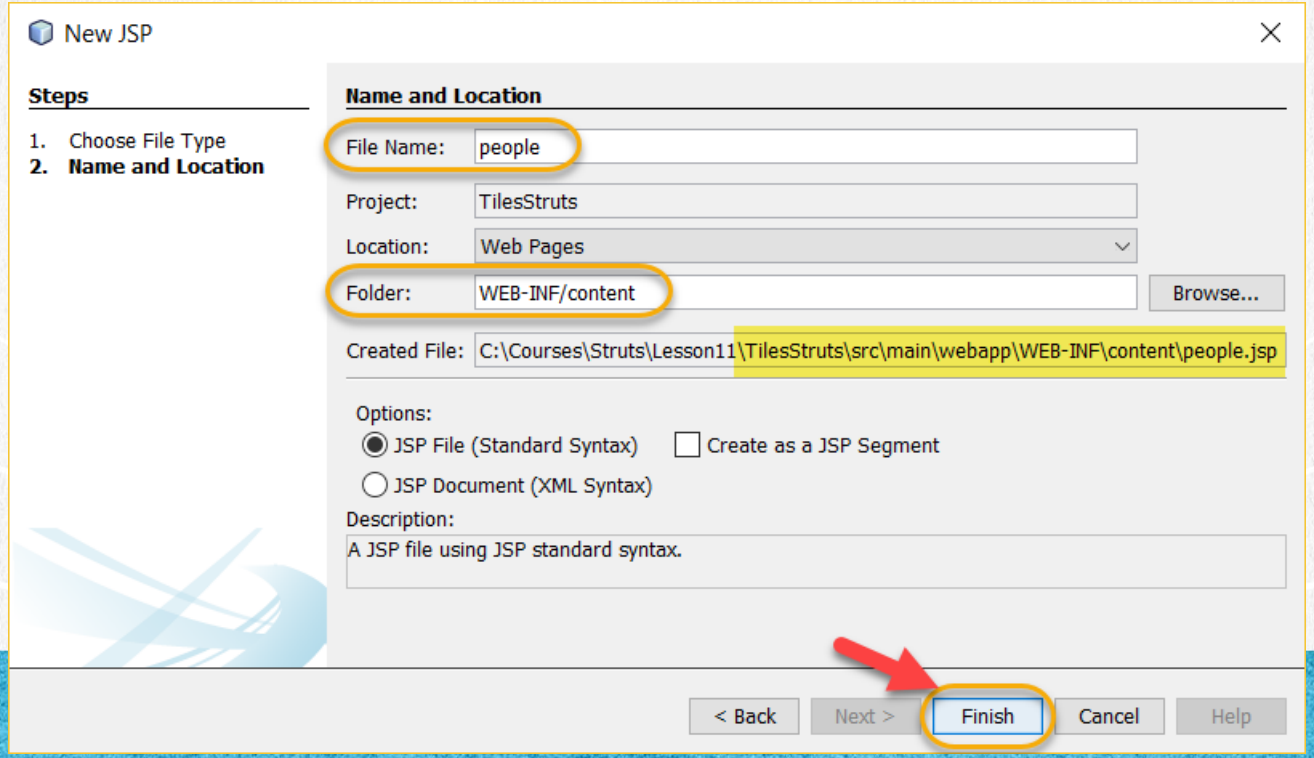
27. CREATE A JSP FILE

- Create the people.jsp file:



27. CREATE A JSP FILE

- We created the file people.jsp in the path shown:



28. MODIFY THE CODE

people.jsp:

Click to download

```
<!DOCTYPE html>
<html>
    <head>
        <title>People</title>
    </head>
    <body>
        <h1>People</h1>
    </body>
</html>
```

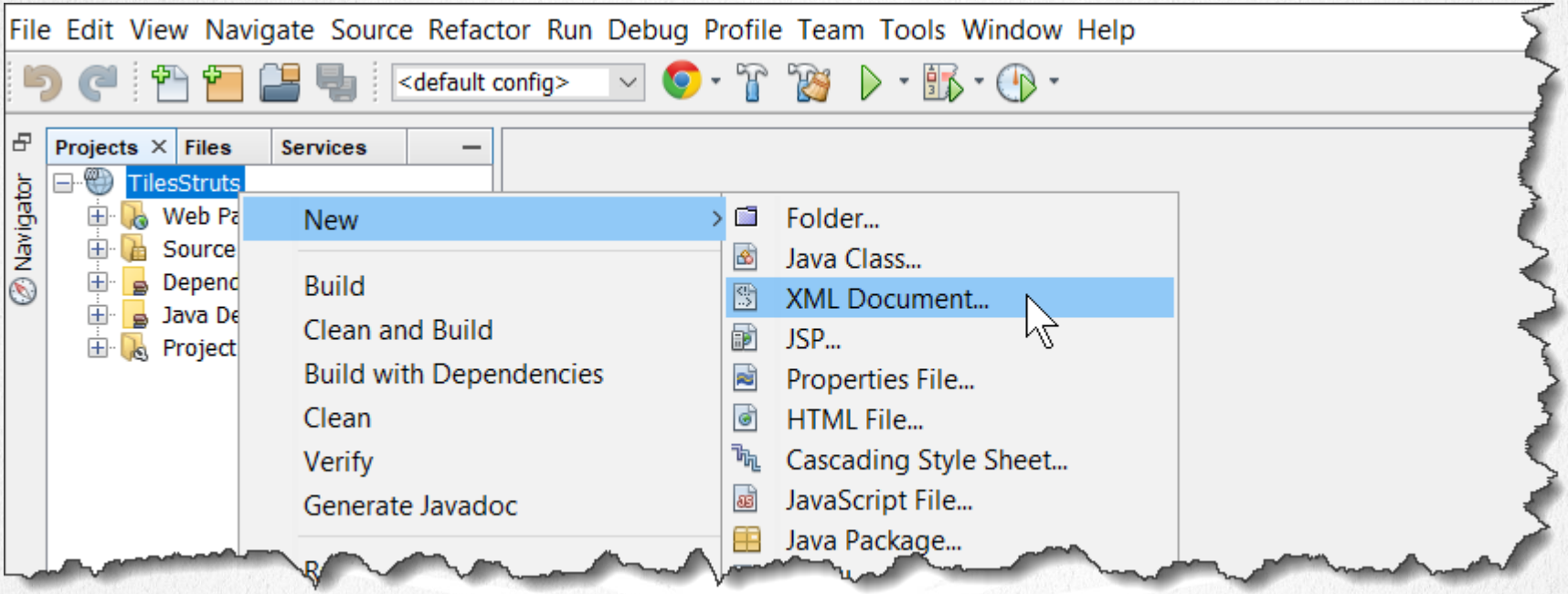
29. CREATE THE LOG4J2.XML

We create a log4j2.xml file. The log4j API allows us to manage the log or log of Java application in a simpler way.

We place this file in the resource path of the maven project. If maven is not used then the file must be deposited at the root level of the Java code src.

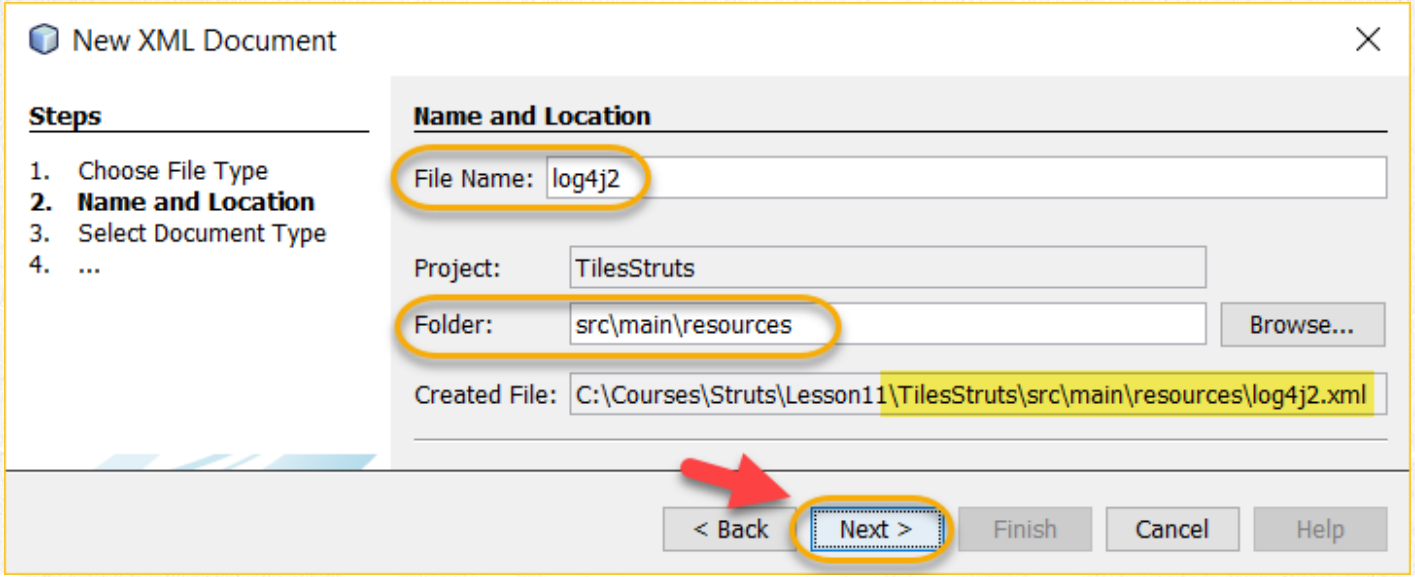
29. CREATE THE LOG4J2.XML

- We create the log4j2.xml file as follows:



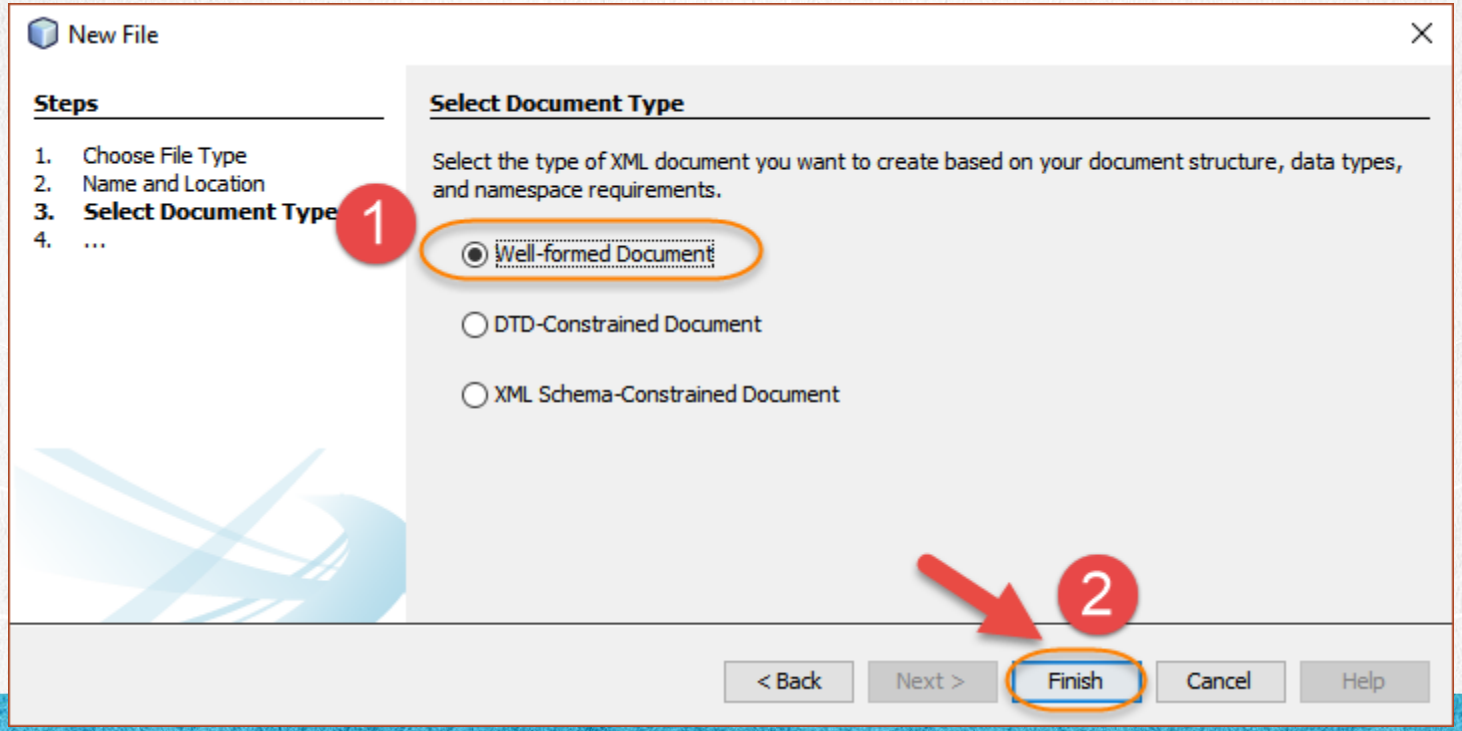
29. CREATE THE LOG4J2.XML

- We deposit the file in the resources folder as shown:



29. CREATE THE LOG4J2.XML

- We select the option shown:



30. MODIFY THE CODE

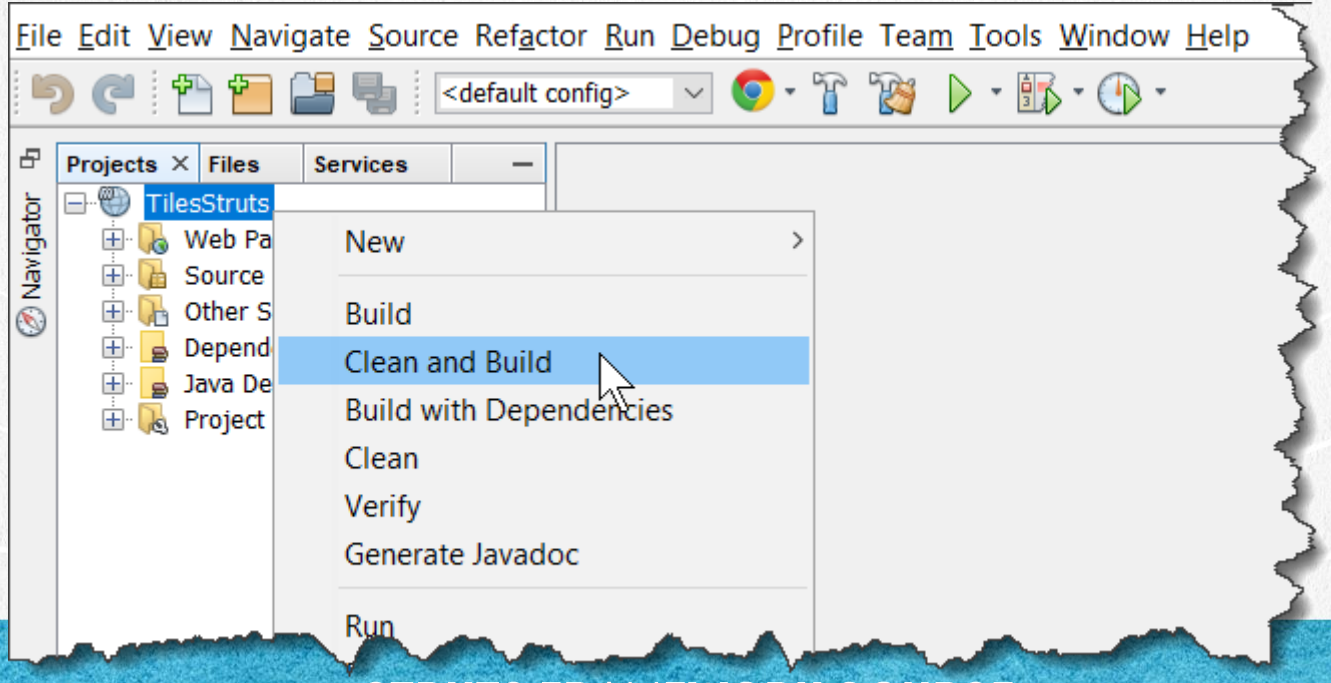
log4j2.xml:

Click to download

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration>
    <Appenders>
        <Console name="STDOUT" target="SYSTEM_OUT">
            <PatternLayout pattern="(%F:%L) - %m%n"/>
        </Console>
    </Appenders>
    <Loggers>
        <Logger name="com.opensymphony.xwork2" level="info"/>
        <Logger name="org.apache.struts2" level="info"/>
        <Root level="info">
            <AppenderRef ref="STDOUT"/>
        </Root>
    </Loggers>
</Configuration>
```

31. EXECUTE CLEAN & BUILD

- We execute the Clean & Build command as shown, to obtain the latest version of each file:



31. EXECUTE CLEAN & BUILD

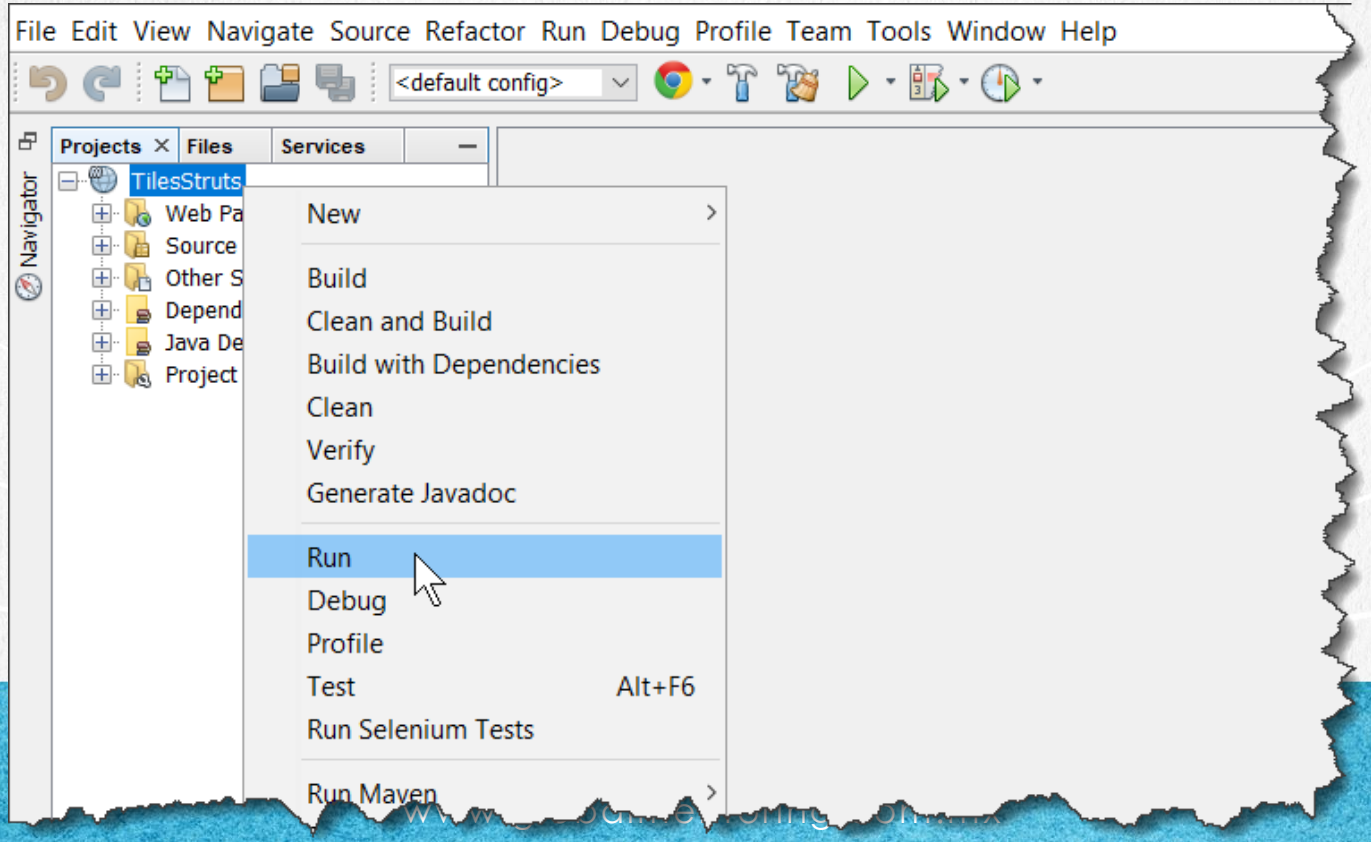
- We must observe a result similar to the following:

```
Output - Build (TilesStruts)
--- maven-war-plugin:2.3:war (default-war) @ TilesStruts ---
Packaging webapp
Assembling webapp [TilesStruts] in [C:\Courses\Struts\Lesson11\TilesStruts\target\TilesStruts-1]
Processing war project
Copying webapp resources [C:\Courses\Struts\Lesson11\TilesStruts\src\main\webapp]
Webapp assembled in [1277 msecs]
Building war: C:\Courses\Struts\Lesson11\TilesStruts\target\TilesStruts-1.war

--- maven-install-plugin:2.3.1:install (default-install) @ TilesStruts ---
Installing C:\Courses\Struts\Lesson11\TilesStruts\target\TilesStruts-1.war to C:\Users\user\.m2\repository\web\TilesStruts\1\TilesStruts-1.war
Installing C:\Courses\Struts\Lesson11\TilesStruts\pom.xml to C:\Users\user\.m2\repository\web\TilesStruts\1\TilesStruts-1.pom
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 4.432s
Finished at: Fri Aug 31 20:21:48
Final Memory: 21M/403M
------------------------------------------------------------------------
```

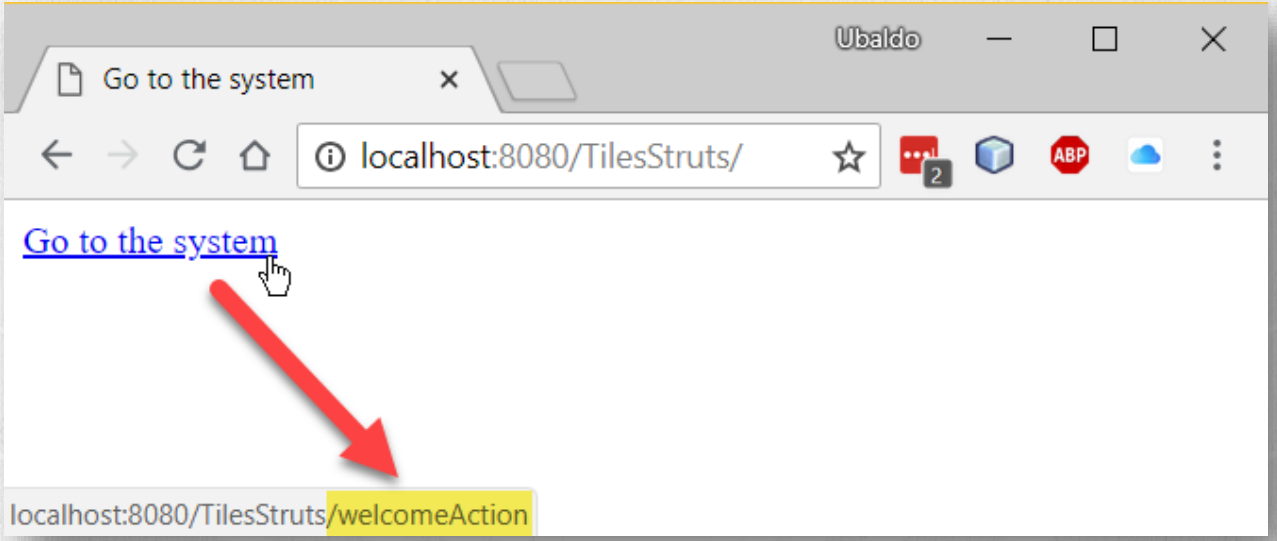
32. EXECUTE THE APPLICATION

- Execute the application as follows:



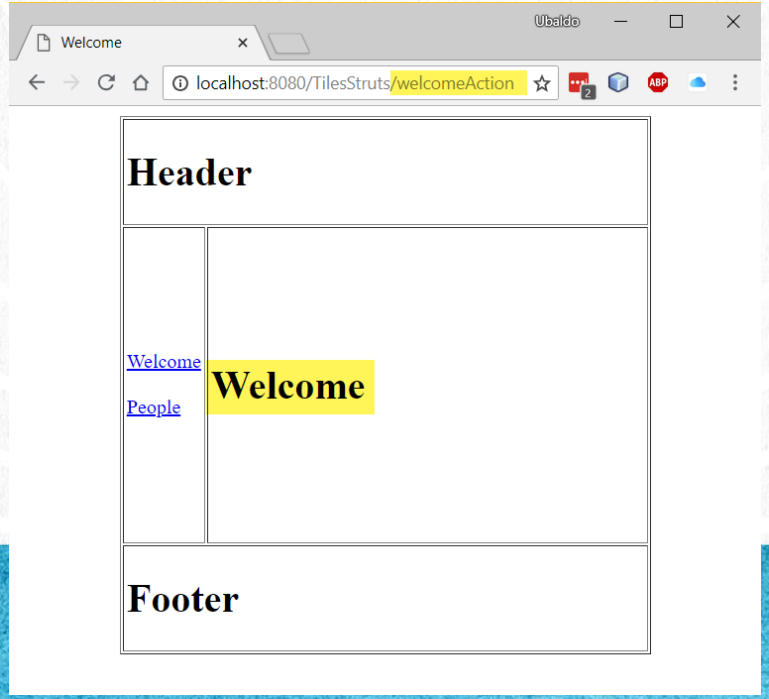
32. EXECUTE THE APPLICATION

- Execute the application as follows:



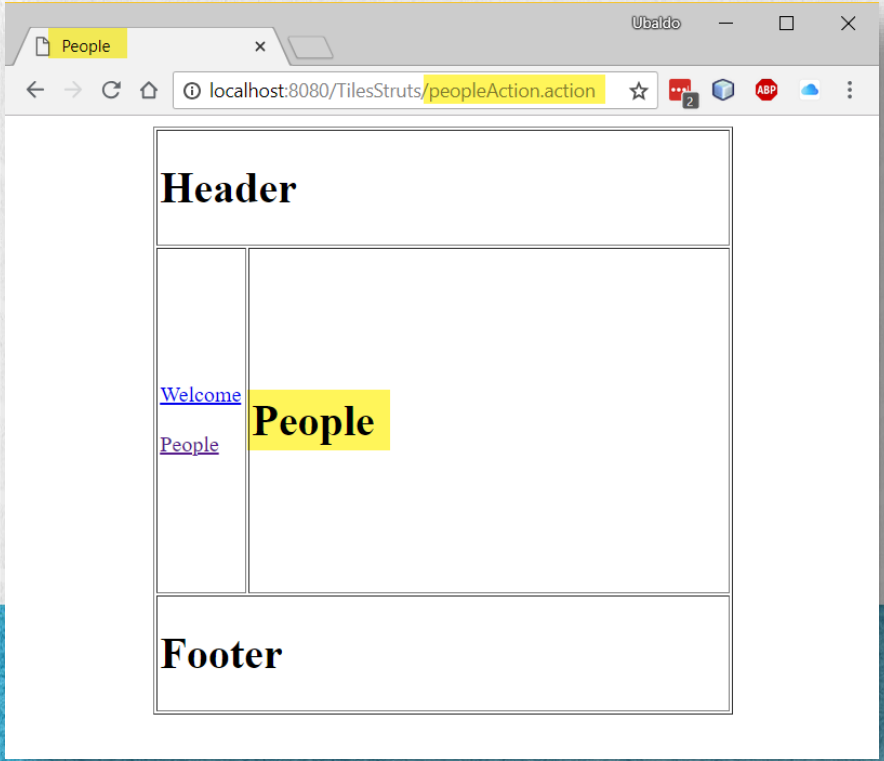
32. EXECUTE THE APPLICATION

- We observe the first view of type tile, as a result of having executed the action welcomeAction. The only parts that have been modified from the original template is the title (tab) and the body (body) of the template.



32. EXECUTE THE APPLICATION

- If we click on the People Link, this is the second view of type tile called Tile people, and we note that the only change of new account is the title and body, the other elements are the same, so we are reusing a lot of code when using and applying the concept of tiles:



FINAL RECOMMENDATIONS

If for some reason the exercise fails, several things can be done to correct it:

1. Stop the Glassfish server
2. Make a Clean & Build project to have the most recent version compiled
3. Restart the project (deploy the project to the server again)

If the above does not work, you can try loading the resolved project which is 100% functional and rule out configuration problems in your environment or any other code error.

EXERCISE CONCLUSION

With this exercise we put into practice the concept of Tiles with Struts 2.

Although in this project we could have applied more features, such as the handling of messages, validations, errors, etc., we left it as something optional so that you can put it into practice. In this exercise we have left this project as simple as possible so that the use of Tiles de Struts 2 is clear.

We saw how it was possible to reuse a lot of the code of our views, and applying a general layout, that's how we reused header, footer, menu, etc. views.

However, we had to make several configurations in different files such as the file web.xml, struts.xml, tiles.xml, as well as between the action classes and the return type of type tile that we defined.

With this we conclude the theme of Tiles in Struts 2.

ONLINE COURSE

STRUTS 2 FRAMEWORK

By: Eng. Ubaldo Acosta