AJAX 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 Ajax 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 Ajax de Struts. This topic is very extensive, however we will leave the bases so that from this project, and the integration with Jquery and Struts 2 can incorporate the subject of AJAX to their projects.

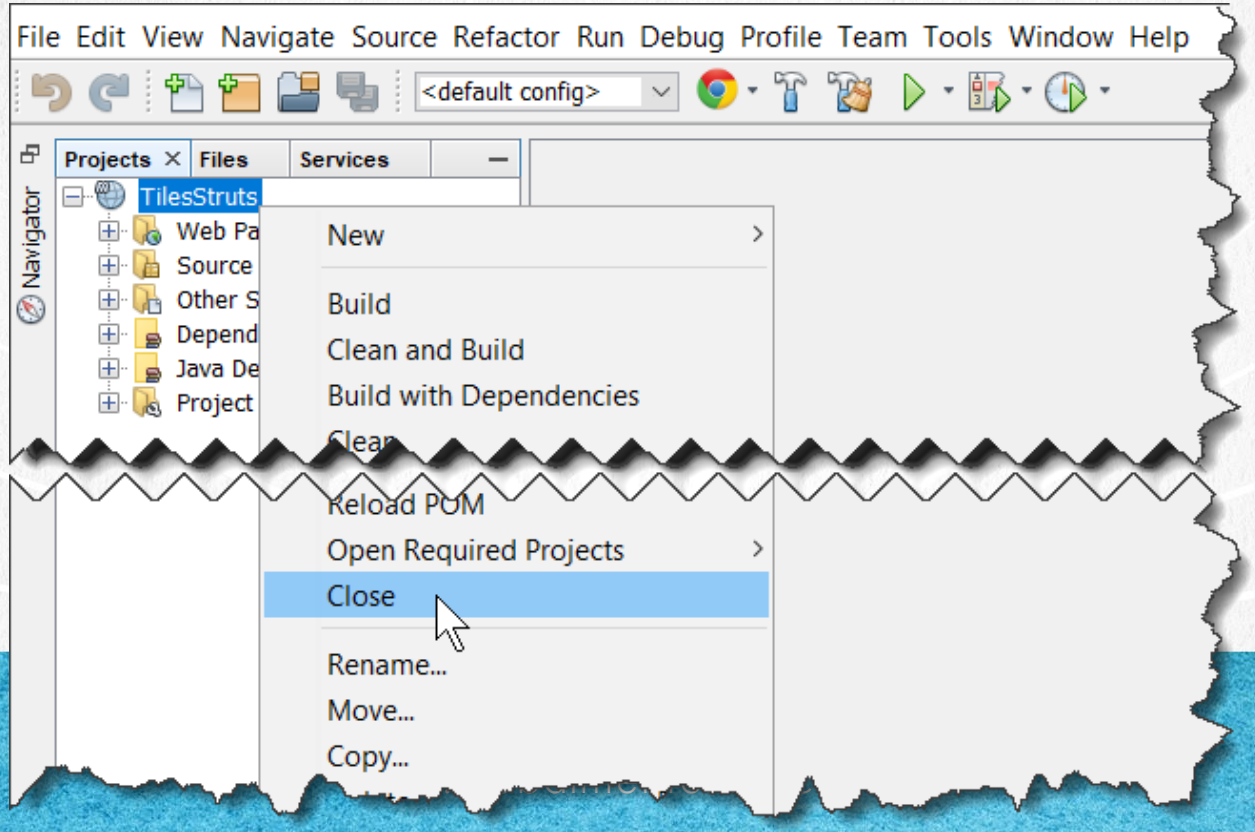
AJAX (Asynchronous JavaScript And XML) is a set of technologies to make asynchronous calls between the client and the server. For this, we will use the Jquery plug-in of Struts 2, and with this we will be able to apply Ajax in a very simple way to our pages created with Struts 2.

For more information about this technology, you can see the following link:

http://struts.jgeppert.com/struts2-jquery-showcase/index.action

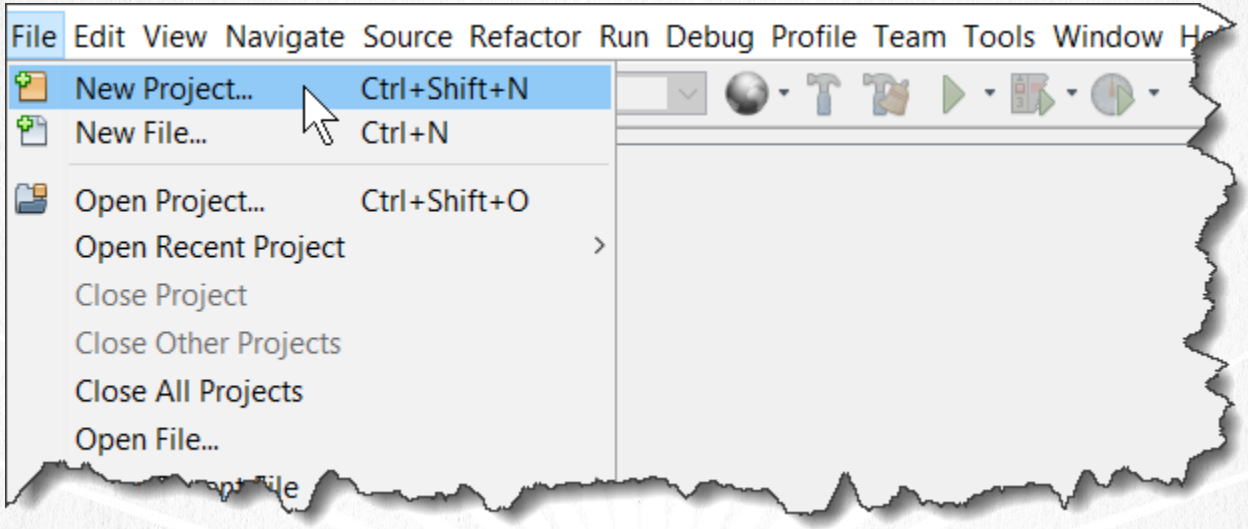
CLOSE PROJECTS THAT WE NO LONGER USE

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



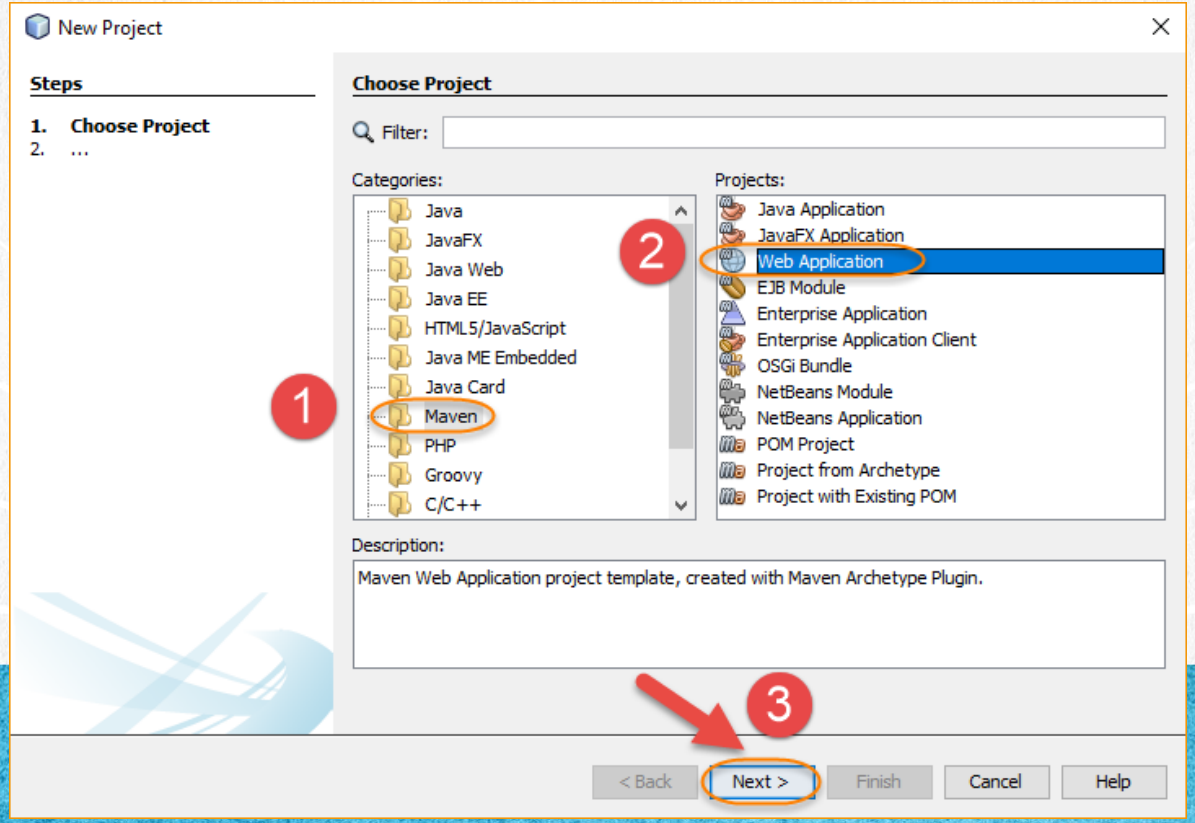
1. CREATE A NEW PROJECT

- We create the new project as shown below:



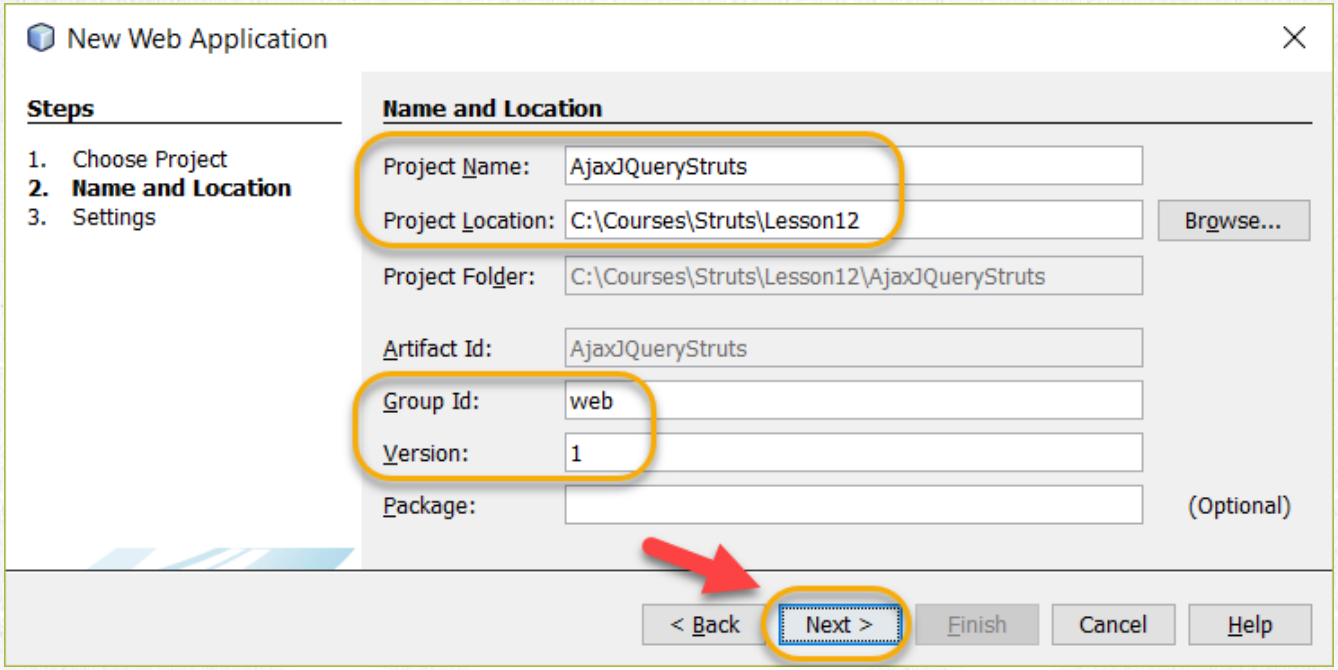
1. CREATE A NEW PROJECT

- We create the new project as shown below:



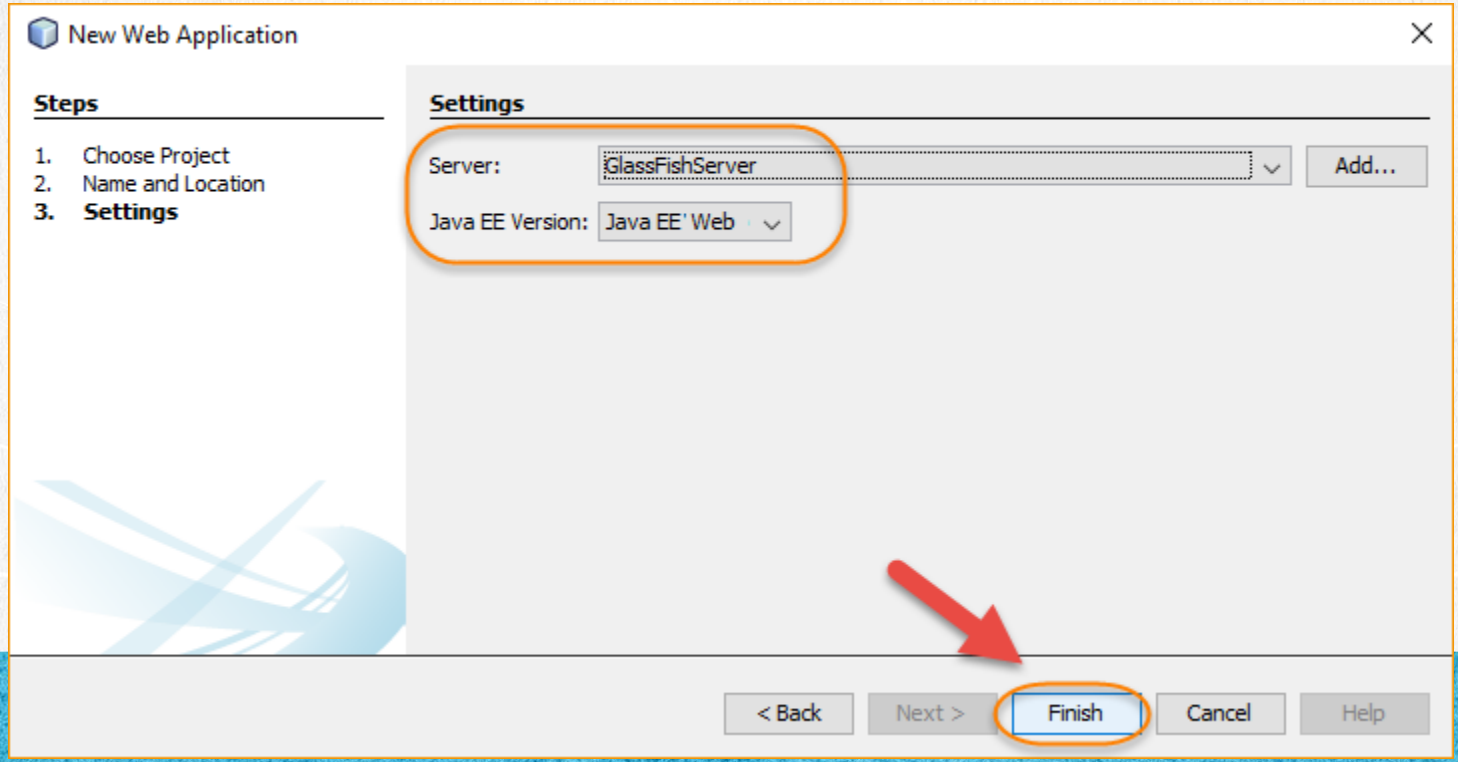
1. CREATE A NEW PROJECT

- We create the new project as shown below:



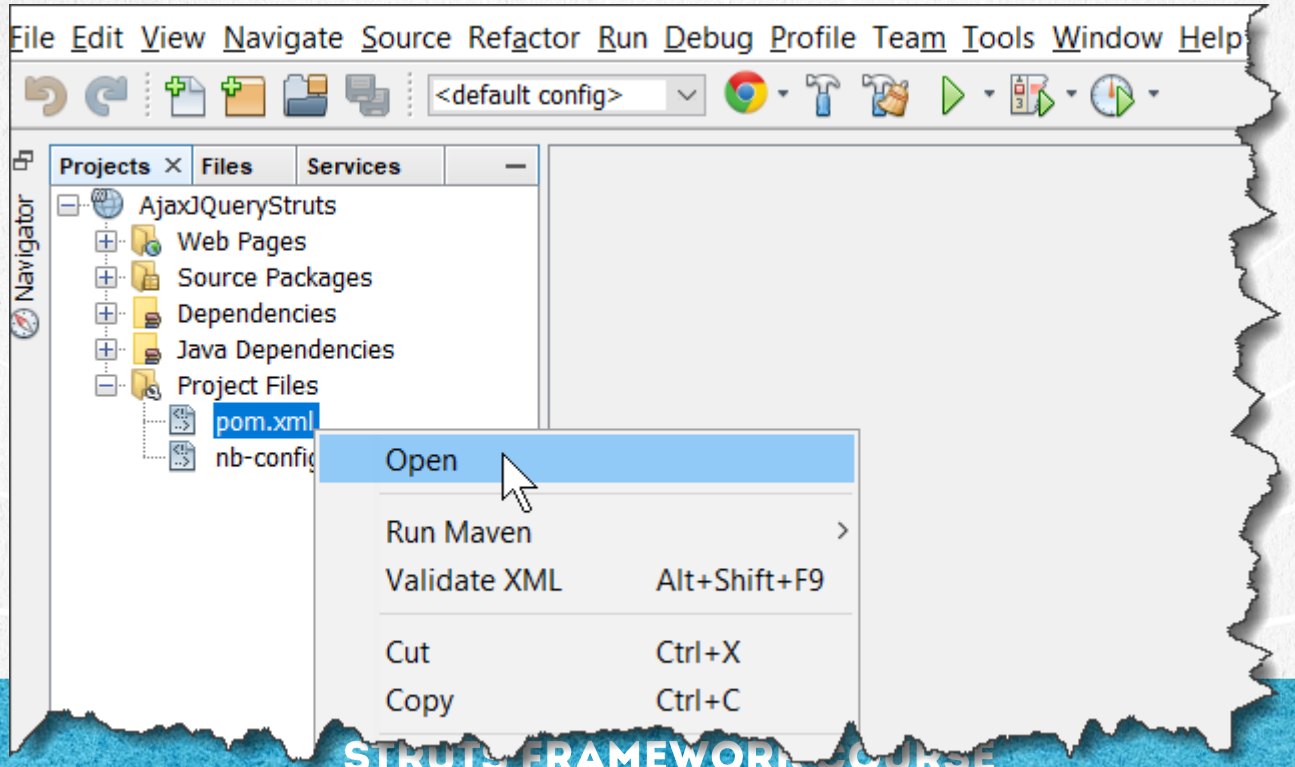
1. CREATE A NEW PROJECT

- We select the values shown:



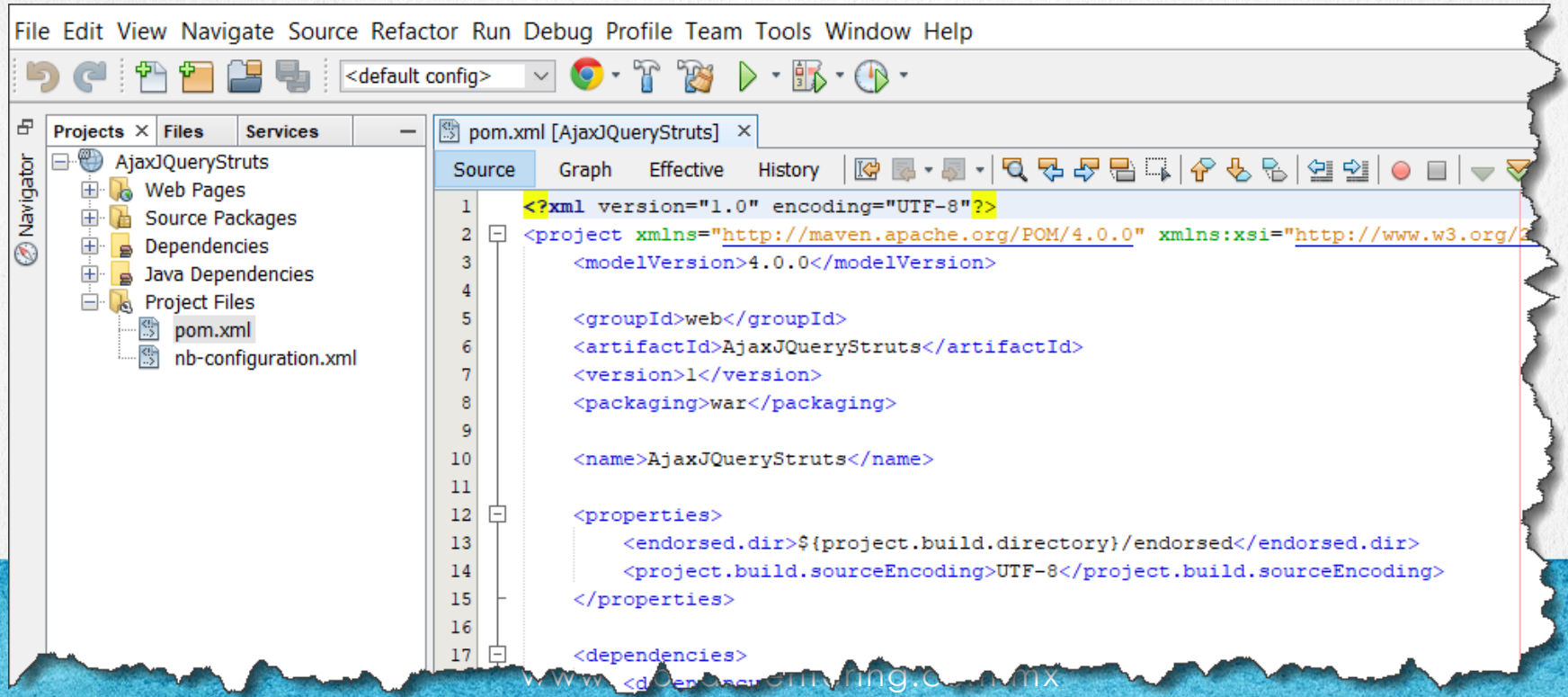
2. OPEN MAVEN'S POM.XML FILE

- 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 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>AjaxJQueryStruts</artifactId>
    <version>1</version>
    <packaging>war</packaging>

    <name>AjaxJQueryStruts</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>
         <dependency>
            <groupId>com.jgeppert.struts2.jquery</groupId>
            <artifactId>struts2-jquery-plugin</artifactId>
            <version>4.0.3</version>
        </dependency>
    </dependencies>
```

3. MODIFY THE FILE

pom.xml:

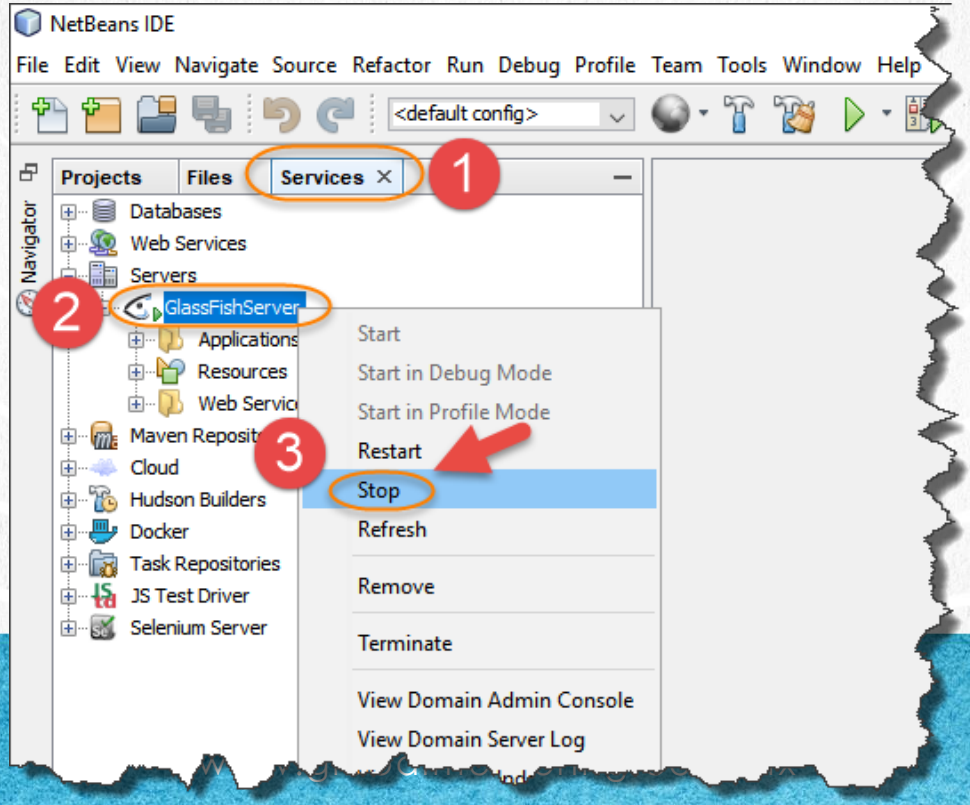
Click to download

```
    <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>
```

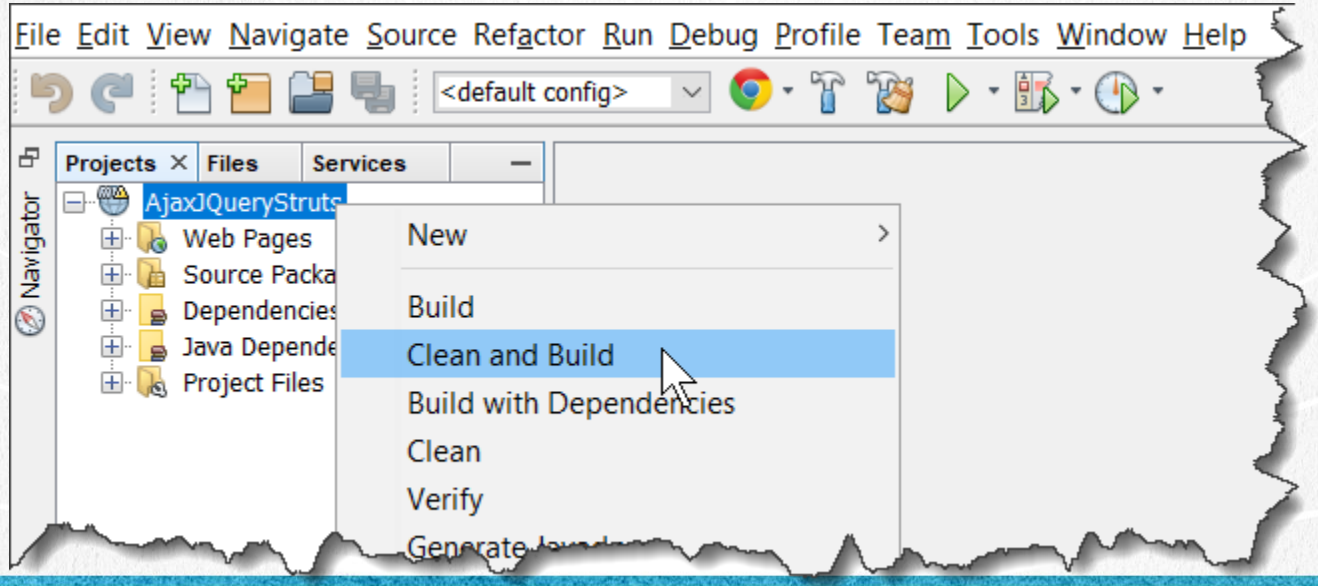
4. STOP GLASSFISH IF IT WAS STARTED

•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 for verification:



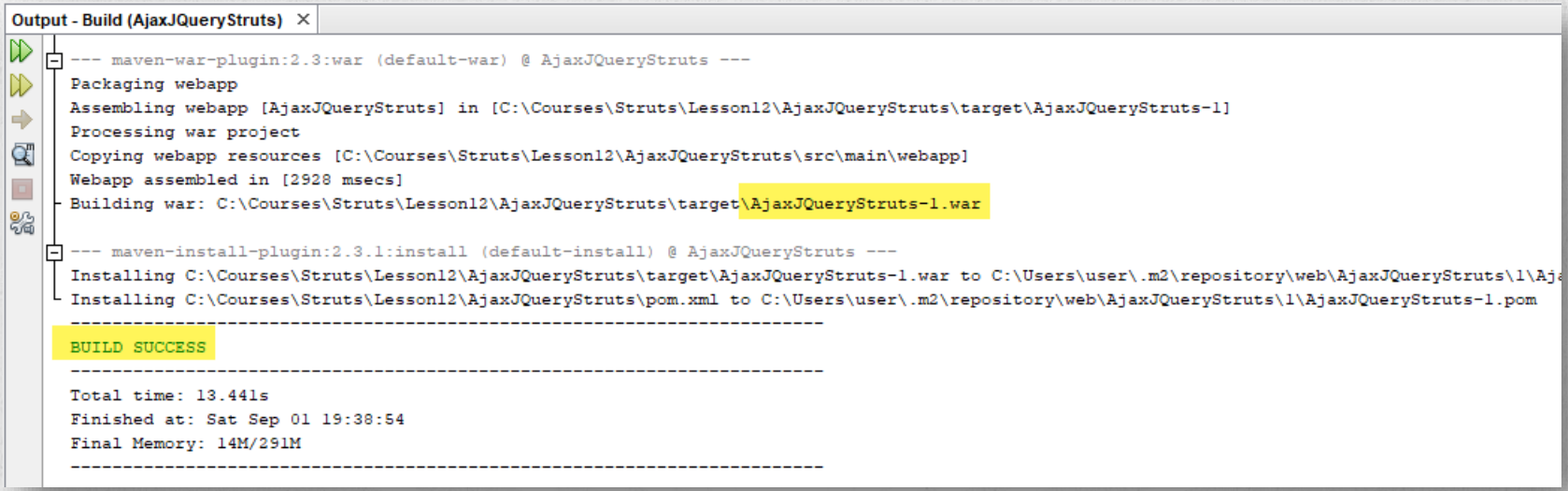
5. EXECUTE CLEAN & BUILD

- 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 :



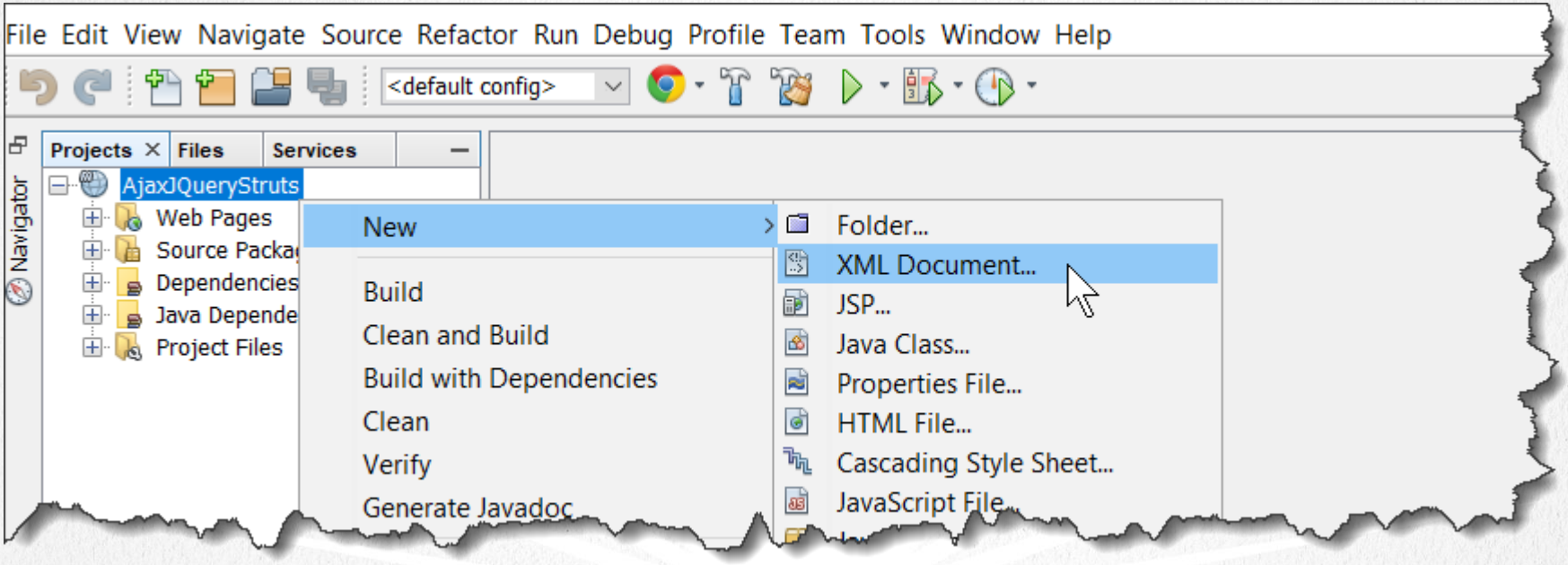
6. CREATE AN XML FILE

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.

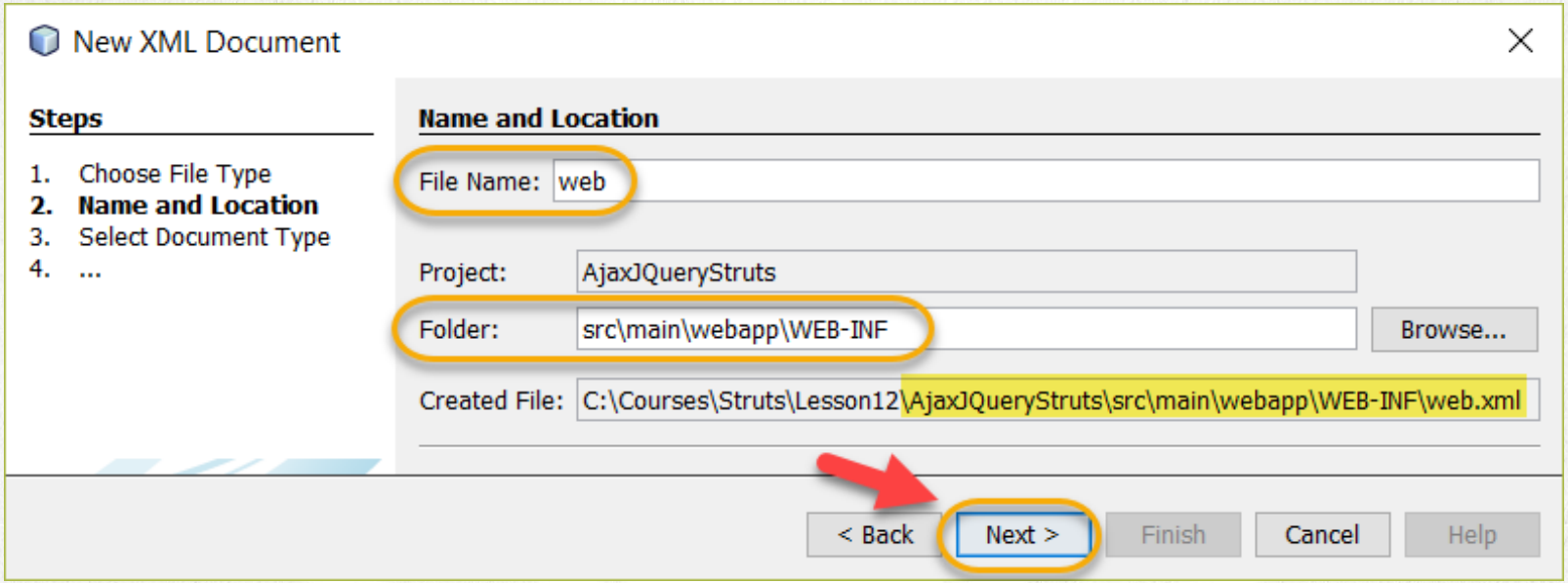
6. CREATE AN XML FILE

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



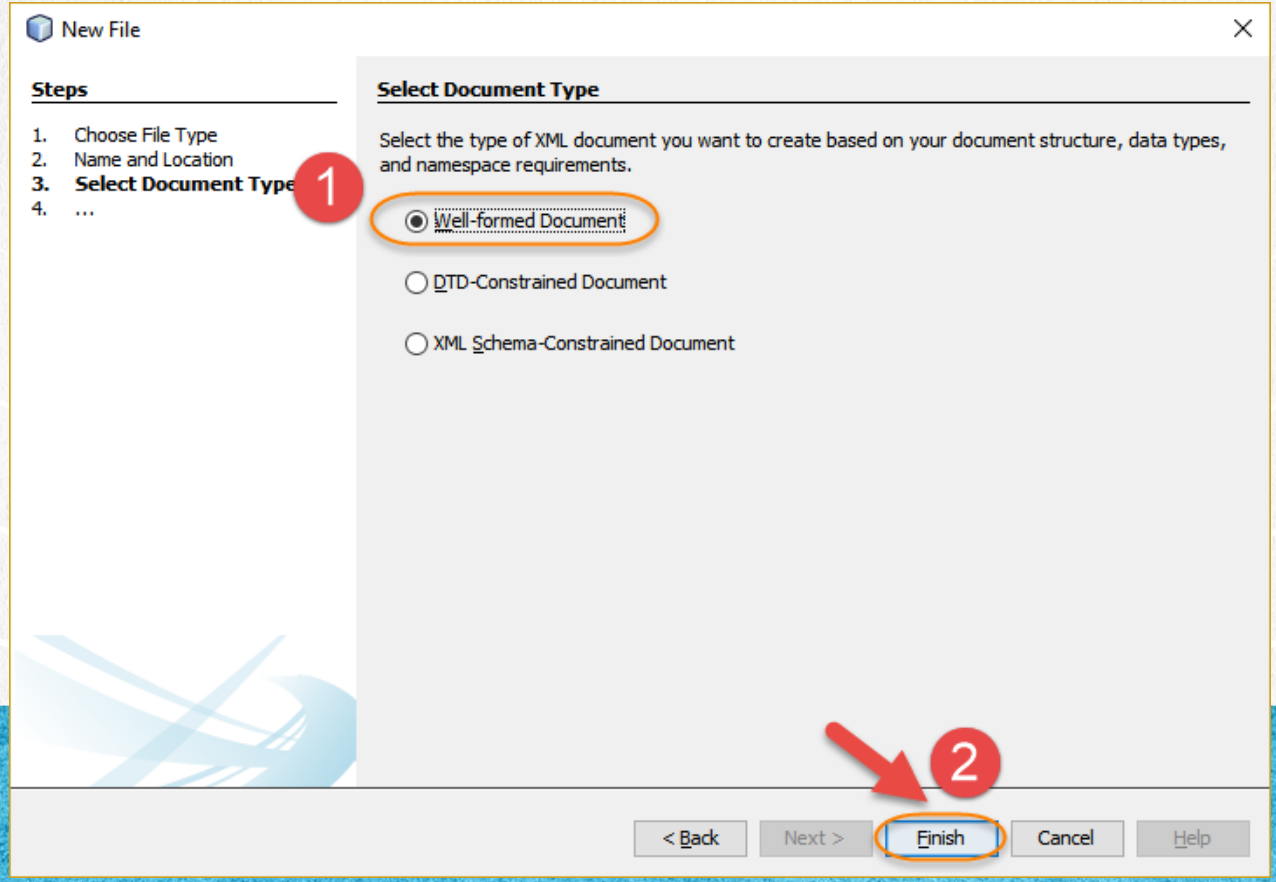
6. CREATE AN XML FILE

- 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 FILE

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>
```

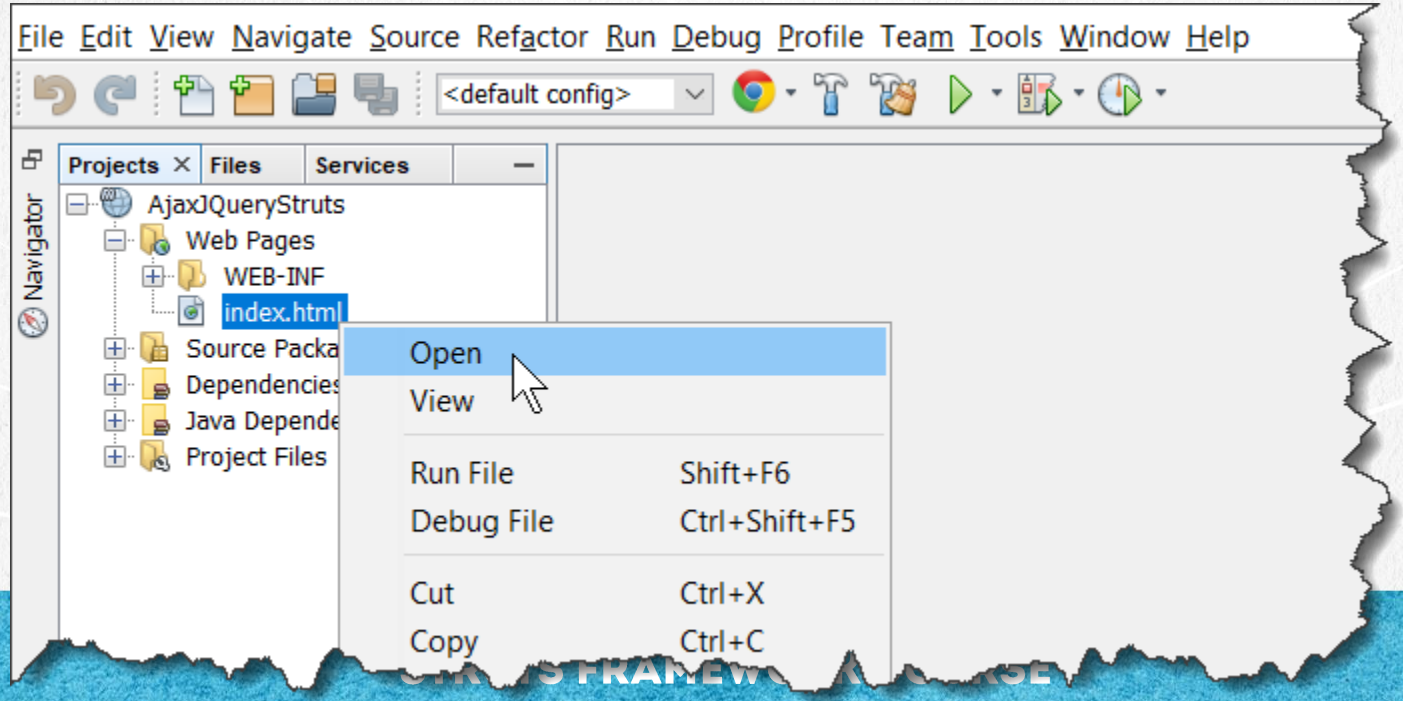
8. MODIFY THE INDEX.HTML FILE

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 is the action, or actions that we wish to execute.

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>Index</title>
    </head>
    <body>
        <a href="ajax">Basic Example with AJAX</a>
        <br/><br/>
        <a href="countries">Autocomplete Example</a>
    </body>
</html>
```

9. CREATE A JAVA CLASS

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

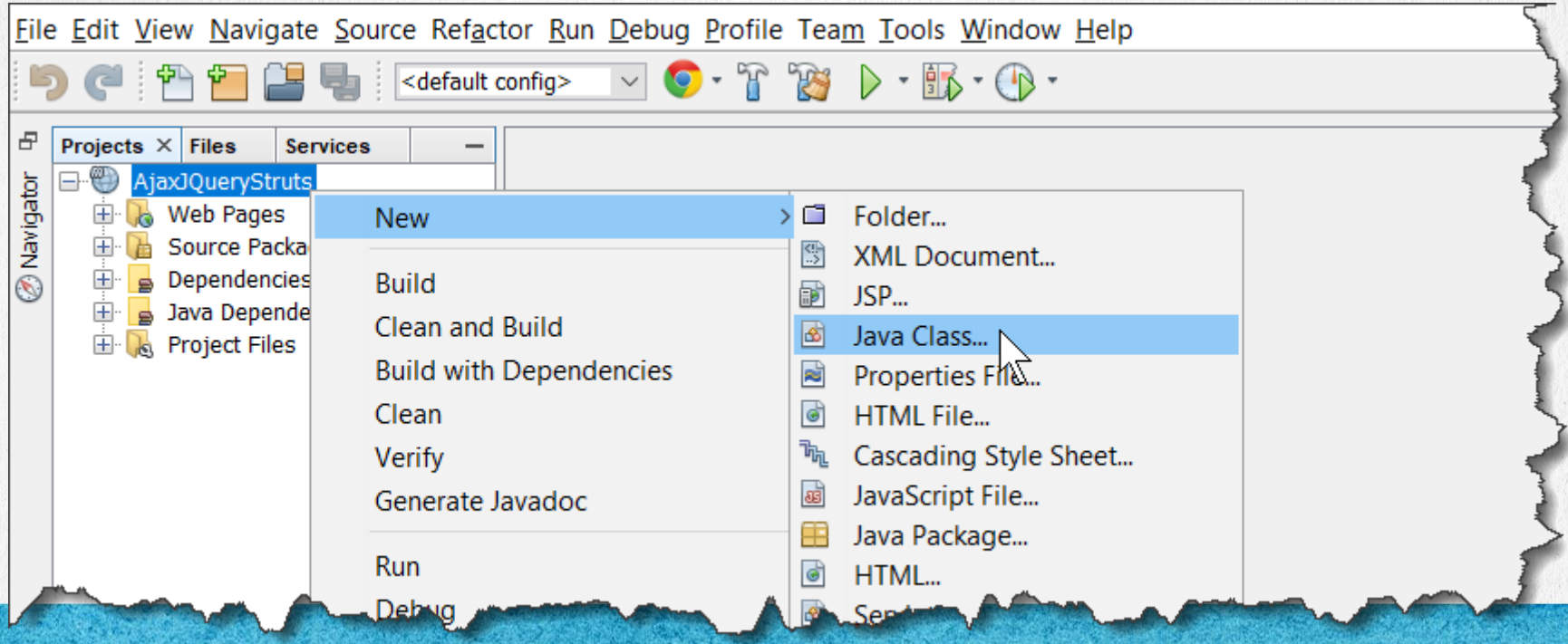
We will extend the ActionSupport class and overwrite the execute method.

We will apply the AJAX theme to show the name value, and from the respective view (JSP) we will configure it so that the request is via AJAX. This means that nothing needs to be modified in our Action type class, we just need to modify our respective JSP to indicate that the request will be via AJAX.

Let's see how our class is.

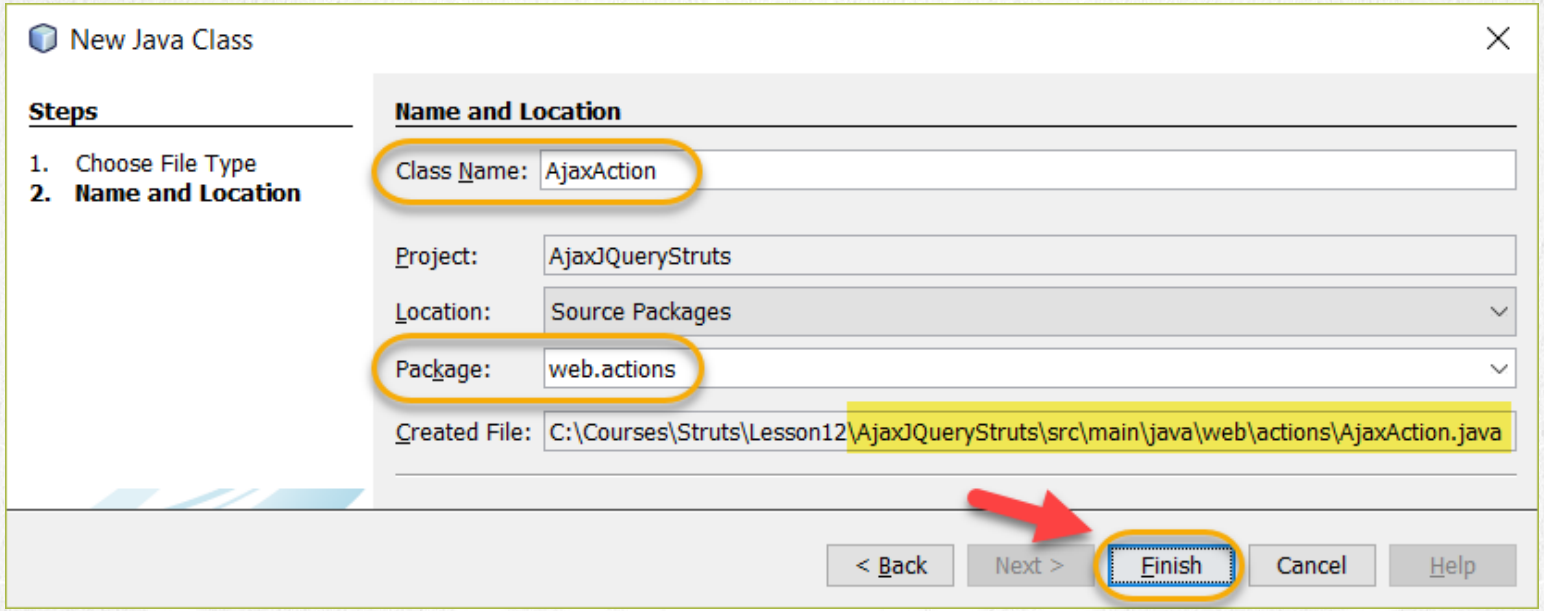
9. CREATE A JAVA CLASS

- We create the AjaxAction.java class:



9. CREATE A JAVA CLASS

- We create the AjaxAction.java class :



10. MODIFY THE CODE

AjaxAction.java:

Click to download

```
package web.actions;

import com.opensymphony.xwork2.ActionSupport;

public class AjaxAction extends ActionSupport {

    private String name;

    @Override
    public String execute() {
        System.out.println("Name:" + name);
        return SUCCESS;
    }

    public String getName() {
        return name;
    }

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

11. CREATE A JAVA CLASS

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

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

We will apply the AJAX theme to show the value a list of values, applying the concept of dropdown menu, where we will see several options as they are in the list, and the user can select these items from the list whether they come from a database or simply from a static list like in this case.

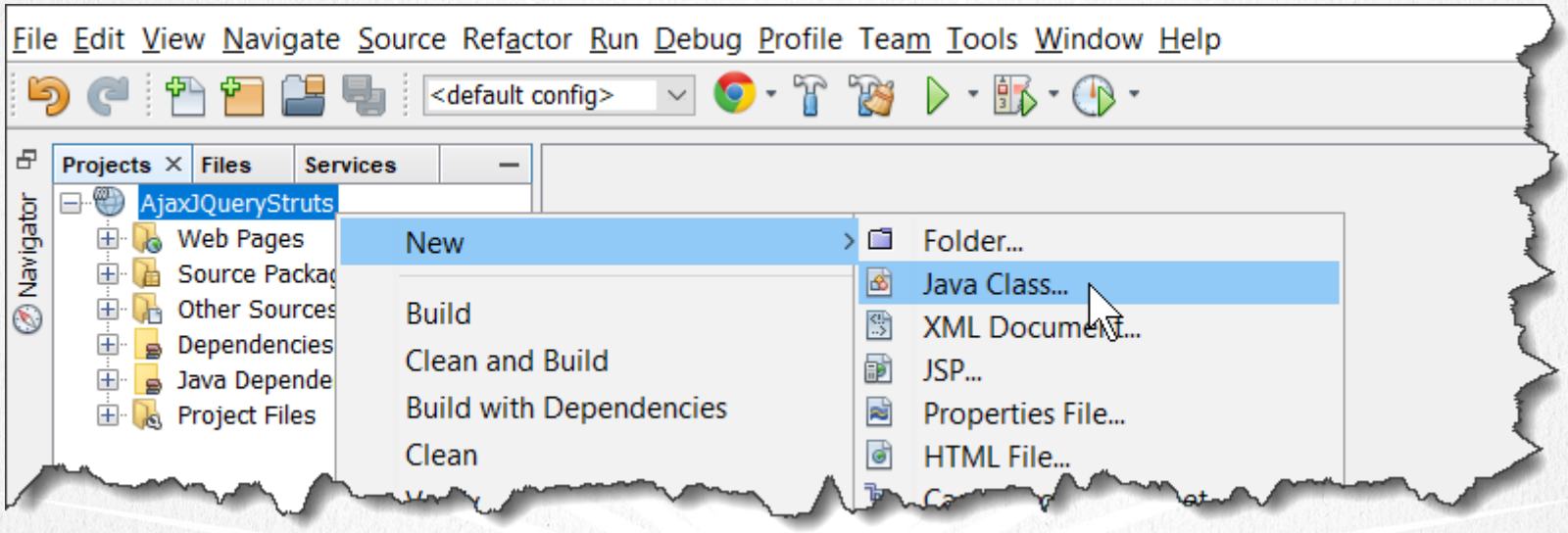
Once selected the value will be displayed. All this functionality is applying the concept of AJAX, since as we can see the data is obtained via AJAX asynchronously, everything is done automatically the plug-in jquery and struts, and this is enough to apply the concept of AJAX to our application with Struts 2.

As part of an Action class in Struts 2, we have the prepare method available, which is executed before the execute method, and we can use this method to initialize any variable or data we need. In this case we will use it to initialize the list of countries.

Let's see how our class is.

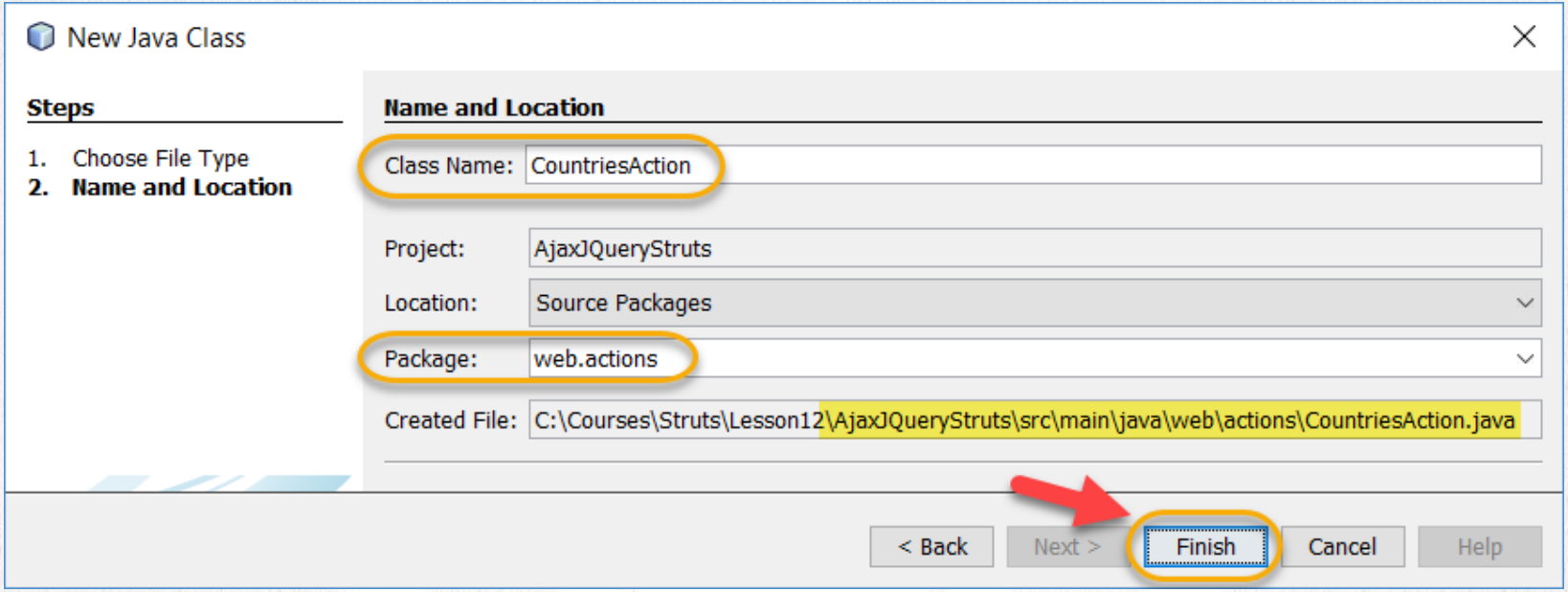
11. CREATE A JAVA CLASS

- We create the CountriesAction.java class :



11. CREATE A JAVA CLASS

- We create the CountriesAction.java class :



12. MODIFY THE CODE

CountriesAction.java:

Click to download

```
package web.actions;

import com.opensymphony.xwork2.*;
import java.util.*;
import org.apache.logging.log4j.*;

public class CountriesAction extends ActionSupport implements Preparable {

    private final String data = "Mexico, United States, Colombia, Argentina, China, Japan";
    private List<String> countries;
    private String country;
    Logger log = LogManager.getLogger(CountriesAction.class);

    @Override
    public void prepare() {
        log.info("We load the list of countries");
        countries = new ArrayList<>();
        StringTokenizer st = new StringTokenizer(data, ",");

        while (st.hasMoreTokens()) {
            countries.add(st.nextToken().trim());
        }
    }
```

12. MODIFY THE CODE

CountriesAction.java:

Click to download

```
    @Override
    public String execute() {
        log.info("Selected country: " + country);
        return SUCCESS;
    }

    public String getCountry() {
        return this.country;
    }

    public List<String> getCountries() {
        return countries;
    }

    public void setCountries(List<String> countries) {
        this.countries = countries;
    }

    public void setCountry(String country) {
        this.country = country;
    }
}
```

13. CREATE A JSP FILE

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

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

In this jsp is that we will include the AJAX theme, and it is very simple. We will add a library called:

```
<%@taglib prefix="sj" uri="/struts-jquery-tags"%>
```

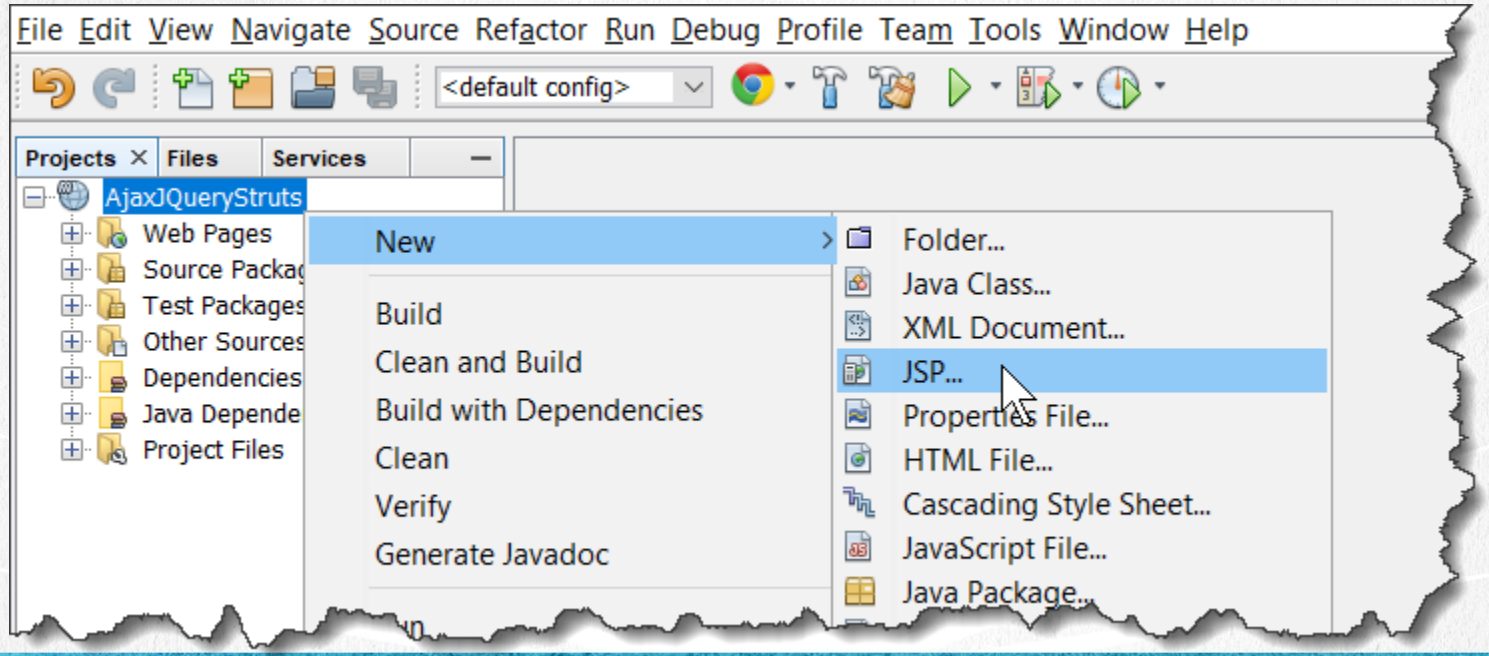
In addition we will also add the tag: `<sj:head />`

The `<sj:submit />` tag is the one that will allow us to send data using AJAX.

With this we will be ready to implement the AJAX concept, using jquery and Struts 2.

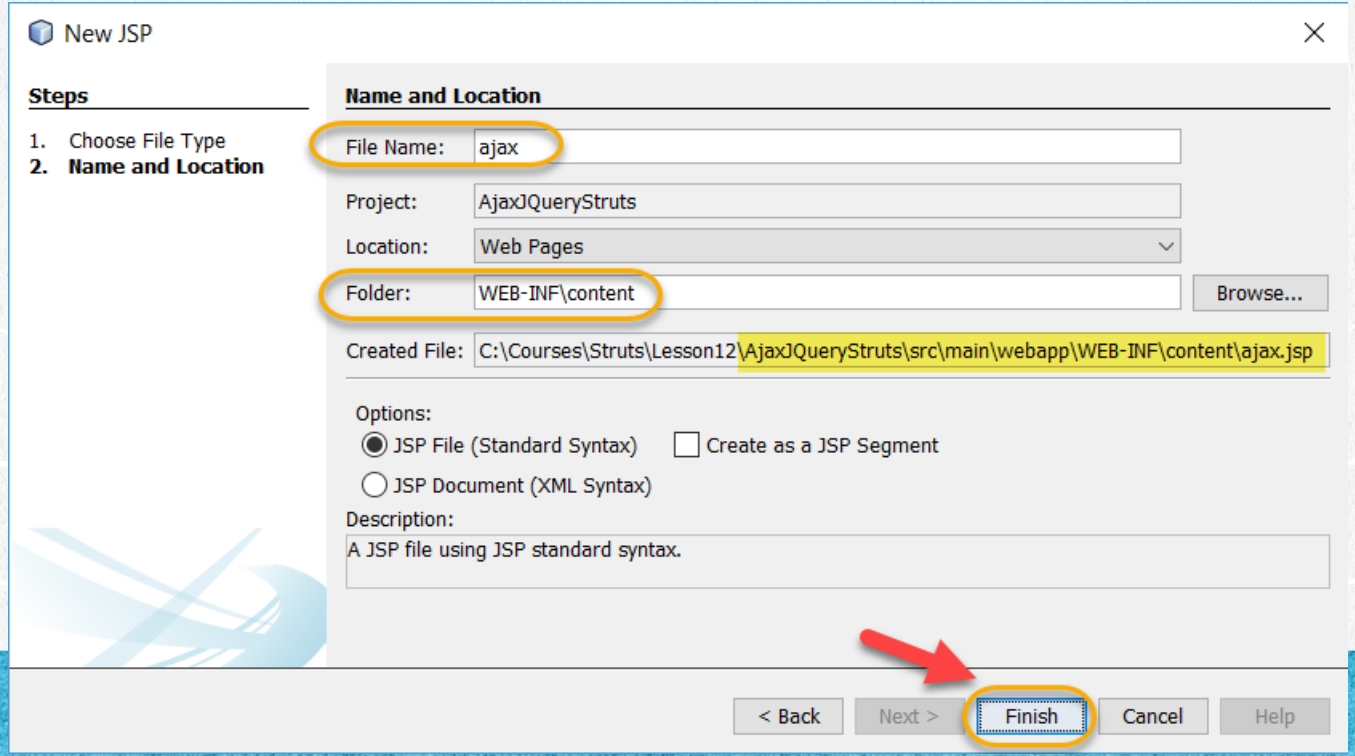
13. CREATE A JSP FILE

- We create the file ajax.jsp:



13. CREATE A JSP FILE

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



14. MODIFY THE CODE

ajax.jsp:

Click to download

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
<%@ taglib prefix="sj" uri="/struts-jquery-tags"%>
<!DOCTYPE html>
<html>
    <head>
        <title>Ajax with Jquery and Struts 2</title>
        <sj:head/>
    </head>
    <body>
        <h2>Introduce your name</h2>
        <s:form >
                <s:textfield name="name" />
            <div class="type-button">
                <sj:submit
                    targets="show"
                    value="Ajax Submit"
                    indicator="indicator"
                    button="true"
                    />
            </div>
        </s:form>
        <div id="show"><h4><s:property value="name"/></h4></div>
    </body>
</html>
```

15. CREATE A JSP FILE

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

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

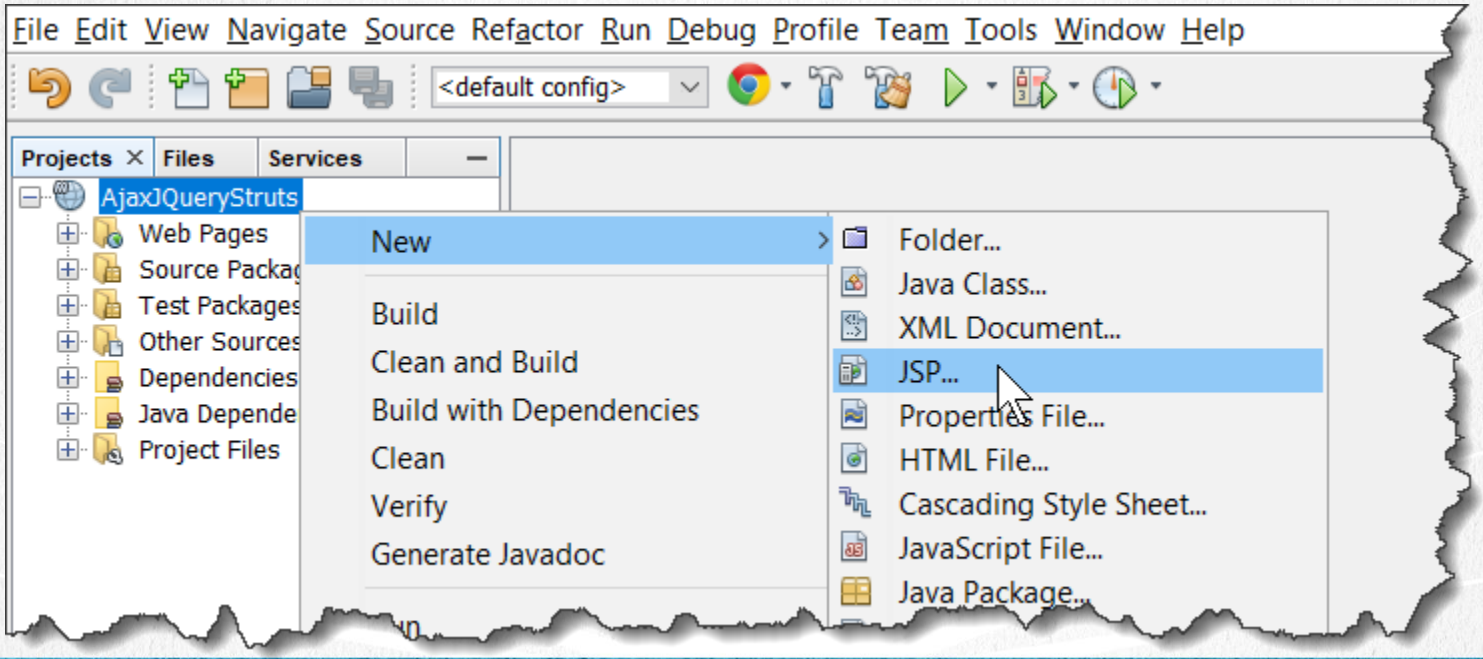
In this jsp is that we will include the AJAX theme, and it is very simple. We will add a library called:
`<%@ taglib prefix="sj" uri="/struts-jquery-tags"%>`

In addition we will also add the tag: `<sj:head />` and the tag: `<sj: autocompleter/>`

With this we will be ready to use the tags that implement the AJAX concept, in particular the dropdown menu, also known as autocompleter using jquery and Struts 2.

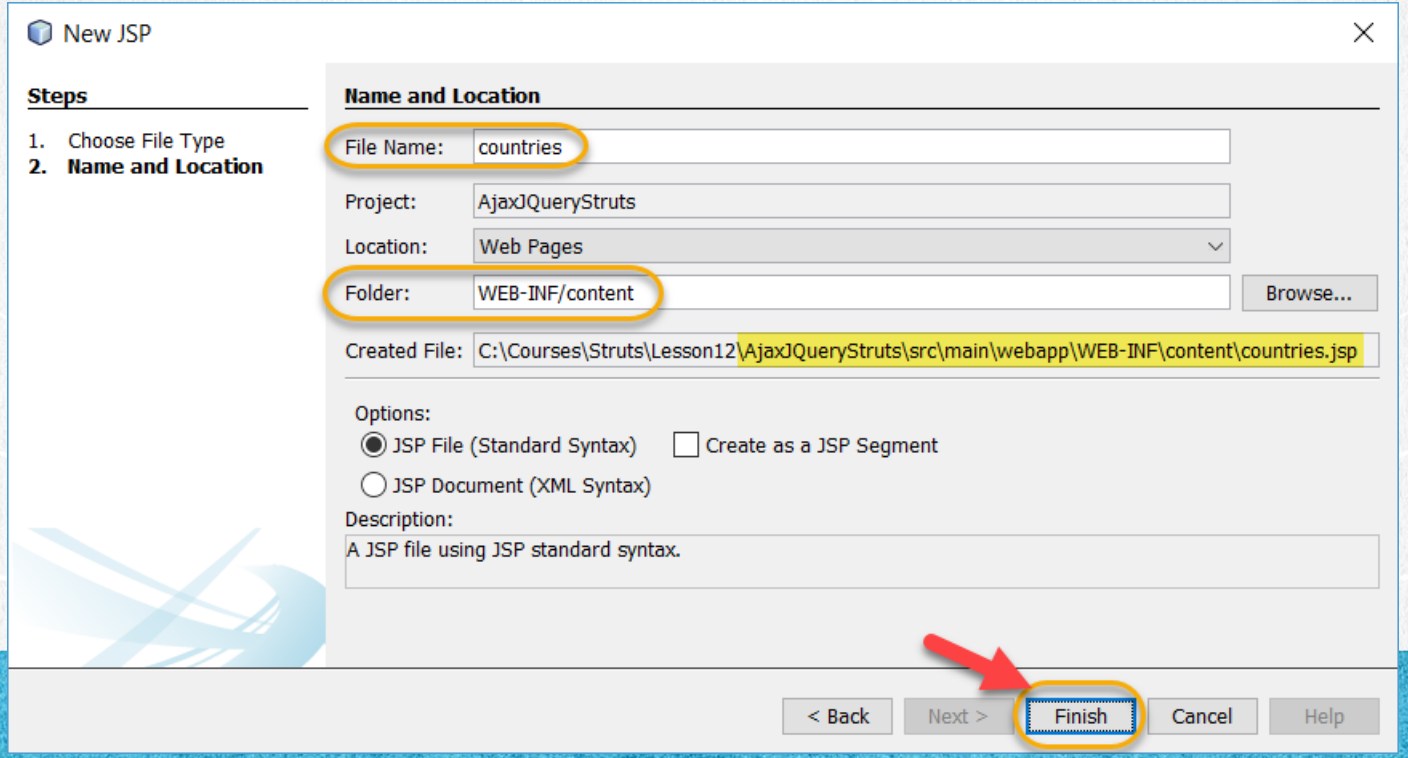
15. CREATE A JSP FILE

- We create the countries.jsp file:



15. CREATE A JSP FILE

- We create the countries.jsp file:



16. MODIFY THE CODE

countries.jsp:

Click to download

```
<%@ page contentType="text/html; charset=UTF-8"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
<%@ taglib prefix="sj" uri="/struts-jquery-tags"%>
<html>
    <head>
        <title>Countries</title>
        <sj:head />
    </head>
    <body>
        <h2>Autocomplete example with Struts 2 and JQuery (Drop down)</h2>
        <s:form >
            Country: <sj:autocompleter size="15" list="countries" name="country"></sj:autocompleter>
                <div class="type-button">
                <sj:submit
                    targets="show"
                    value="Ajax Submit"
                    indicator="indicator"
                    button="true"
                    />
            </div>
        </s:form>
        <br/>
        <div id="show">Selected Country:<s:property value="country"/></div>
    </body>
</html>
```

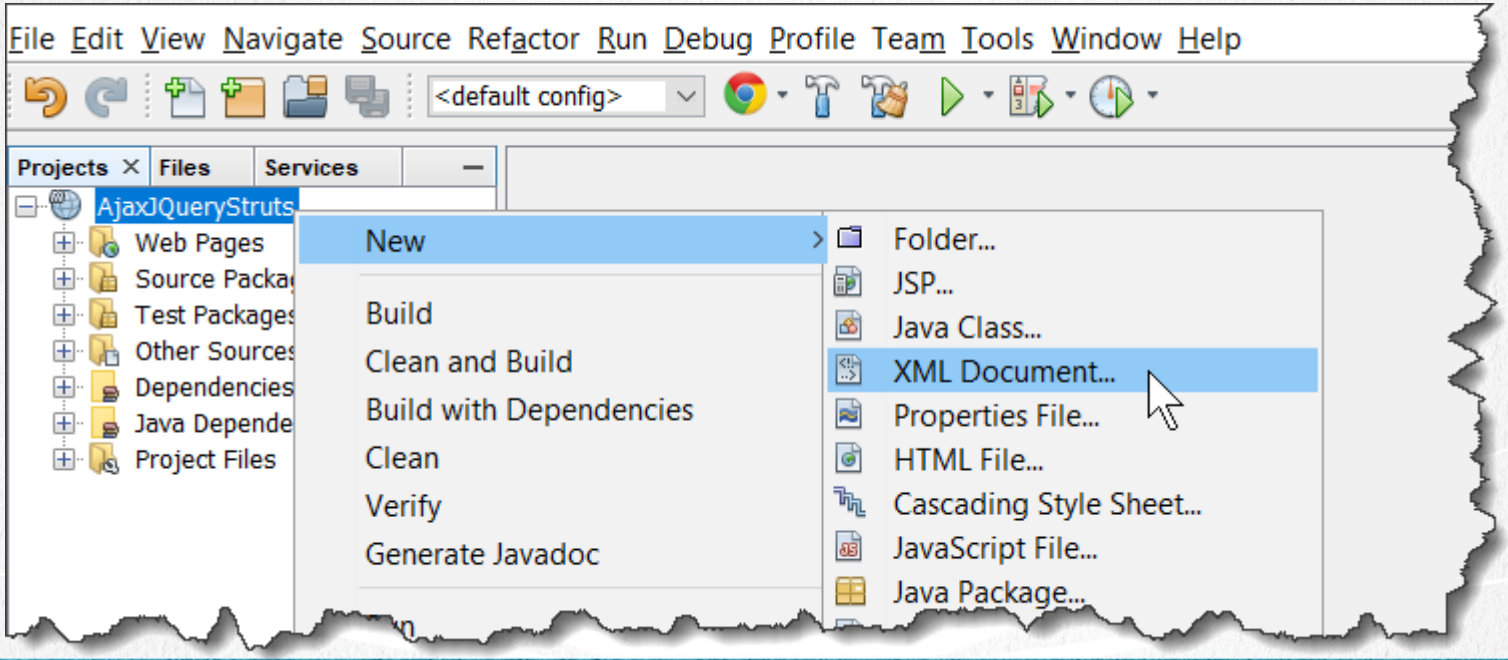
17. CREATE THE LOG4J2.XML FILE

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.

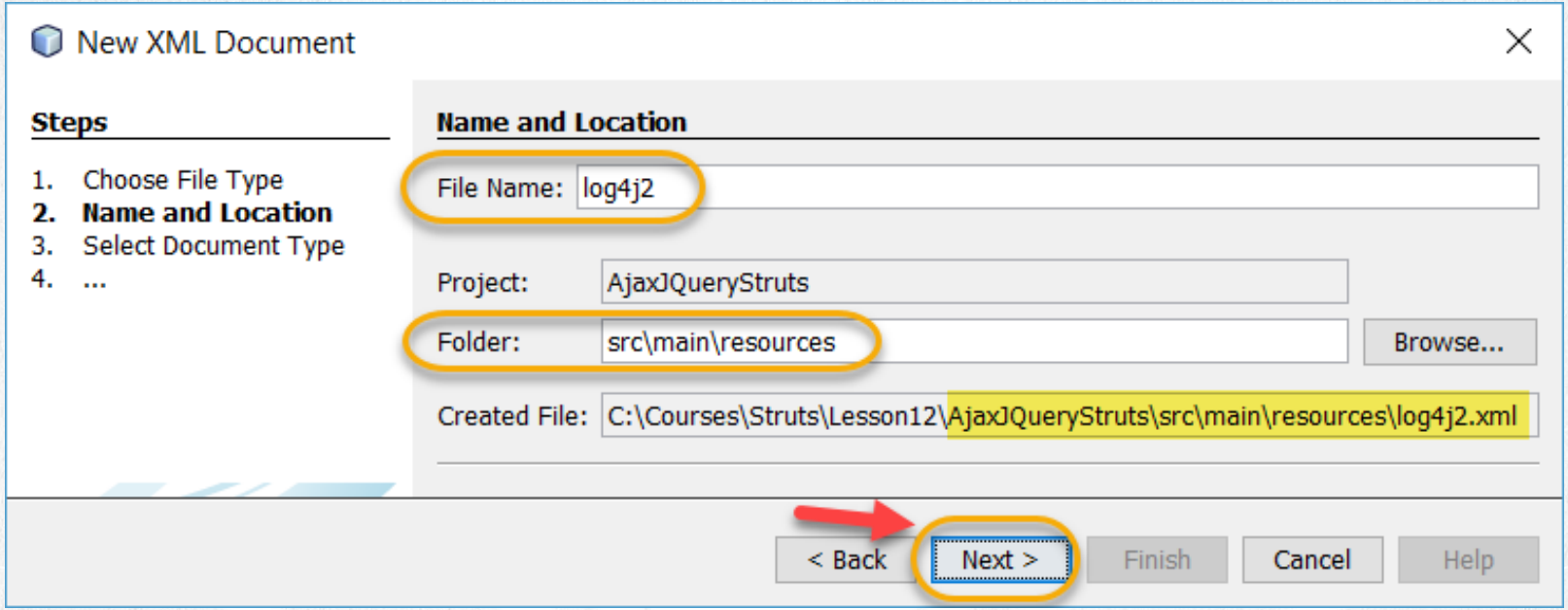
17. CREATE THE LOG4J2.XML FILE

- We create the log4j2.xml file as follows:



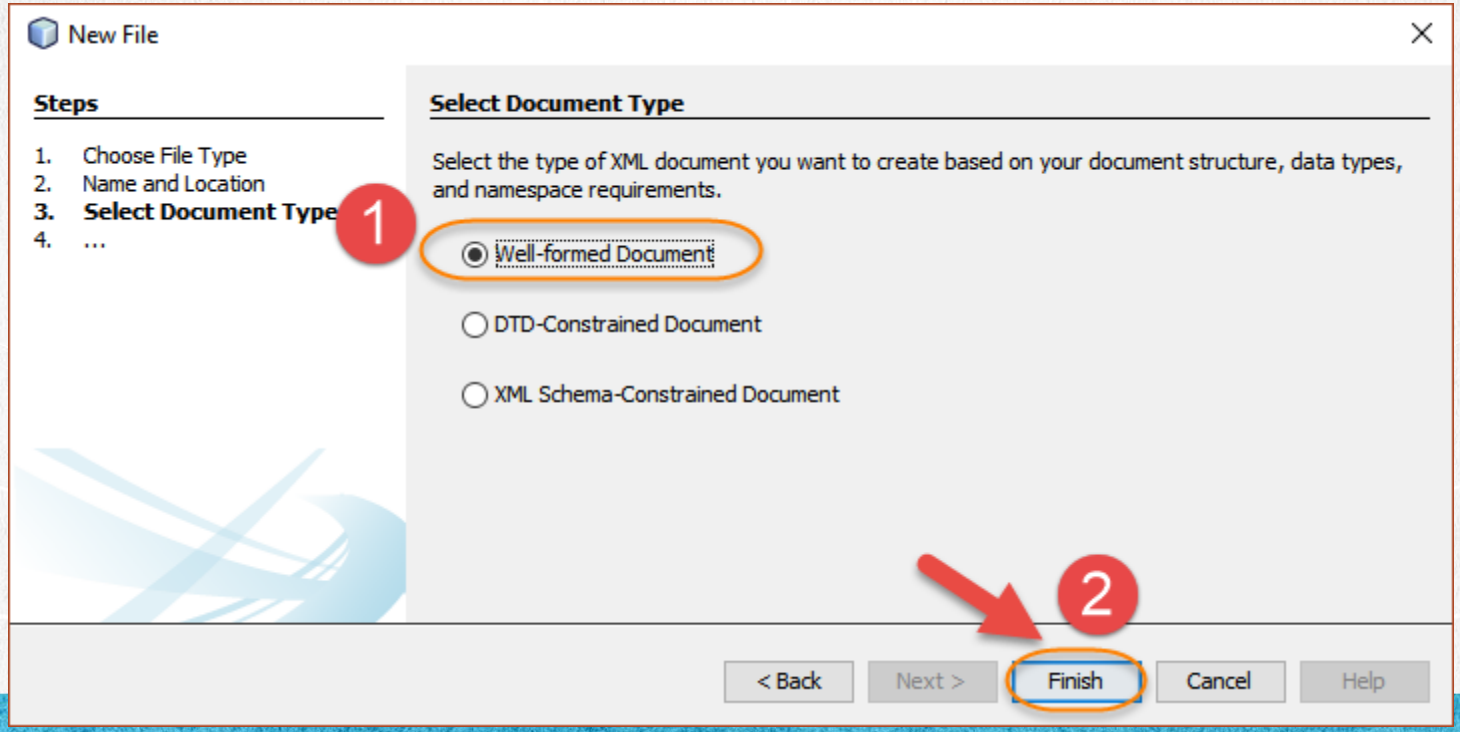
17. CREATE THE LOG4J2.XML FILE

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



17. CREATE THE LOG4J2.XML FILE

- We select the option shown:



18. MODIFY THE CODE

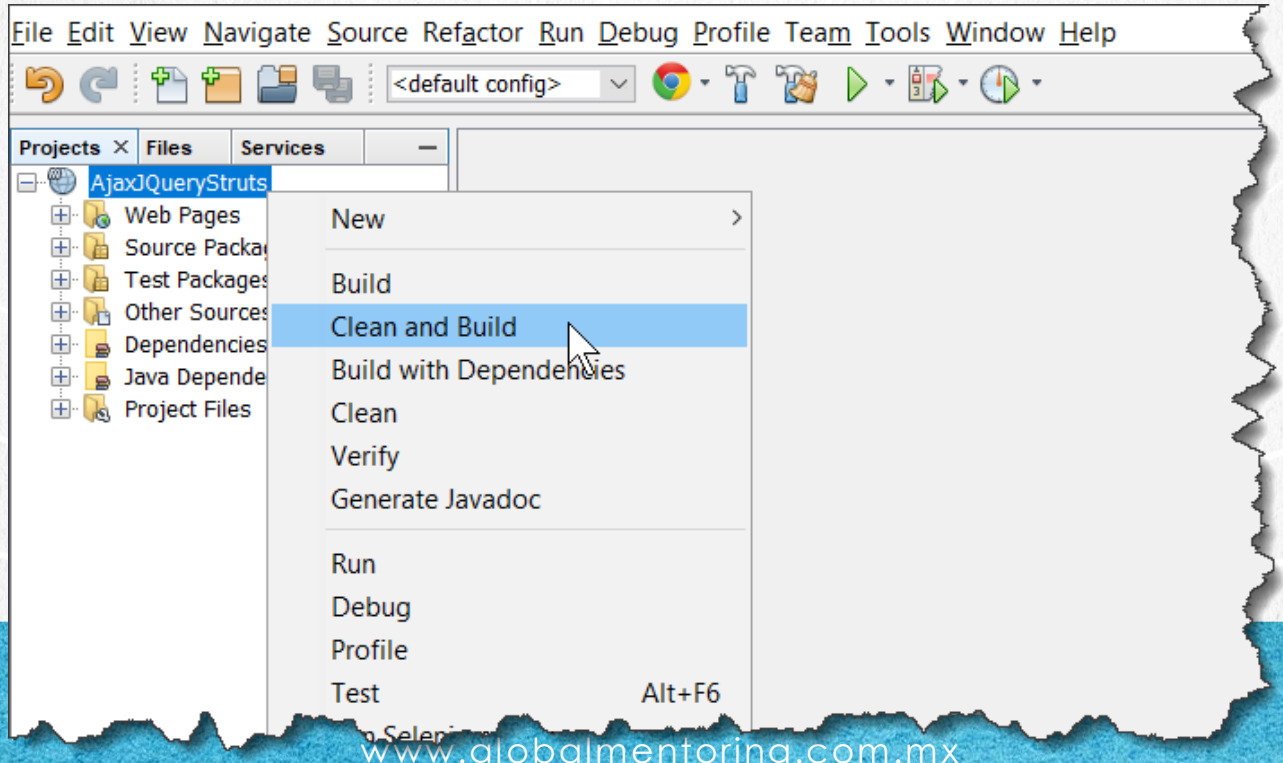
log4j2.xml:

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>
```

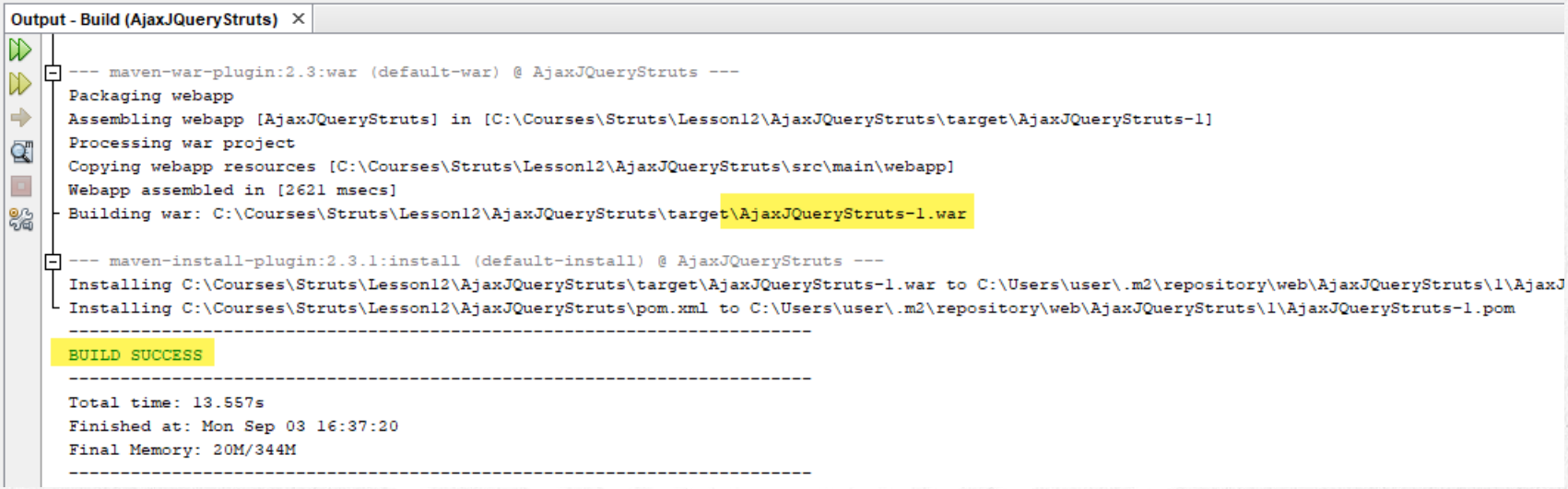
19. EXECUTE CLEAN & BUILD

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



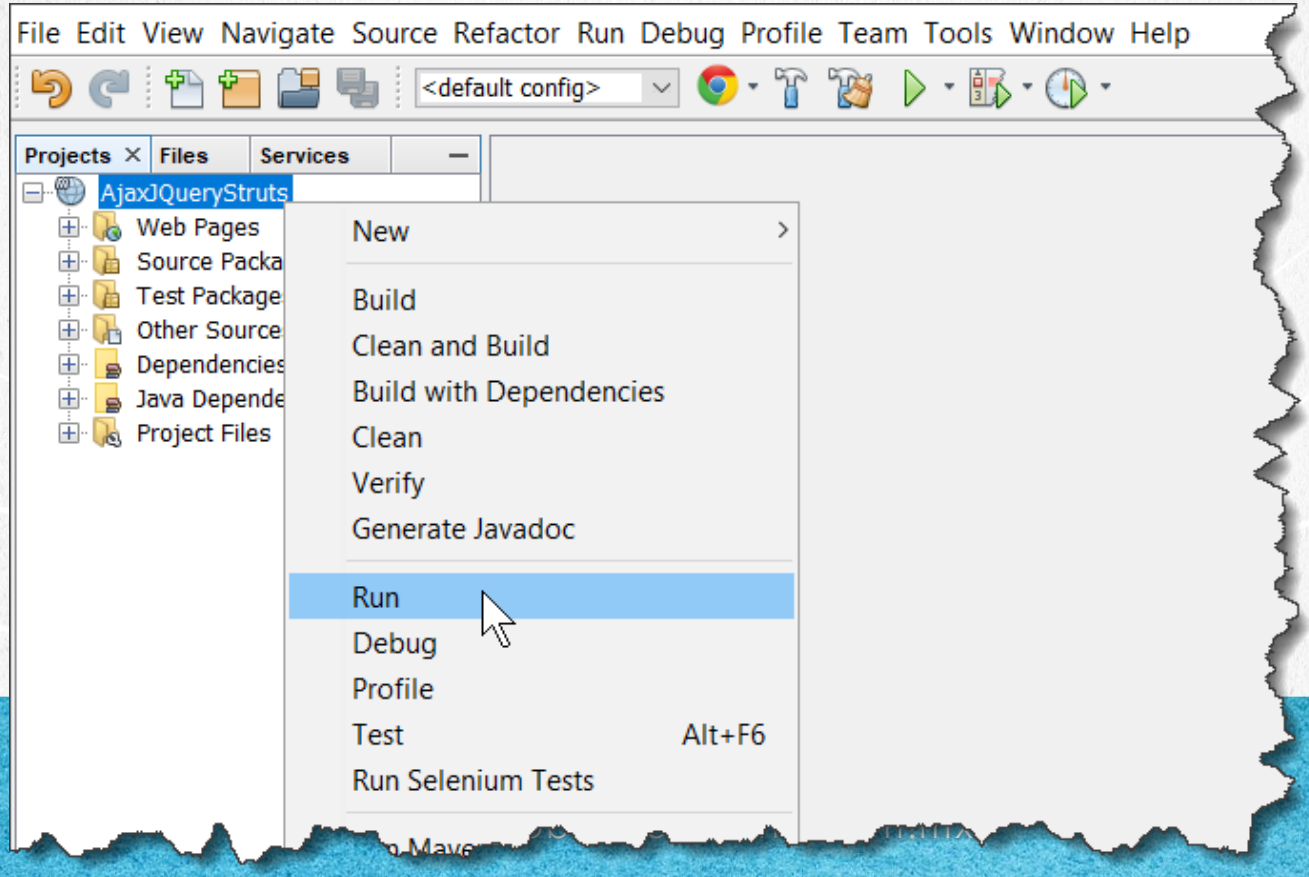
19. EXECUTE CLEAN & BUILD

- We must observe a result similar to the following:



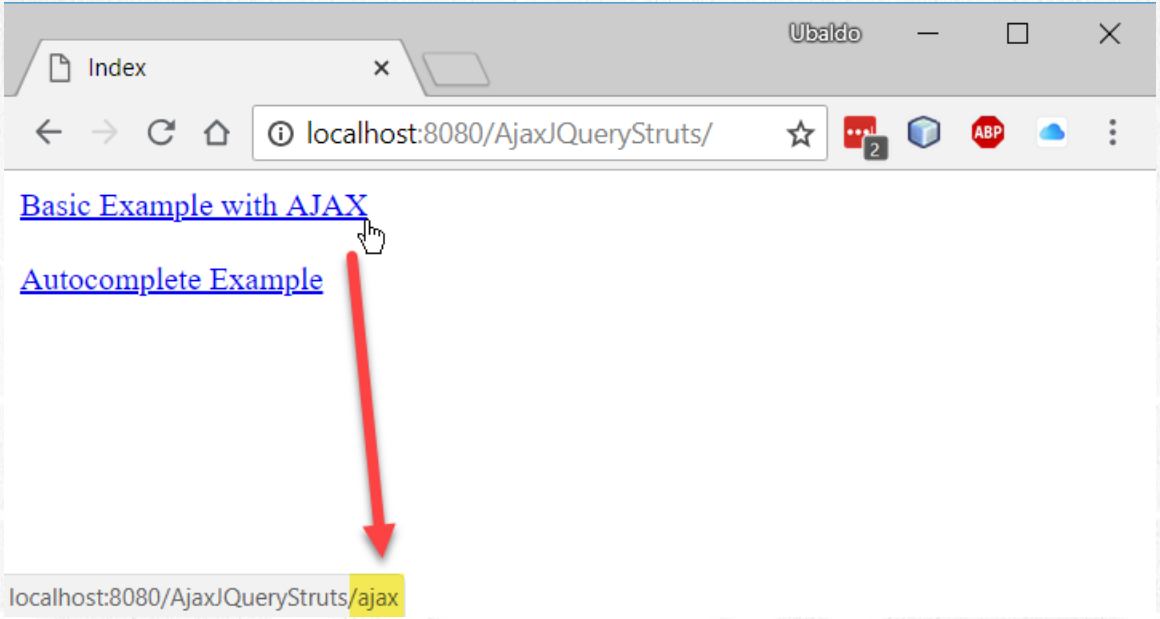
20. EXECUTE THE APPLICATION

- We run the application as follows:



20. EXECUTE THE APPLICATION

- We enter the first option called: Basic Ajax Example:

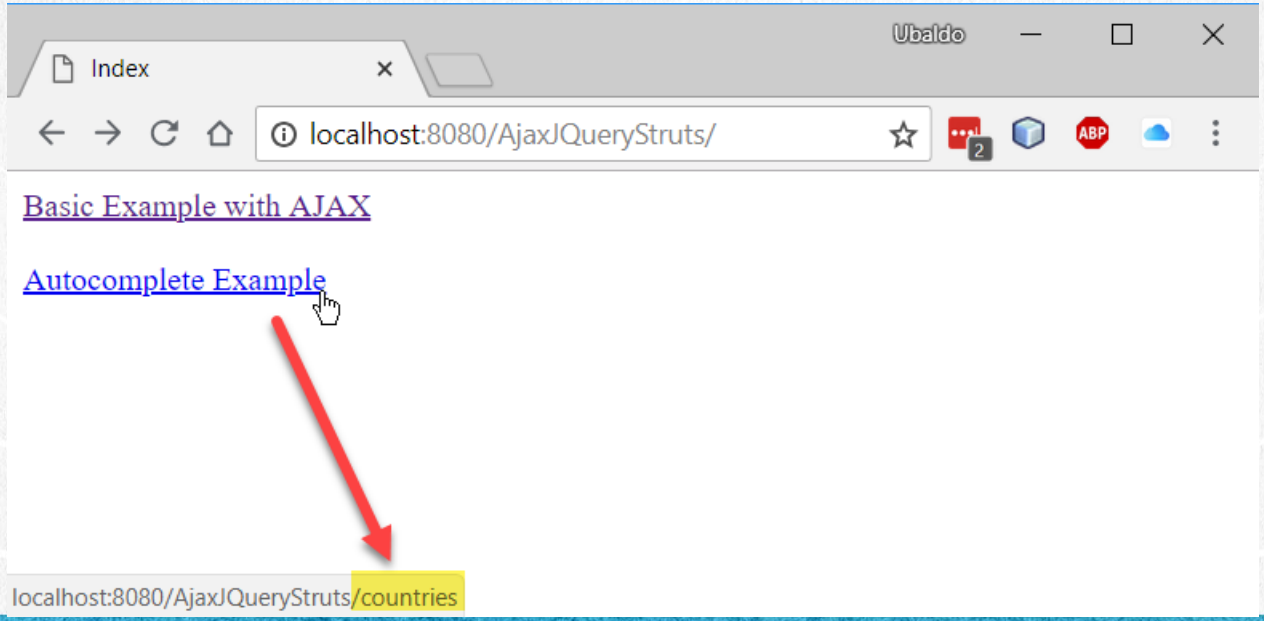


20. EXECUTE THE APPLICATION

- We enter the first option called: Basic Ajax Example:

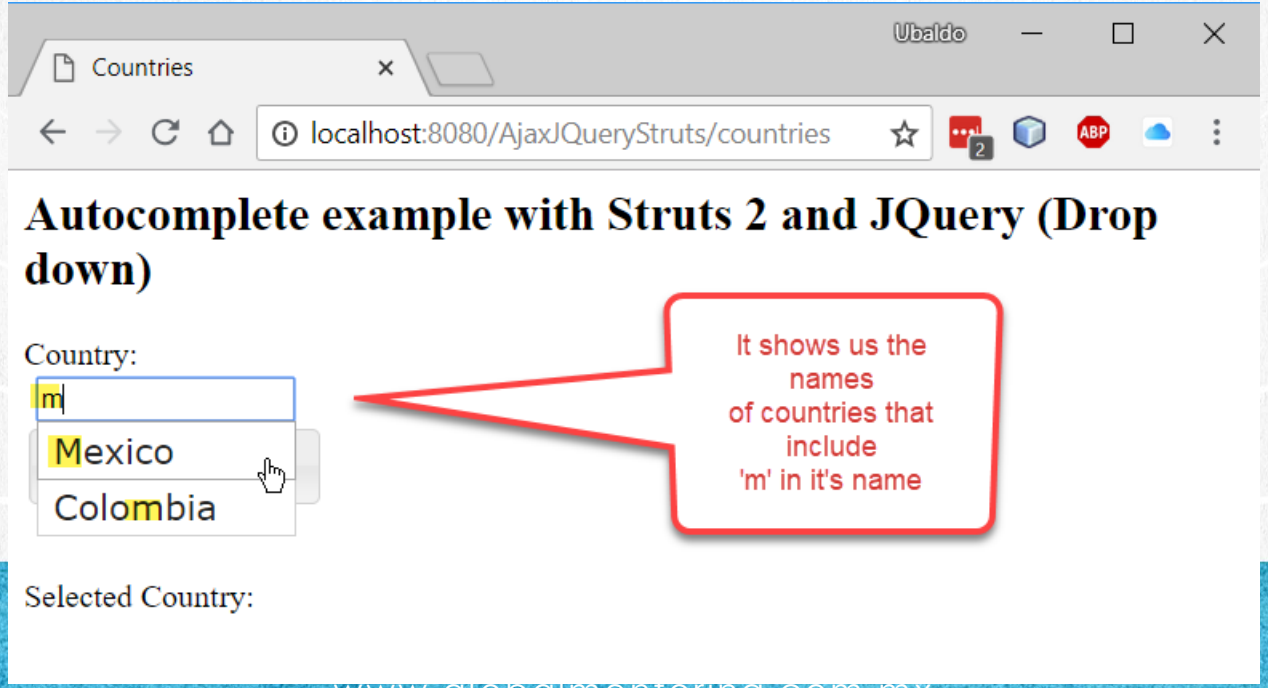
20. EXECUTE THE APPLICATION

- Return to index and enter the second option called: Autocomplete Example:



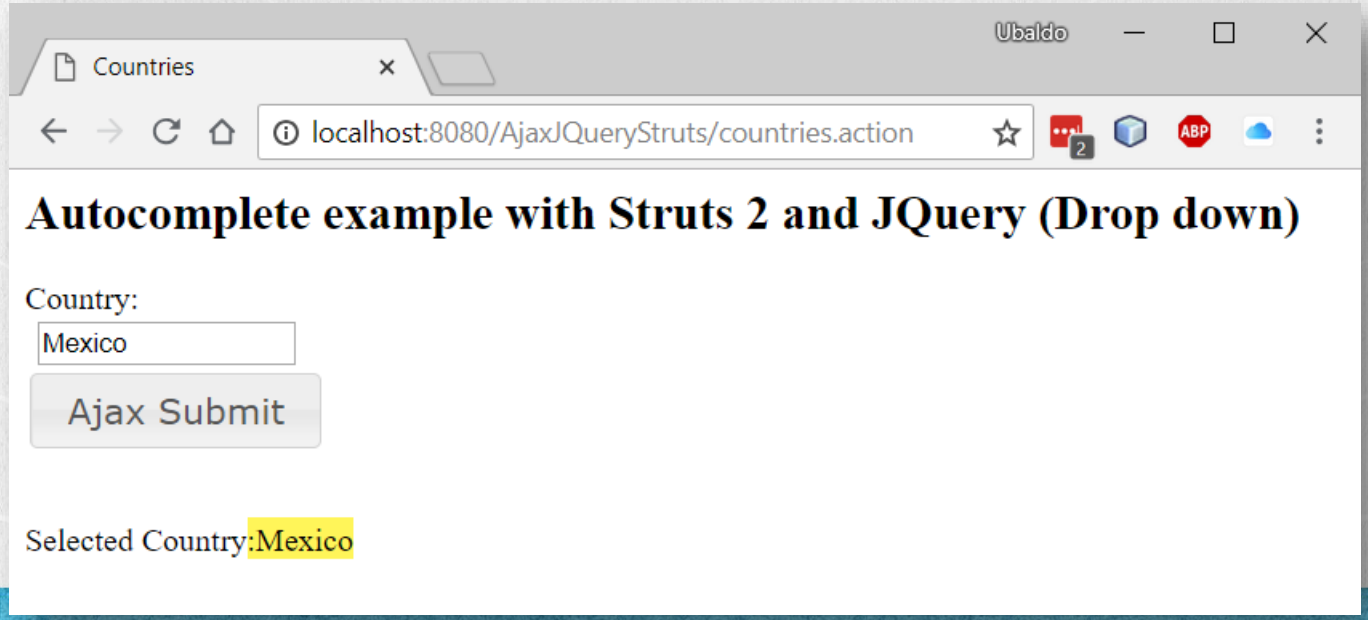
20. EXECUTE THE APPLICATION

- We write a letter or a set of letters, and automatically autocomplete us with the list of countries that include this set of characters. This list of countries is obtained asynchronously using AJAX:



20. EXECUTE THE APPLICATION

- Once the respective country has been selected, we will send the name of the country, and it will display the selected country via AJAX.



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

CONCLUSION OF THE EXERCISE

With this exercise we put the concept of AJAX with the help of the jquery and Struts2 plug-in.

The theme of AJAX is quite extensive, but with the examples we saw and put into practice, it is a very good start to understand how Struts 2 applies the AJAX concept.

There are other technologies that implement AJAX in combination with Struts 2, such as DOJO Toolkit, however, several of these technologies are considered obsolete, so we have only explained the technologies that are still used today, however any other technology that you want to use is very similar to implement as we saw with jquery and Struts 2.

There are many more components that you can use, which you can check in the following link:

http://struts.jgeppert.com/struts2-jquery-showcase/index.action

With this we conclude the AJAX theme in Struts 2.

ONLINE COURSE

STRUTS 2 FRAMEWORK

By: Eng. Ubaldo Acosta