TAGS WITH STRUTS 2 FRAMEWORK

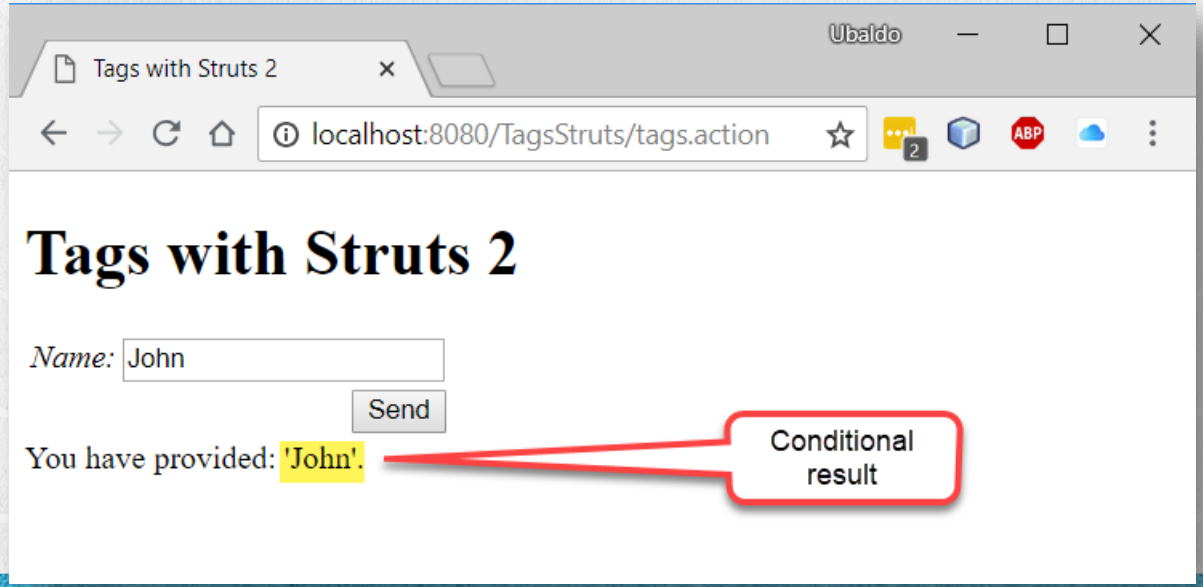
By the expert: Ubaldo Acosta

Global Mentoring

JAVA ONLINE COURSES UNIVERSITY

EXERCISE OBJECTIVE

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



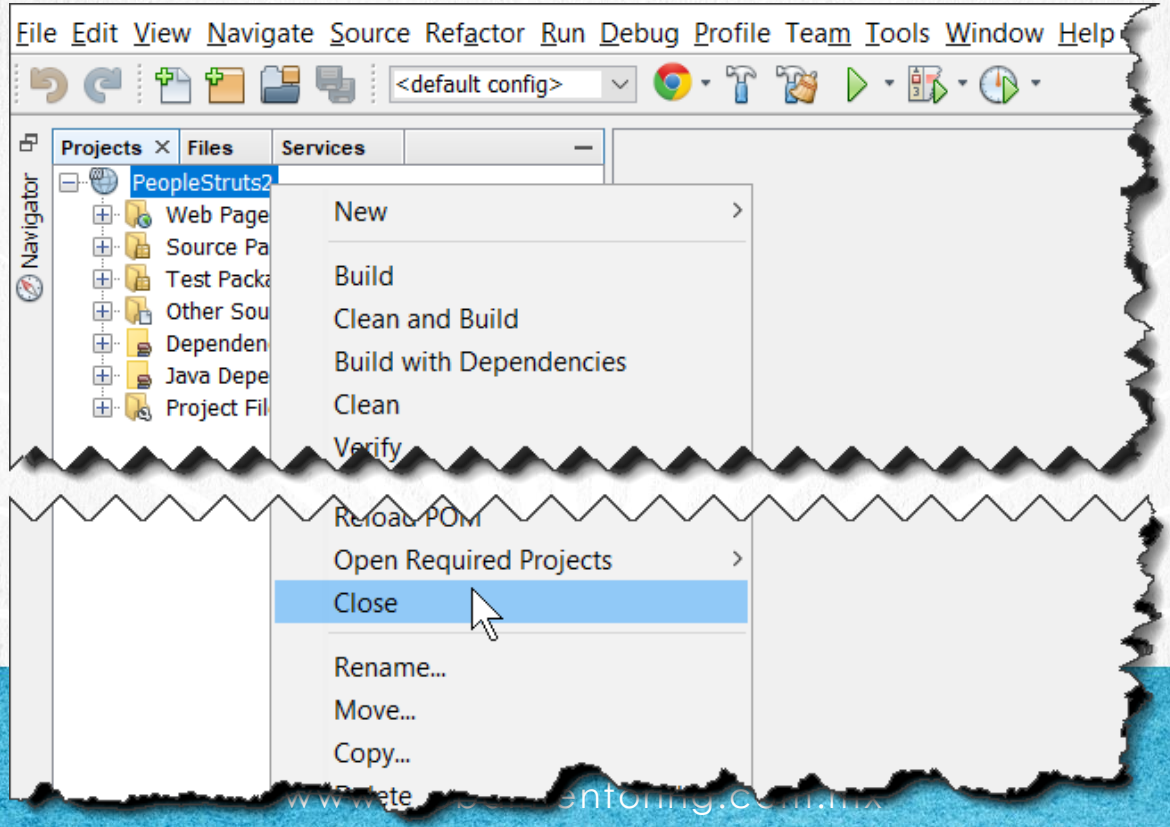
EXERCISE REQUIREMENT

In this project we are going to put into practice the concept of Struts Tags. In particular the conditional tags.

Although we have used several tags before, we are going to add some more so you have more knowledge of this topic in Struts 2.

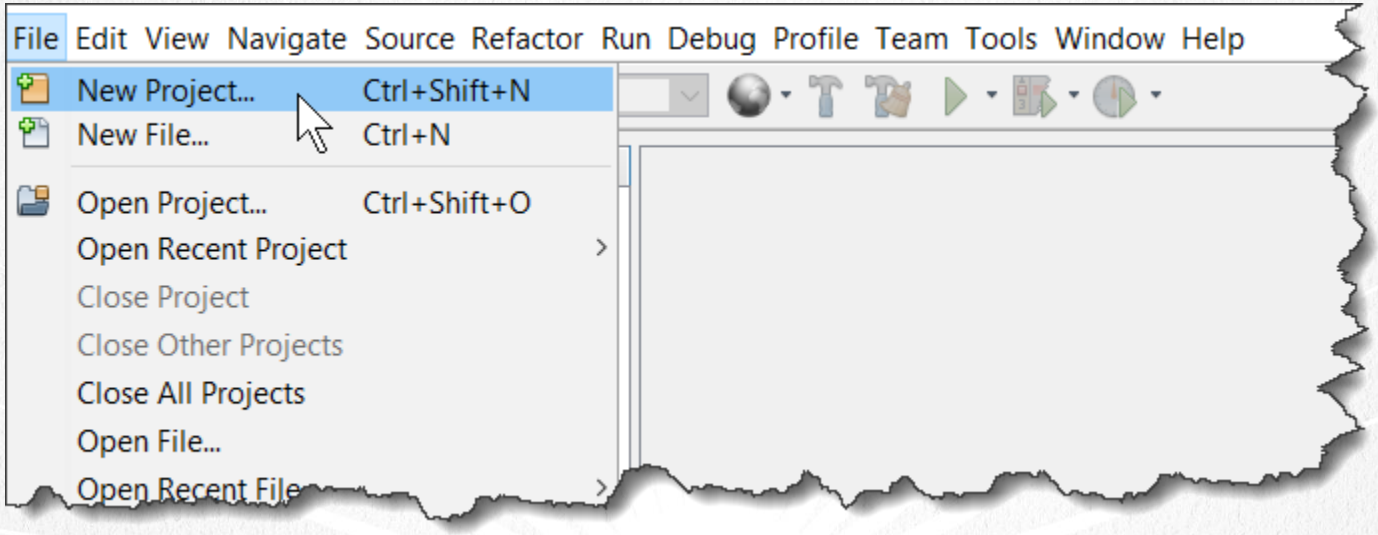
CLOSE PROJECTS THAT WE NO LONGER USE

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



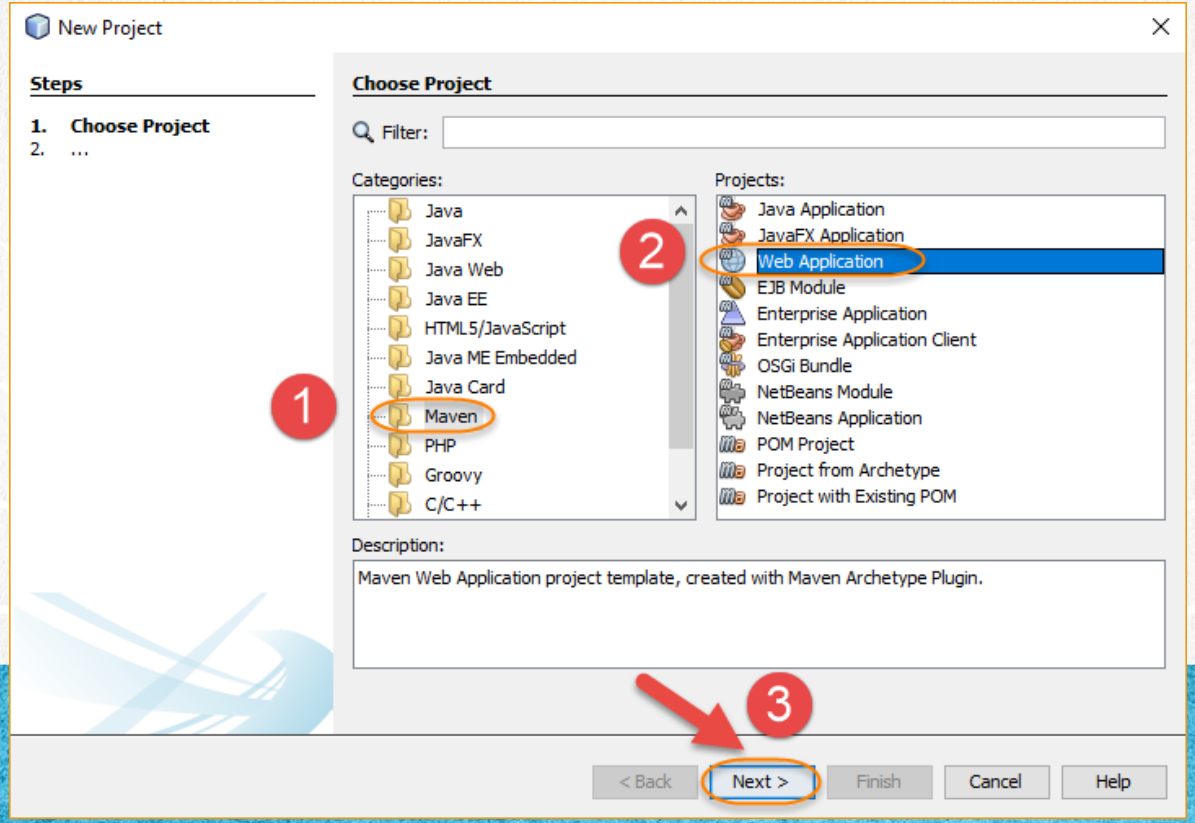
1. CREATE A NEW PROJECT

- We create the new project as shown below:



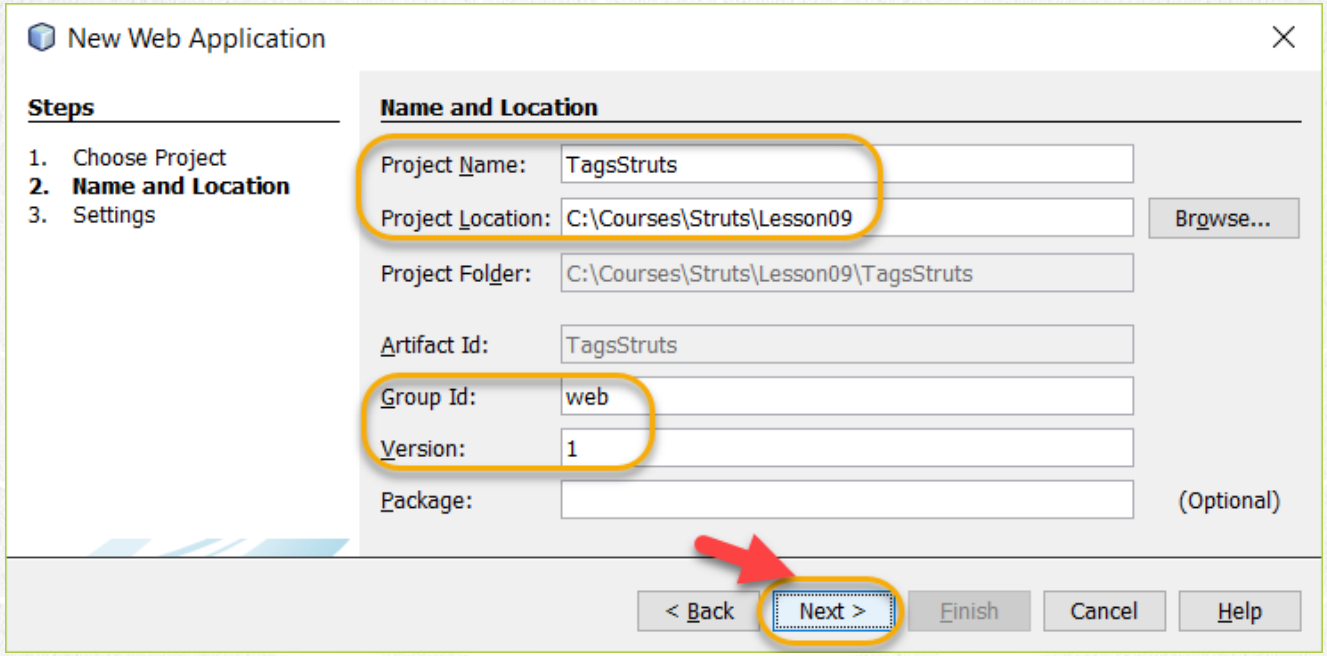
1. CREATE A NEW PROJECT

- We create the new project as shown below:



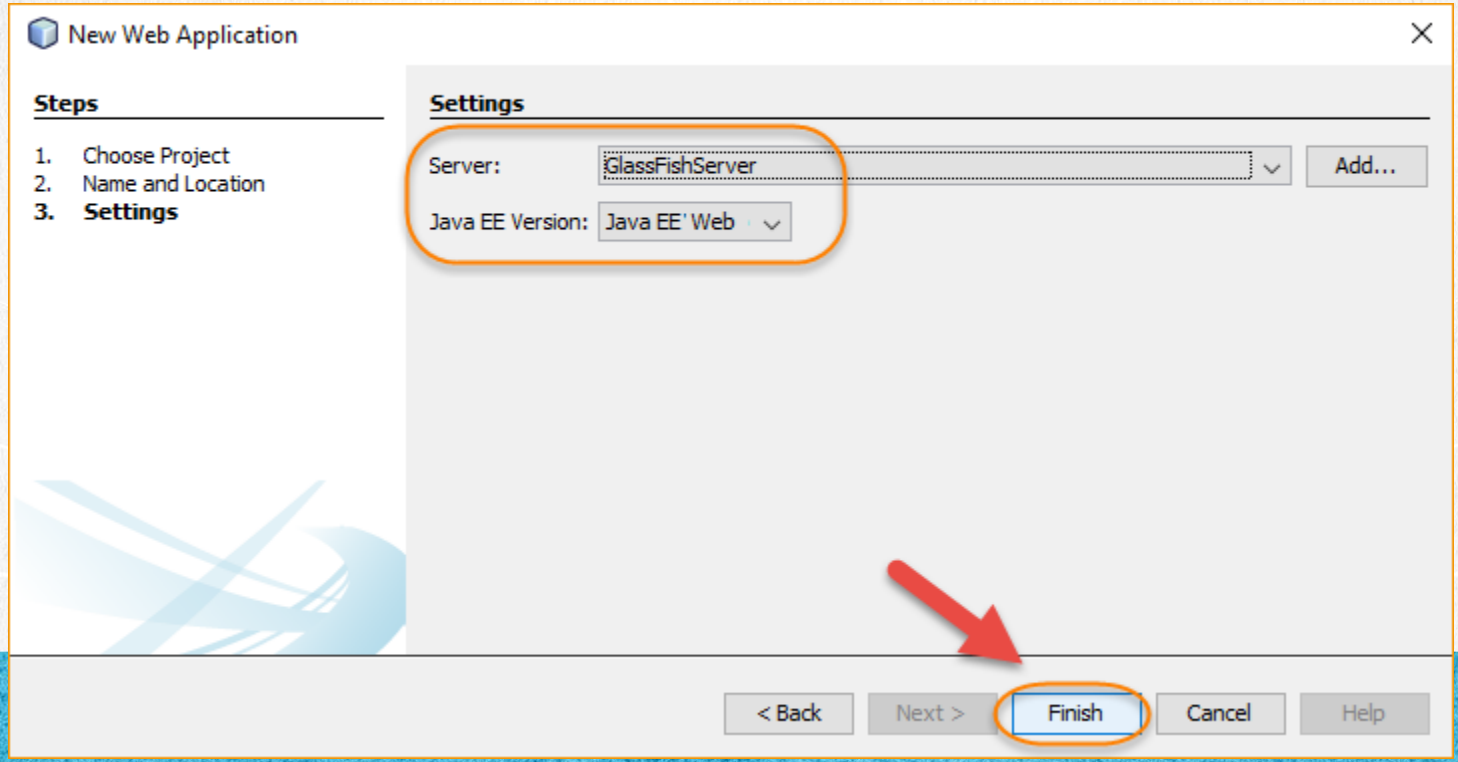
1. CREATE A NEW PROJECT

- We create the new project as shown below: :



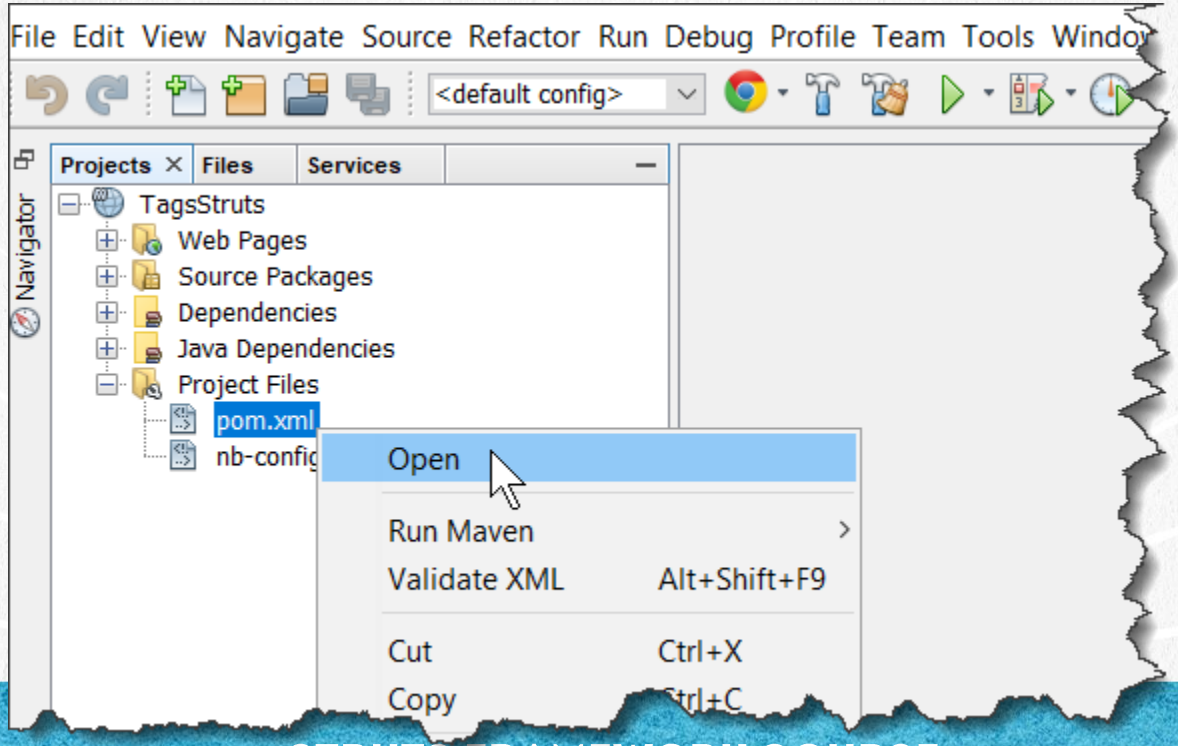
1. CREATE A NEW PROJECT

- We select the values shown:



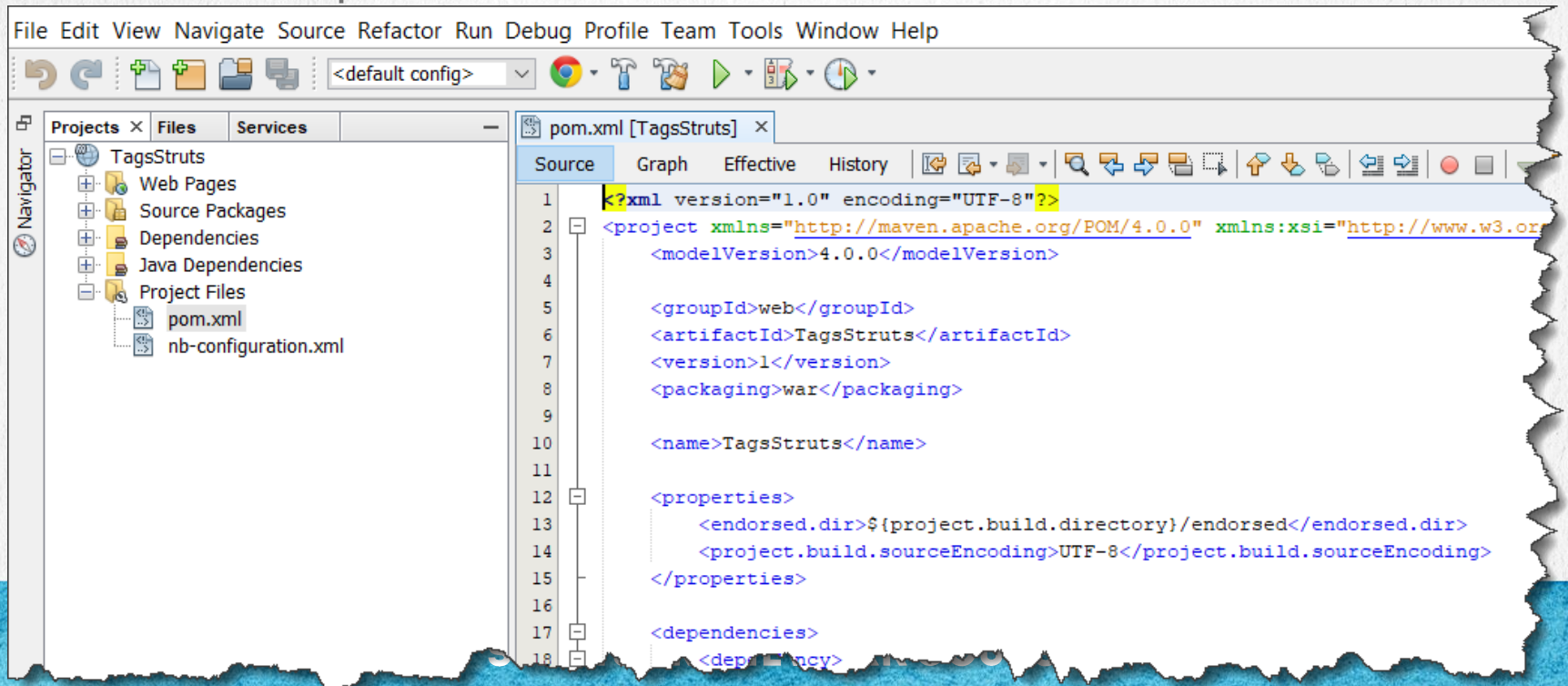
2. OPEN MAVEN'S POM.XML FILE

- 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:



3. MODIFY THE CODE

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>TagsStruts</artifactId>
    <version>1</version>
    <packaging>war</packaging>

    <name>TagsStruts</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 CODE

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 CODE

pom.xml:

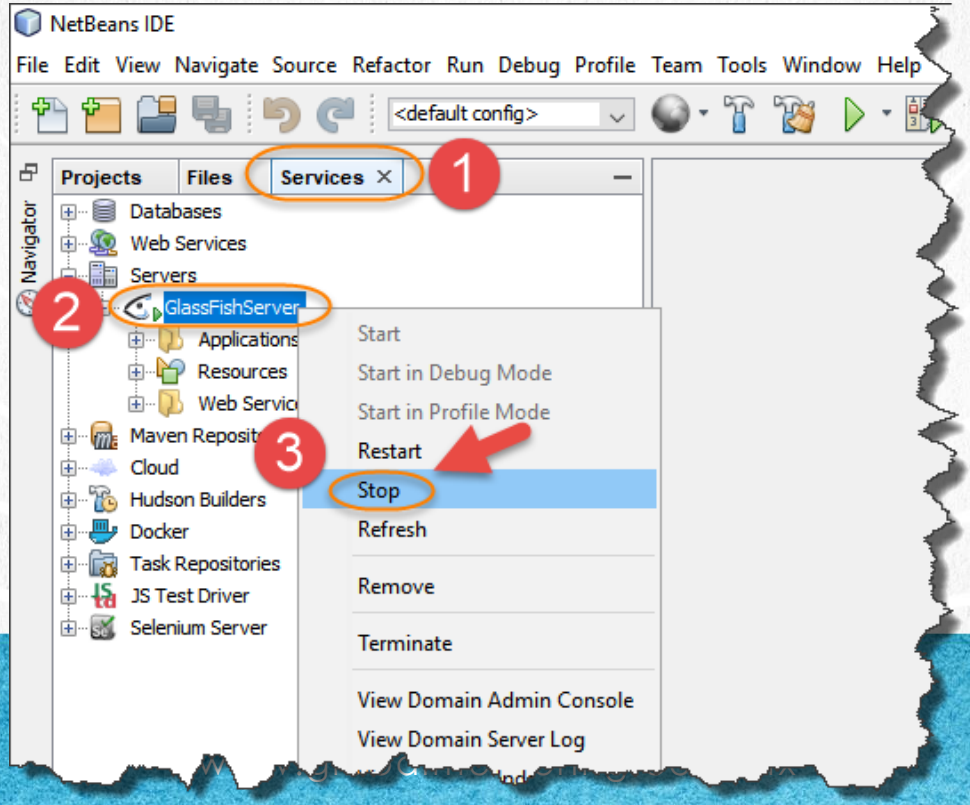
Click to download

```
    <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>
```

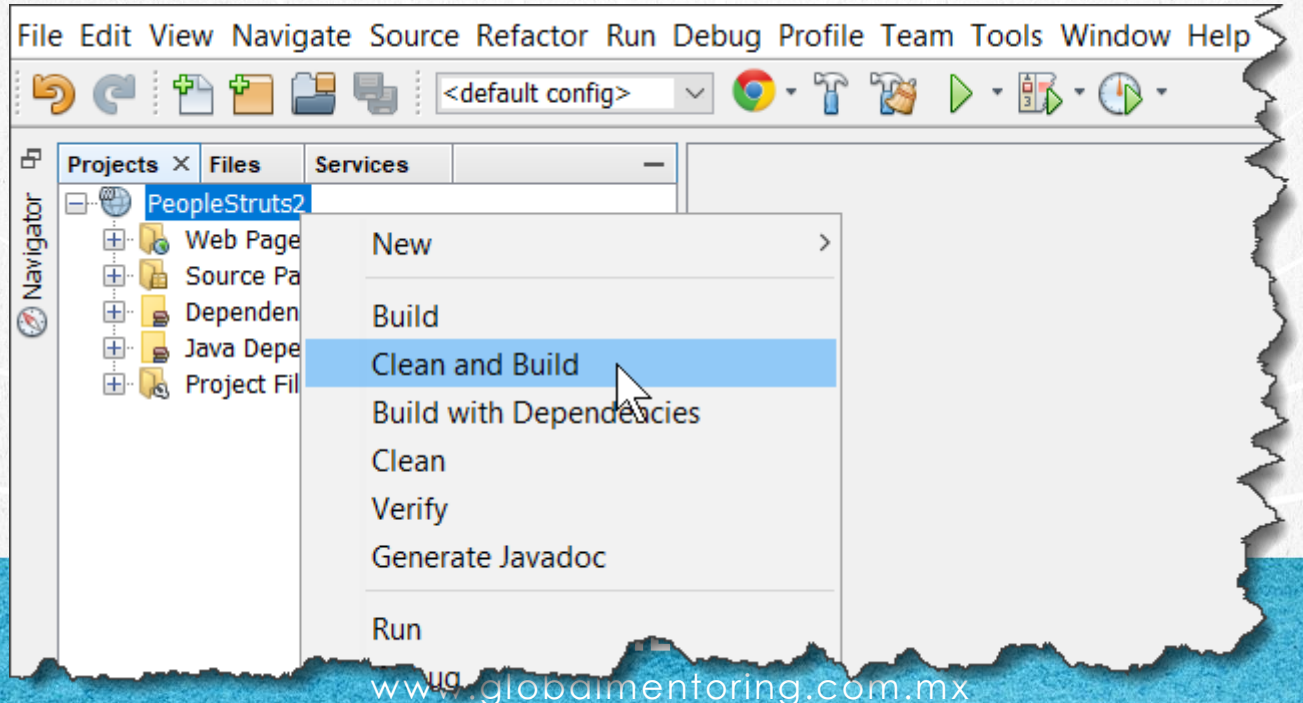
4. STOP GLASSFISH IF IT WAS STARTED

•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 for verification:



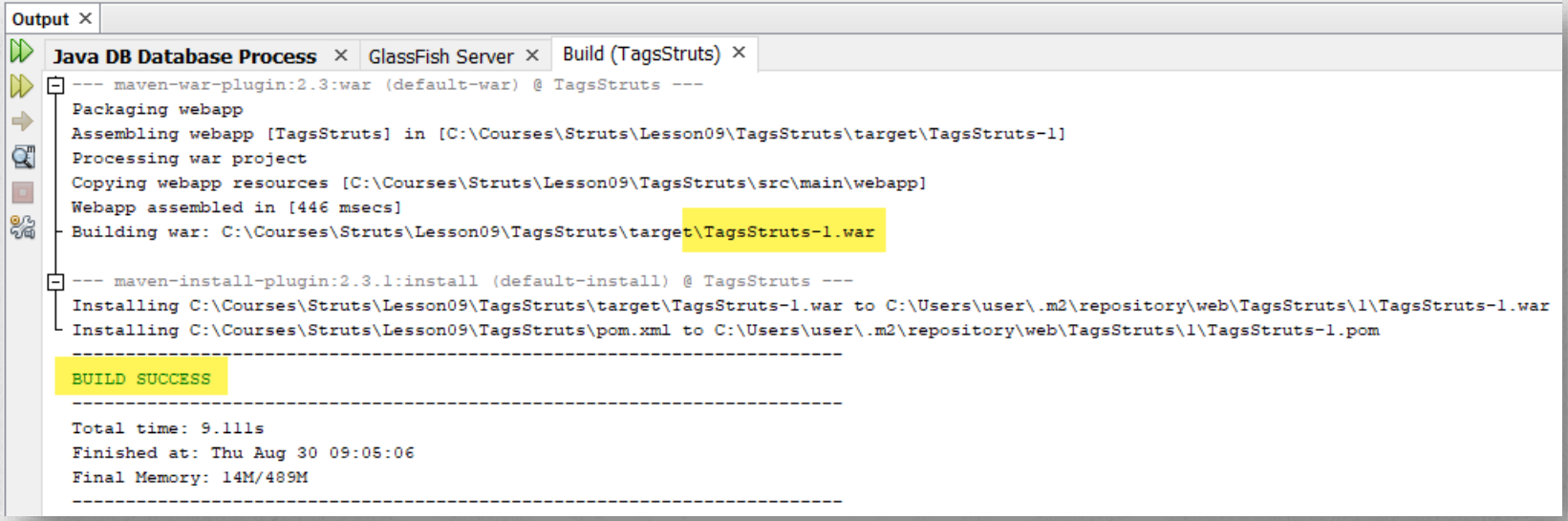
5. EXECUTE CLEAN & BUILD

- 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:



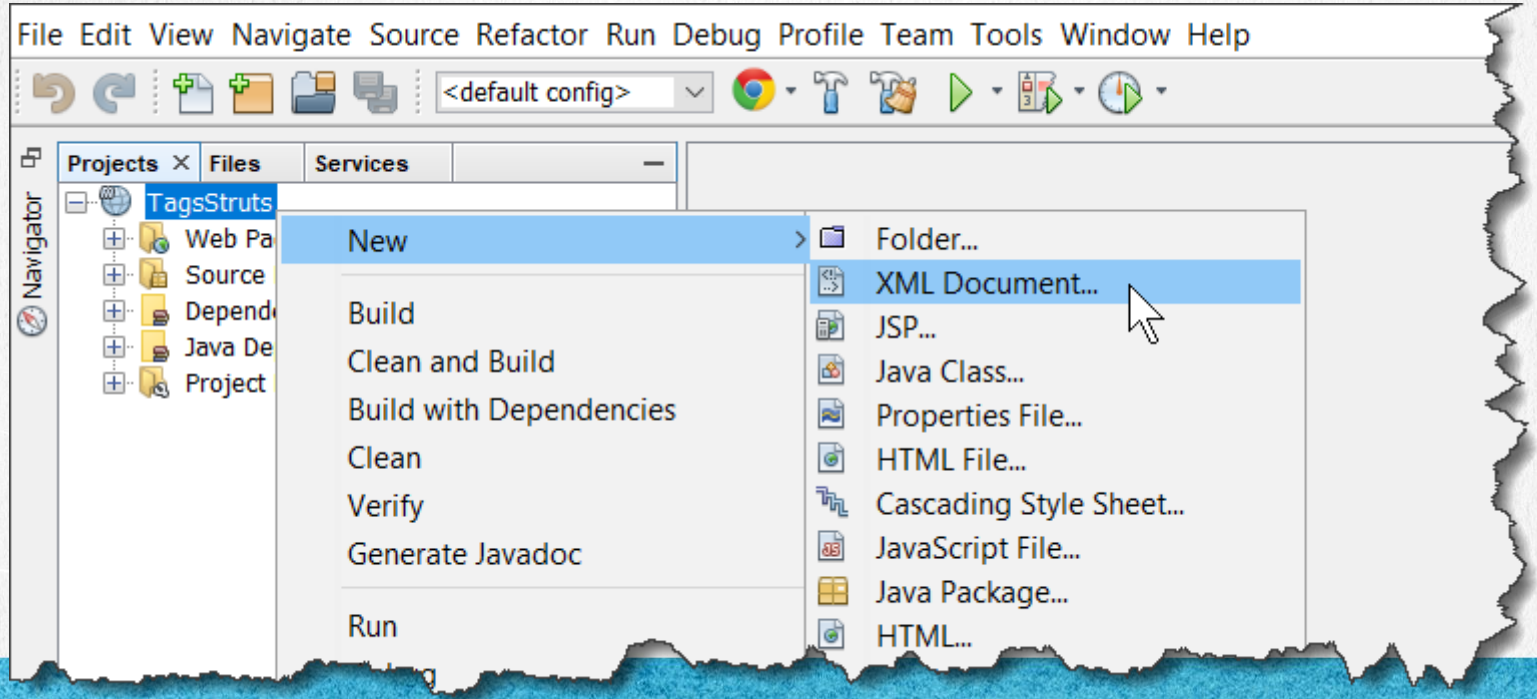
6. CREATE AN XML FILE

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.

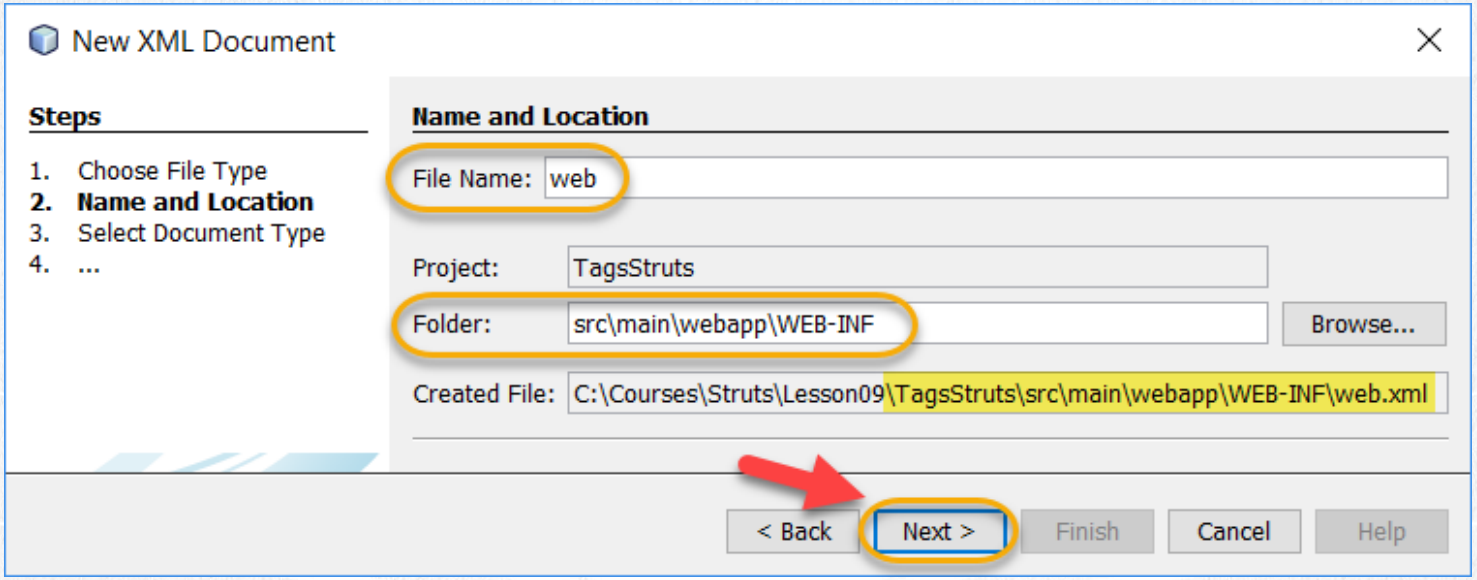
6. CREATE AN XML FILE

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



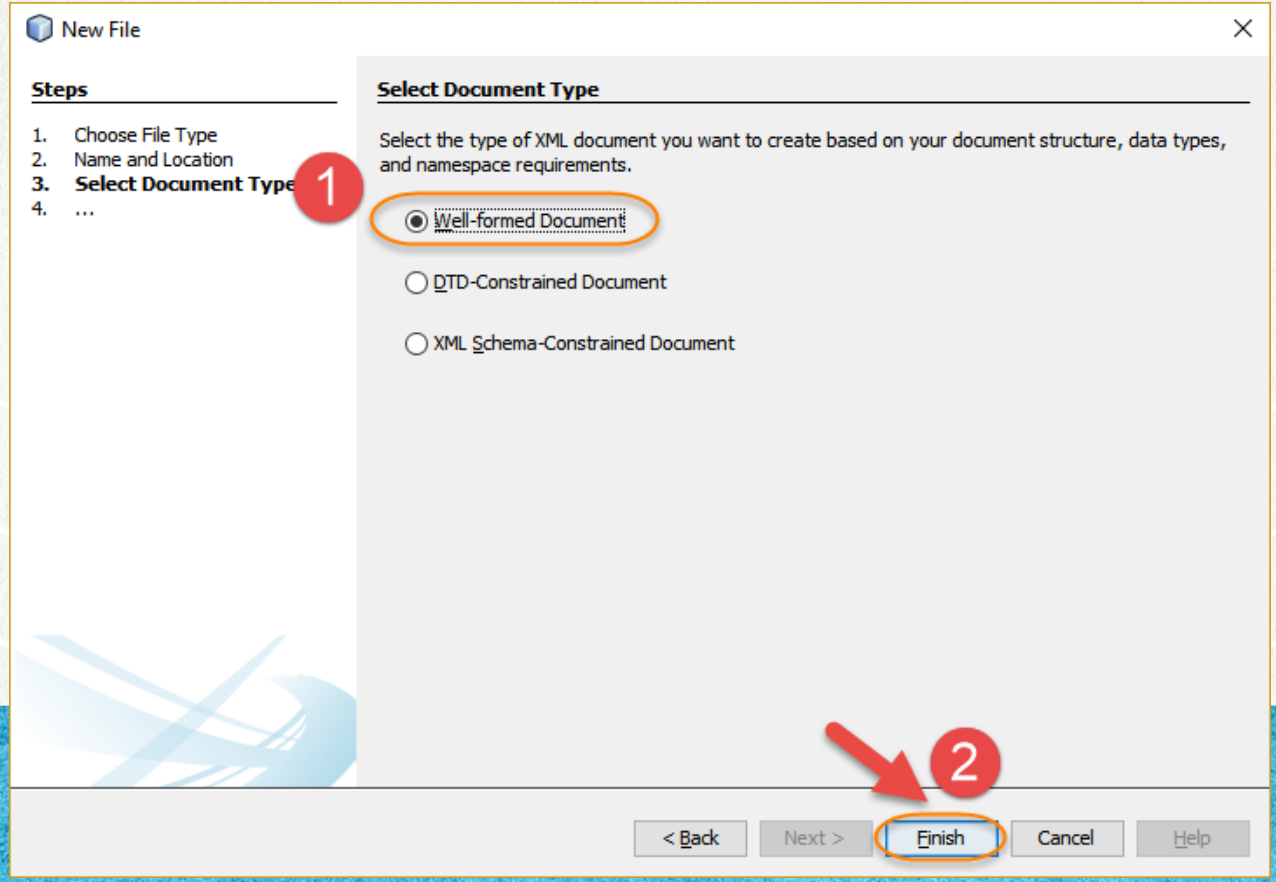
6. CREATE AN XML FILE

- 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:



6. CREATE AN XML FILE

- We select the indicated type and click on finish.



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

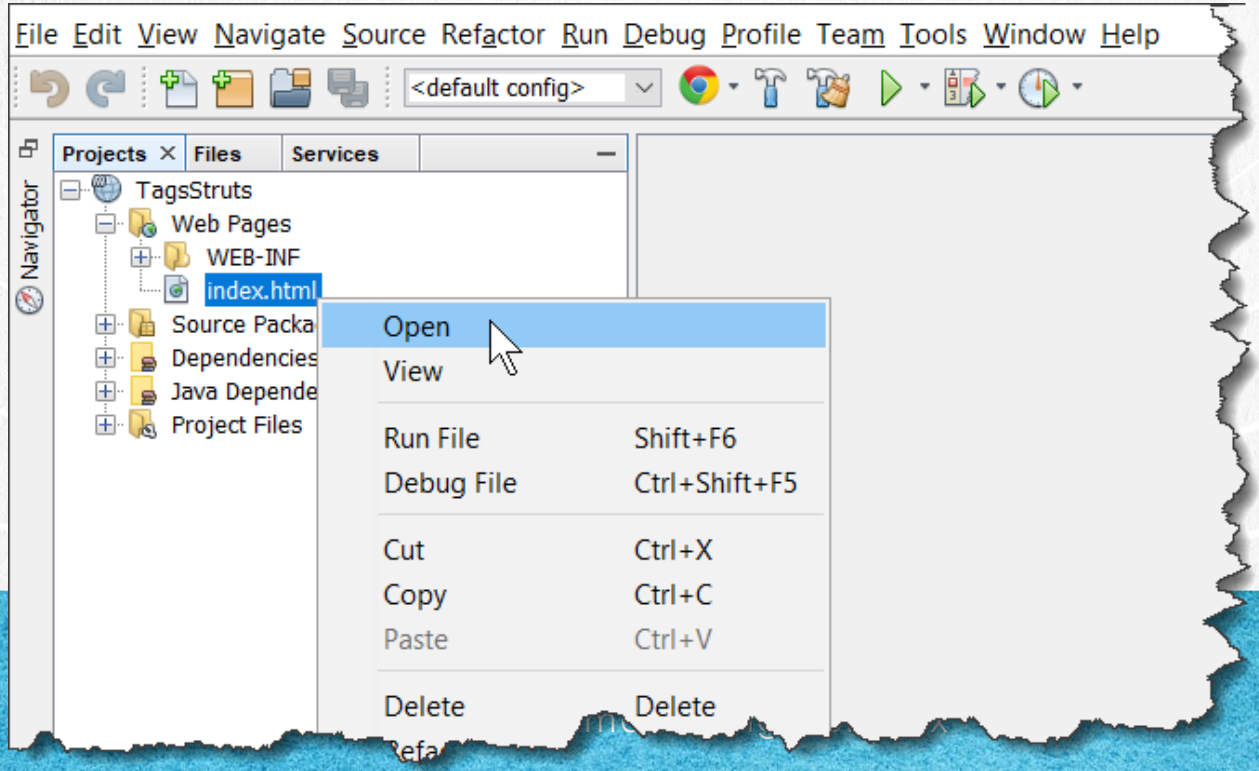
In automatic the IDE adds a file called index.html. However, if this file is not created we must add it to the project at the root level of Web Pages.

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: tags

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="tags">Go to the system</a>
    </body>
</html>
```

9. CREATE A NEW CLASS

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

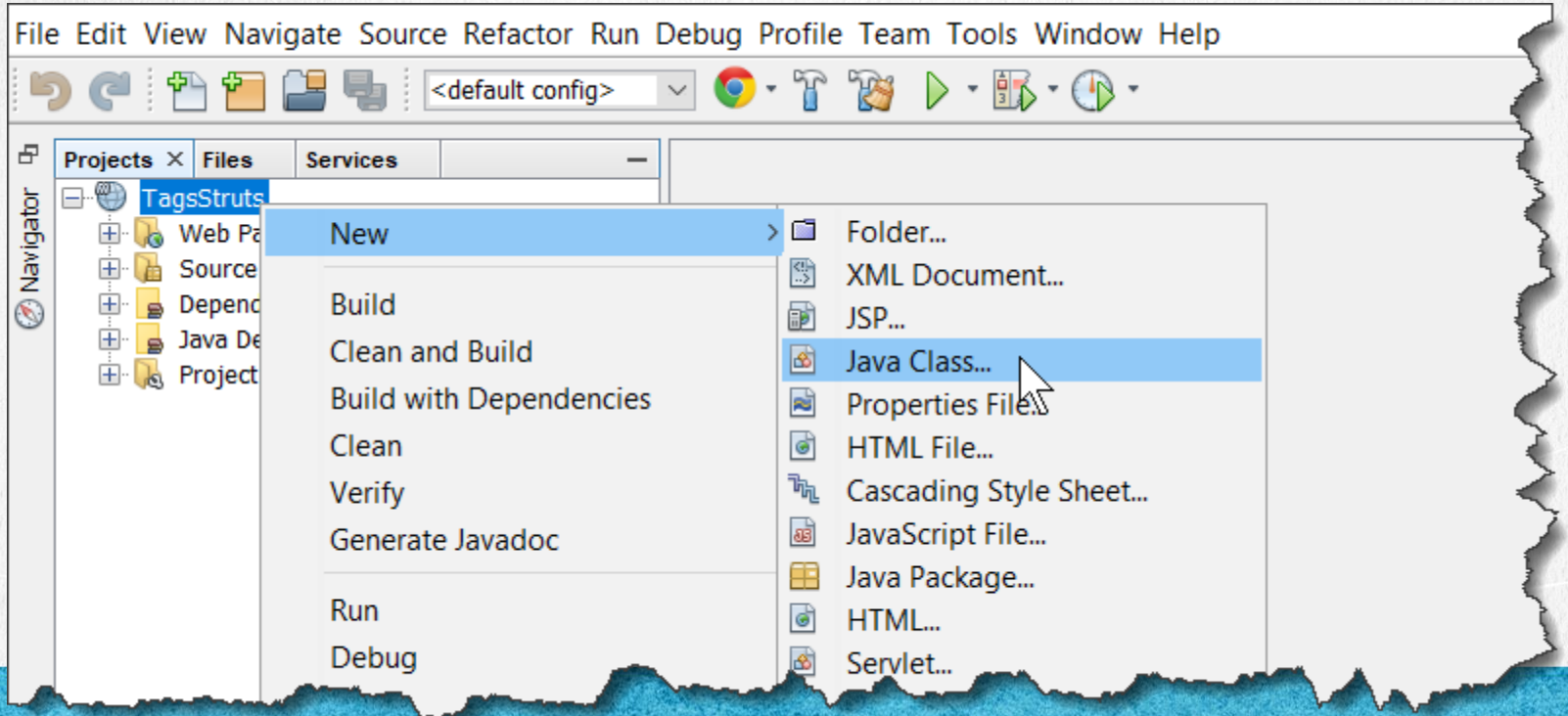
We will extend the ActionSupport class and overwrite the execute method.

We are going to apply the topic of conventions by name of Struts to configure our class of type Action and the associated JSP view.

Let's see how our class is.

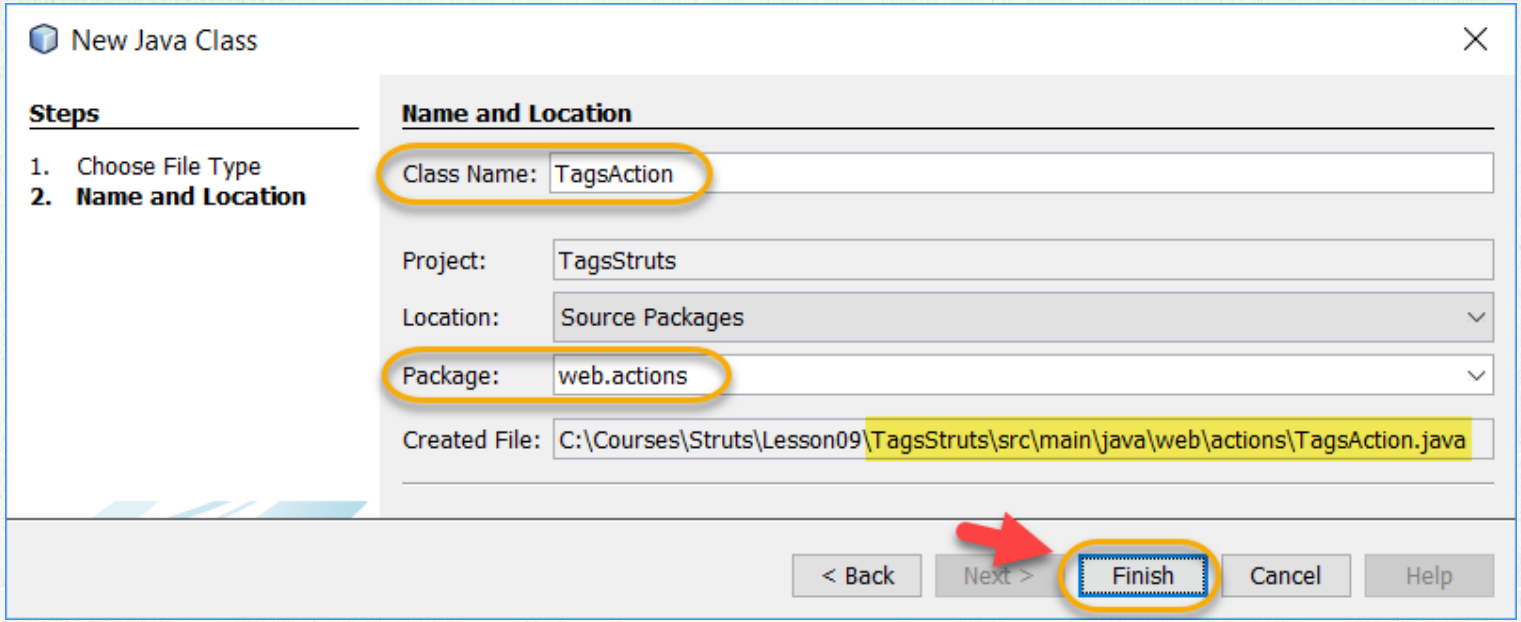
9. CREATE A NEW JAVA CLASS

- Create the TagsActions.java class:



9. CREATE A NEW JAVA CLASS

- Create the TagsActions.java class:



10. MODIFY THE CODE

TagsAction.java:

Click to download

```
package web.actions;

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

public class TagsAction extends ActionSupport {

    Logger log = LogManager.getLogger(TagsAction.class);

    private String name;

    @Override
    public String execute() {
        log.info("Name value:" + name);
        return SUCCESS;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

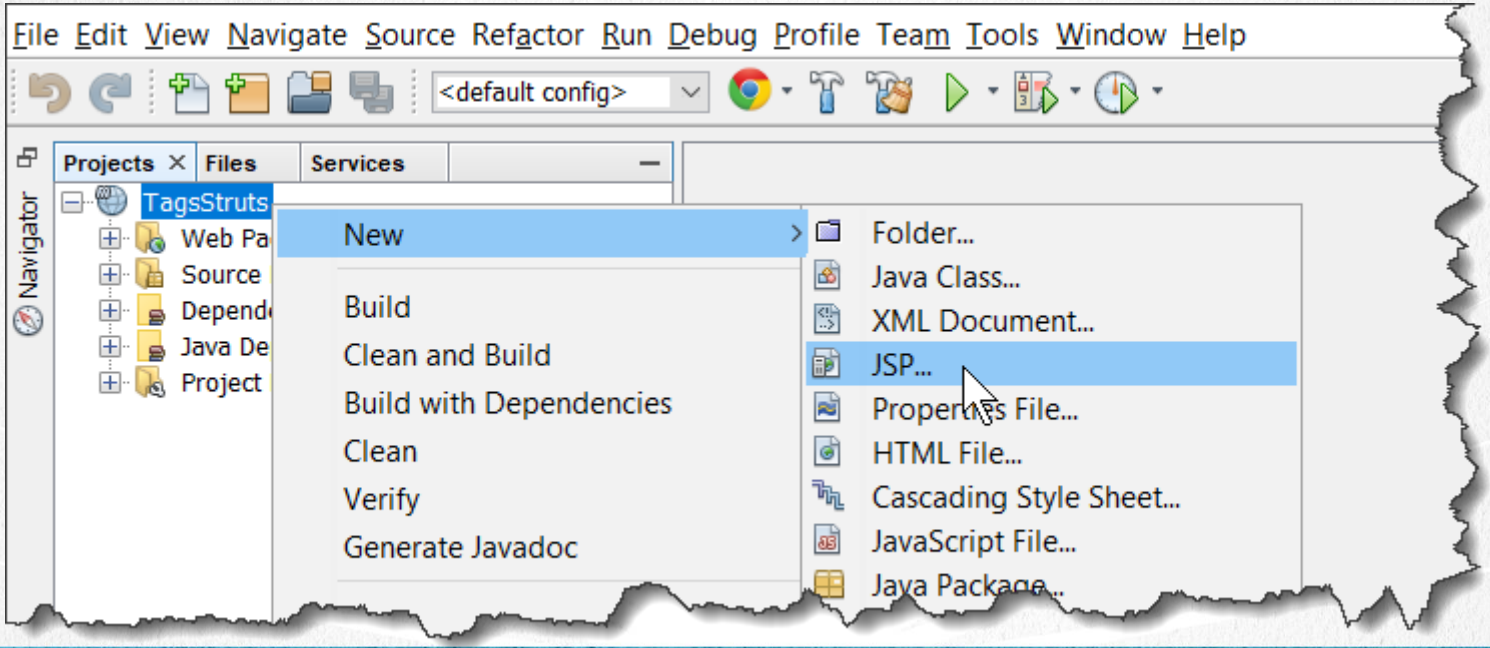
11. CREATE A NEW JSP FILE

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

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

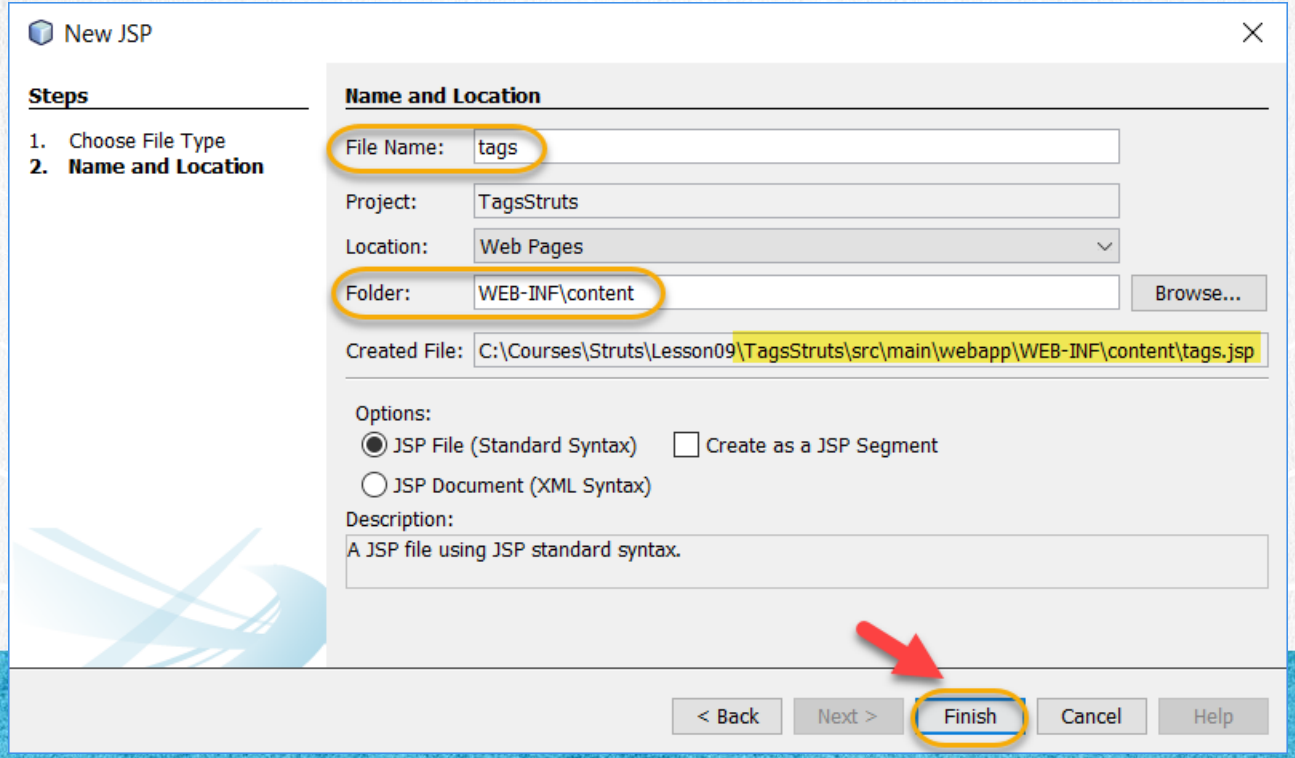
11. CREATE A NEW JSP FILE

- Create the tags.jsp file:



11. CREATE A NEW JSP FILE

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



12. MODIFY THE CODE

tags.jsp:

Click to download

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib  prefix="s" uri="/struts-tags" %>
<!DOCTYPE html>
<html>
    <head>
        <title>Tags with Struts 2</title>
        <s:head />
    </head>
    <body>
        <h1>Tags with Struts 2</h1>
        <s:form>
            <s:textfield label="Name" name="name" />
            <s:submit value="Send"/>
        </s:form>
```

12. MODIFY THE CODE

tags.jsp:

Click to download

```
        <s:if test="name!=null">
            <s:if test = "name == 'John'">
                You have provided: 'John'.
            </s:if>
            <s:elseif test = "name == 'Katty'">
                You have provided: 'Katty'
            </s:elseif>
            <s:else>
                You have provided a different name: <s:property value="name" />
            </s:else>
        </s:if>
    </body>
</html>
```

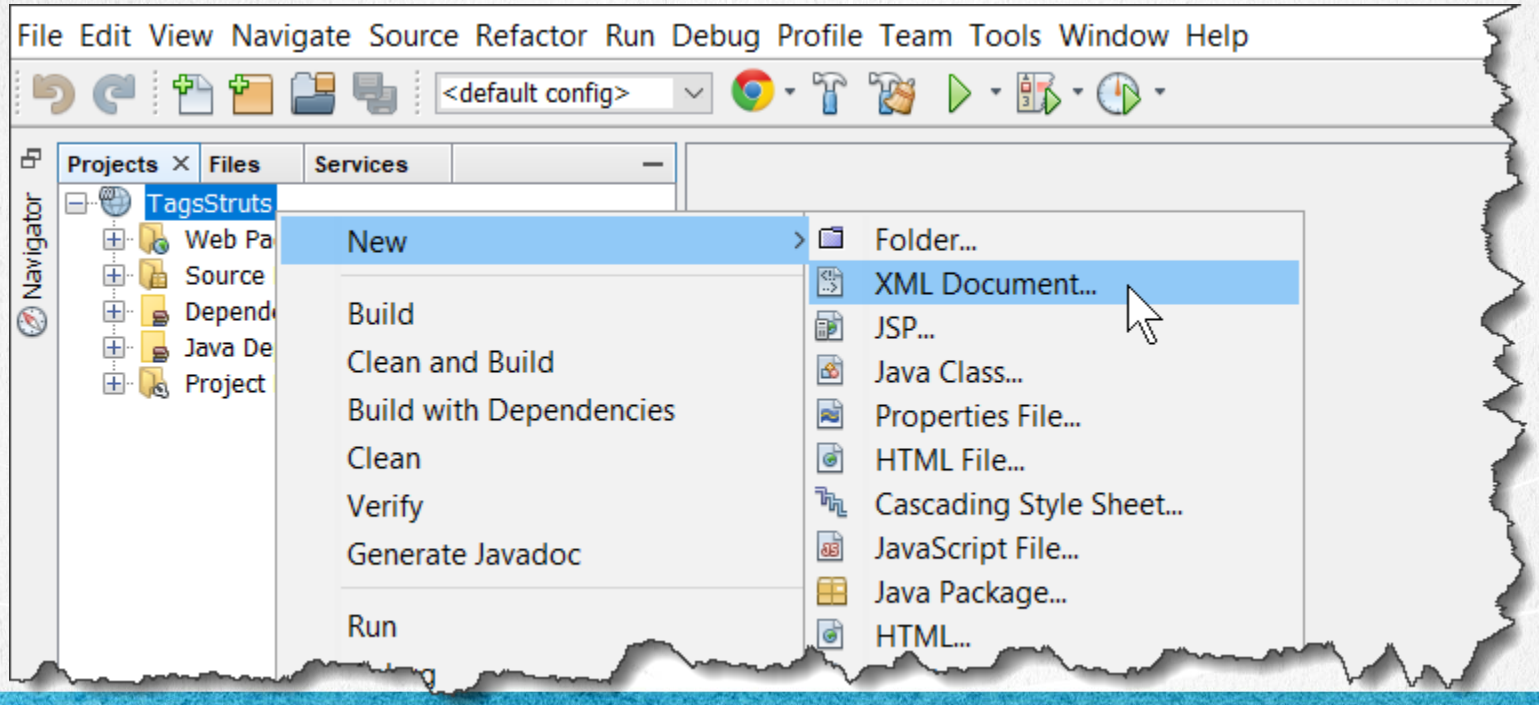
13. CREATE THE LOG4J2.XML FILE

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

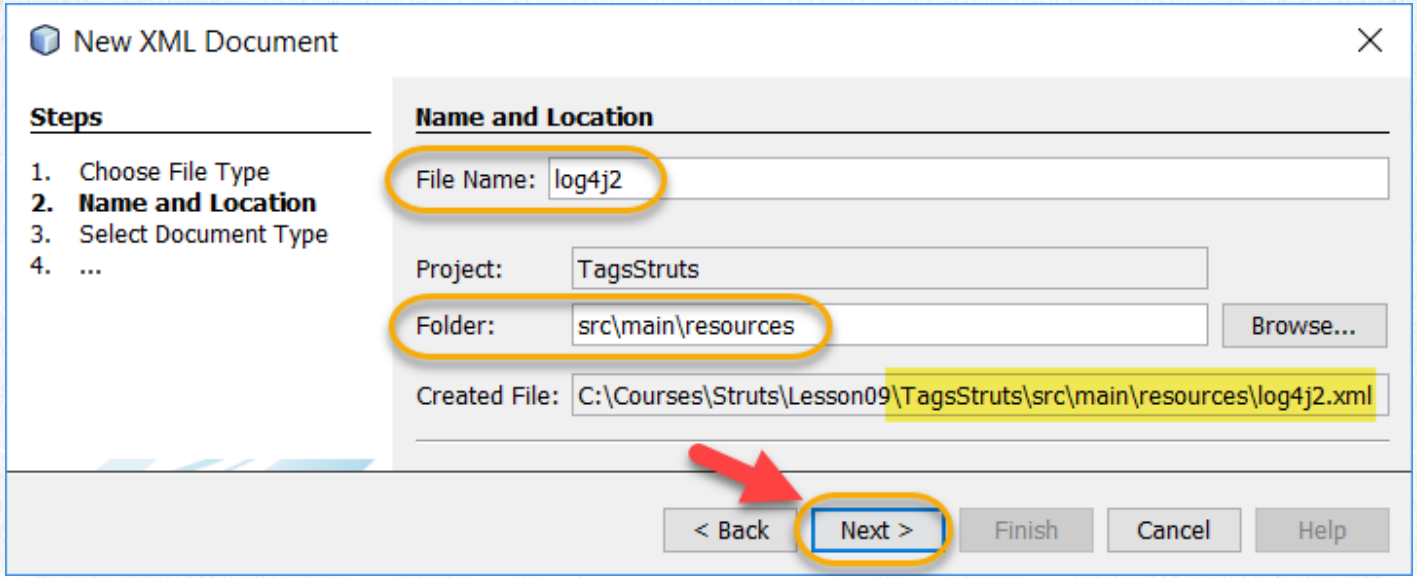
13. CREATE THE LOG4J2.XML FILE

- Create the log4j2.xml as follows:



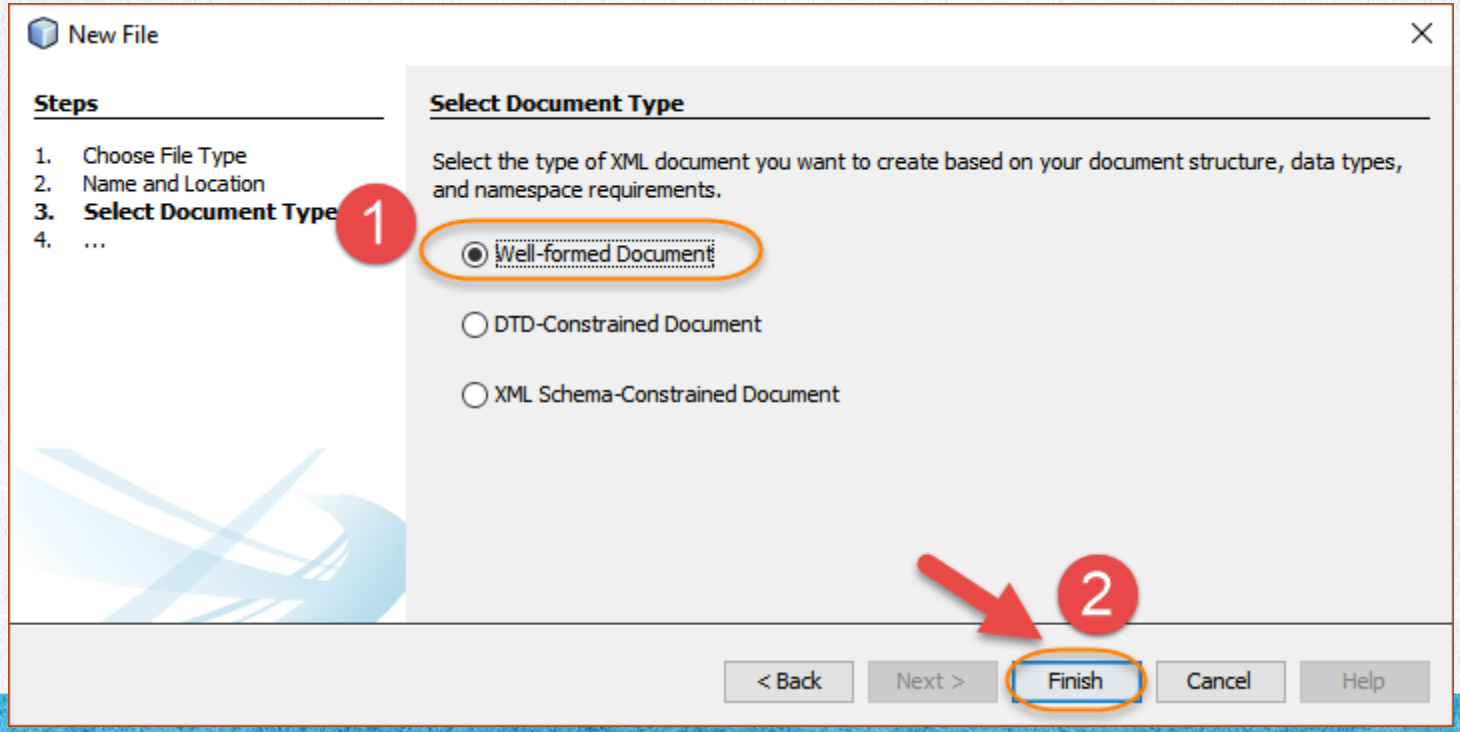
13. CREATE THE LOG4J2.XML FILE

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



13. CREATE THE LOG4J2.XML FILE

- We select the option shown:



14. MODIFY THE CODE

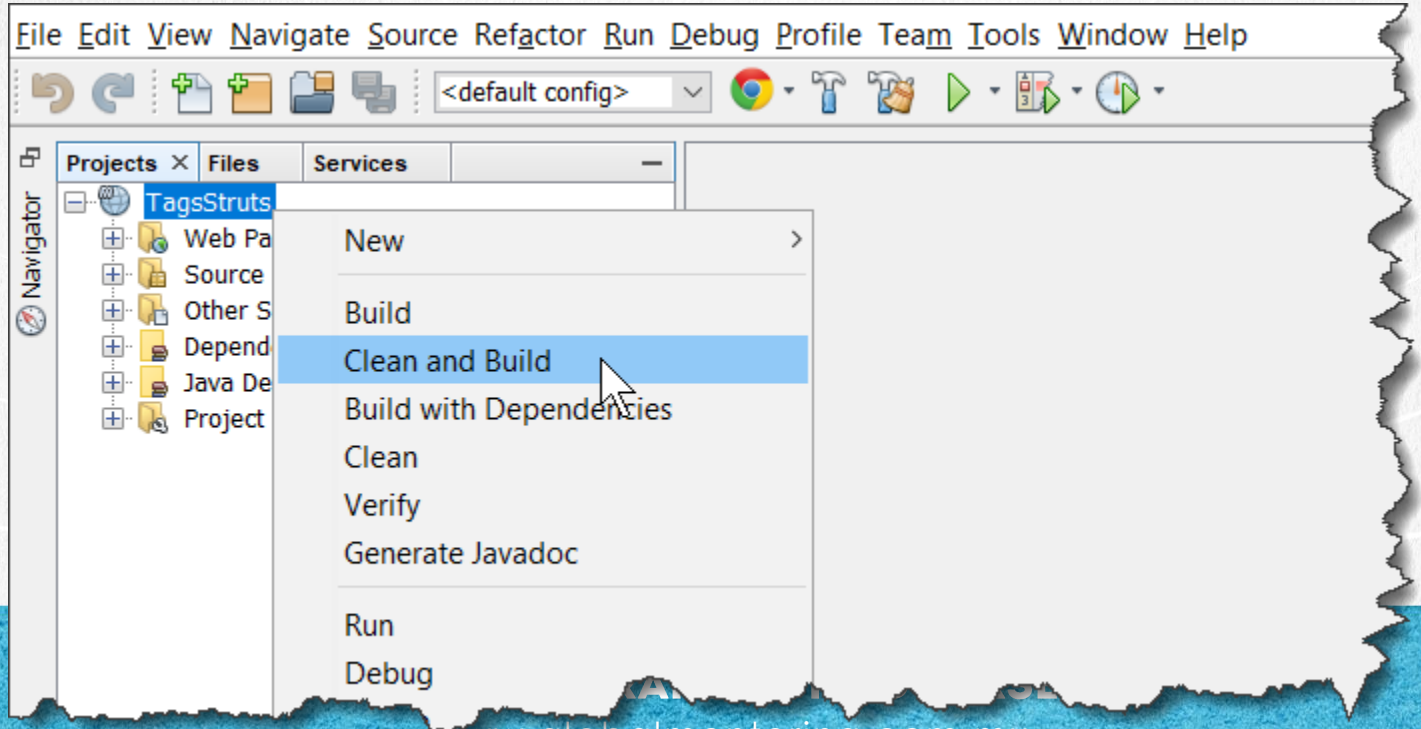
log4j2.xml:

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>
```

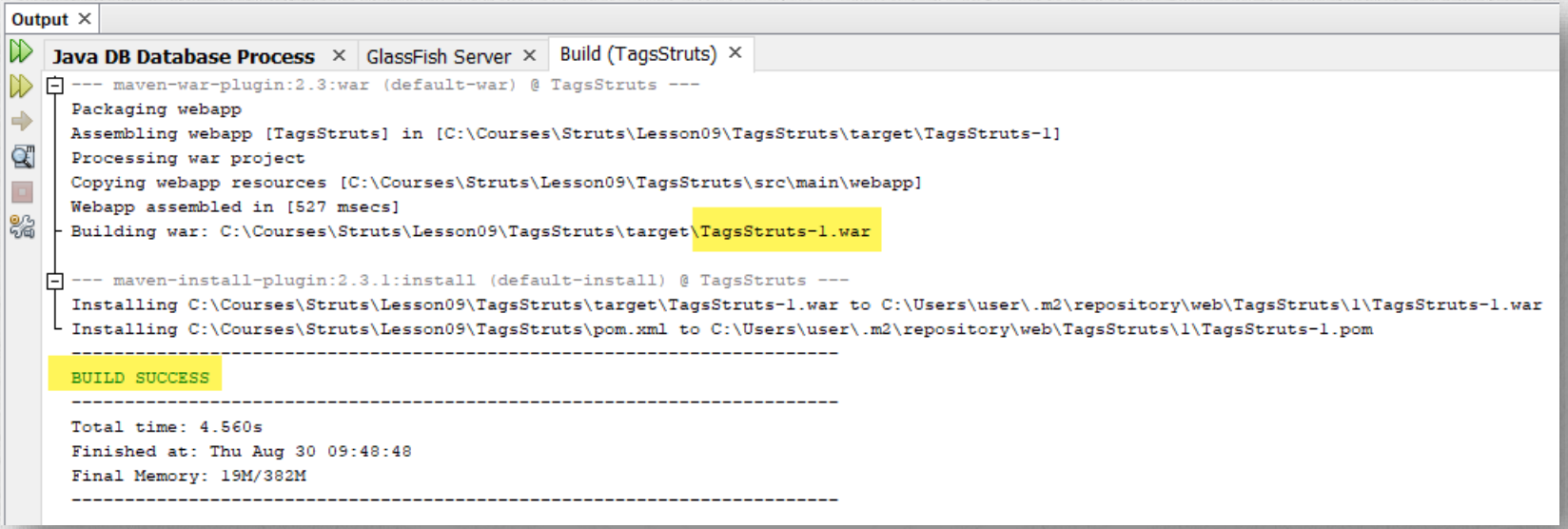
15. EXECUTE CLEAN & BUILD

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



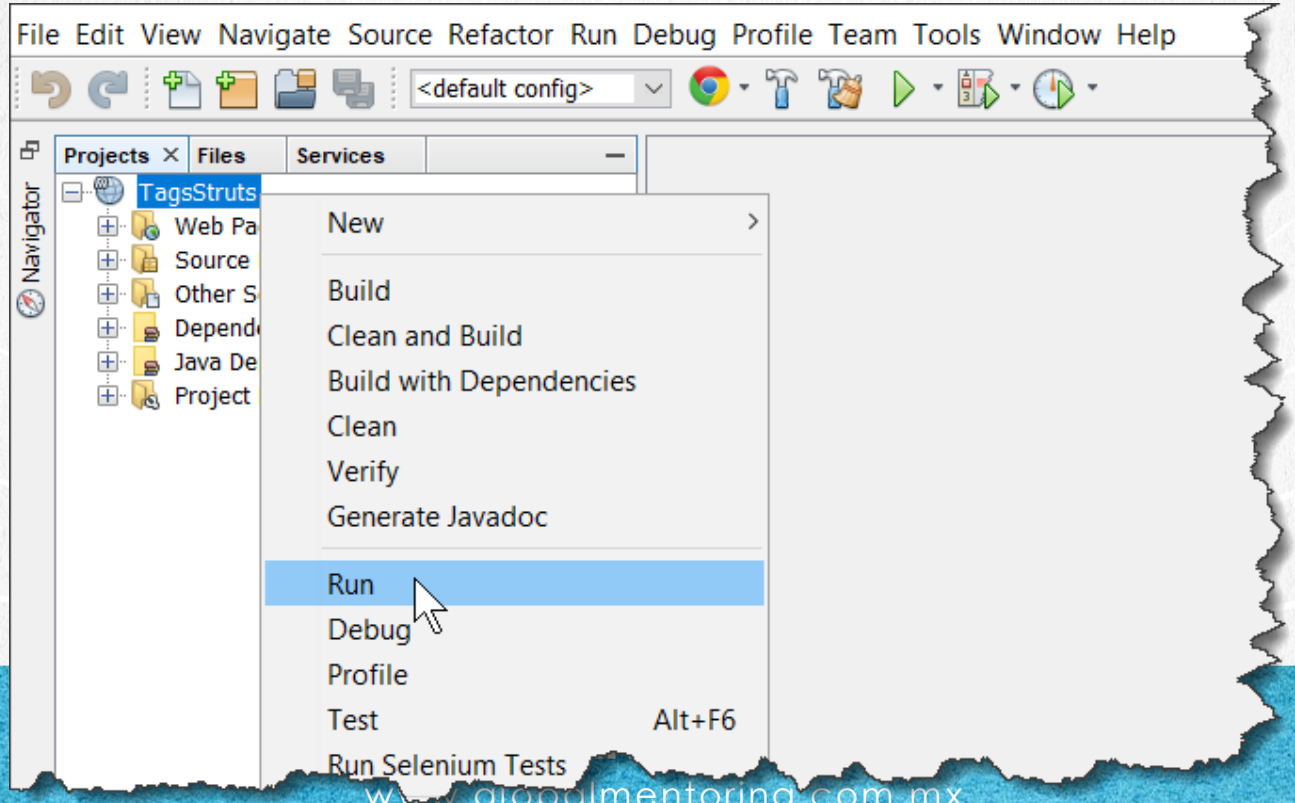
15. EXECUTE CLEAN & BUILD

- We must observe a result similar to the following:



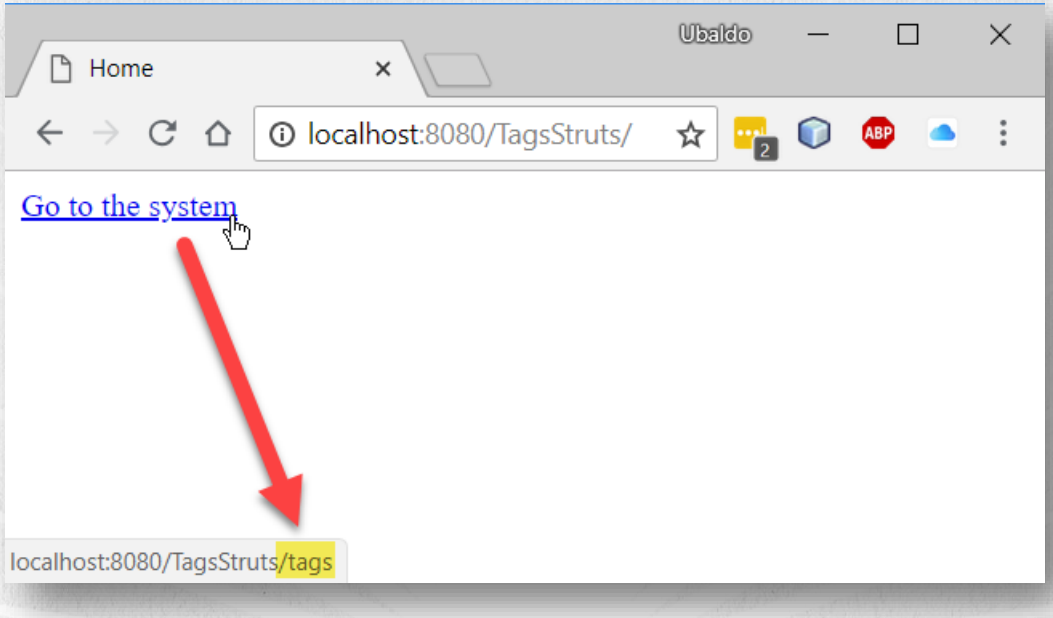
16. EXECUTE THE APPLICATION

- We run the TagsStruts application as follows:



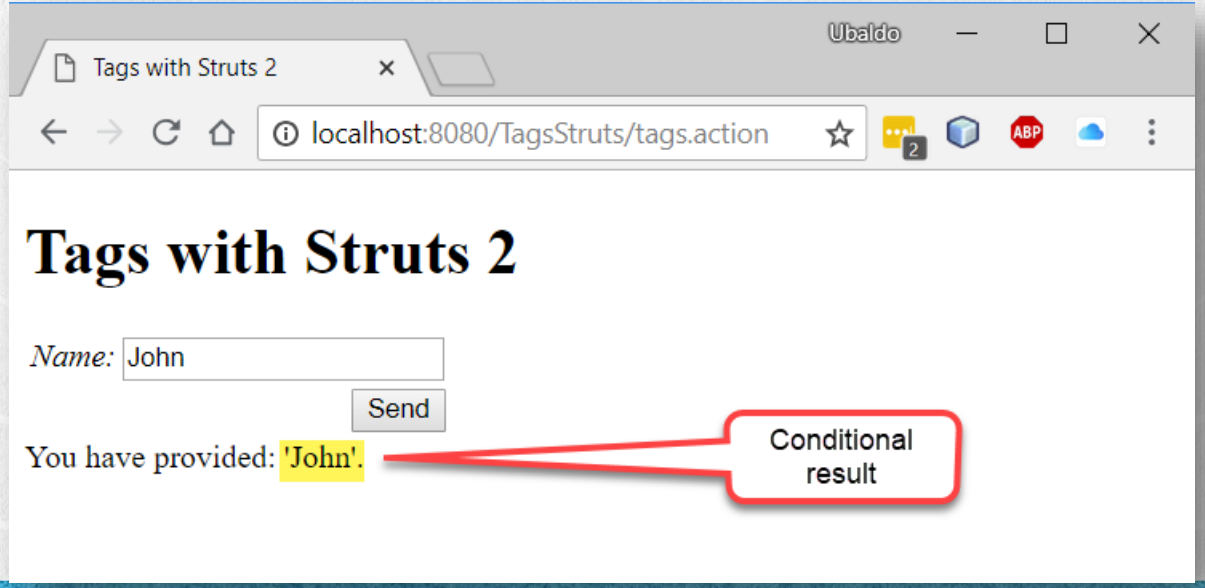
16. EXECUTE THE APPLICATION

- We run the application as follows :



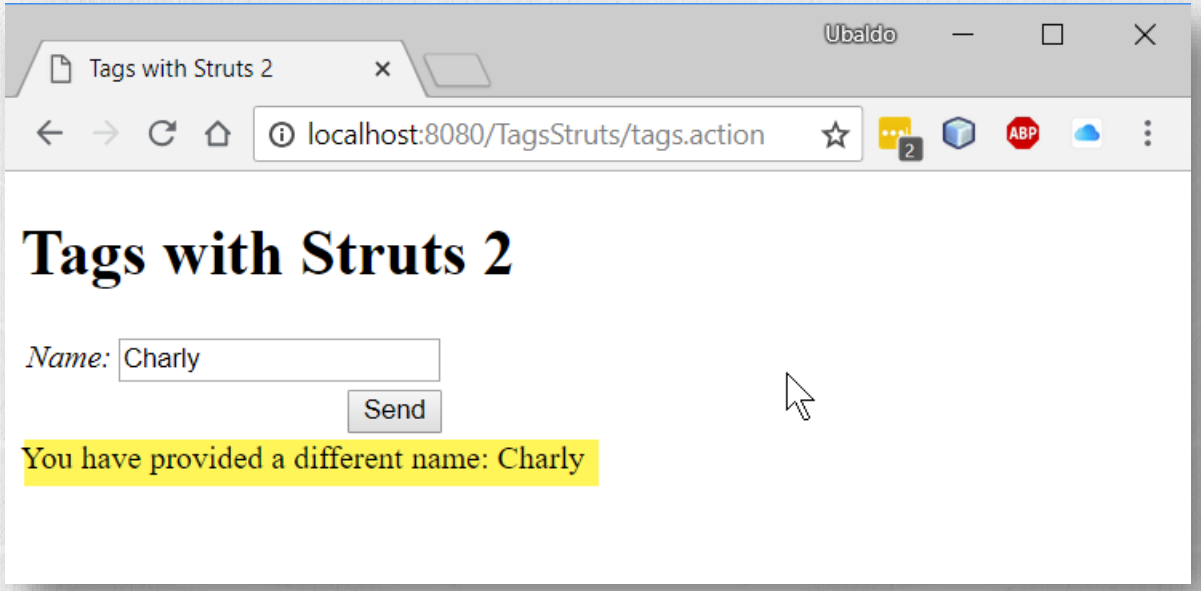
16. EXECUTE THE APPLICATION

- Fill the form and click on Send, and with this we will have applied the concept of conditional tags and some others:



16. EXECUTE THE APPLICATION

- We fill in the form we provide any other value than John or Katty, and we observe a different text :



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.

Experiencia y Conocimiento para tu vida

EXERCISE CONCLUSION

With this exercise we put into practice the concept of tags, in particular the conditional tags, but in the same way we can start using any other type of tags or labels of 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 it is clear the use of Tags since it is one of the most important of Struts 2.

We saw how there was no need to complicate our code to handle conditions in our JSP. However, it is not good practice to load our views too logically, so the handling of labels should be just as necessary and the logic that is the responsibility of the business layer as far as possible, but we know that there are requirements that do need of logic in the view, so we will use it but as little as possible.

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

ONLINE COURSE

STRUTS 2 FRAMEWORK

By: Eng. Ubaldo Acosta