STRUTS FRAMEWORK COURSE

ANNOTATIONS WITH STRUTS 2 FRAMEWORK

By the expert: Ubaldo Acosta

Global Mentoring

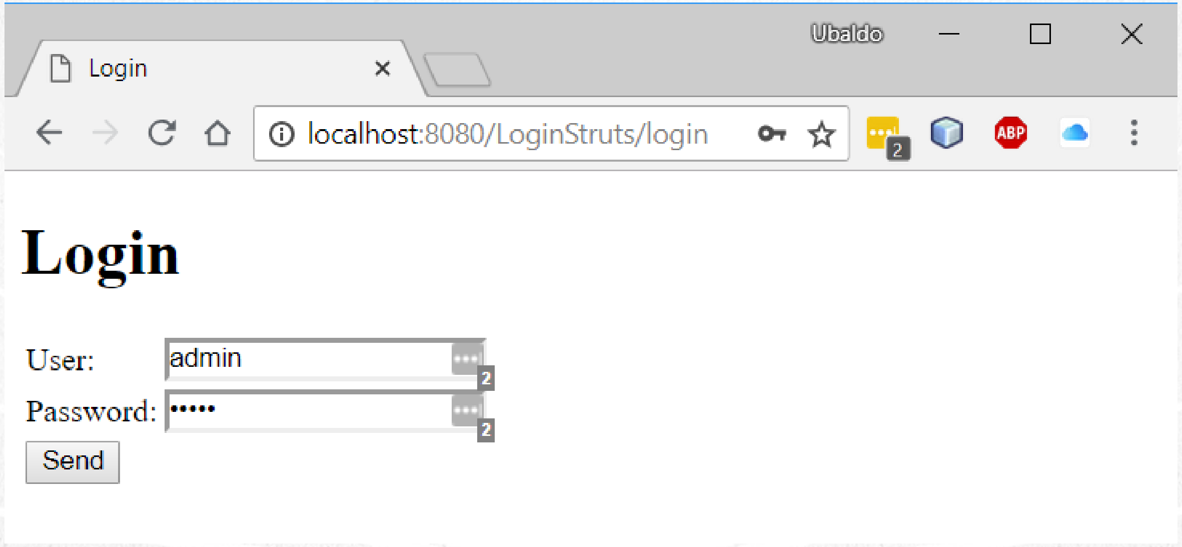
JAVA ONLINE COURSES UNIVERSITY

STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

EXERCISE OBJECTIVE

Create an application to implement the use of Annotations with Struts2. At the end we should observe the following:



Login

User: admin

Password: •••••

Send

EXERCISE REQUIREMENT

In this project we will obtain the same answer as in the lesson of Result with Struts 2. However, in this case we are going to apply the concept of annotations with Struts 2.

Although we can start from the previous project, we are going to create a completely new one, since it requires several changes that we are going to apply, and we don't want to add any confusion to these new changes.

EXERCISE REQUIREMENT

Create an application that does the following:

- It must allow to capture the attributes of user and password of type String.

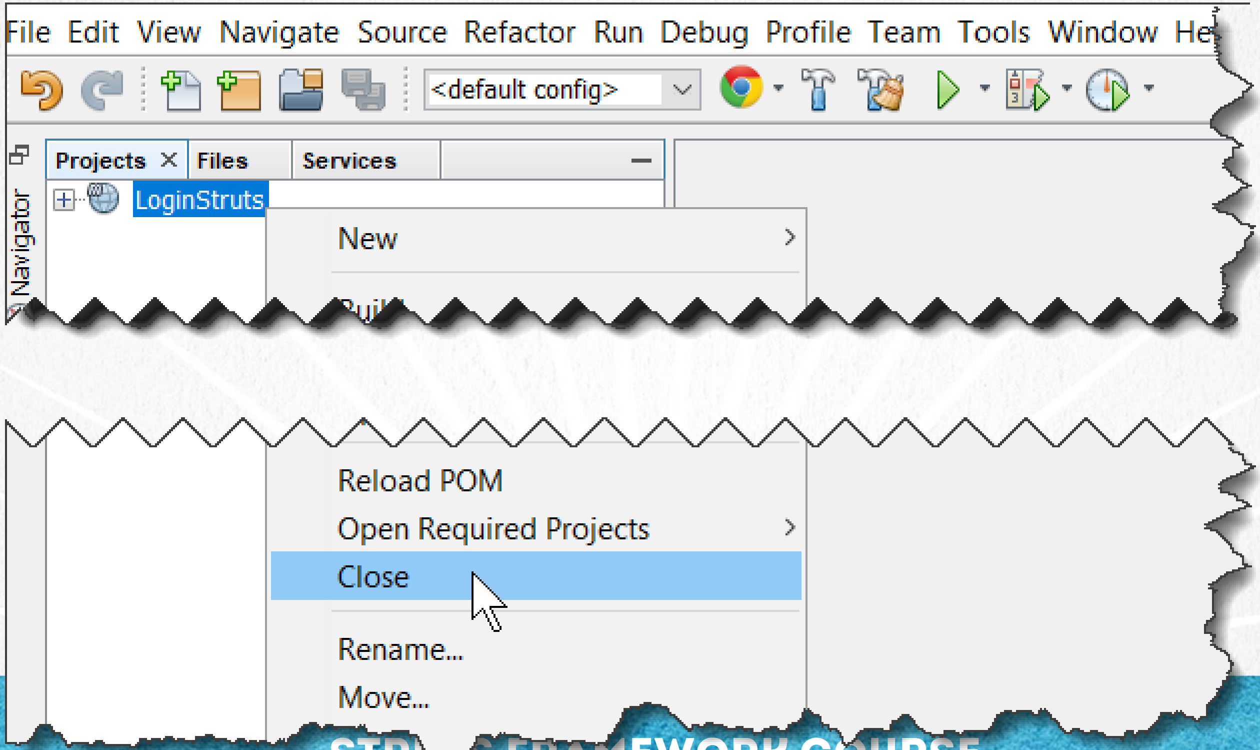
- The application will be called: LoginStruts

- There will be two path's mainly in the application:
  - ✓ The first one should show the login page. The path is: /login
  - ✓ The second should show the welcome page if the user is valid. The path is: /validateUser

- Although we can apply the management of conventions to simplify several configurations, in this exercise we will apply the concept of Annotations, so that we will explicitly indicate both the annotation of @Action and @Result although it is not necessary, since the idea is to put into practice the use of annotations. After this exercise it is optional the use of annotations or preferably apply the use of conventions by name to avoid configuration even by annotations or the struts.xml file.

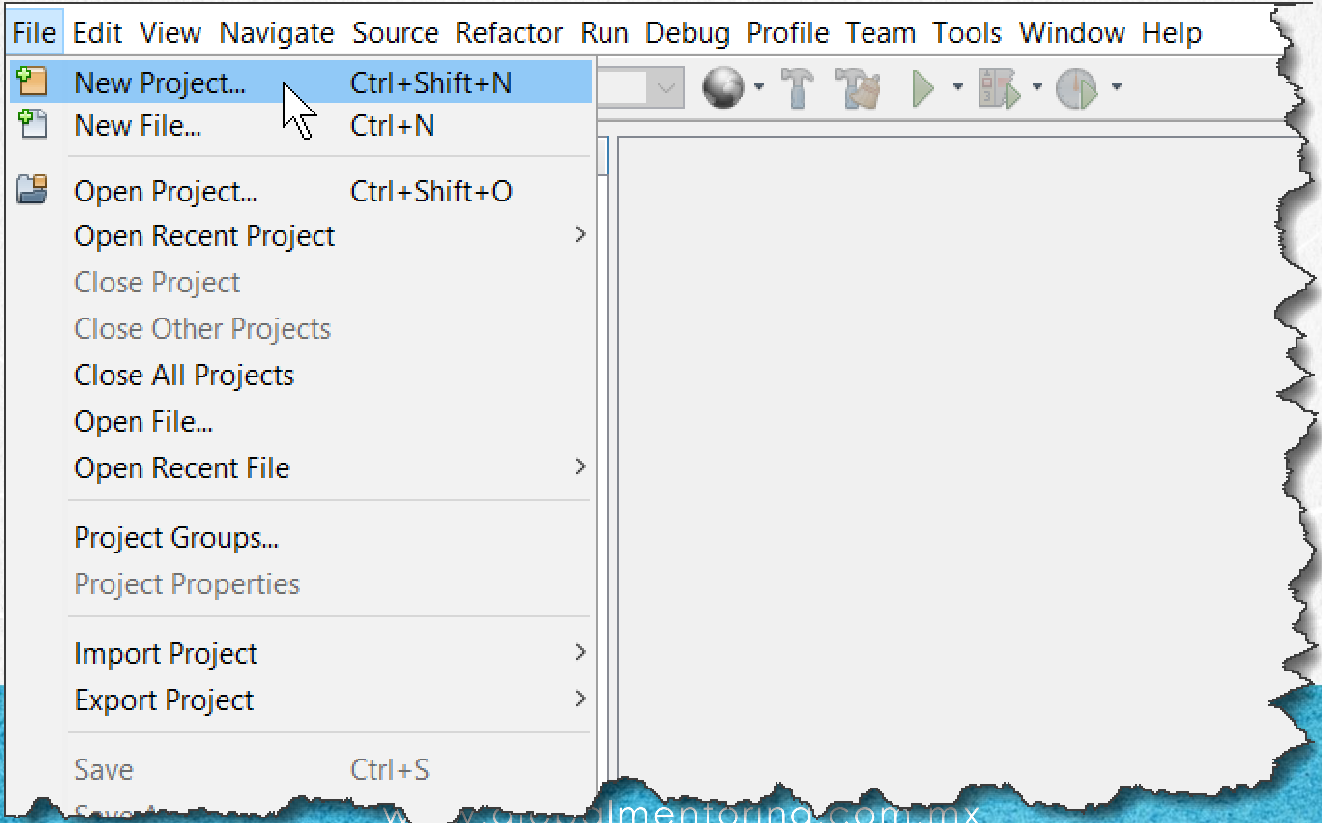
CLOSE PROJECTS THAT WE NO LONGER USE

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



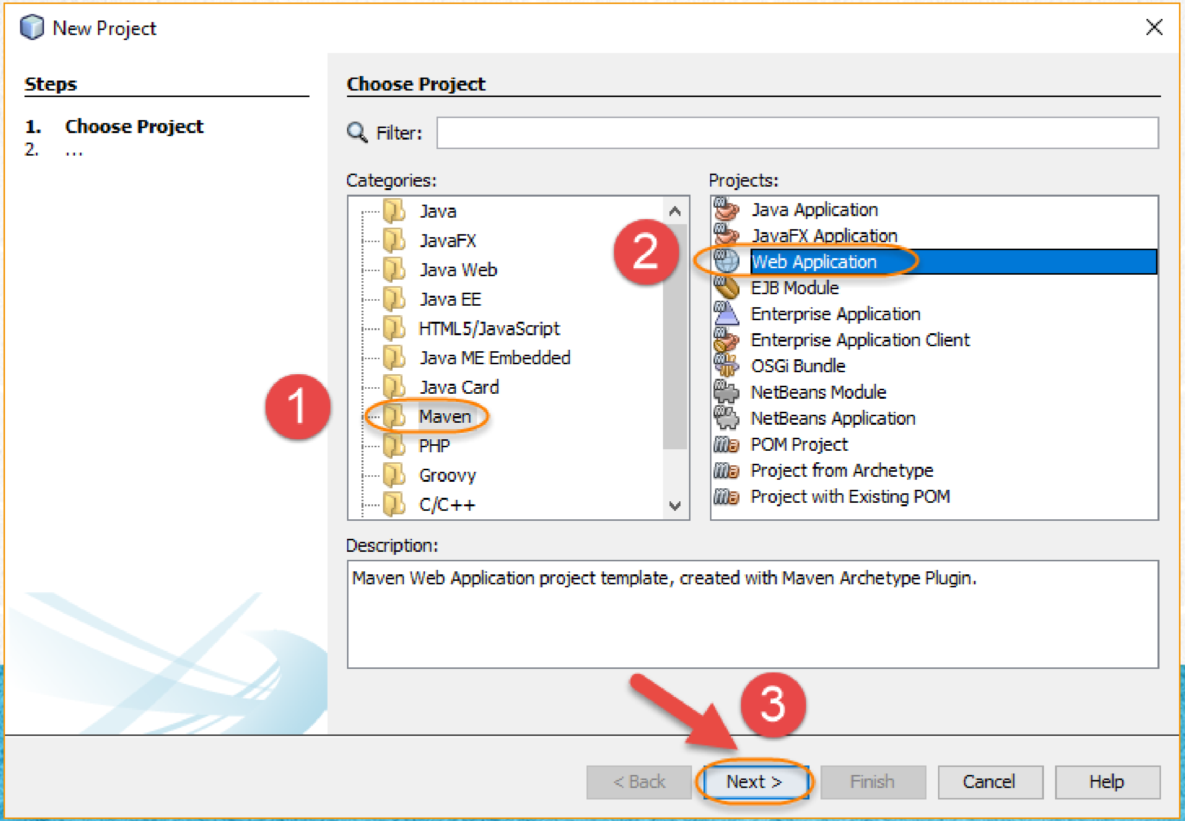
1. CREATE A NEW PROJECT

- We create the new project as shown below:



1. CREATE A NEW PROJECT

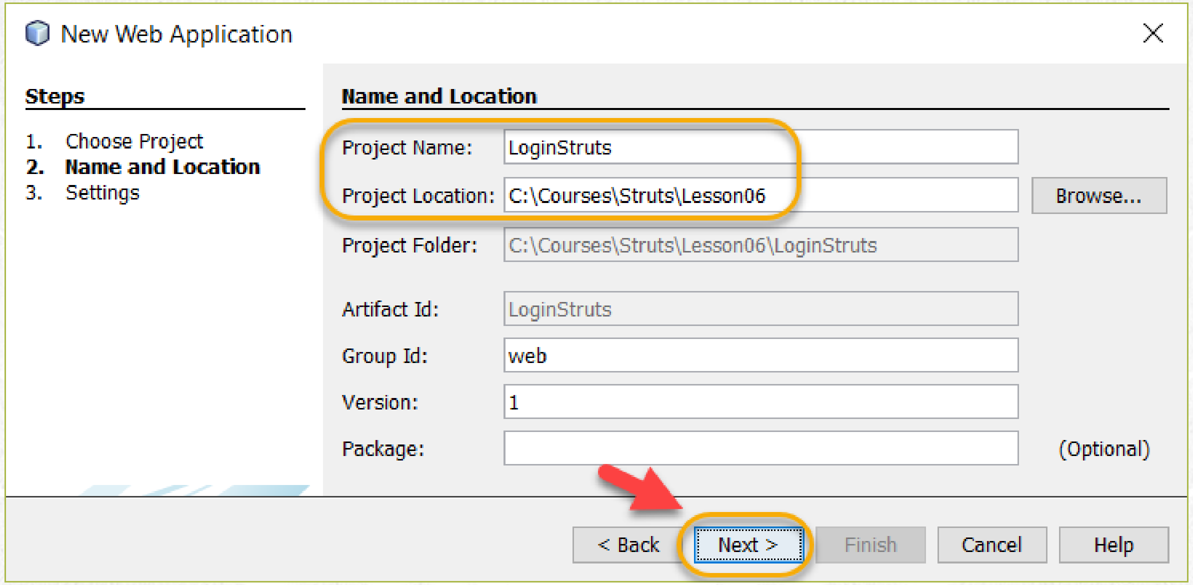
- We create the new project as shown below:



Maven Web Application project template, created with Maven Archetype Plugin.

1. CREATE A NEW PROJECT

- We create the new project as shown below:


New Web Application

Steps

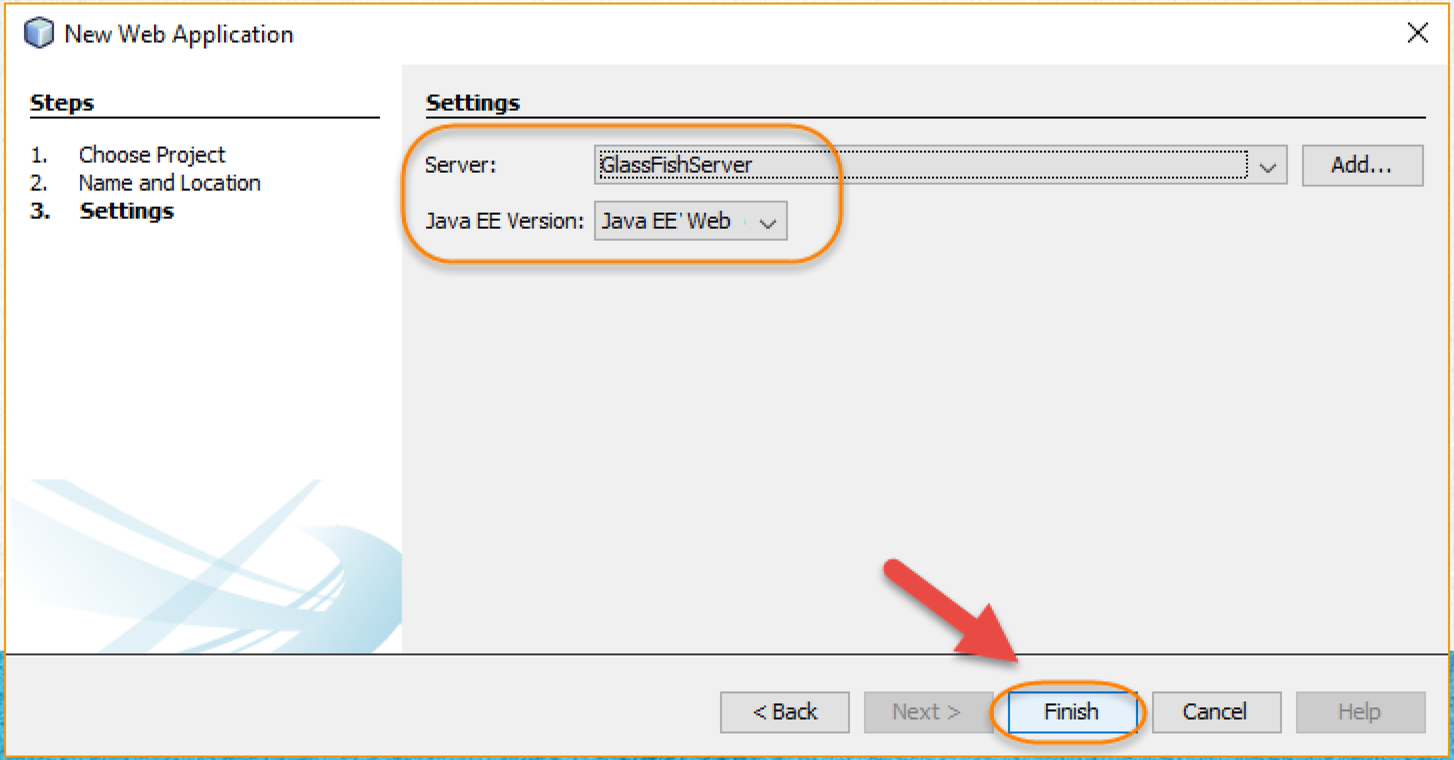
1. Choose Project
2. **Name and Location**
3. Settings

Name and Location

Project Name:	LoginStruts
Project Location:	C:\Courses\Struts\Lesson06
Project Folder:	C:\Courses\Struts\Lesson06\LoginStruts
Artifact Id:	LoginStruts
Group Id:	web
Version:	1
Package:	(Optional)

1. CREATE A NEW PROJECT

- We select the values shown:



New Web Application

Steps

1. Choose Project
2. Name and Location
3. **Settings**

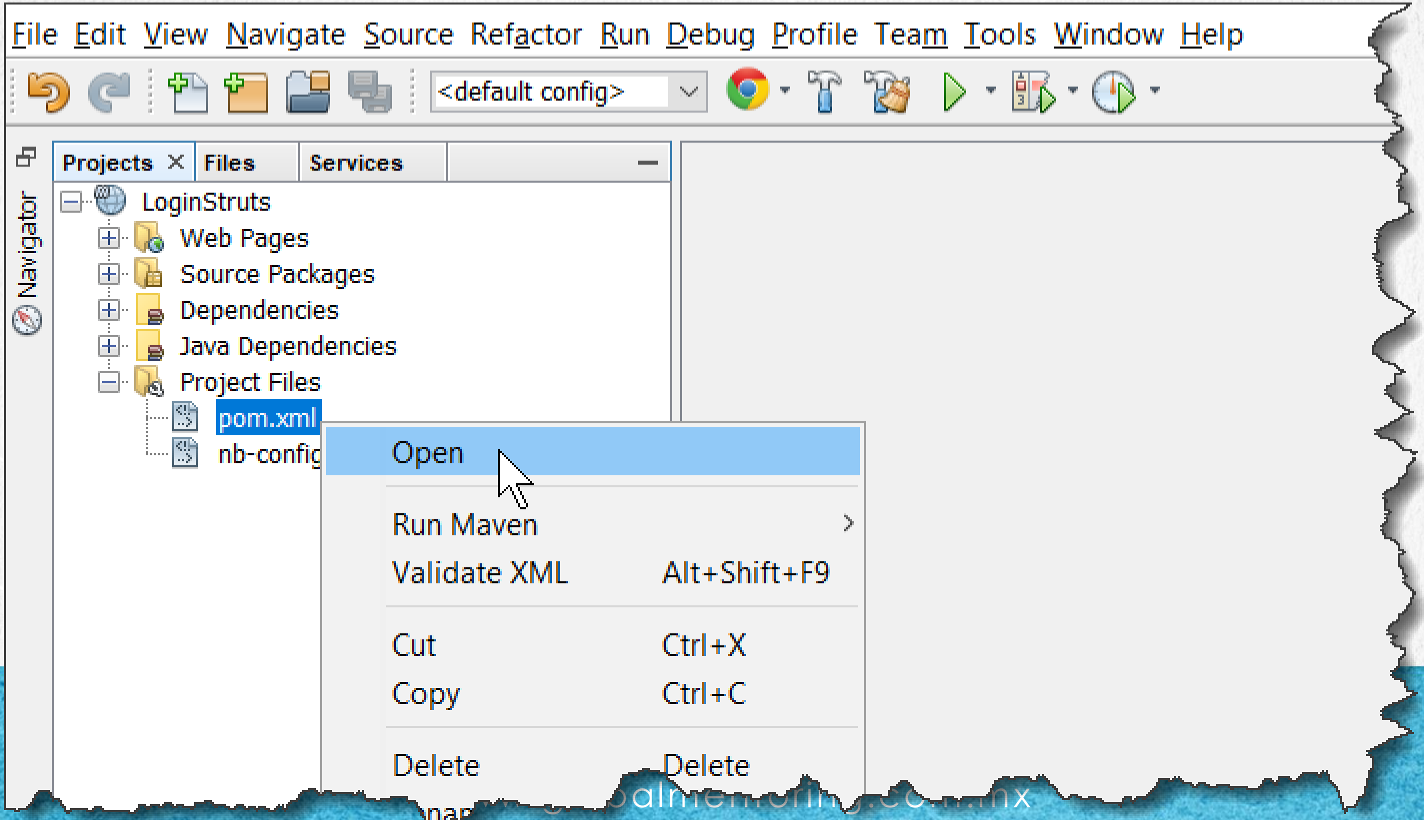
Settings

Server: GlassFishServer

Java EE Version: Java EE Web

2. OPEN MAVEN'S POM.XML FILE

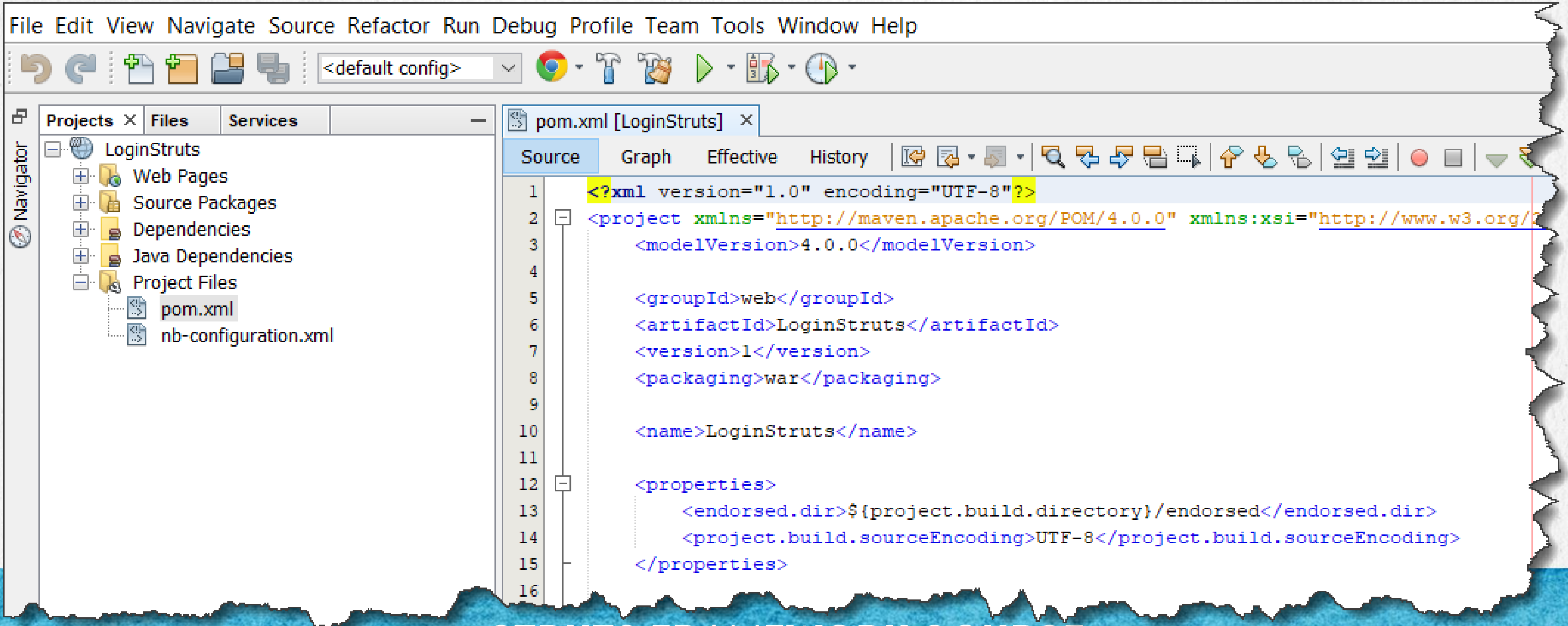
- The maven pom.xml file manages the Java libraries we will use :



www.globalmentoring.com.mx

2. OPEN MAVEN'S POM.XML FILE

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



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

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

    <name>LoginStruts</name>

    <properties>
        <endorsed.dir>${project.build.directory}/endorsed</endorsed.dir>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>
```

3. MODIFY THE FILE

pom.xml:

Click to download

```
<?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>LoginStruts</artifactId>
    <version>1</version>
    <packaging>war</packaging>

    <name>LoginStruts</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>
    </dependencies>
```

3. MODIFY THE FILE

pom.xml:

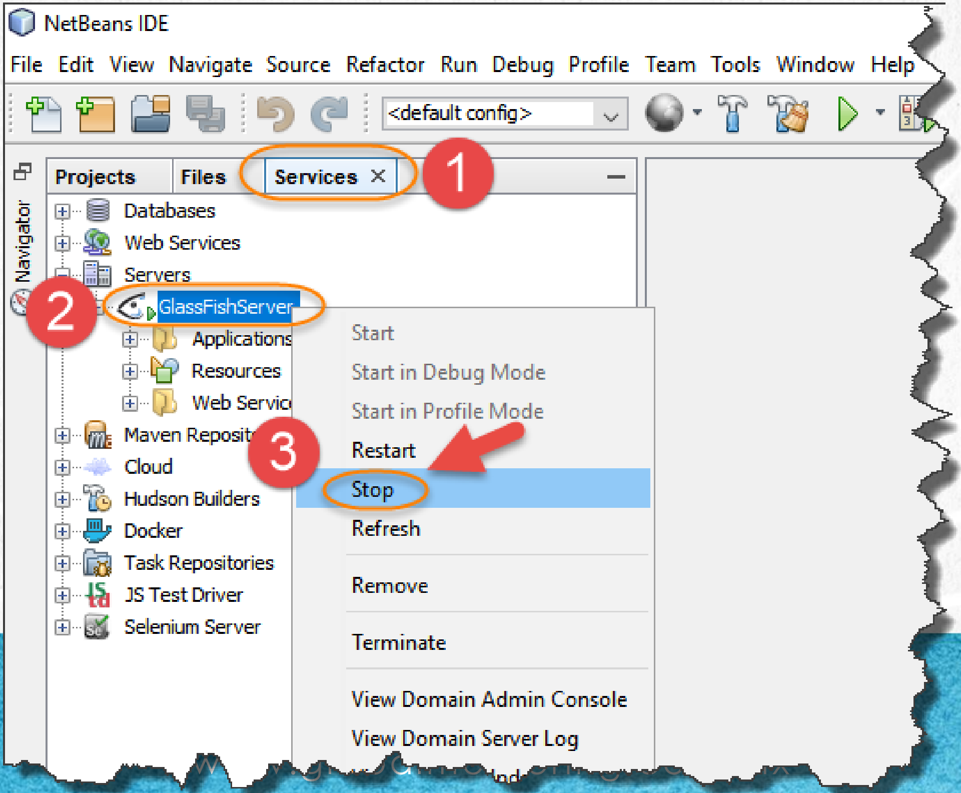
Click to download

```
    <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>
```

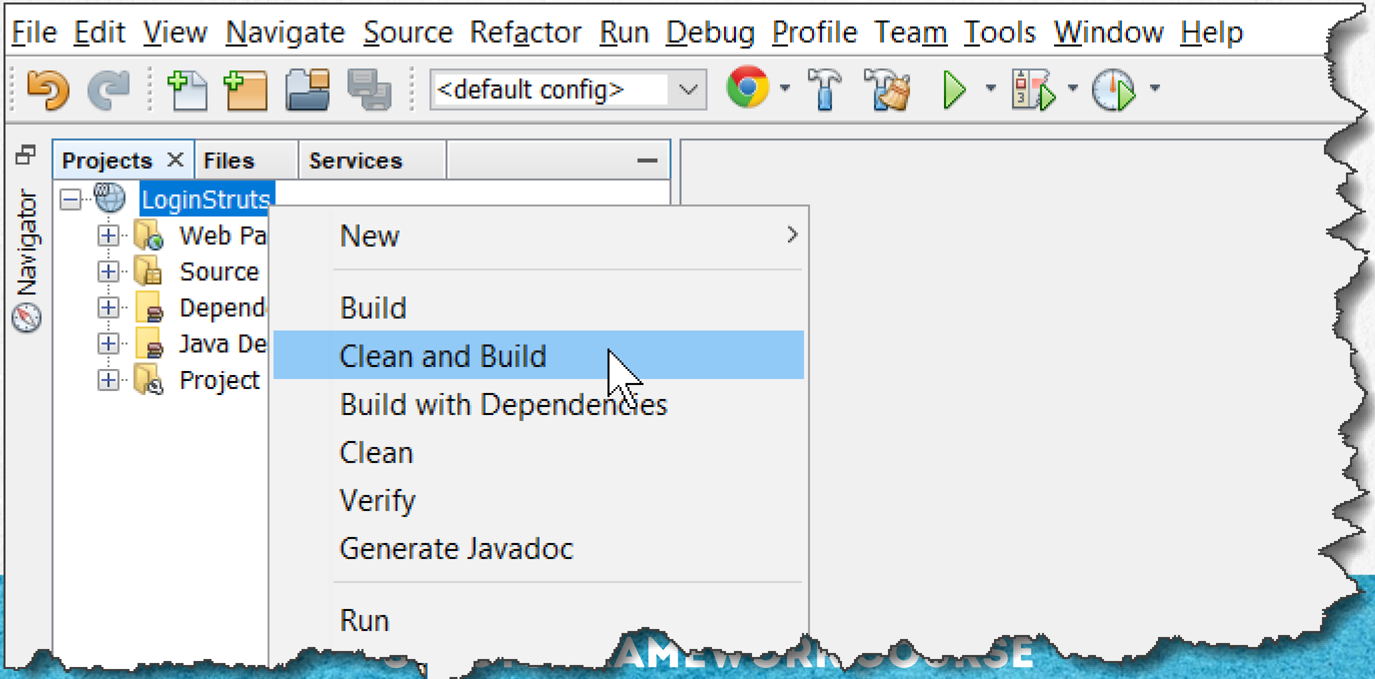
4. STOP GLASSFISH IF IT WAS STARTED

•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:



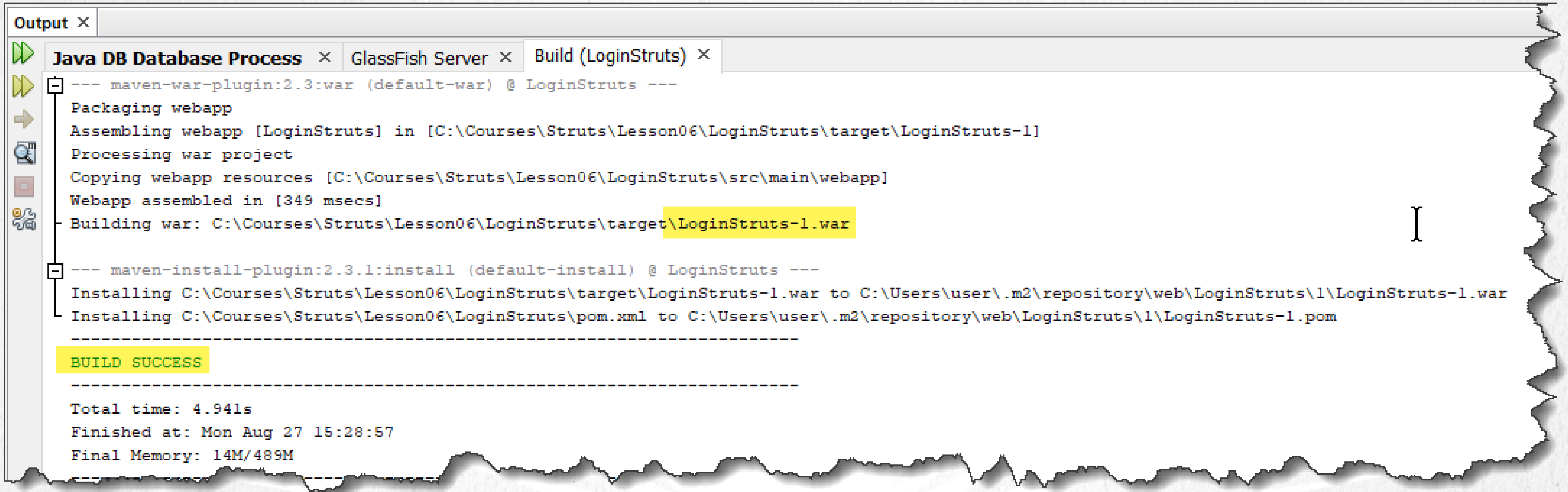
5. EXECUTE CLEAN & BUILD

•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:



```
--- maven-war-plugin:2.3:war (default-war) @ LoginStruts ---
Packaging webapp
Assembling webapp [LoginStruts] in [C:\Courses\Struts\Lesson06\LoginStruts\target\LoginStruts-1]
Processing war project
Copying webapp resources [C:\Courses\Struts\Lesson06\LoginStruts\src\main\webapp]
Webapp assembled in [349 msecs]
Building war: C:\Courses\Struts\Lesson06\LoginStruts\target\LoginStruts-1.war

--- maven-install-plugin:2.3.1:install (default-install) @ LoginStruts ---
Installing C:\Courses\Struts\Lesson06\LoginStruts\target\LoginStruts-1.war to C:\Users\user\.m2\repository\web\LoginStruts\1\LoginStruts-1.war
Installing C:\Courses\Struts\Lesson06\LoginStruts\pom.xml to C:\Users\user\.m2\repository\web\LoginStruts\1\LoginStruts-1.pom
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 4.941s
Finished at: Mon Aug 27 15:28:57
Final Memory: 14M/489M
```

6. CREATE AN XML FILE

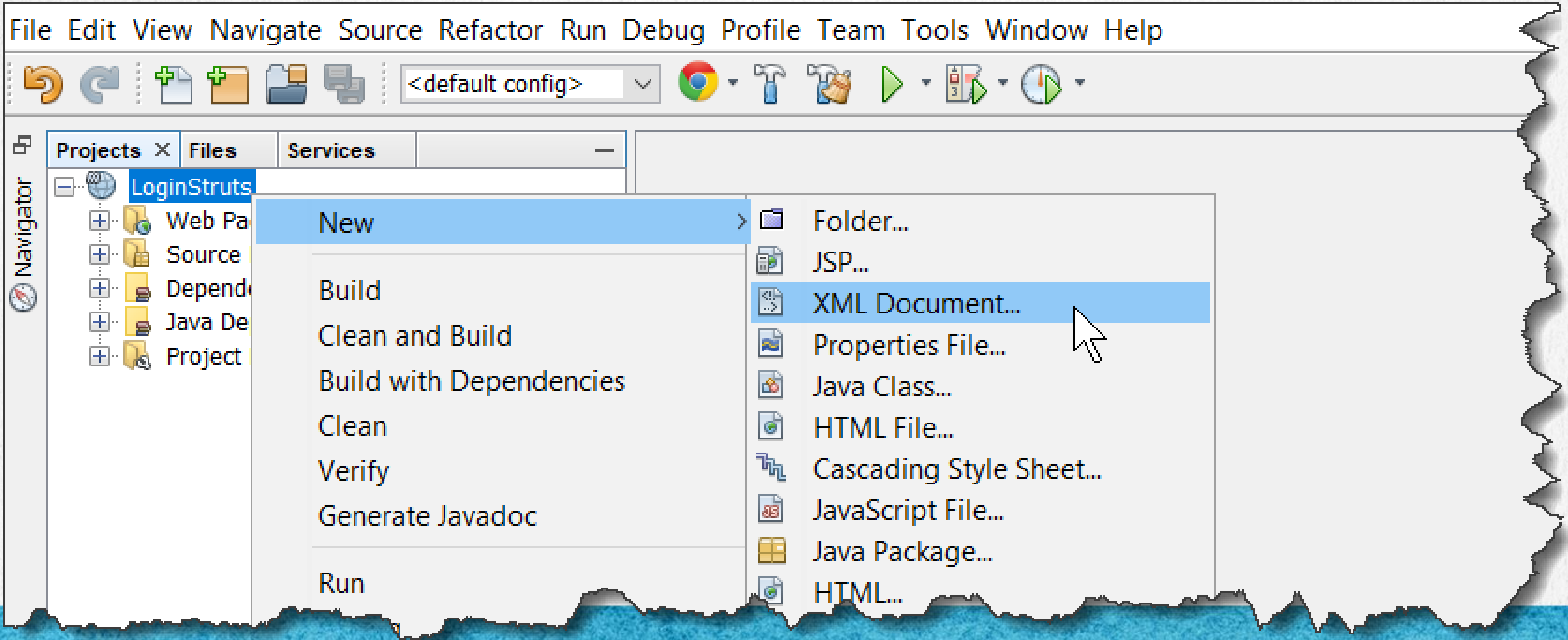
We are going to create the web.xml file below

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.

Experiencia y Conocimiento para tu vida

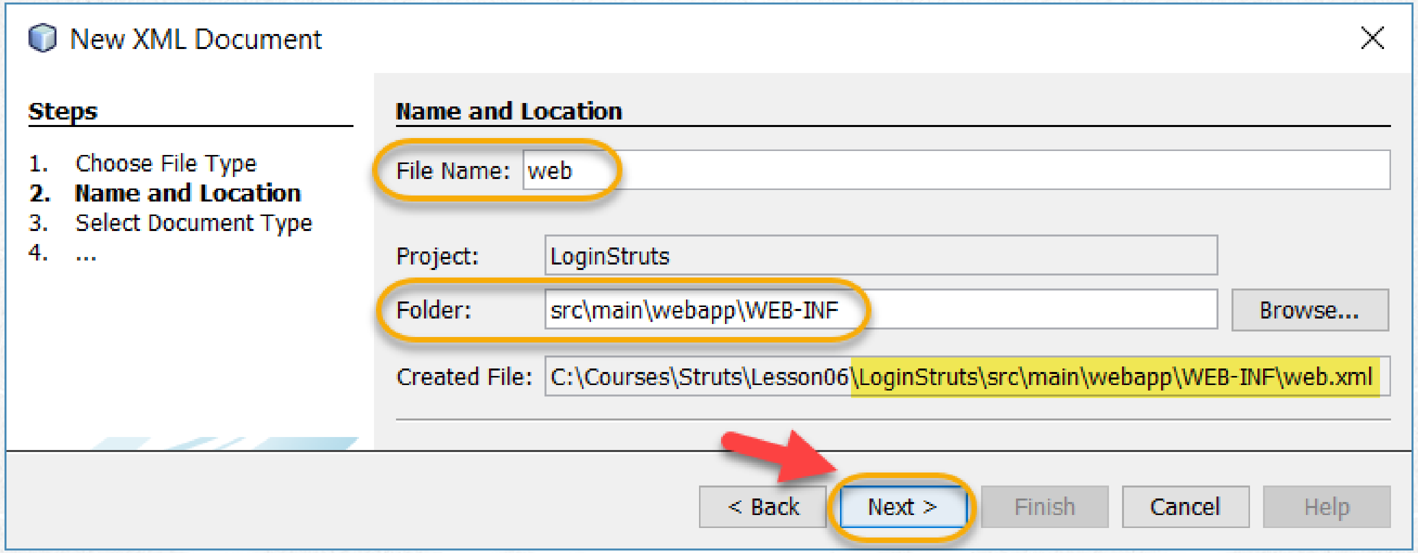
6. CREATE AN XML FILE

- 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 route shown:



New XML Document

Steps

1. Choose File Type
2. **Name and Location**
3. Select Document Type
4. ...

Name and Location

File Name: web

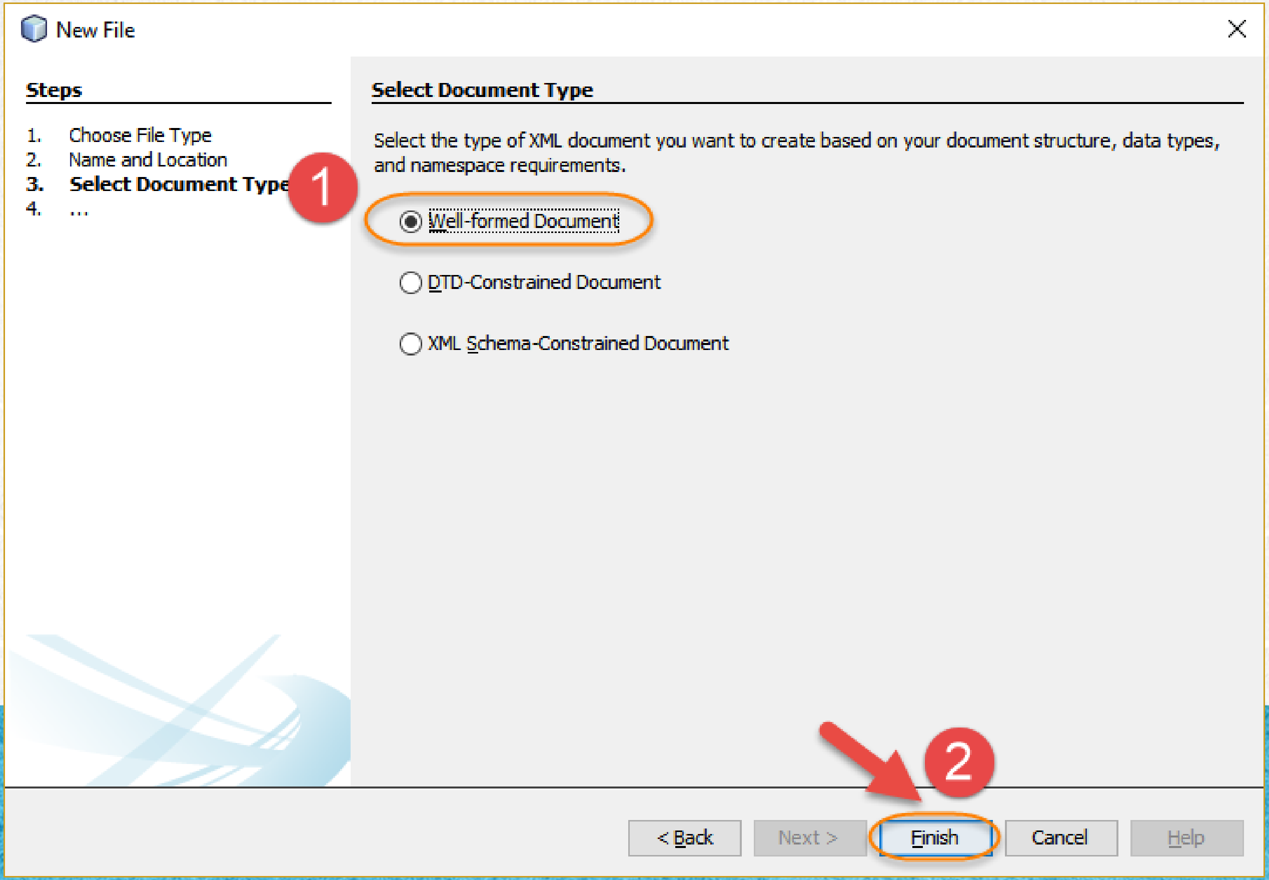
Project: LoginStruts

Folder: src\main\webapp\WEB-INF

Created File: C:\Courses\Struts\Lesson06\LoginStruts\src\main\webapp\WEB-INF\web.xml

6. Create an XML file

- We select the indicated type and click on finish.



New File

Steps

1. Choose File Type
2. Name and Location
3. **Select Document Type**
4. ...

Select Document Type

Select the type of XML document you want to create based on your document structure, data types, and namespace requirements.

(●) Well-formed Document

( ) DTD-Constrained Document

( ) XML Schema-Constrained Document

7. MODIFY THE CODE

web.xml:

Click to download

```
<?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">

    <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. MODIFY THE INDEX.HTML FILE

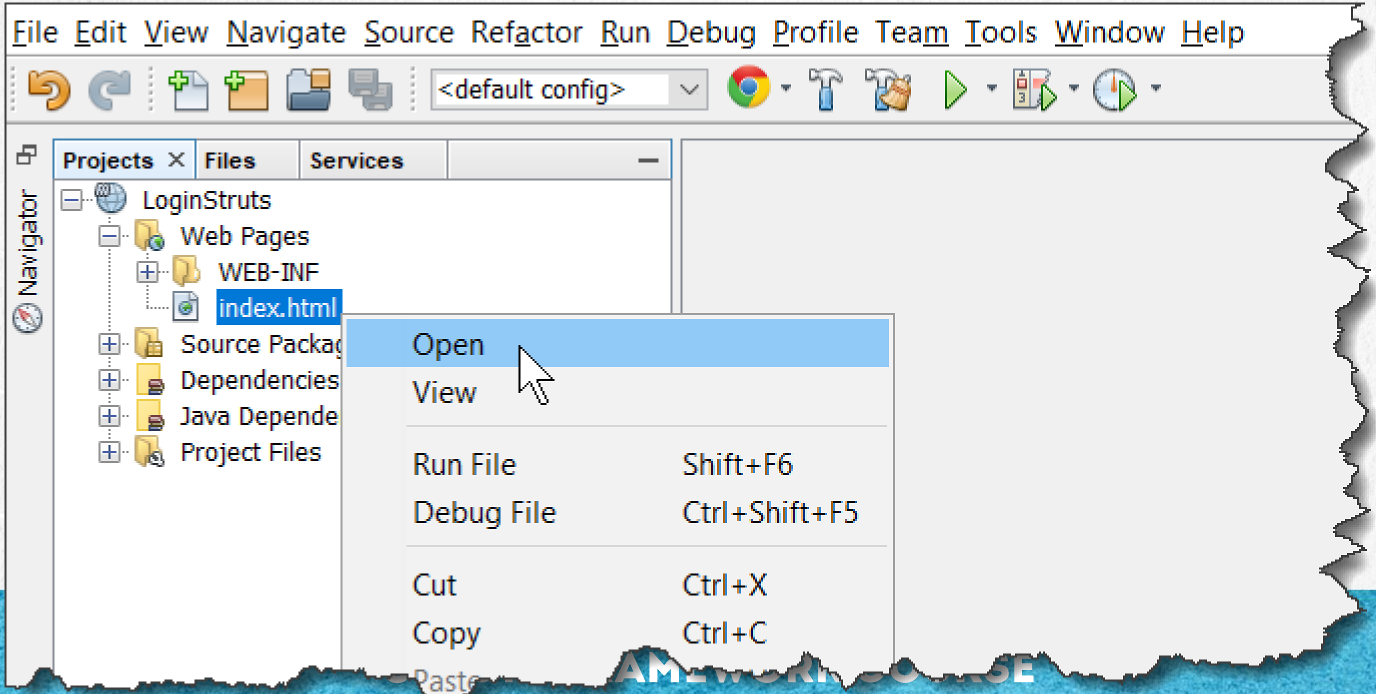
In automatic the IDE adds a file called index.html. However, if this file is not created we must add it to the project at the root level of Web Pages.

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: /login

8. 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 create it, as shown:



8. MODIFY THE CODE

index.html:

Click to download

```
<!DOCTYPE html>
<html>
    <head>
        <title>Home</title>
        <meta charset="UTF-8">
    </head>
    <body>
        <a href="login">Go to the system</a>
    </body>
</html>
```

9. CREATE A JAVA CLASS

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

We are going to extend the ActionSupport class and to overwrite the execute method.

This class could be omitted as we saw in the previous exercise, since the configuration could be added to the struts.xml file, however the objective of this exercise is to use the concept of annotations explicitly to avoid the use of the struts.xml file .

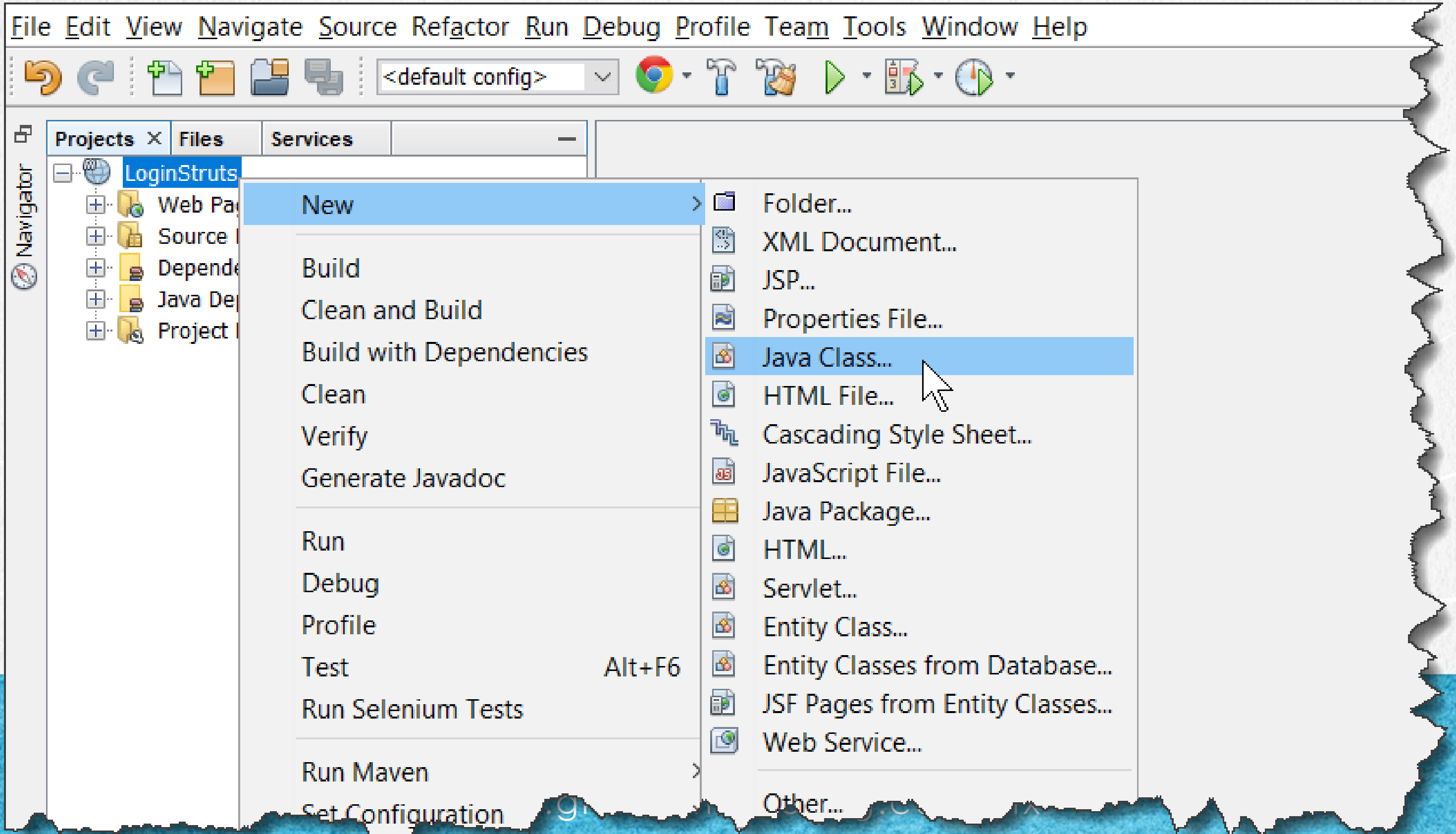
Therefore, this class will only redirect to the login JSP.

Both the annotation of @Action and the notation of @Result is optional to use them since we are handling the concept of Struts 2 conventions. However, we are going to implement the use of Struts 2 annotations.

Let's see how our class is.

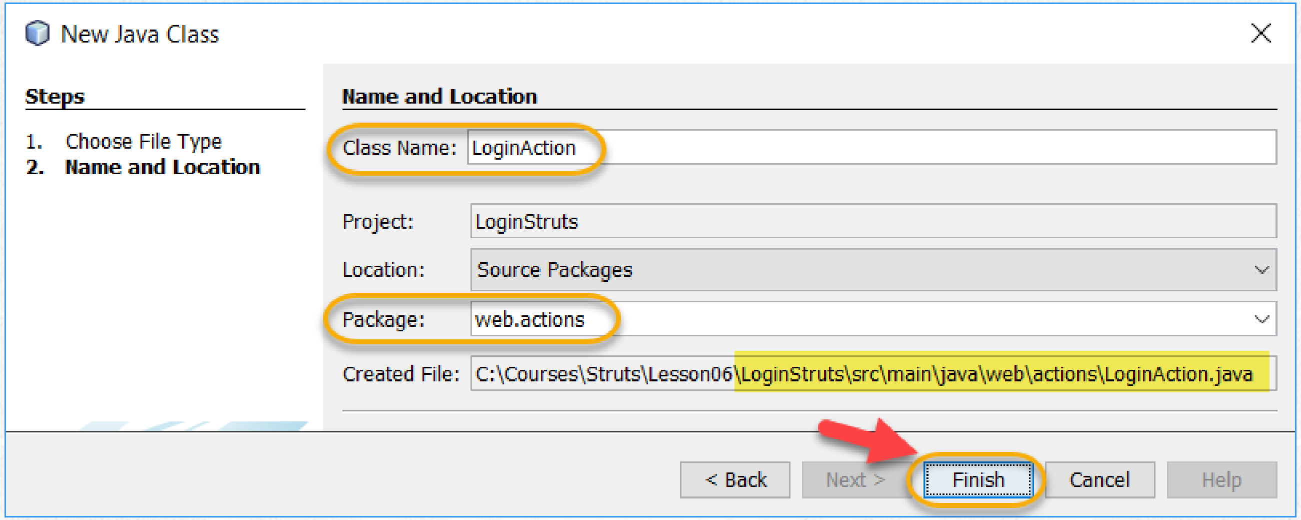
9. Create a Java class

- We create the class LoginAction.java:



9. CREATE A JAVA CLASS

- We create the class LoginAction.java:



New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name: LoginAction

Project: LoginStruts

Location: Source Packages

Package: web.actions

Created File: C:\Courses\Struts\Lesson06\LoginStruts\src\main\java\web\actions\LoginAction.java

10. MODIFY THE CODE

LoginAction.java:

Click to download

```java
package web.actions;

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

@Result(name = "success", location = "/WEB-INF/content/login.jsp")
public class LoginAction extends ActionSupport {

    @Action("login")
    @Override
    public String execute() {
        return SUCCESS;
    }
}
```

11. CREATE A PROPERTIES FILE

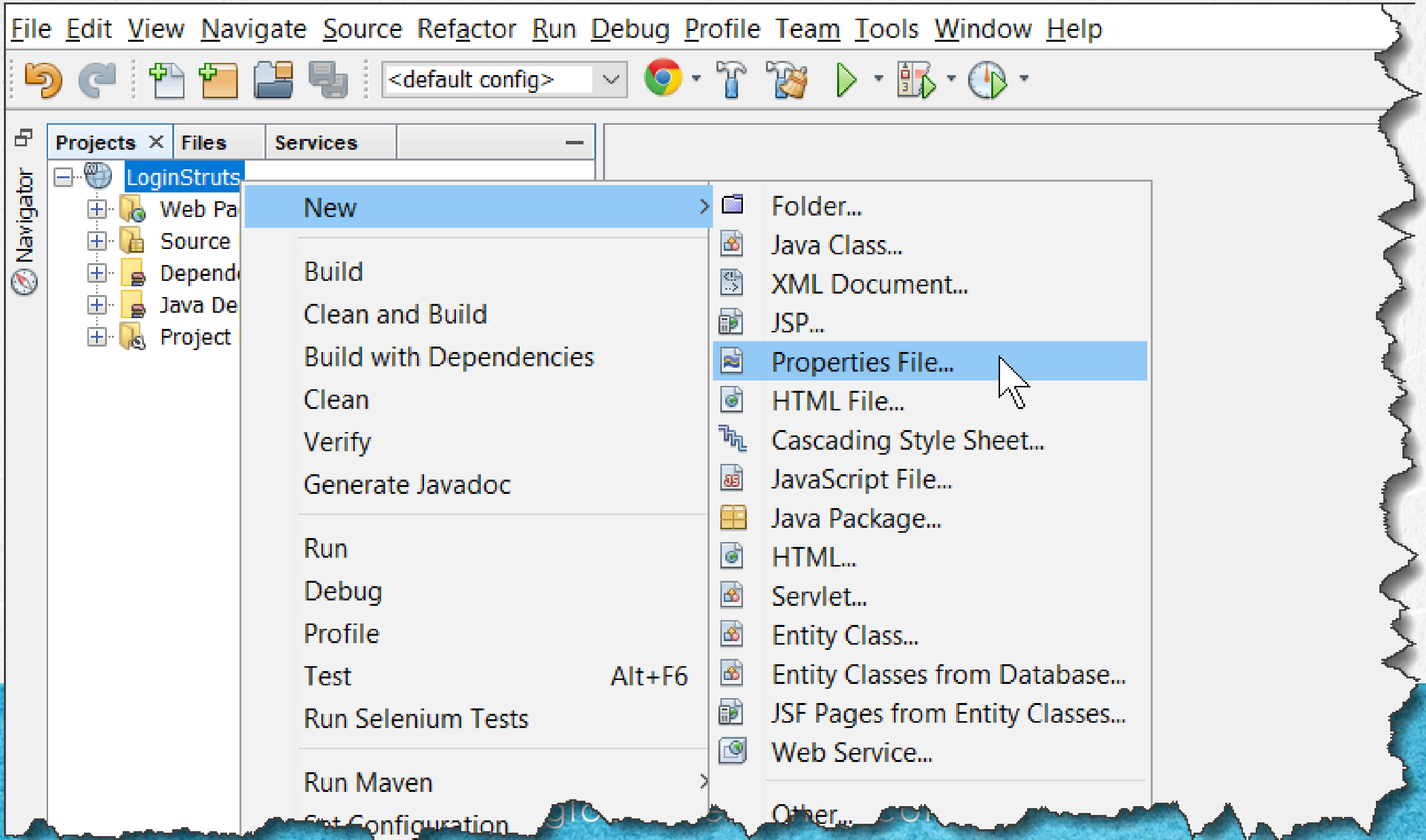
We are going to create a property file that will contain the messages that we are going to handle for the JSP associated with the response of the Action class defined above.

This file will handle the configuration by conventions, so it must be called equal to the Action class that we have created. Therefore the file will be called: LoginAction.properties

Because we are using Maven, this file must be deposited in the resources folder in a package identical to where the Action class is located, let's see how:

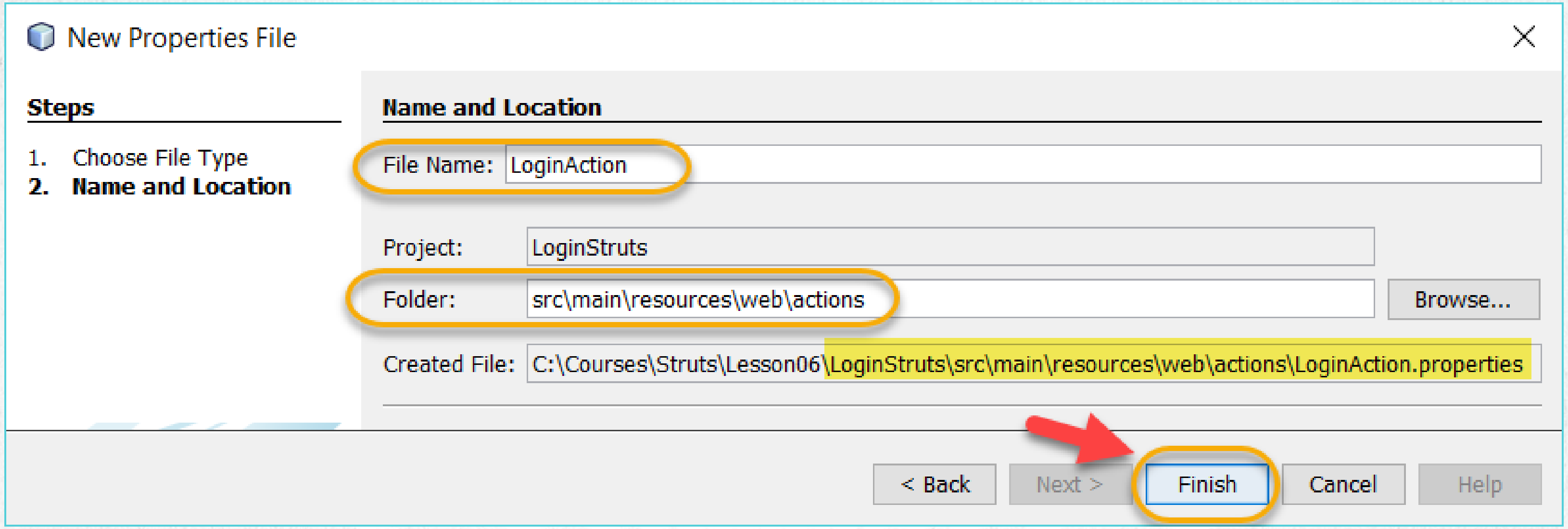
11 . CREATE A PROPERTIES FILE

- Create the LoginAction.properties file:



11 . CREATE A PROPERTIES FILE

- We provide the values as shown. The path is the following:



New Properties File

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

File Name: LoginAction

Project: LoginStruts

Folder: src\main\resources\web\actions

Created File: C:\Courses\Struts\Lesson06\LoginStruts\src\main\resources\web\actions\LoginAction.properties

12. MODIFY THE CODE

LoginAction.properties:

Click to download

```
form.user: User
form.password: Password
form.button: Send
form.title: Login
```

13. CREATE A NEW JSP FILE

Now we create the file: login.jsp. Remember that this name corresponds to the path that will be used to call the corresponding action LoginAction.java

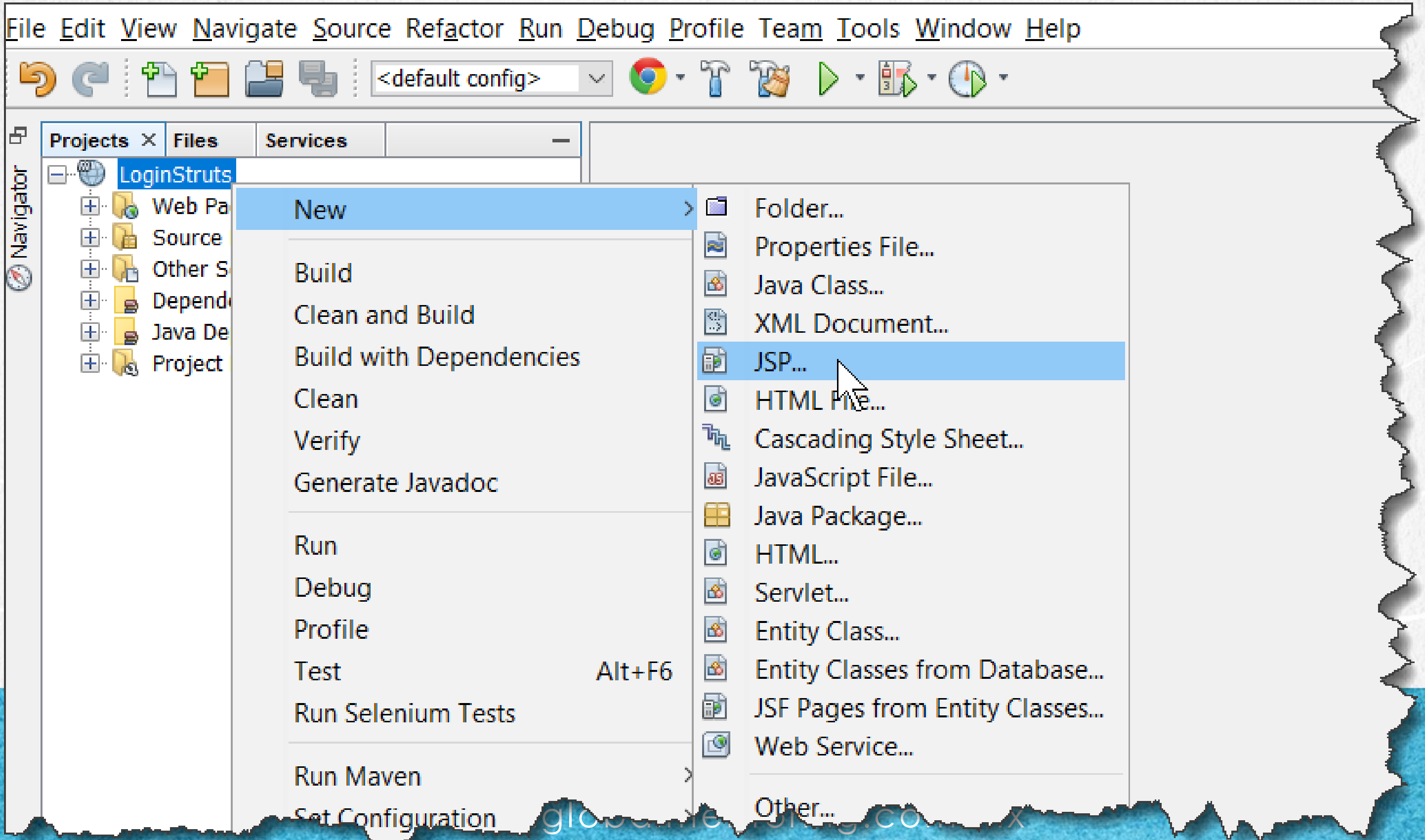
We must also deposit this JSP in the folder /WEB-INF/content as we have seen in the Struts 2 conventions topic.

However, within our LoginAction.java class we have used the @Result annotation which overwrites the default behavior of the handling of conventions by name to implement the concept of annotations.

It is optional to use the @Result annotation if the use of conventions by name of Struts 2 is being used as we have studied previously.

13. Create a new JSP

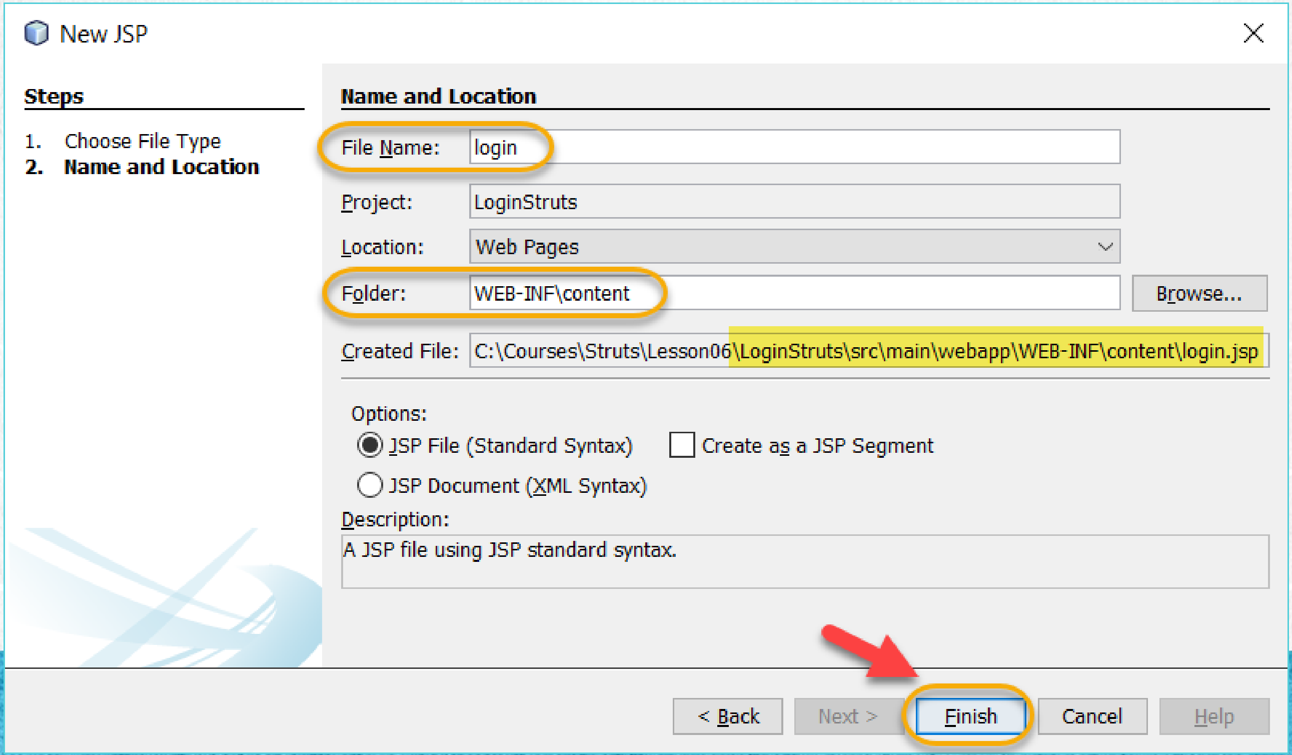
- Create the login.jsp file:



globalnerdy.com

13. CREATE A NEW JSP

- Create the login.jsp file in the path shown below:


New JSP

Steps
1. Choose File Type
2. **Name and Location**

Name and Location

File Name: login

Project: LoginStruts

Location: Web Pages

Folder: WEB-INF\content

Created File: C:\Courses\Struts\Lesson06\LoginStruts\src\main\webapp\WEB-INF\content\login.jsp

Options:
[● JSP File (Standard Syntax)] ☐ Create as a JSP Segment
○ JSP Document (XML Syntax)

Description:
A JSP file using JSP standard syntax.

14. MODIFY THE CODE

login.jsp:

Click to download

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib  prefix="s" uri="/struts-tags" %>
<!DOCTYPE html>
<html>
    <head>
        <title><s:text name="form.title" /></title>
    </head>
    <body>

        <h1><s:text name="form.title" /></h1>

        <%--The url is of the form is: / validateUser--%>
        <s:form action="validateUser">
            <s:textfield key="form.user" name="user" />
            <s:password key="form.password" name="password" />
            <s:submit key="form.button" name="submit" />
        </s:form>
    </body>
</html>
```

15. CREATE A NEW JAVA CLASS

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

We are going to extend the ActionSupport class and to overwrite the execute method.

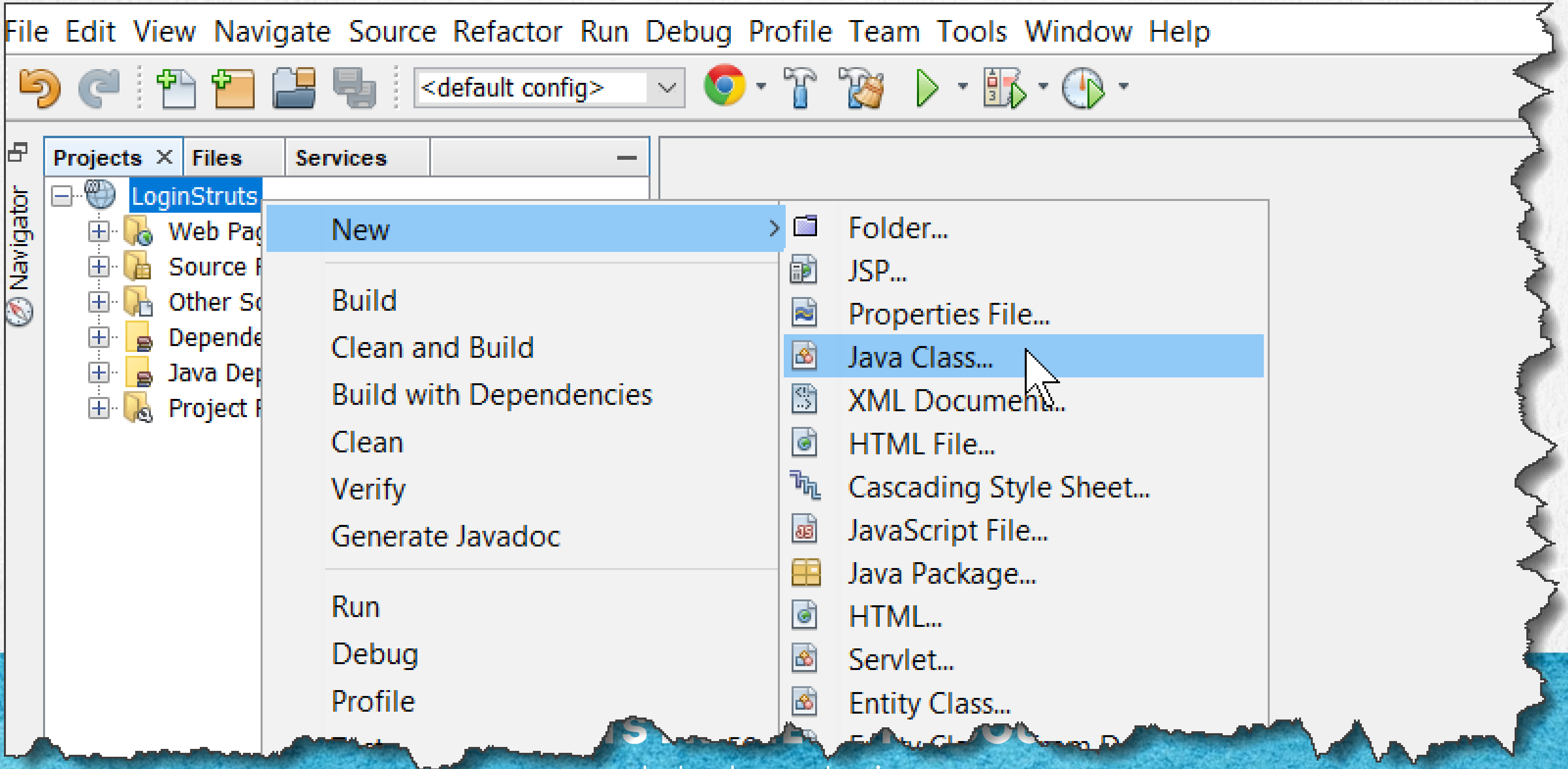
In this class we could apply the theme of conventions by Struts 2 name. However, we will apply the annotations of @Action, @Results and @Result to implement these annotations.

Recall that one of the objectives of using annotations is to avoid the use of this configuration by XML in the struts.xml file, therefore the configuration remains within our class of type Action.

Let's see how our class is.

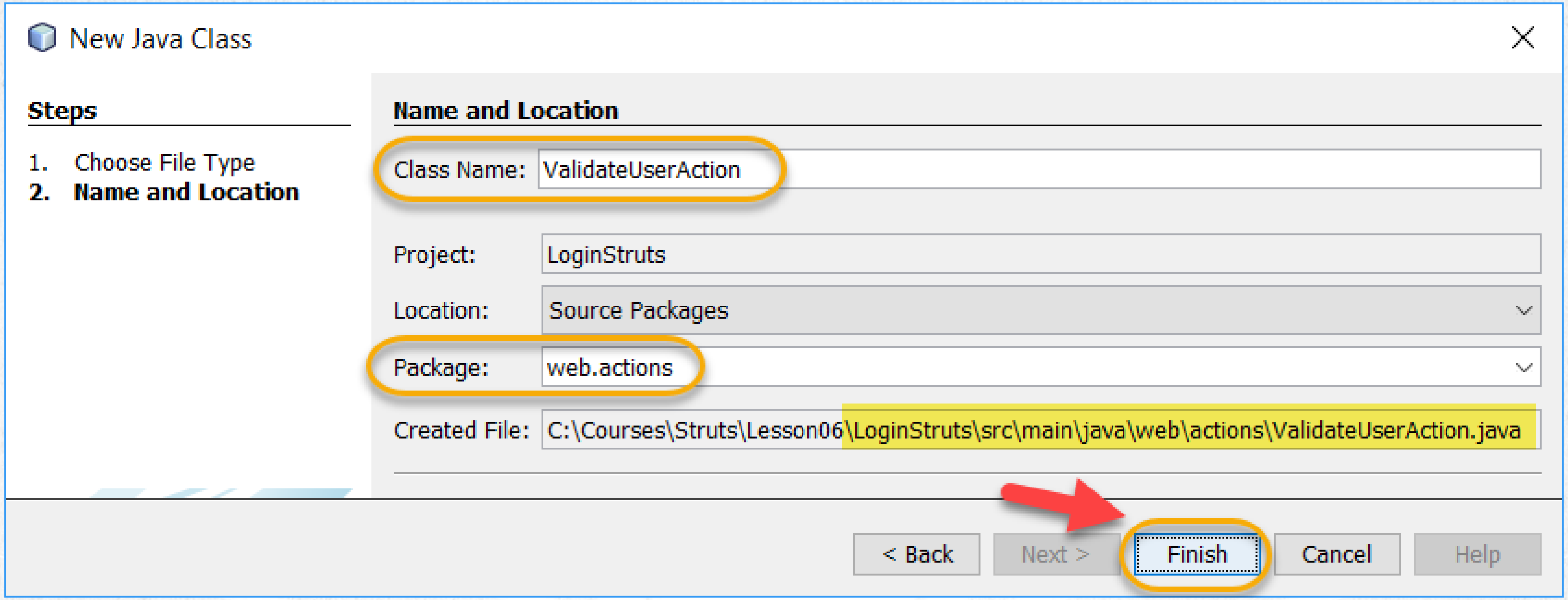
15. CREATE A NEW JAVA CLASS

- Create the ValidateUserAction.java class:



15. CREATE A NEW JAVA CLASS

- Create the ValidateUserAction.java class:



New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name: ValidateUserAction

Project: LoginStruts

Location: Source Packages

Package: web.actions

Created File: C:\Courses\Struts\Lesson06\LoginStruts\src\main\java\web\actions\ValidateUserAction.java

16. MODIFY THE CODE

ValidateUserAction.java:

Click to download

```
package web.actions;

import com.opensymphony.xwork2.ActionSupport;
import org.apache.logging.log4j.*;
import org.apache.struts2.convention.annotation.*;

@Results({
    @Result(name="success", location="/WEB-INF/content/welcome.jsp"),
    @Result(name="input", location="login", type="redirectAction")
})
public class ValidateUserAction extends ActionSupport {

    private String user;
    private String password;
    Logger log = LogManager.getLogger(ValidateUserAction.class);

    @Action("validateUser")
    @Override
    public String execute() {
        log.info("User:" + this.user);
        //If you are a valid user, we show the welcome.jsp page
        if ("admin".equals(this.user)) {
            return SUCCESS;
        } else {
            //If you are an invalid user, we return to login
            return INPUT;
        }
    }
```

16. MODIFY THE CODE

ValidateUserAction.java:

Click to download

```
    public String getUser() {
        return user;
    }

    public void setUser(String user) {
        this.user = user;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

17. Create a properties file

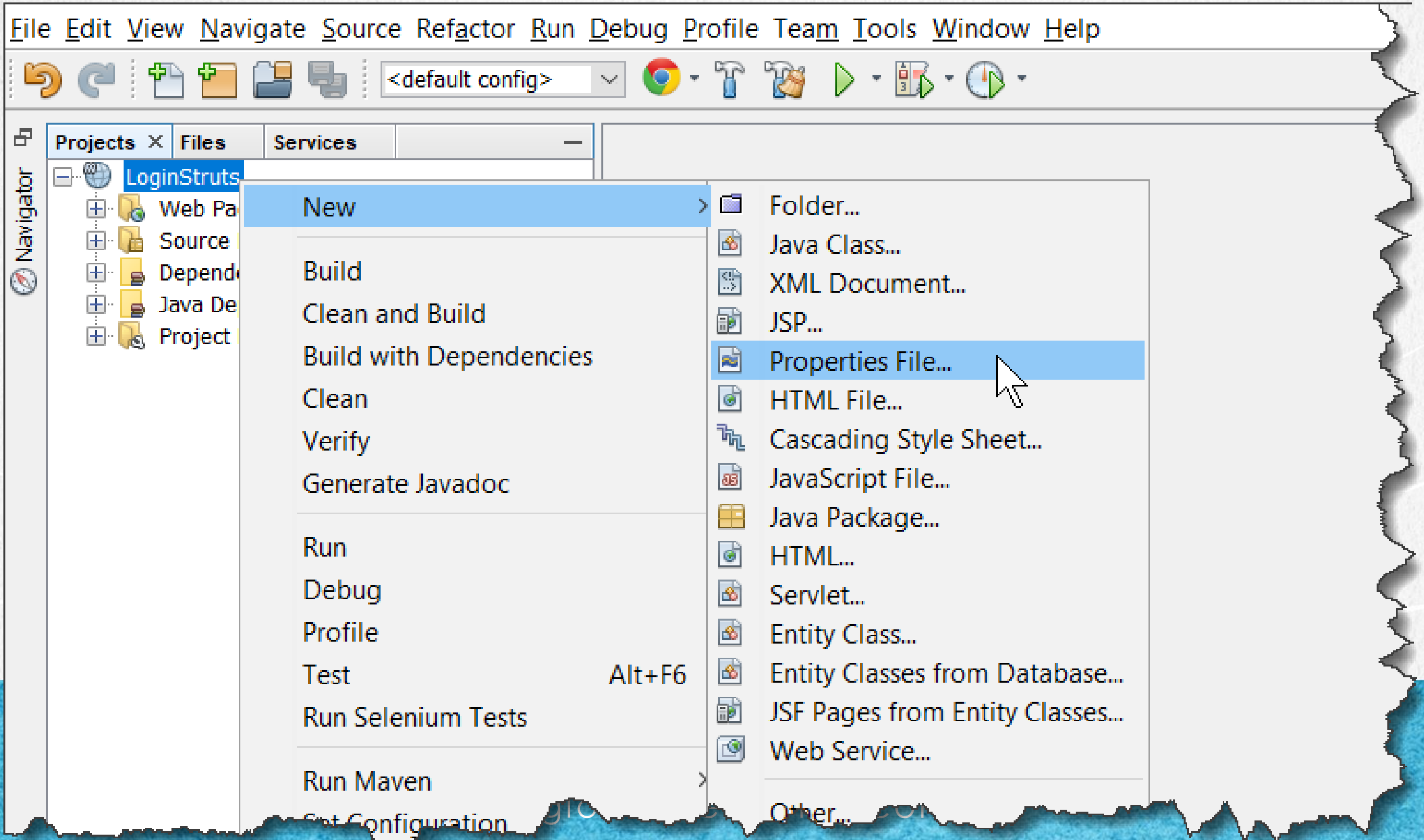
We are going to create a property file that will contain the messages that we are going to handle for the JSP associated with the response of the Action class defined above.

This file will handle the configuration by conventions, so it must be called equal to the Action class that we have created. Therefore the file will be called: ValidateUserAction.properties

Because we are using Maven, this file must be deposited in the resources folder in a package identical to where the Action class is located, let's see how:

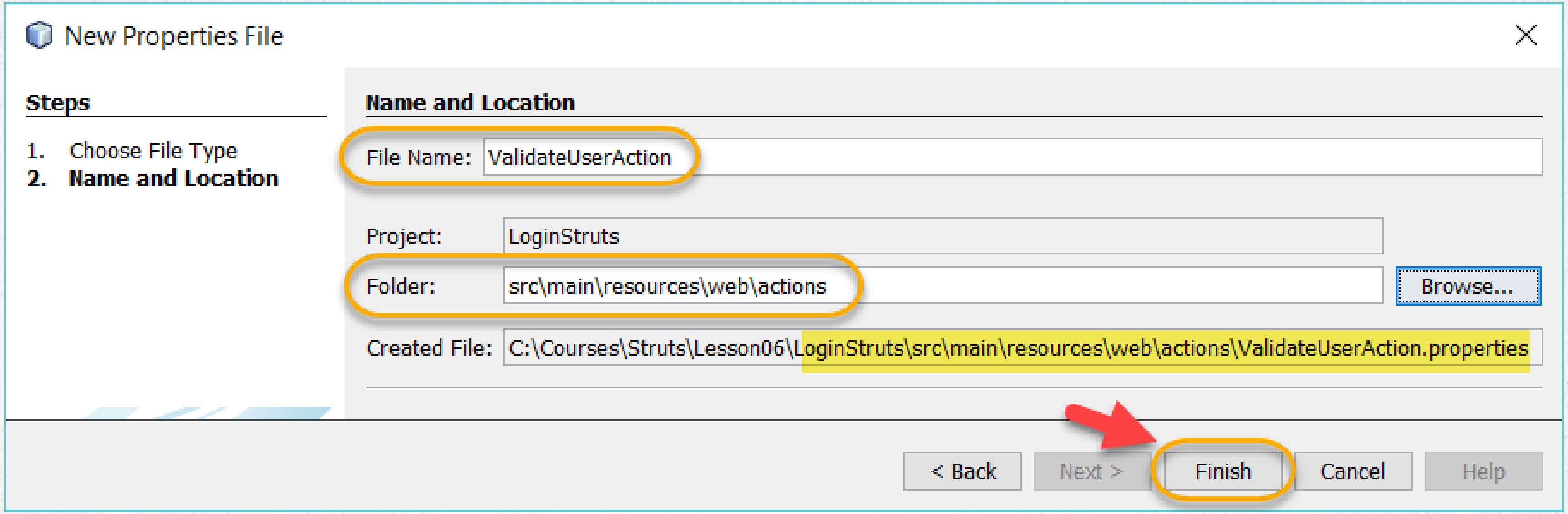
17. Create a Properties File

- We create the file ValidateUserAction.properties :



17. CREATE A PROPERTIES FILE

- We provide the values as shown. The path te is the following :



New Properties File

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

File Name: ValidateUserAction

Project: LoginStruts

Folder: src\main\resources\web\actions

Created File: C:\Courses\Struts\Lesson06\LoginStruts\src\main\resources\web\actions\ValidateUserAction.properties

18. MODIFY THE CODE

ValidateUserAction.properties:

Click to download

```
welcome.title: Welcome
welcome.message: Correct User
welcome.return: Return
```

19. CREATE A NEW JSP

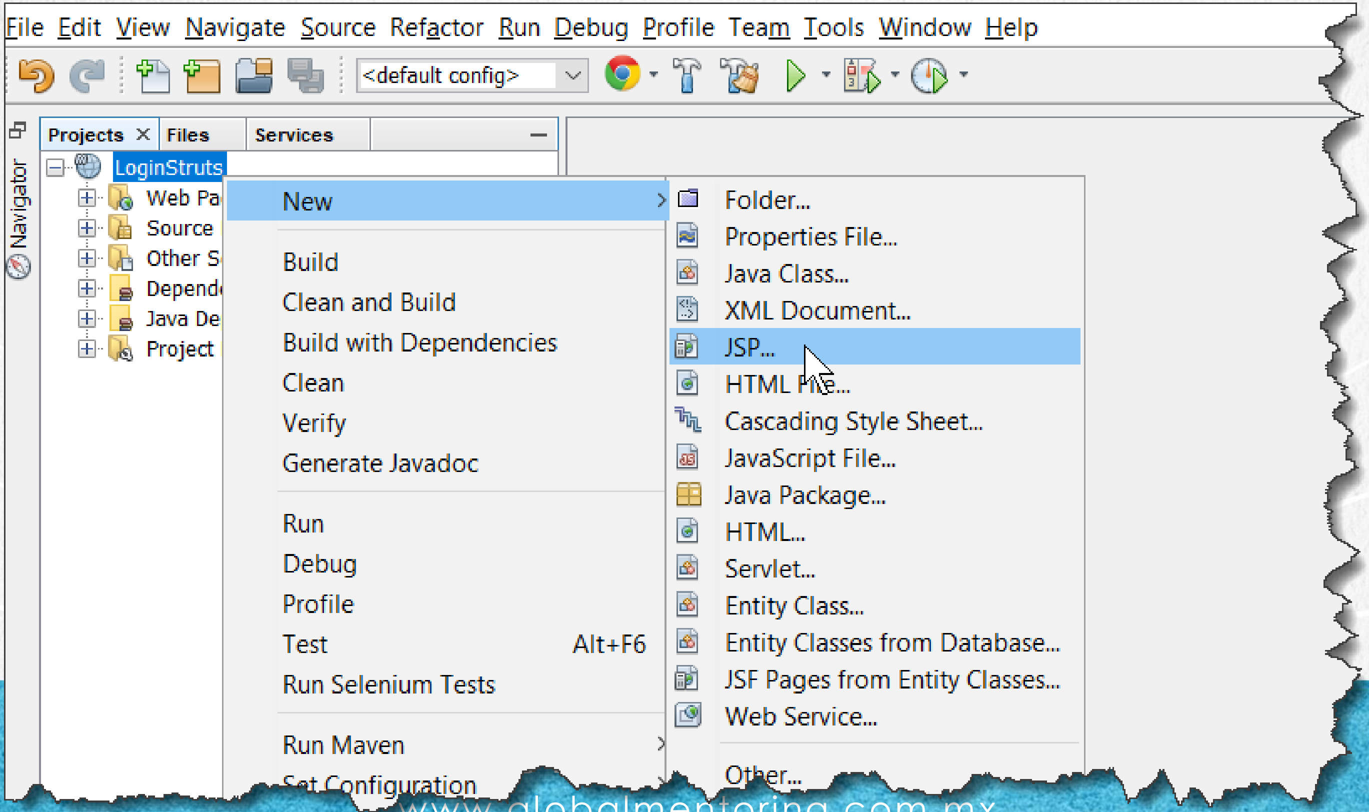
Now we create the file: welcome.jsp.

In this case we are not using the concept of conventions by name of Struts, so in this case it is necessary to specify which is the JSP path to be displayed in the annotation of @Result in the class ValidateUserAction.java

Therefore, it is optional to deposit this JSP in the WEB-INF/content folder, however we will deposit it there to have our JSPs organized.

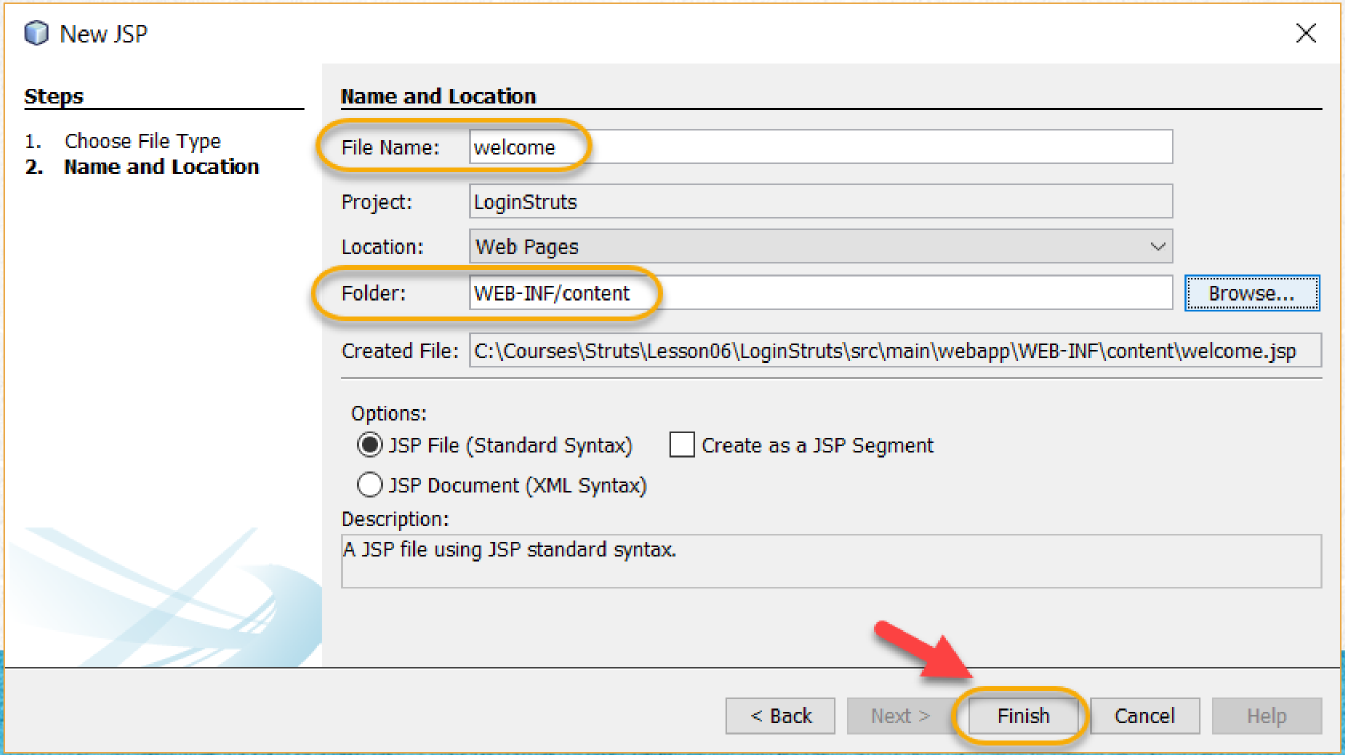
19. CREATE A NEW JSP

- Create a new welcome.jsp file:



19. CREATE A NEW JSP

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



New JSP

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

File Name: welcome

Project: LoginStruts

Location: Web Pages

Folder: WEB-INF/content

Created File: C:\Courses\Struts\Lesson06\LoginStruts\src\main\webapp\WEB-INF\content\welcome.jsp

Options:

(●) JSP File (Standard Syntax) ☐ Create as a JSP Segment

( ) JSP Document (XML Syntax)

Description:

A JSP file using JSP standard syntax.

20. MODIFY THE CODE

welcome.jsp:

Click to download

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ taglib prefix="s" uri="/struts-tags" %>
<html>
    <head>
        <title><s:text name="welcome.title" /></title>
    </head>
    <body>
        <h1><s:text name="welcome.title" /></h1>
        <h2>
            <s:text name="welcome.message" />: <s:property value="user"/>
        </h2>
        <div>
            <a href="<s:url action="login"/>"><s:text name="welcome.return" /></a>
        </div>
    </body>
</html>
```

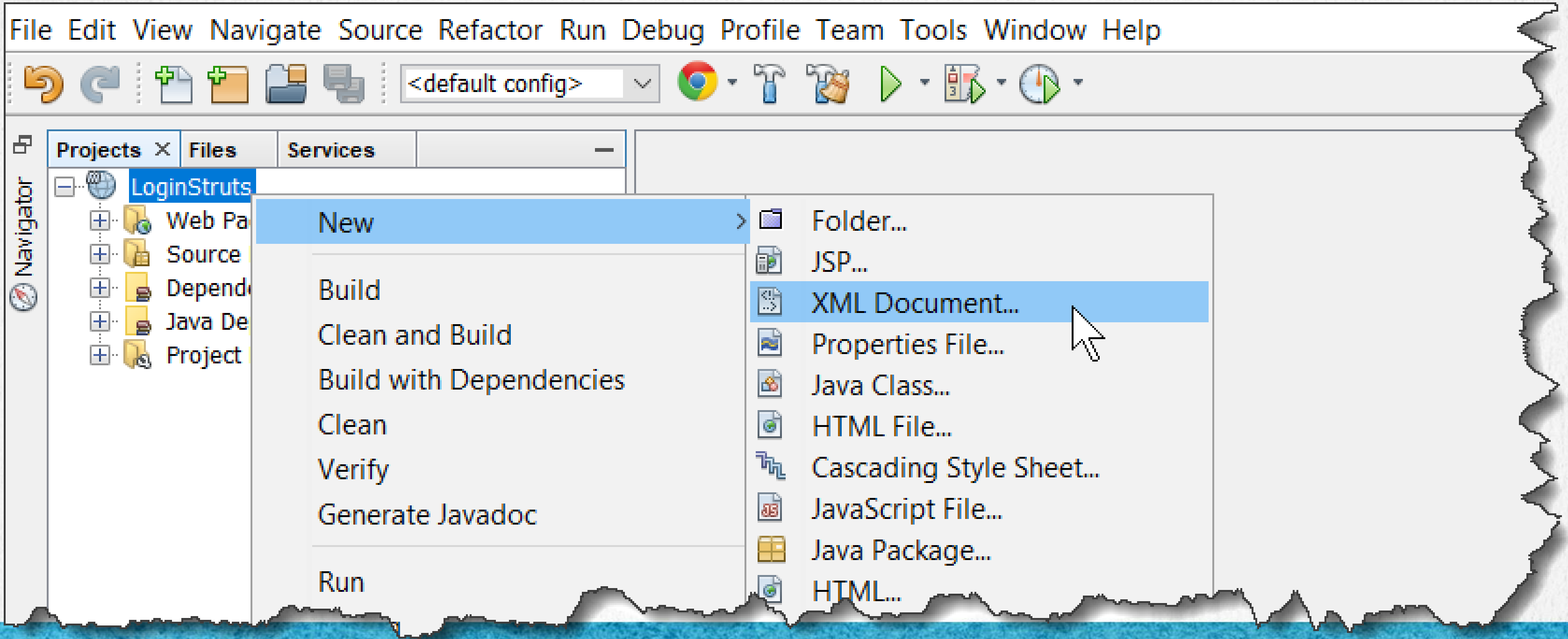
21. CREATE THE LOG4J2.XML FILE

We create a log4j2.xml file. The log4j API allows us to manage the log of a 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.

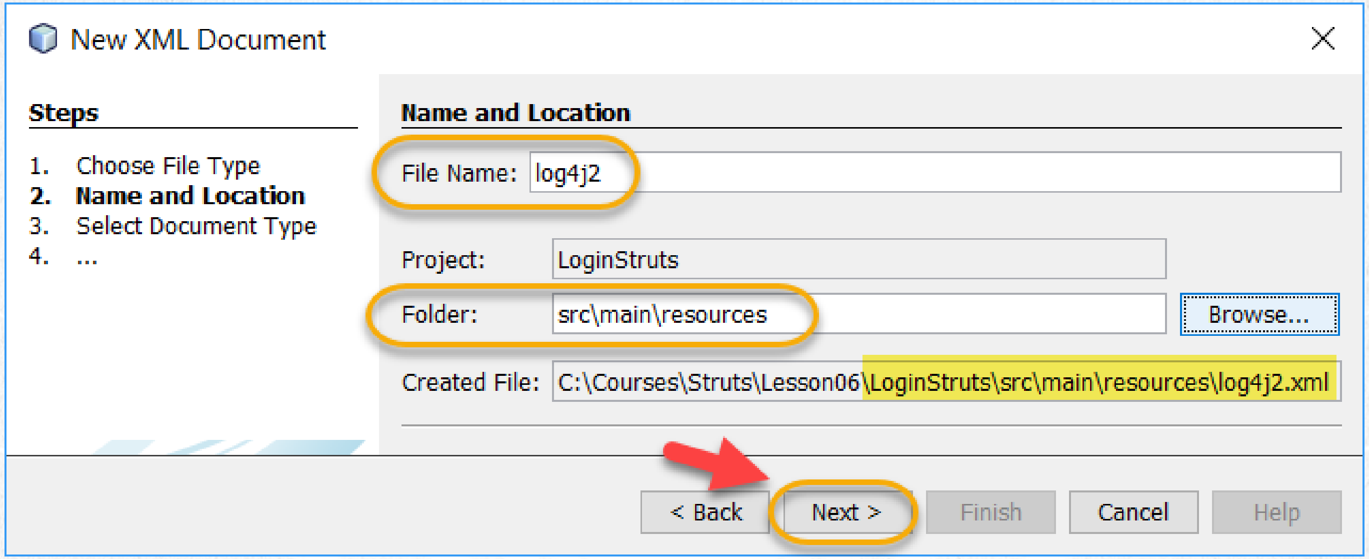
21. CREATE THE LOG4J2.XML FILE

- We create the log4j2.xml file as follows:



21. CREATE THE LOG4J2.XML FILE

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



New XML Document

Steps

1. Choose File Type
2. **Name and Location**
3. Select Document Type
4. ...

Name and Location

File Name: log4j2

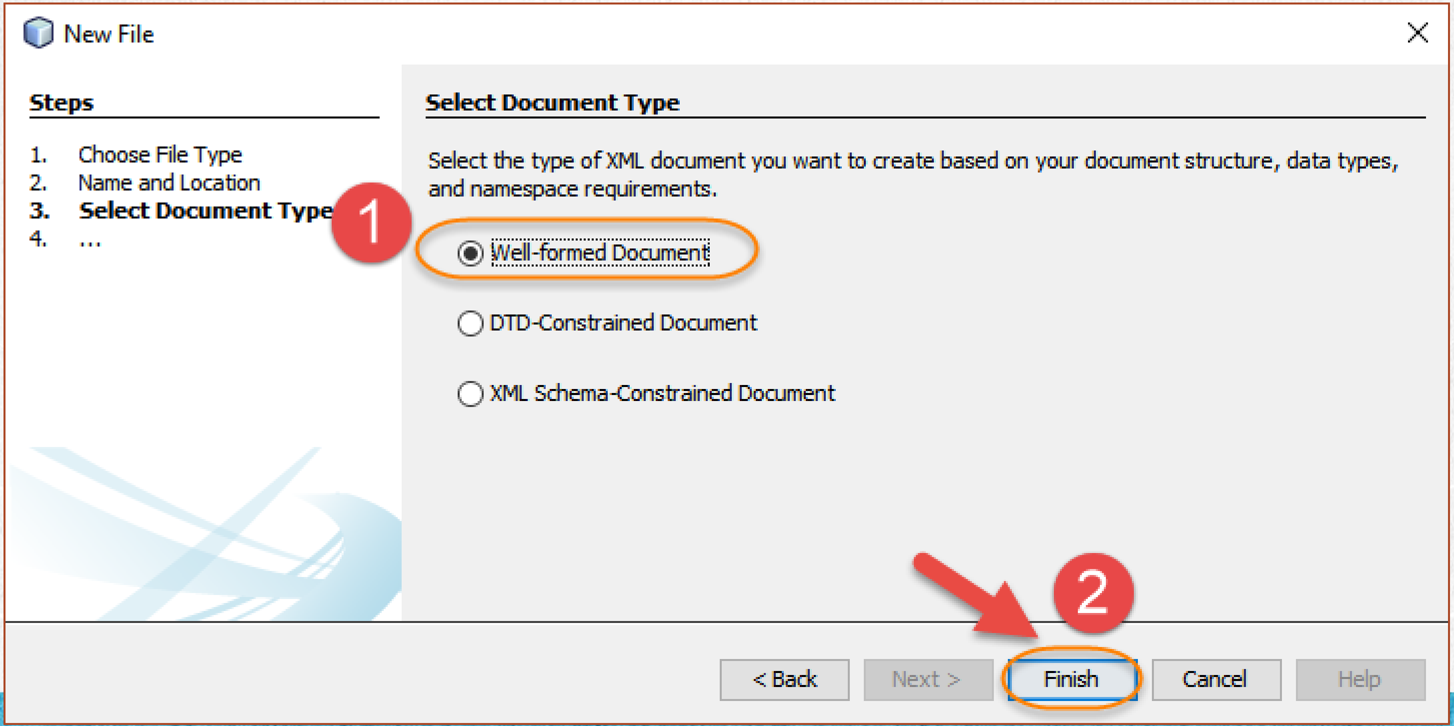
Project: LoginStruts

Folder: src\main\resources

Created File: C:\Courses\Struts\Lesson06\LoginStruts\src\main\resources\log4j2.xml

21. CREATE THE LOG4J2.XML FILE

- We select the option shown:



New File

Steps

1. Choose File Type
2. Name and Location
3. **Select Document Type**
4. ...

Select Document Type

Select the type of XML document you want to create based on your document structure, data types, and namespace requirements.

- [✓ Well-formed Document]
- DTD-Constrained Document
- XML Schema-Constrained Document

22. MODIFY THE CODE

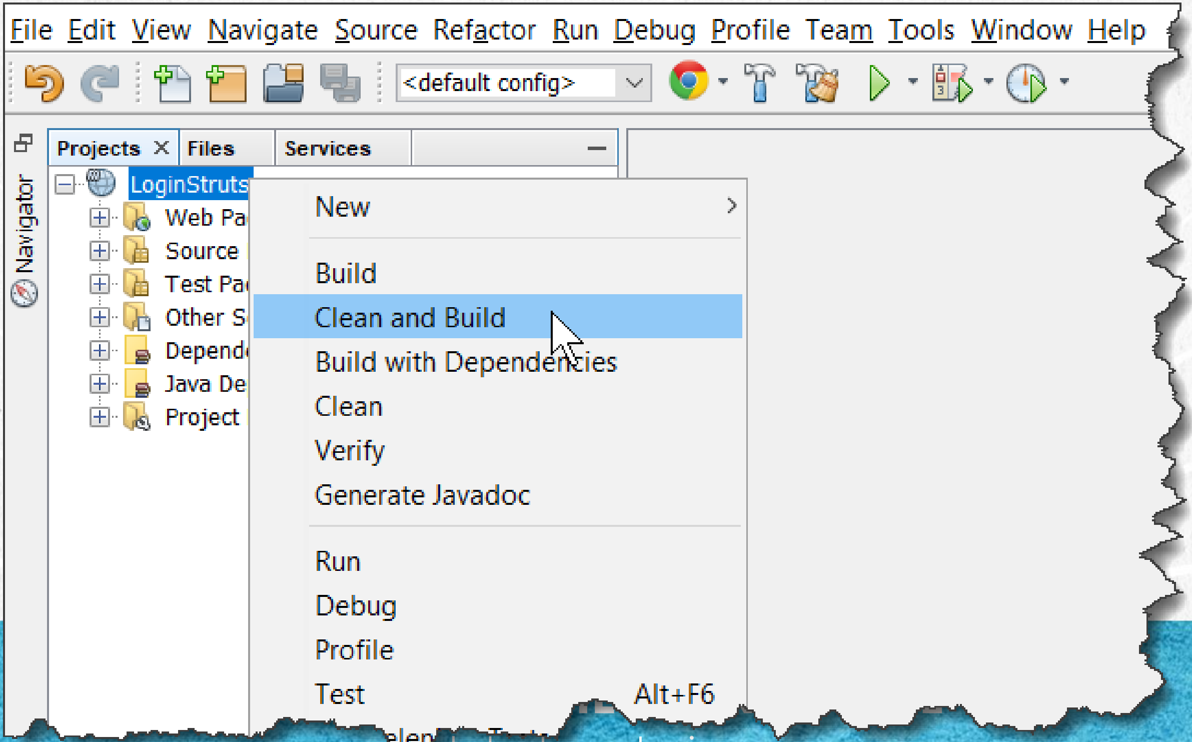
log4j2.xml:

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>
```

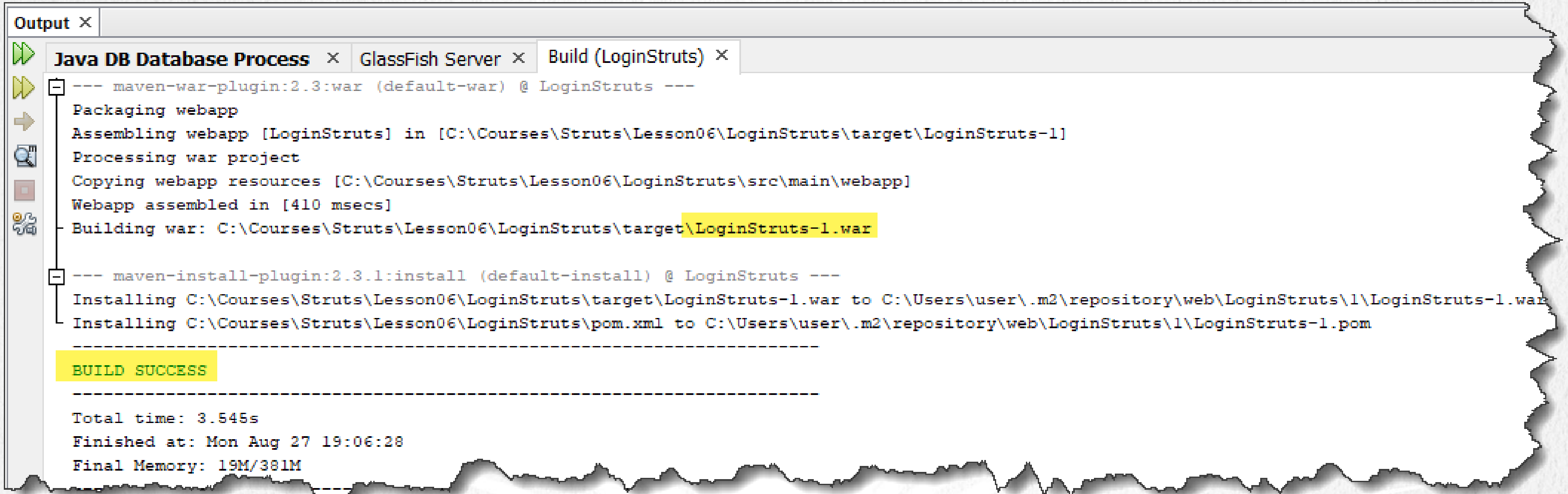
23. EXECUTE CLEAN & BUILD

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



23. EXECUTE CLEAN & BUILD

- We must observe a result similar to the following :

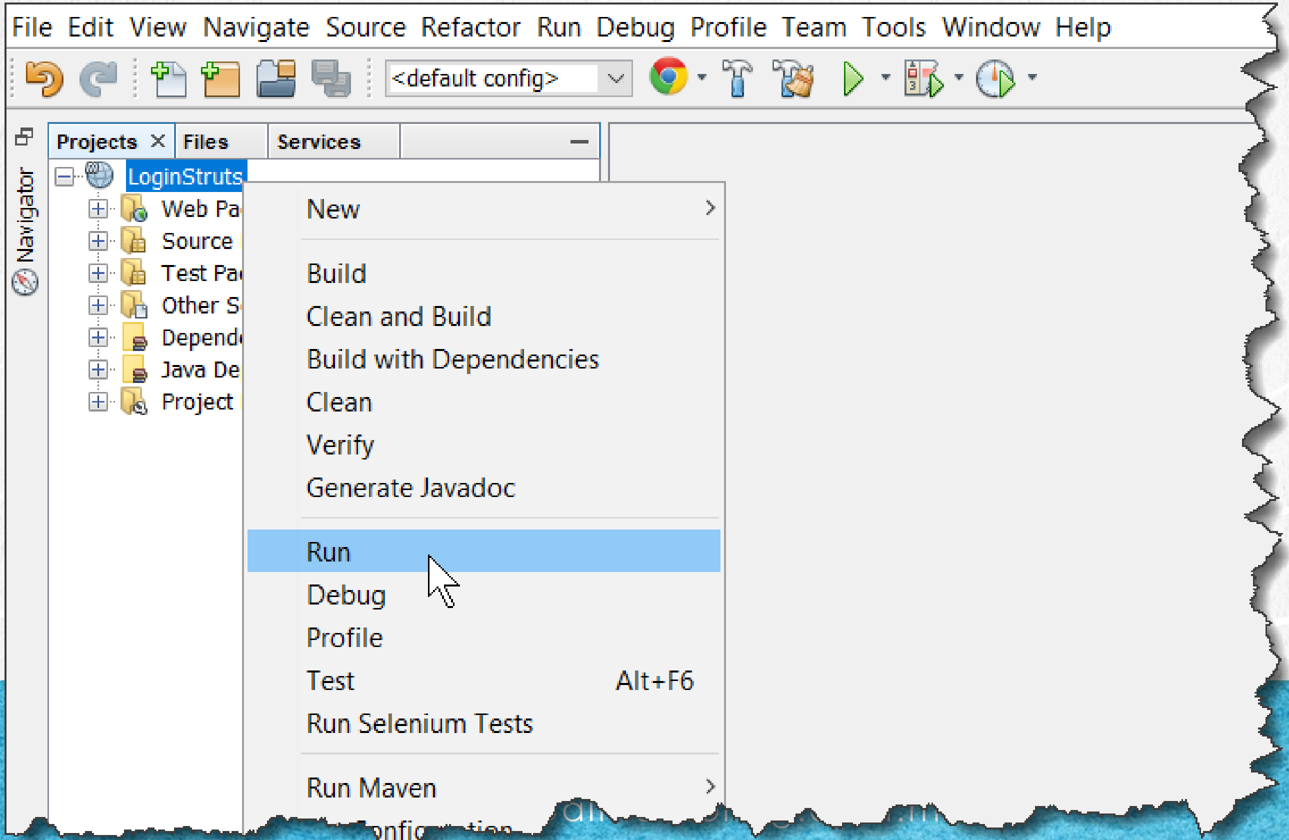


```
--- maven-war-plugin:2.3:war (default-war) @ LoginStruts ---
Packaging webapp
Assembling webapp [LoginStruts] in [C:\Courses\Struts\Lesson06\LoginStruts\target\LoginStruts-1]
Processing war project
Copying webapp resources [C:\Courses\Struts\Lesson06\LoginStruts\src\main\webapp]
Webapp assembled in [410 msecs]
Building war: C:\Courses\Struts\Lesson06\LoginStruts\target\LoginStruts-1.war

--- maven-install-plugin:2.3.1:install (default-install) @ LoginStruts ---
Installing C:\Courses\Struts\Lesson06\LoginStruts\target\LoginStruts-1.war to C:\Users\user\.m2\repository\web\LoginStruts\1\LoginStruts-1.war
Installing C:\Courses\Struts\Lesson06\LoginStruts\pom.xml to C:\Users\user\.m2\repository\web\LoginStruts\1\LoginStruts-1.pom
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 3.545s
Finished at: Mon Aug 27 19:06:28
Final Memory: 19M/381M
```

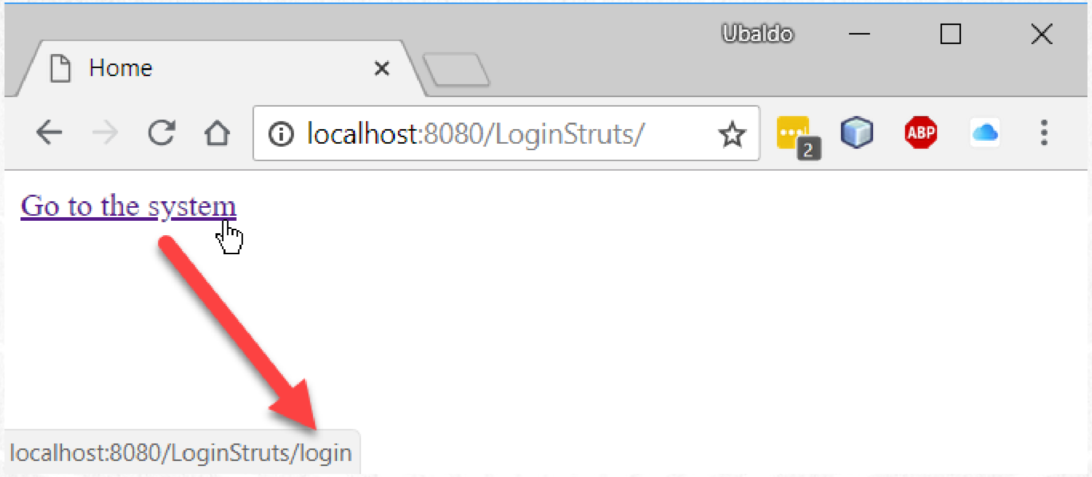
24. Execute the Application

- We execute the LoginStruts application as follows:



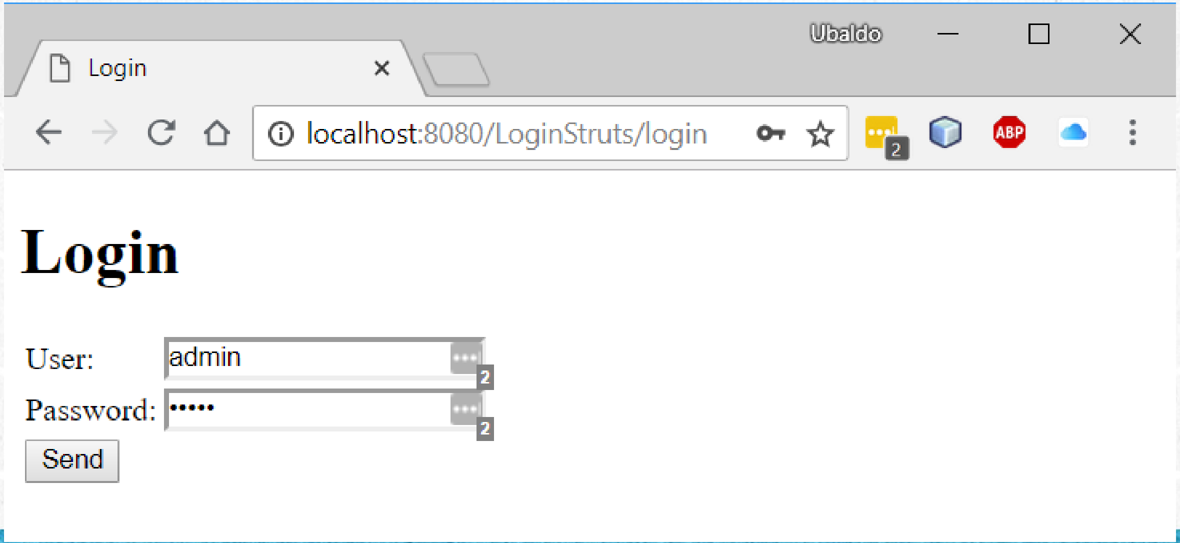
24. EXECUTE THE APPLICATION

- We run the application as follows:



24. EXECUTE THE APPLICATION

- We look at the form, and if we provide the value of "admin" in the user field it will direct us to the path of /validateUser and the result will be "success" showing us as a result the welcome.jsp view. The result is the same as in the previous exercise, the only difference was the use of annotations instead of the struts.xml file:



Login

localhost:8080/LoginStruts/login

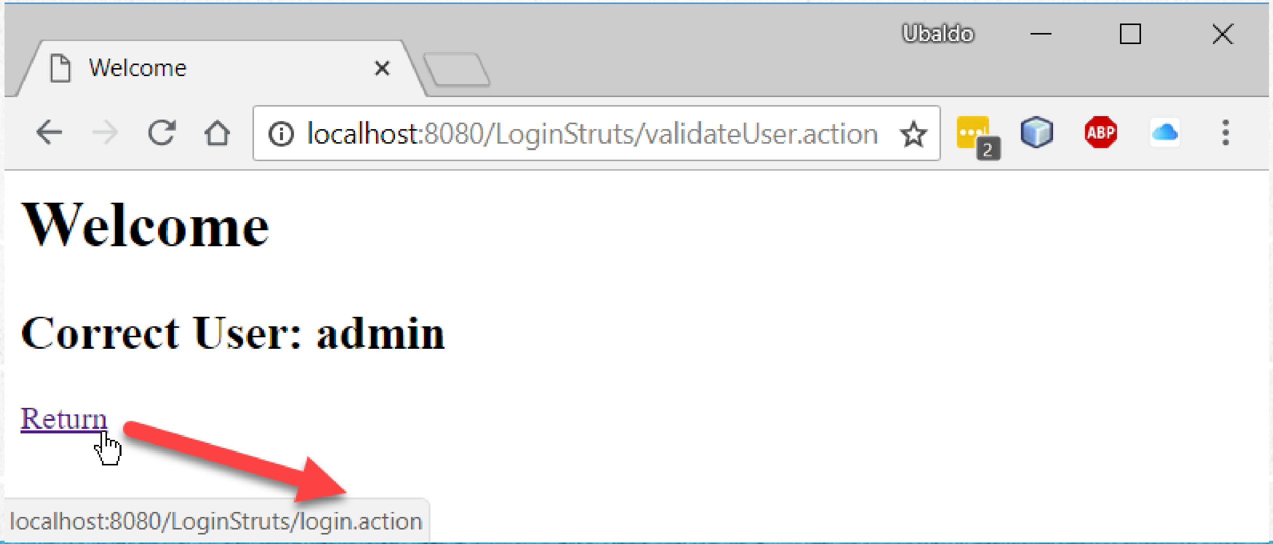
Login

User:	admin
Password:	•••••

Send

24. EXECUTE THE APPLICATION

- We observe the value provided by the user, and we see that the action that was executed is the path of: validateUser, showing us the view of welcome.jsp. We have also added a link to return to the login, but not directly to the JSP, but by calling the login path:



Welcome

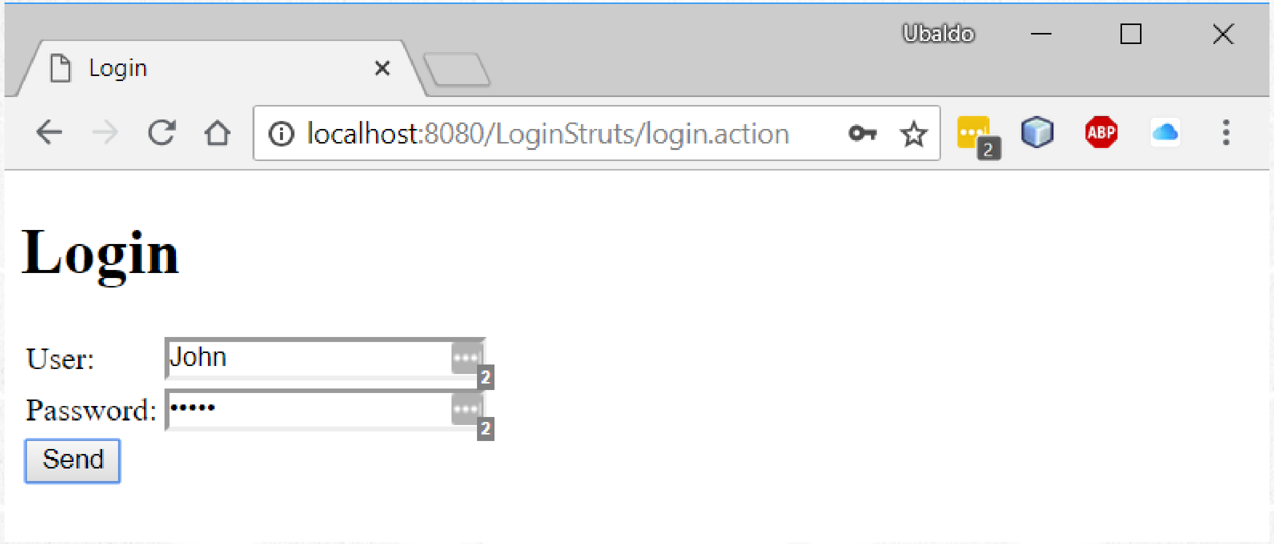
Correct User: admin

Return

localhost:8080/LoginStruts/login.action

24. EXECUTE THE APPLICATION

- We look at the form, and if we provide the value other than "admin" in the user field, it will direct us to the path of /validateUser and the result will be "input" showing us the result of login.jsp again:



Login

localhost:8080/LoginStruts/login.action

Login

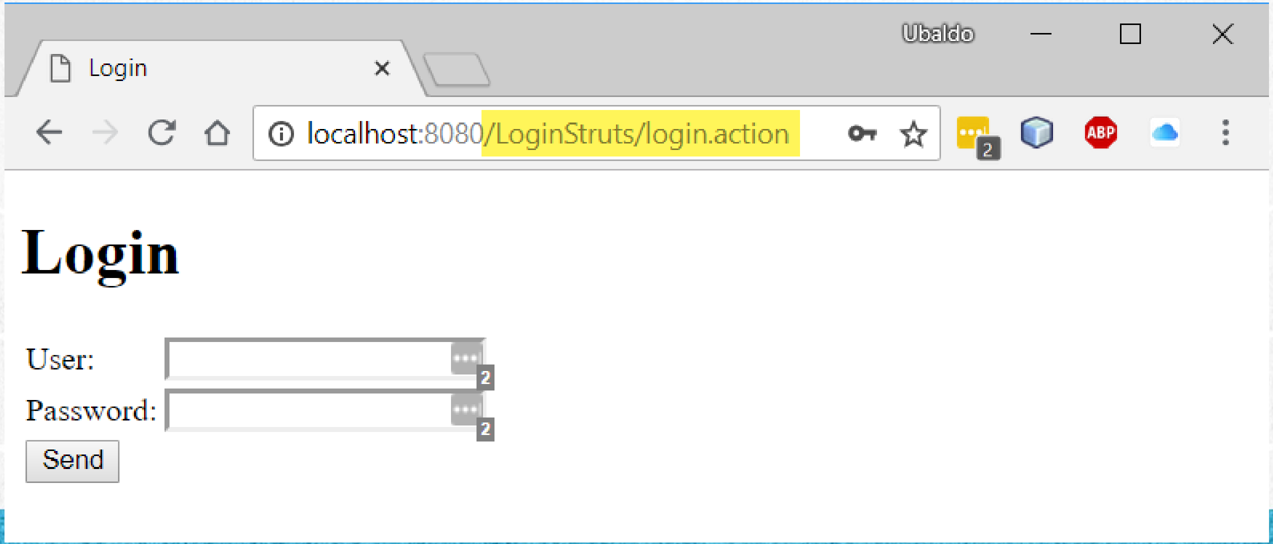
User: John

Password: •••••

Send

24. EXECUTE THE APPLICATION

•We see that the action that was executed is the path of: validateUser, and because the user value was not equal to "admin", then it returned as a result the "input" string, showing us again the login.jsp view. To do this, we configured a redirect in the struts.xml file to the "login" path, and that again showed the JSP of login.jsp, and modifying the URL to show the path of login.action, instead of validateUser. action which is the path that would have been shown if we had not specified the redirect to the "login" action:



Login

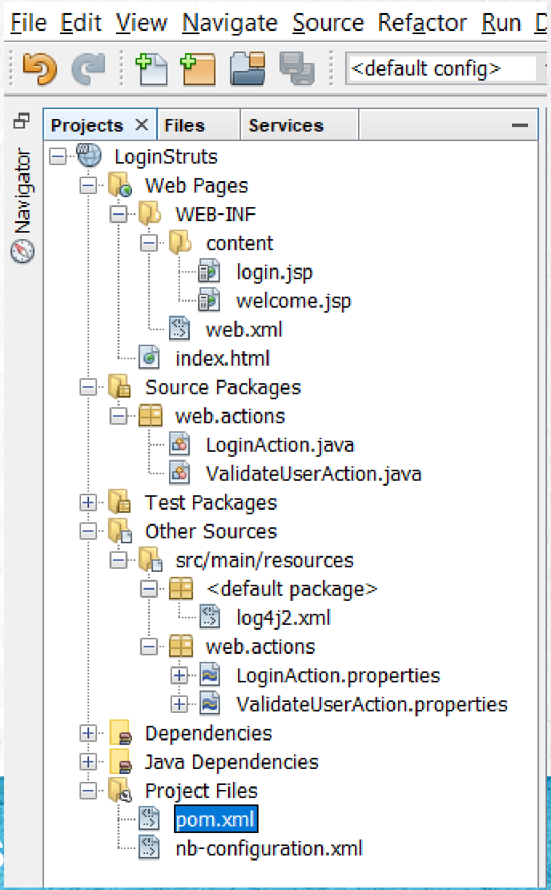
User:

Password:

Send

FINAL STRUCTURE OF THE PROJECT

At the end of the exercise the structure should be as follows:



```
LoginStruts
  Web Pages
    WEB-INF
      content
        login.jsp
        welcome.jsp
      web.xml
    index.html
  Source Packages
    web.actions
      LoginAction.java
      ValidateUserAction.java
  Test Packages
  Other Sources
    src/main/resources
      <default package>
        log4j2.xml
      web.actions
        LoginAction.properties
        ValidateUserAction.properties
  Dependencies
  Java Dependencies
  Project Files
    pom.xml
    nb-configuration.xml
```

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 management of Annotations in Struts 2.

There are several types of annotations that we can use in our Action classes in Struts 2, however we have used two of the most important: @Action and @Result.

With this exercise we are ready to add more features, such as validation issues, error handling, among several other concepts that we will see later.

ONLINE COURSE

STRUTS 2 FRAMEWORK

By: Eng. Ubaldo Acosta

Global Mentoring

JAVA
ONLINE COURSES
UNIVERSITY

STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx