STRUTS FRAMEWORK COURSE

OGNL WITH STRUTS 2 FRAMEWORK

By the expert: Ubaldo Acosta

Global Mentoring

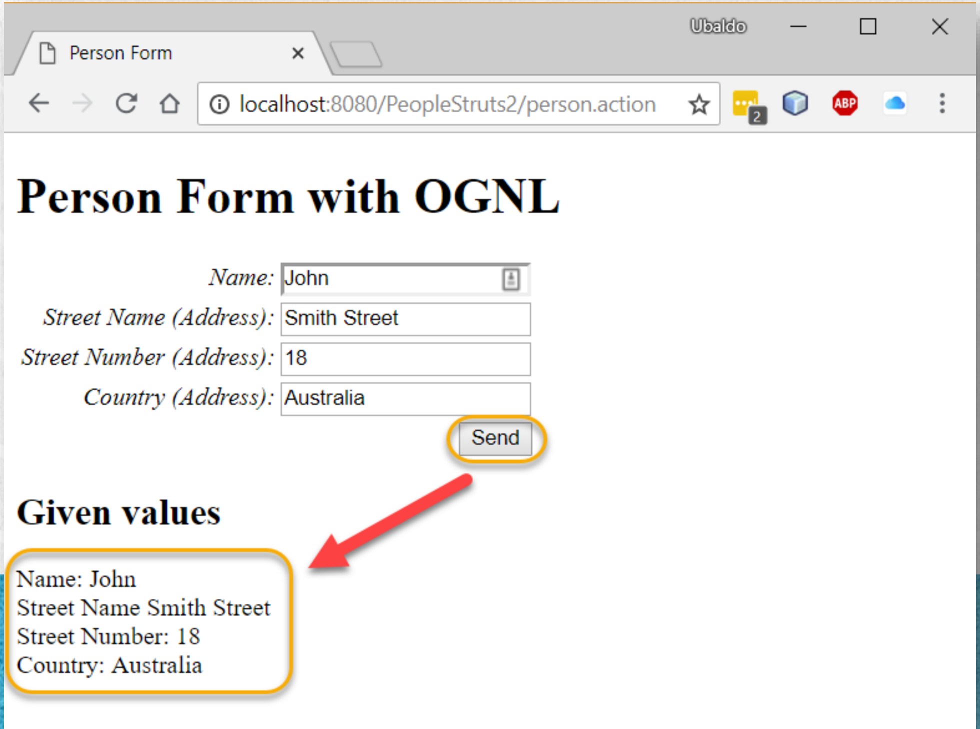
JAVA ONLINE COURSES UNIVERSITY

STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

EXERCISE OBJECTIVE

Create an application to implement the use of OGNL (Object-Graph Navigation Language) with Struts2. At the end we should observe the following:



Person Form with OGNL

Name:	John
Street Name (Address):	Smith Street
Street Number (Address):	18
Country (Address):	Australia

Given values

Name: John
Street Name Smith Street
Street Number: 18
Country: Australia

EXERCISE REQUIREMENT

In this project we are going to put into practice the concept of OGNL (Object-Graph Navigation Language).

To do this we are going to create an object called Person that will have the following properties:

- String name
- Address address

And we will create an object called Address that will have the following properties:

- String streetName;
- int streetNumber;
- String country

EXERCISE REQUIREMENT

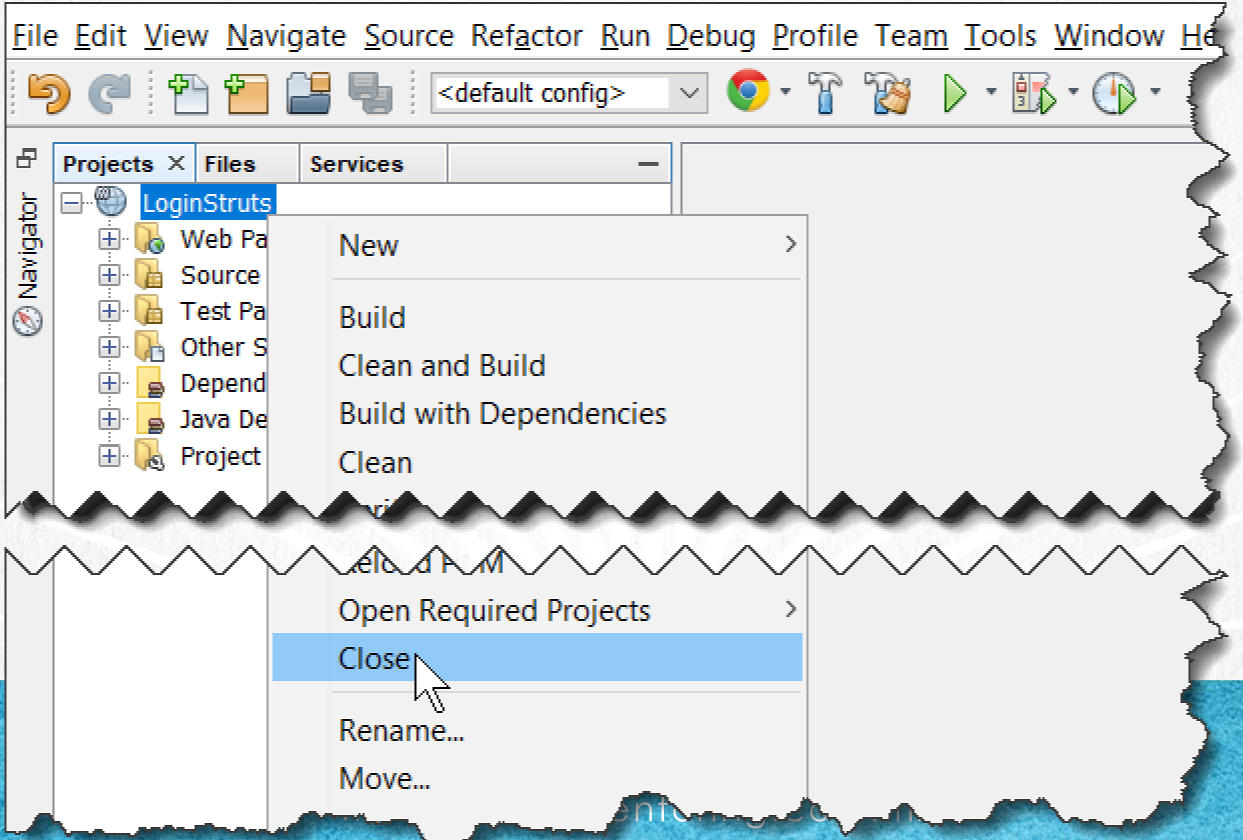
Create an application that does the following:

- The application will be called: PeopleStruts2
- It must allow capturing the attributes of both the Person class and the Address class using the Struts 2 OGNL concept.
- We will apply several of the topics seen so far. However, we will leave this exercise as simple as possible to focus on the use and application of OGNL. Later, more features can be applied to the project to add concepts such as validation, error handling, or any other characteristic seen so far if desired.

Global Mentoring

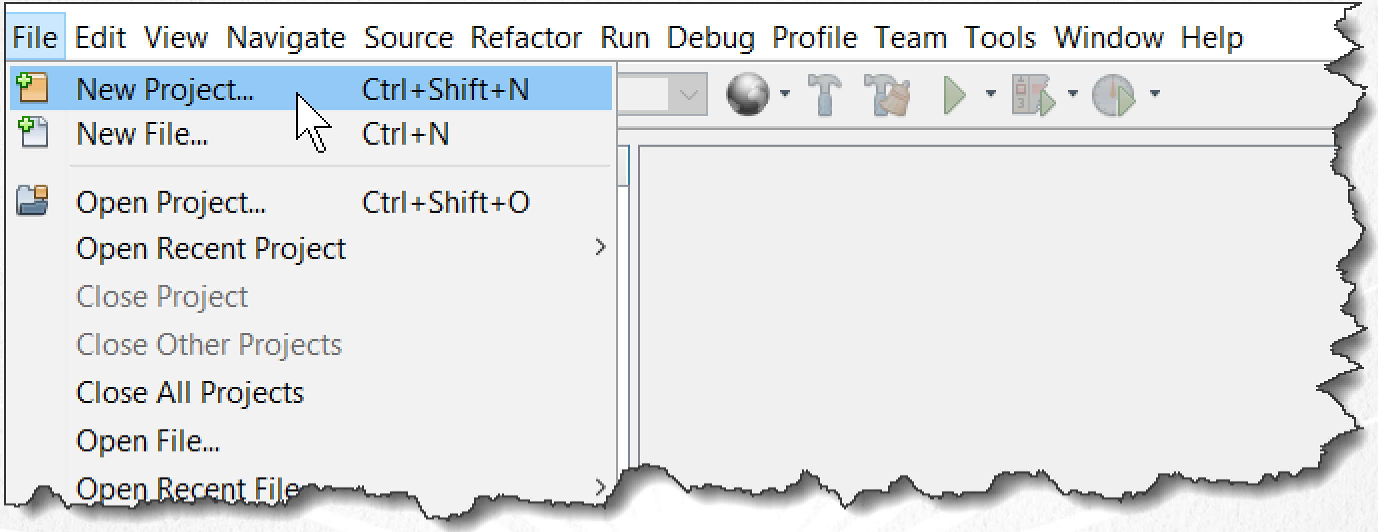
CLOSE PROJECTS THAT WE NO LONGER USE

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



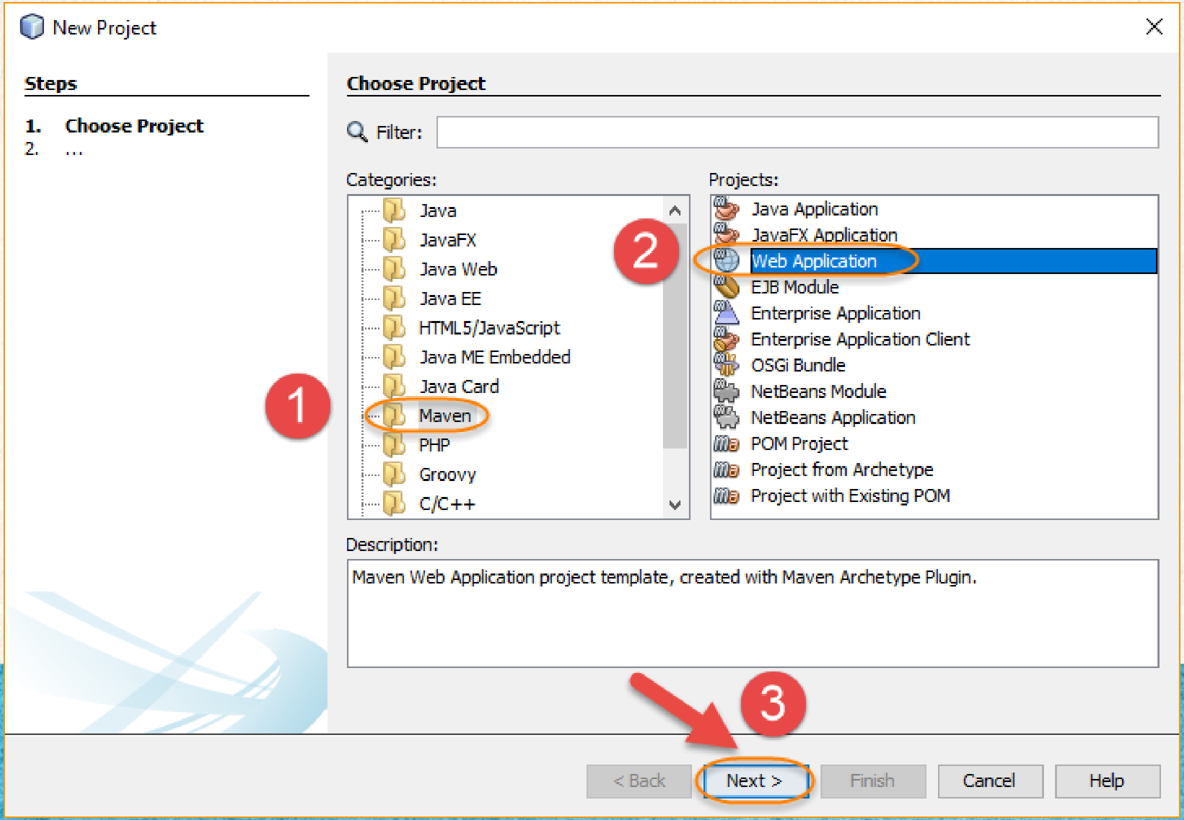
1. CREATE A NEW PROJECT

- We create the new project as shown below:


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
New Project... Ctrl+Shift+N
New File... Ctrl+N
Open Project... Ctrl+Shift+O
Open Recent Project
Close Project
Close Other Projects
Close All Projects
Open File...
Open Recent File

1. CREATE A NEW PROJECT

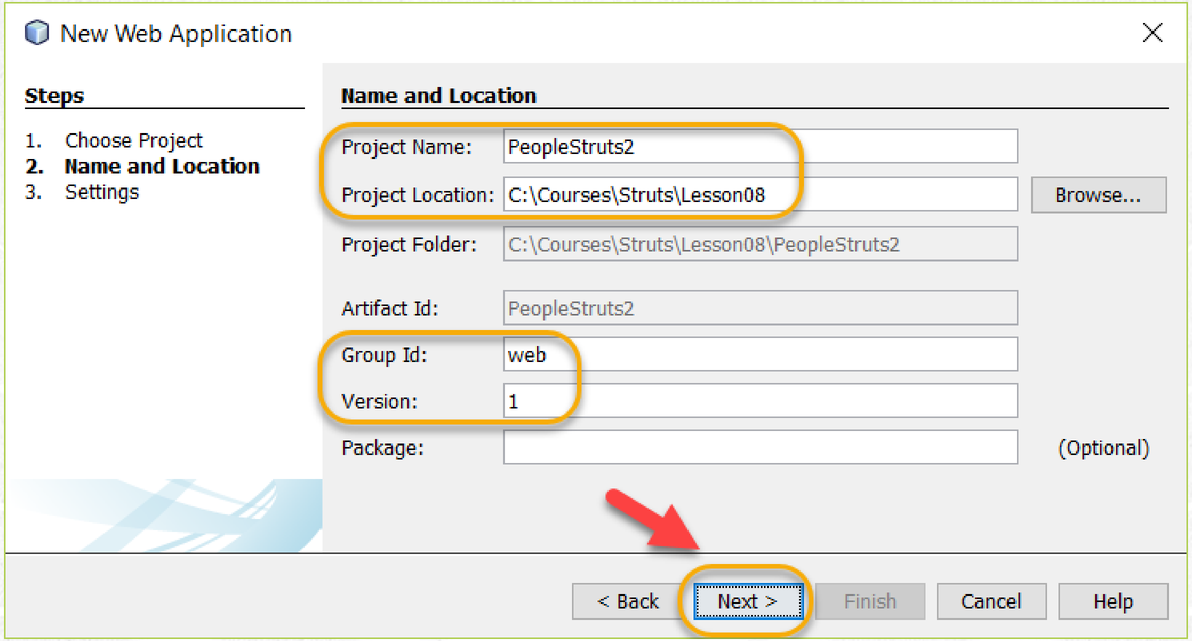
- We create the new project as shown below:



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: PeopleStruts2

Project Location: C:\Courses\Struts\Lesson08

Project Folder: C:\Courses\Struts\Lesson08\PeopleStruts2

Artifact Id: PeopleStruts2

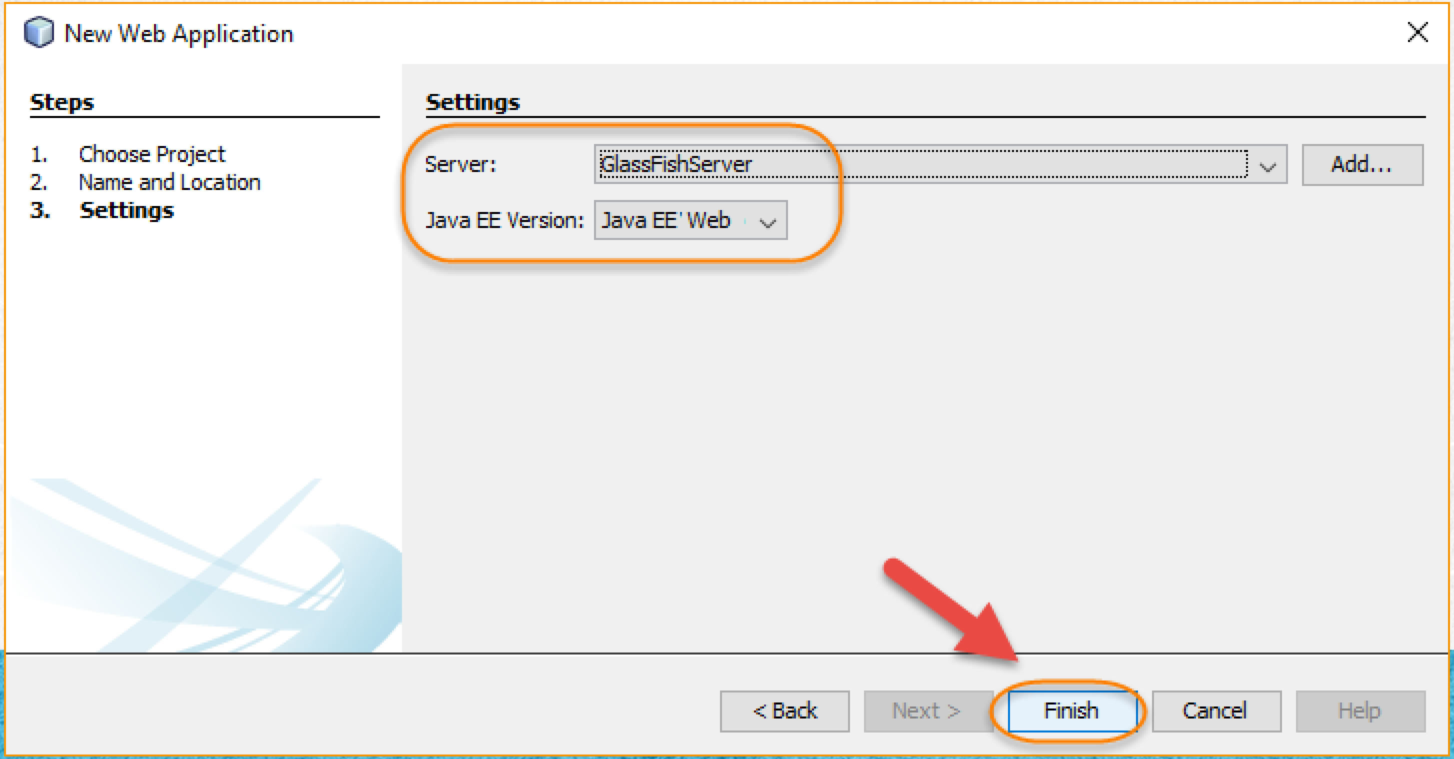
Group Id: web

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**

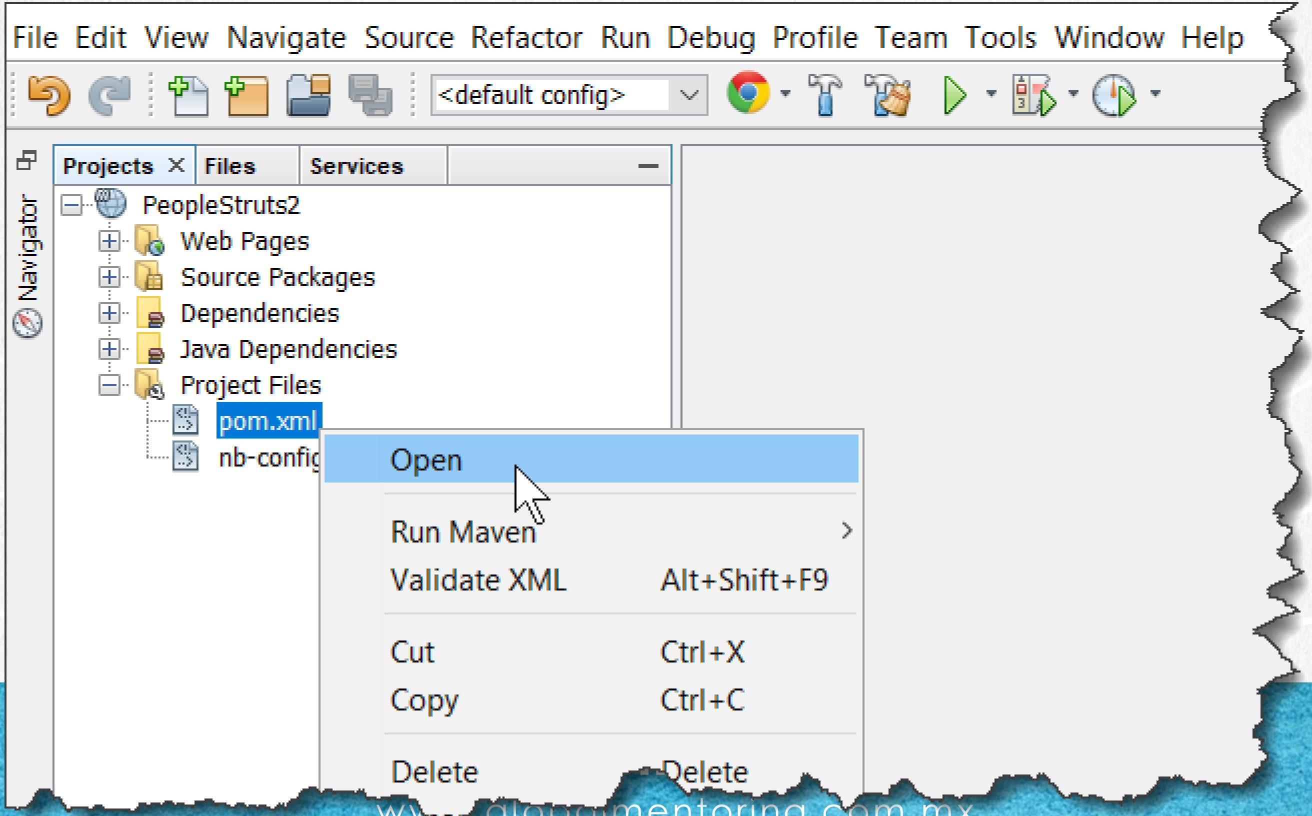
Settings

Server: GlassFishServer

Java EE Version: Java EE' Web

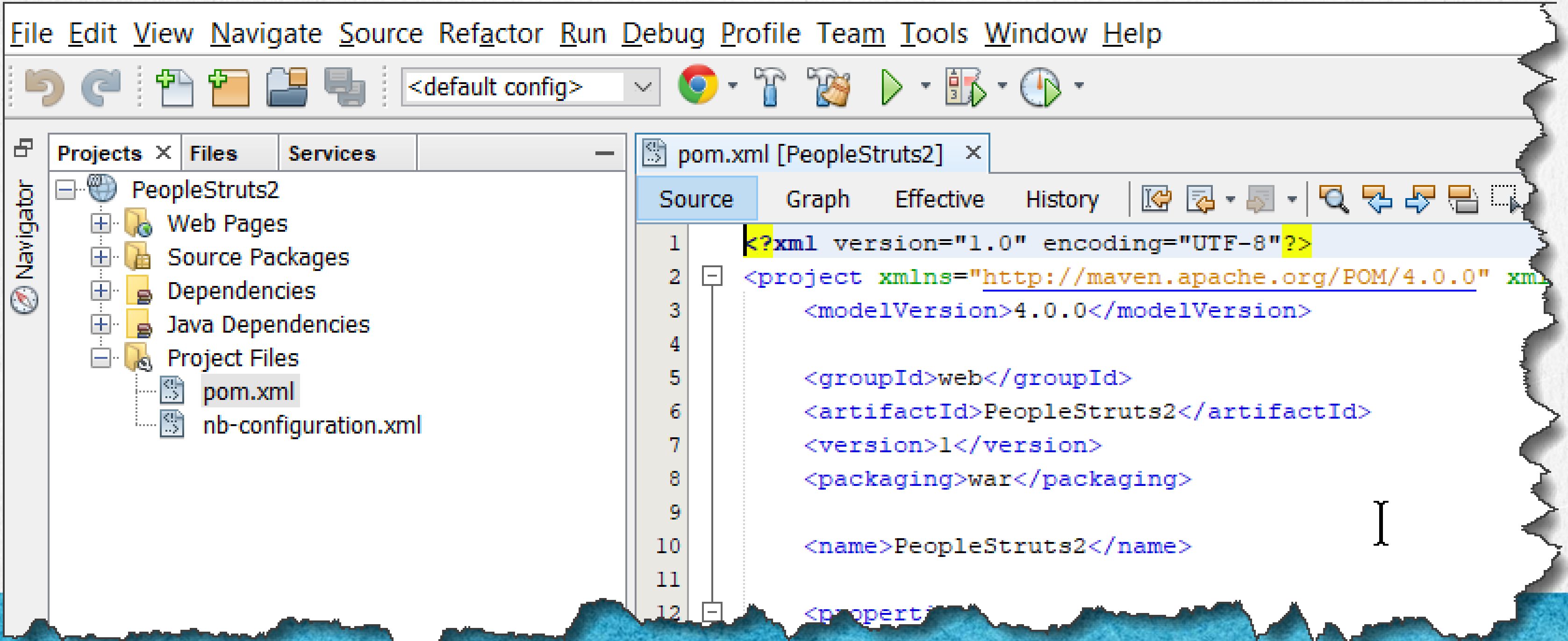
2. Open Maven's pom.xml file

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



2. OPEN MAVEN'S POM.XML FILE

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



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

    <groupId>web</groupId>
    <artifactId>PeopleStruts2</artifactId>
    <version>1</version>
    <packaging>war</packaging>

    <name>PeopleStruts2</name>

    <propert
```

3. MODIFY THE CODE

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>PeopleStruts2</artifactId>
    <version>1</version>
    <packaging>war</packaging>

    <name>PeopleStruts2</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

```xml
        <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:

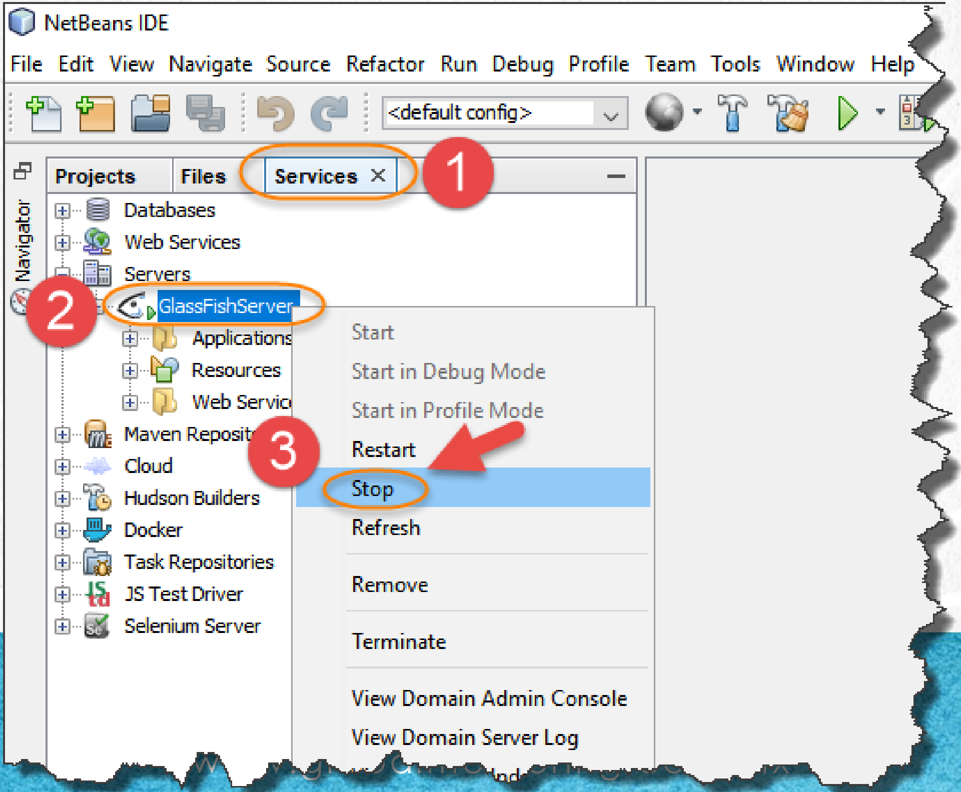
Click to download

```
    <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>
```

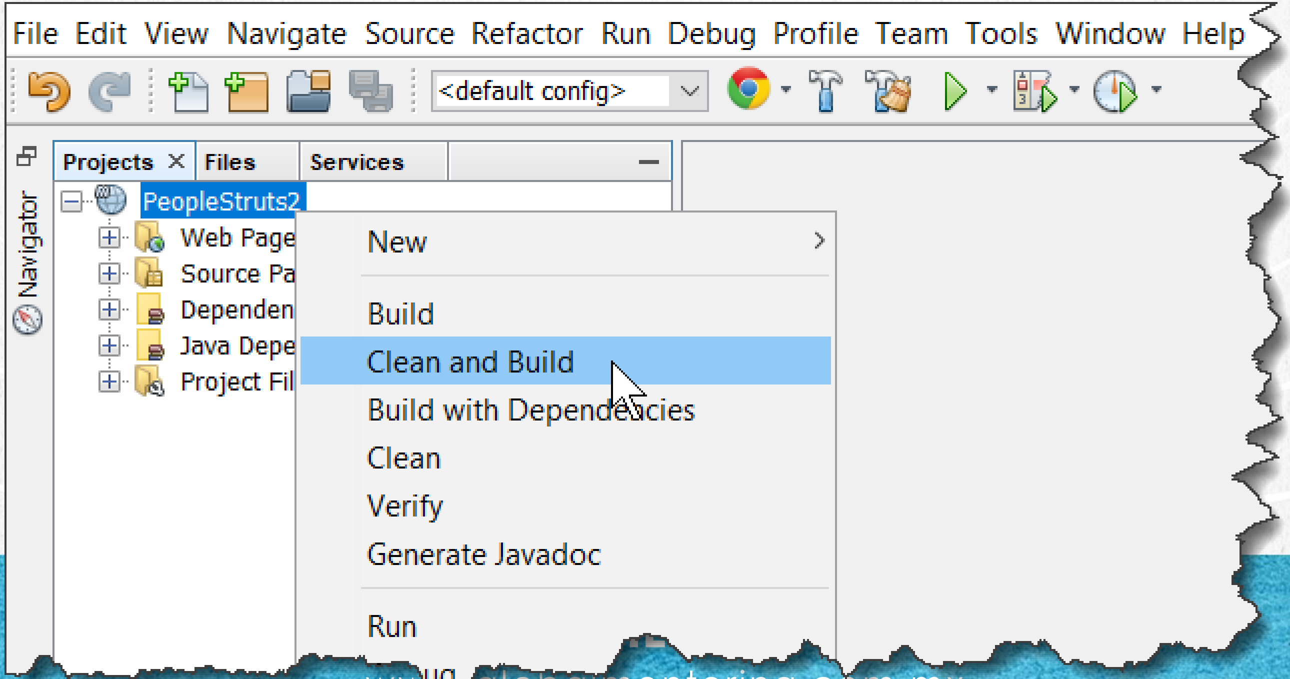
4. STOP GLASSFISH IF IT WAS STARTED

- 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:



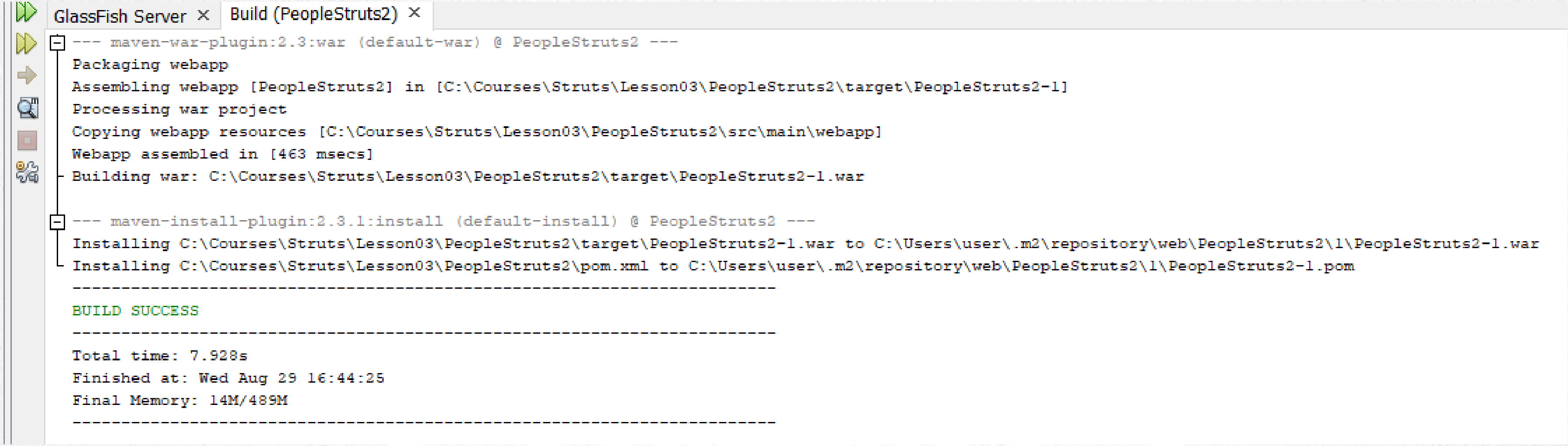
5. EXECUTE CLEAN & BUILD

- 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:



```
--- maven-war-plugin:2.3:war (default-war) @ PeopleStruts2 ---
Packaging webapp
Assembling webapp [PeopleStruts2] in [C:\Courses\Struts\Lesson03\PeopleStruts2\target\PeopleStruts2-1]
Processing war project
Copying webapp resources [C:\Courses\Struts\Lesson03\PeopleStruts2\src\main\webapp]
Webapp assembled in [463 msecs]
Building war: C:\Courses\Struts\Lesson03\PeopleStruts2\target\PeopleStruts2-1.war

--- maven-install-plugin:2.3.1:install (default-install) @ PeopleStruts2 ---
Installing C:\Courses\Struts\Lesson03\PeopleStruts2\target\PeopleStruts2-1.war to C:\Users\user\.m2\repository\web\PeopleStruts2\1\PeopleStruts2-1.war
Installing C:\Courses\Struts\Lesson03\PeopleStruts2\pom.xml to C:\Users\user\.m2\repository\web\PeopleStruts2\1\PeopleStruts2-1.pom
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 7.928s
Finished at: Wed Aug 29 16:44:25
Final Memory: 14M/489M
------------------------------------------------------------------------
```

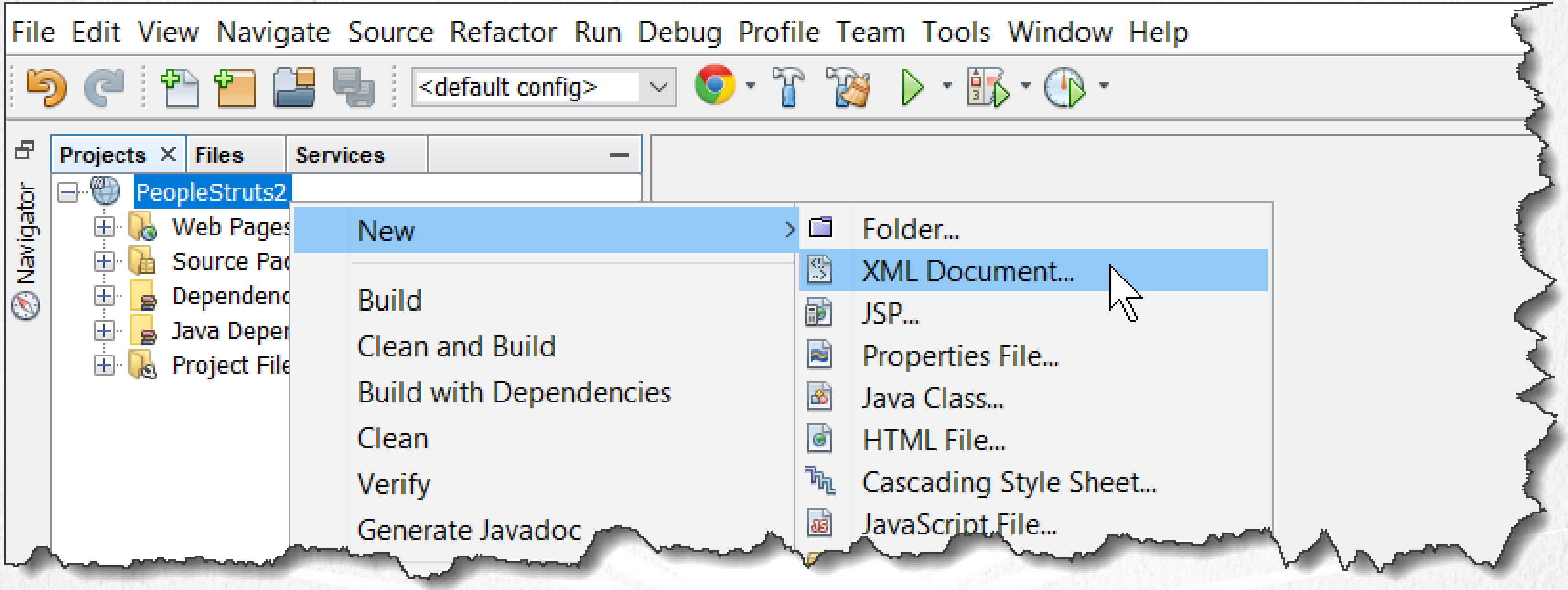
6. Create an XML file

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.

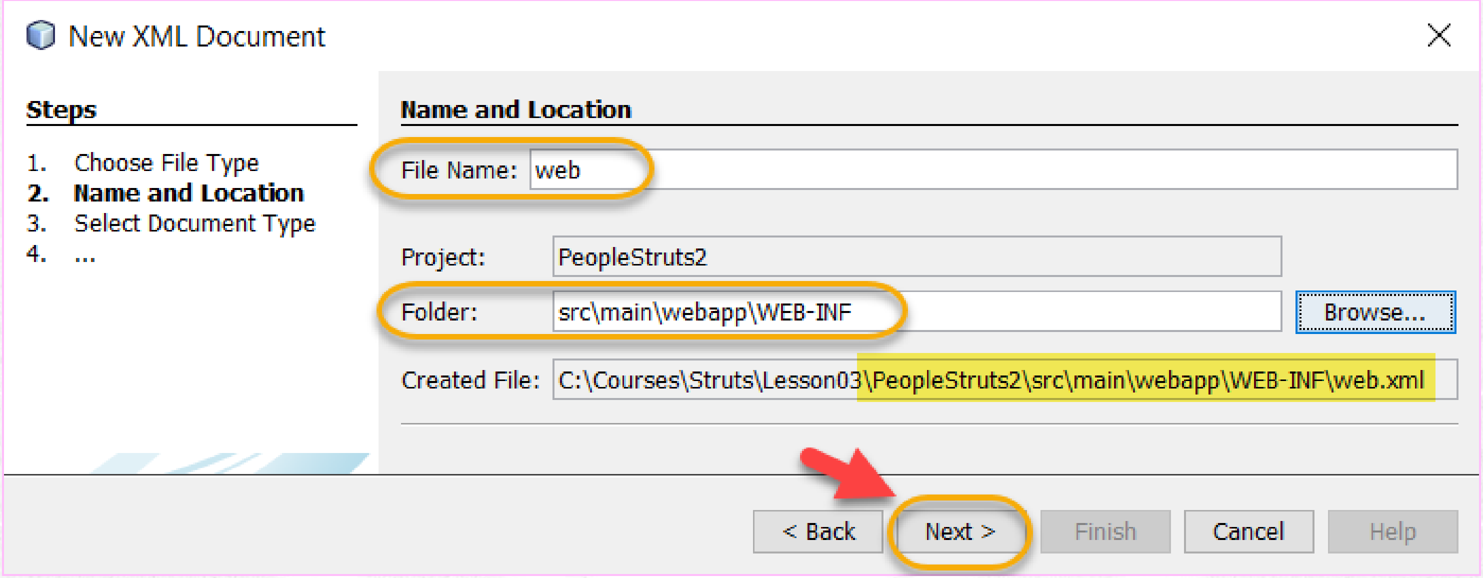
6. CREATE AN XML FILE

- 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

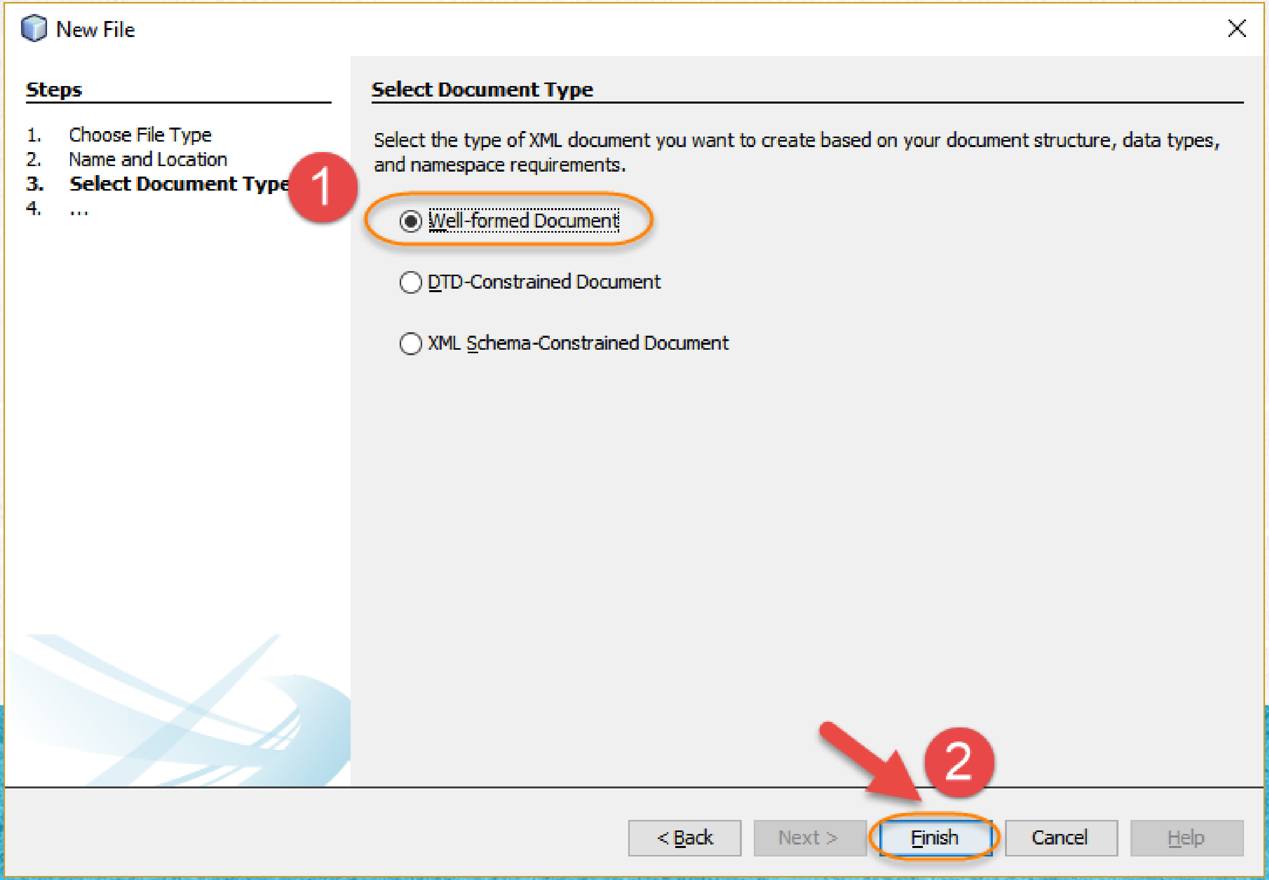
Project: PeopleStruts2

Folder: src\main\webapp\WEB-INF

Created File: C:\Courses\Struts\Lesson03\PeopleStruts2\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

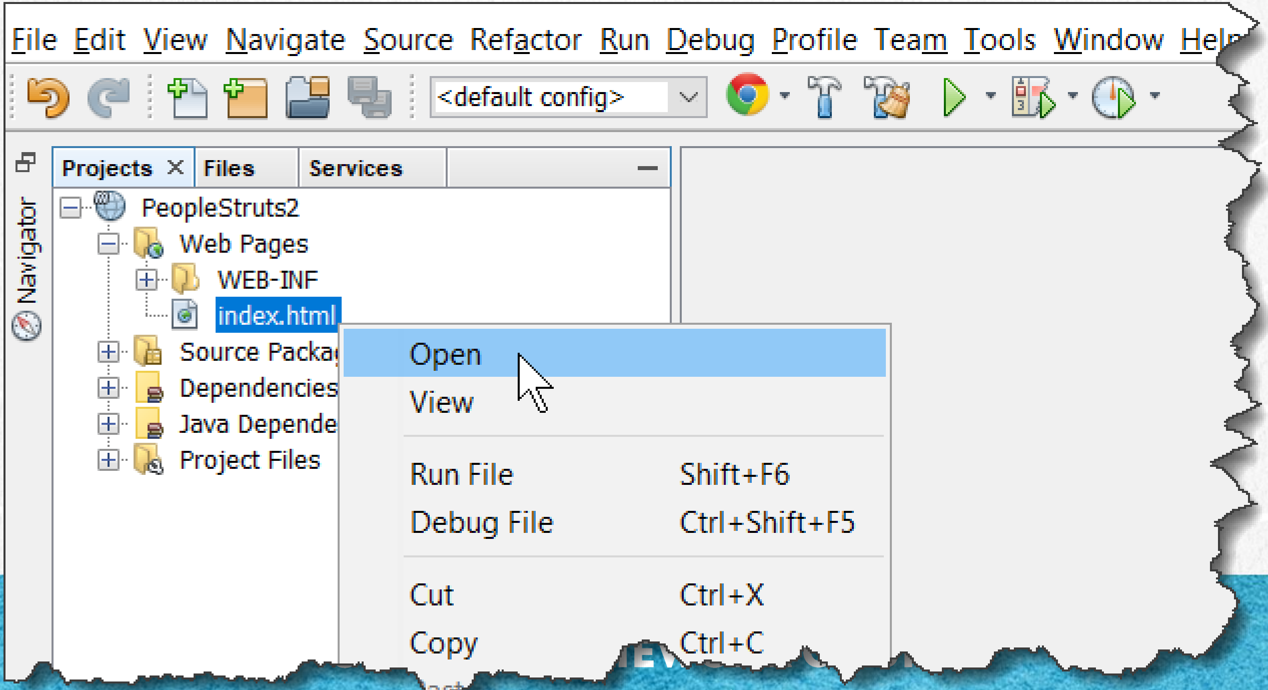
In automatic the IDE adds a file called index.html. However, if this file is not created we must add it to the project at the root level of Web Pages.

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: person

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

9. CREATE A JAVA CLASS

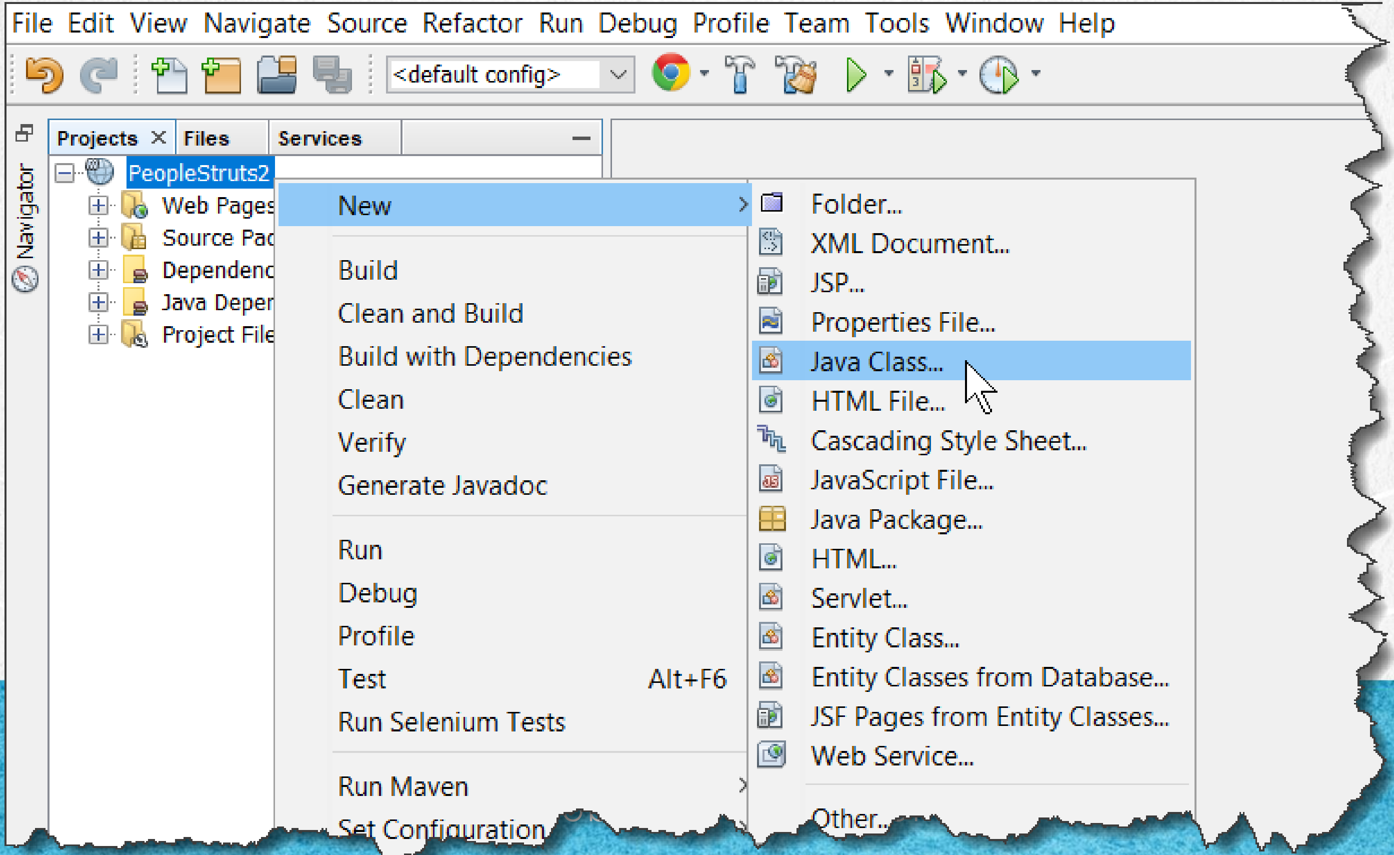
We are going to create a class called Address, which will be associated later with the Person class and will contain some attributes of a person's address.

We will use the ONGL technology to access this information later from the respective JSP view.

Let's see how our class Address.java is

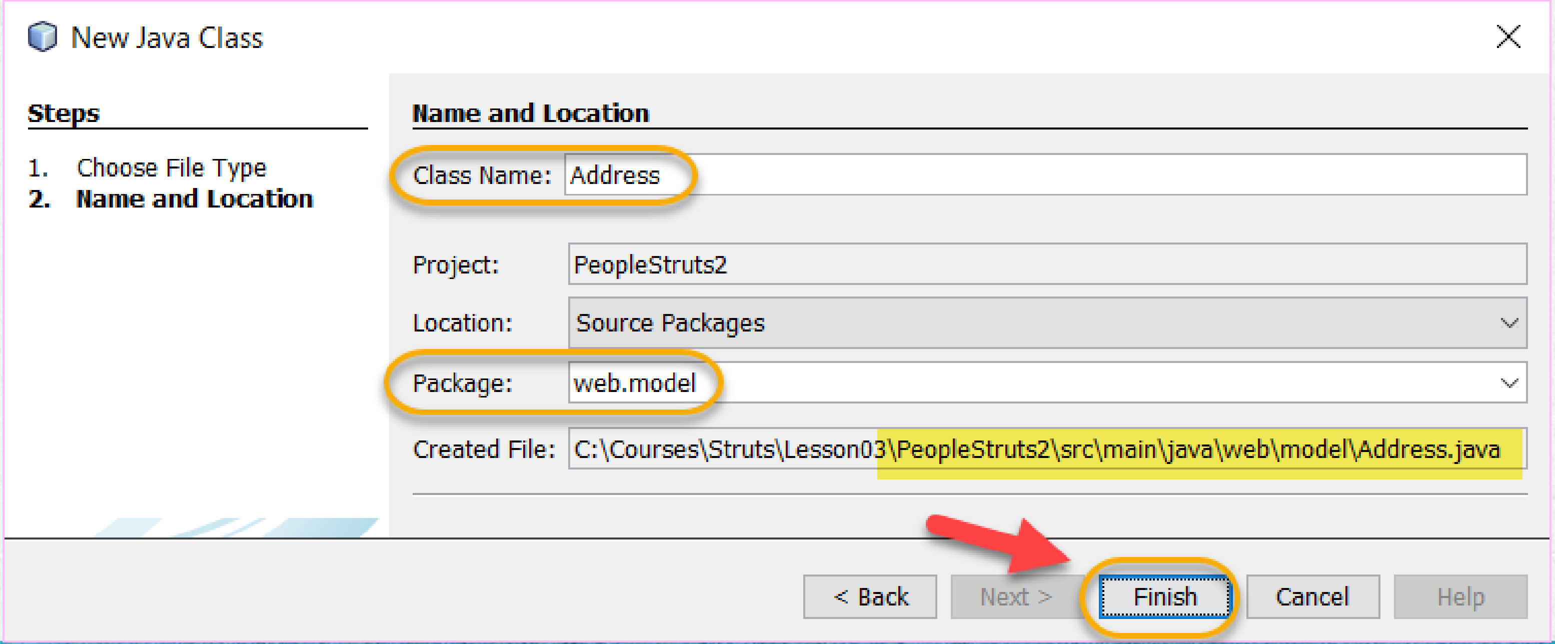
9. Create a new class

- Create the Address.java class:



9. CREATE A NEW CLASS

- Create the Address.java class:



New Java Class

Steps

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

Name and Location

Class Name: Address

Project: PeopleStruts2

Location: Source Packages

Package: web.model

Created File: C:\Courses\Struts\Lesson03\PeopleStruts2\src\main\java\web\model\Address.java

10. MODIFY THE CODE

Address.java:

Click to download

```
package web.model;

public class Address {

    private String streetName;
    private int streetNumber;
    private String country;

    public String getStreetName() {
        return streetName;
    }

    public void setStreetName(String streetName) {
        this.streetName = streetName;
    }

    public int getStreetNumber() {
        return streetNumber;
    }

    public void setStreetNumber(int streetNumber) {
        this.streetNumber = streetNumber;
    }
```

10. MODIFY THE CODE

Address.java:

Click to download

```
    public String getCountry() {
        return country;
    }

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

    @Override
    public String toString() {
        return "Address{" + "streetName=" + streetName + ", streetNumber=" + streetNumber + ", 
country=" + country + '}';
    }

}
```

11. CREATE A JAVA CLASS

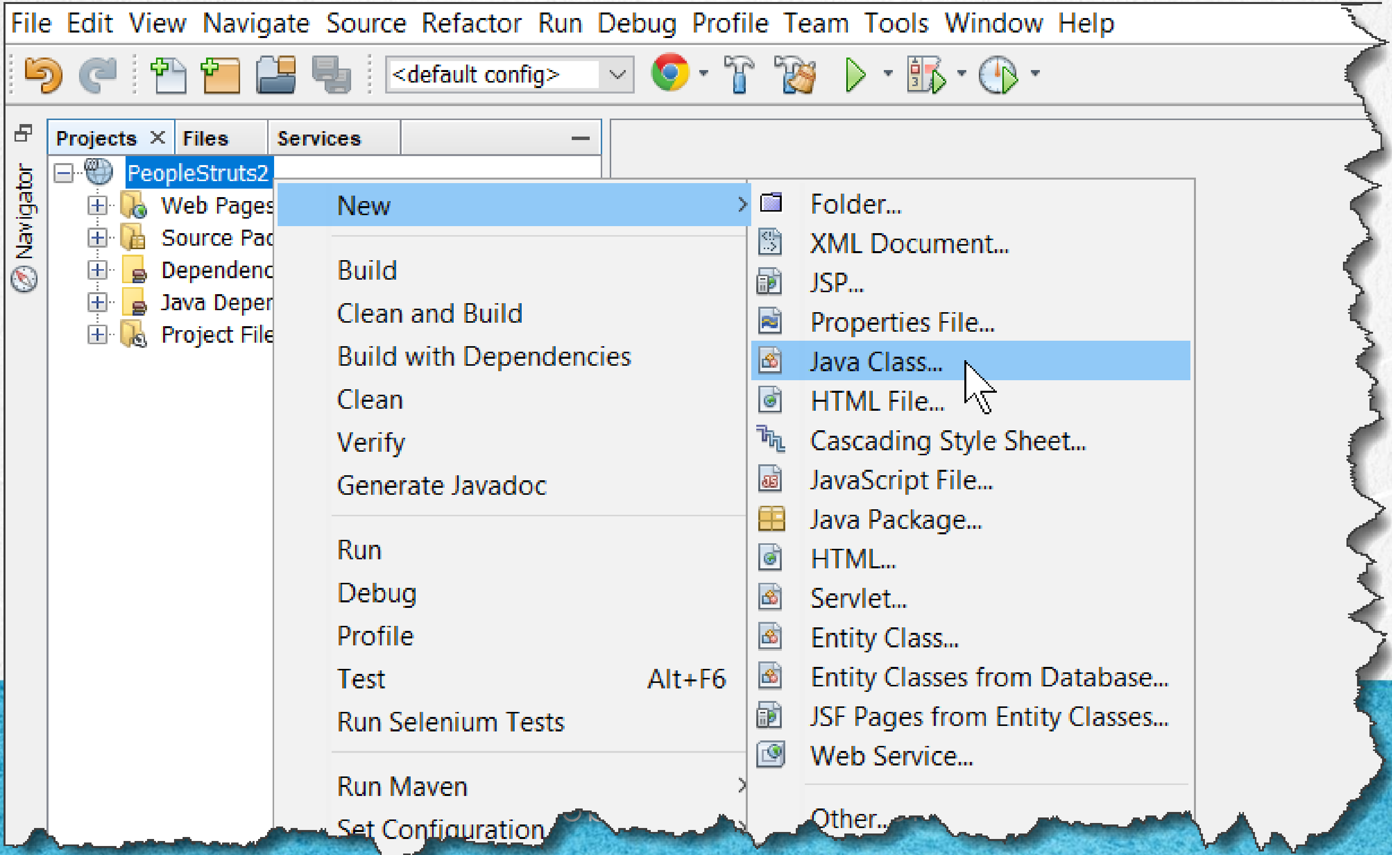
We are going to create a class called Person.java, it contains some attributes, including an attribute called address of type Address.

We will use the ONGL technology to access this information later from the respective JSP view.

Let's see how our class Person.java is

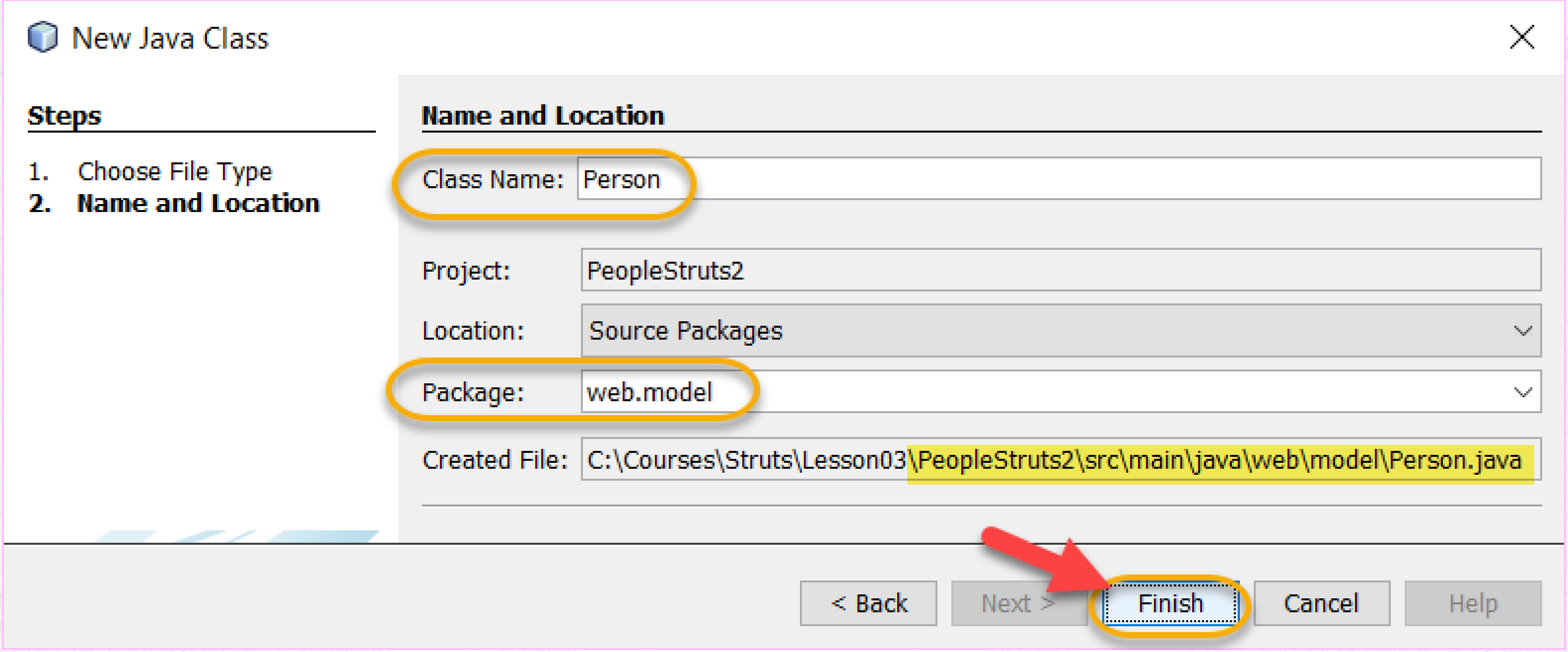
11. CREATE A NEW JAVA CLASS

- Create the Person.java class:



11. CREATE A NEW JAVA CLASS

- Create the Person.java class:



New Java Class

Steps

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

Name and Location

Class Name: Person

Project: PeopleStruts2

Location: Source Packages

Package: web.model

Created File: C:\Courses\Struts\Lesson03\PeopleStruts2\src\main\java\web\model\Person.java

11. MODIFY THE CODE

Person.java:

Click to download

```java
package web.model;

public class Person {
    private String name;
    private Address address;

    public String getName() {
        return name;
    }

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

    public Address getAddress() {
        return address;
    }

    public void setAddress(Address domicilio) {
        this.address = domicilio;
    }

    @Override
    public String toString() {
        return "Person{" + "name=" + name + ", address=" + address + '}';
    }
}
```

12. CREATE A NEW JAVA CLASS

The PersonAction.java class that we are going to create next will act as Controller (Action), but not as a model, since the Model (Bean) will be the Person and Address classes that we created previously.

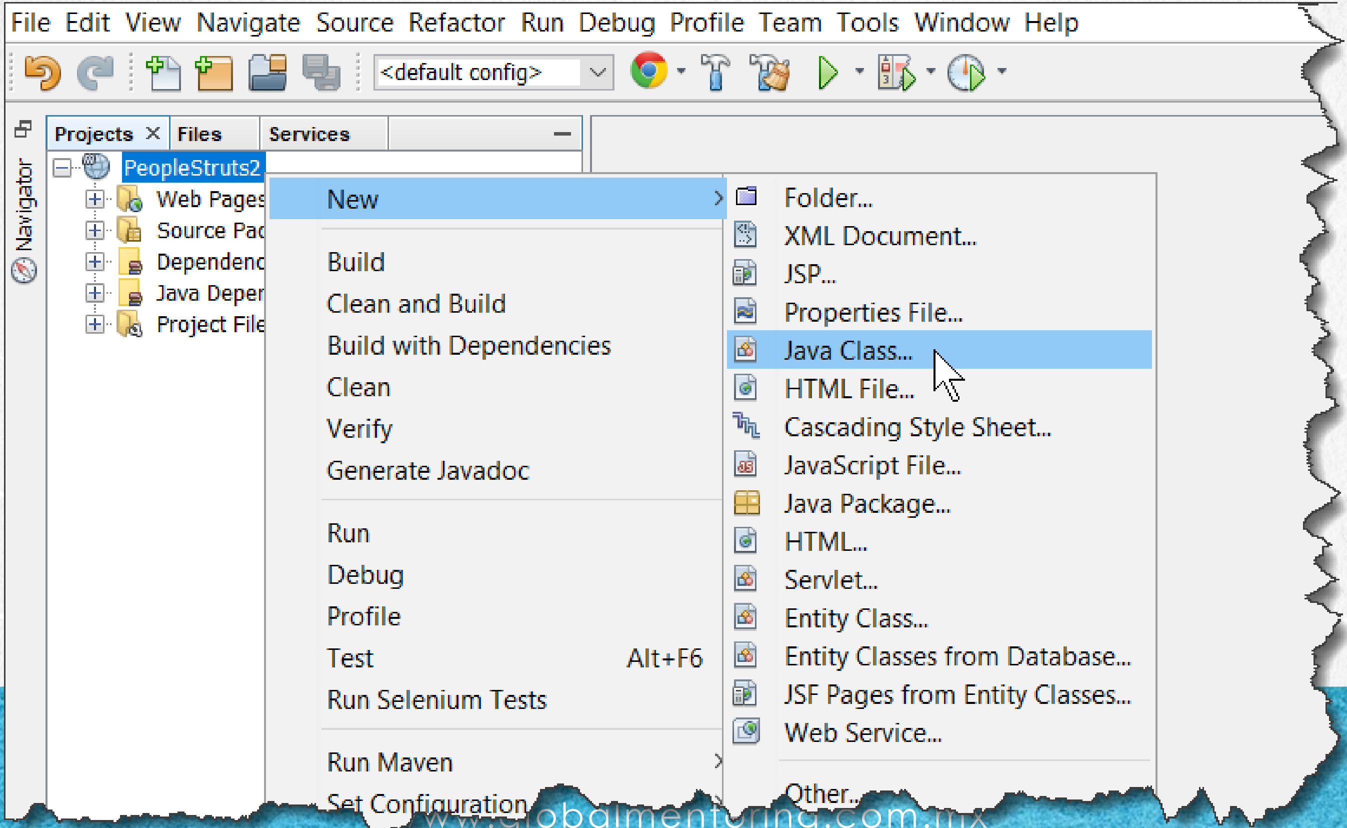
We will extend the ActionSupport class and overwrite the execute method.

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.

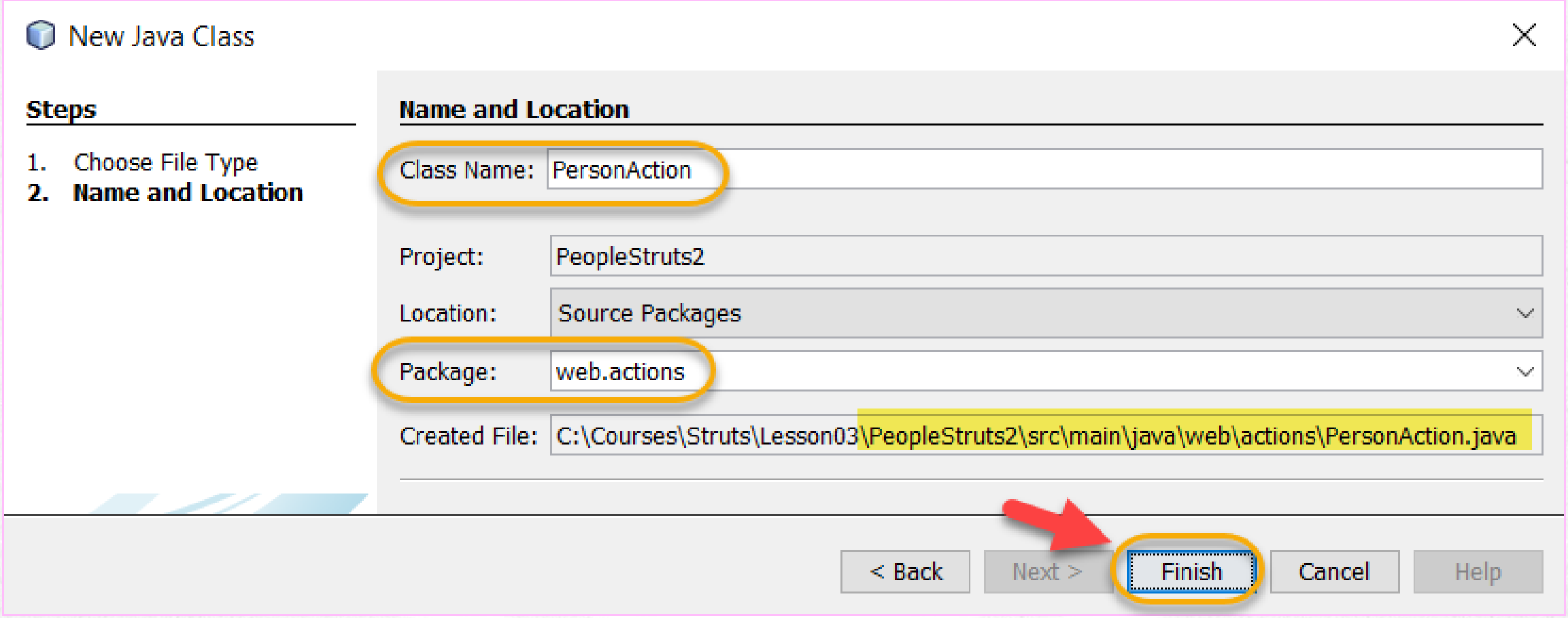
12. CREATE A NEW JAVA CLASS

- Create the PersonAction.java class:



12. CREATE A NEW JAVA CLASS

- Create the PersonAction.java class:



New Java Class

Steps

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

Name and Location

Class Name: PersonAction

Project: PeopleStruts2

Location: Source Packages

Package: web.actions

Created File: C:\Courses\Struts\Lesson03\PeopleStruts2\src\main\java\web\actions\PersonAction.java

13. MODIFY THE CODE

PersonAction.java:

Click to download

```java
package web.actions;

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

public class PersonAction extends ActionSupport {

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

    private Person person;
```

13. MODIFY THE CODE

PersonAction.java:

Click to download

```
    @Override
    public String execute() {
        if (person != null) {
            log.info("\n");
            log.info("person:" + person.getName());
            log.info("Street Name:" + person.getAddress().getStreetName());
            log.info("Street Number:" + person.getAddress().getStreetNumber());
            log.info("Country:" + person.getAddress().getCountry());
        }
        else{
            log.info("Person with null value ");
        }

        return SUCCESS;
    }

    public Person getPerson() {
        return person;
    }

    public void setPerson(Person person) {
        this.person = person;
    }
}
```

14. CREATE A NEW JSP FILE

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

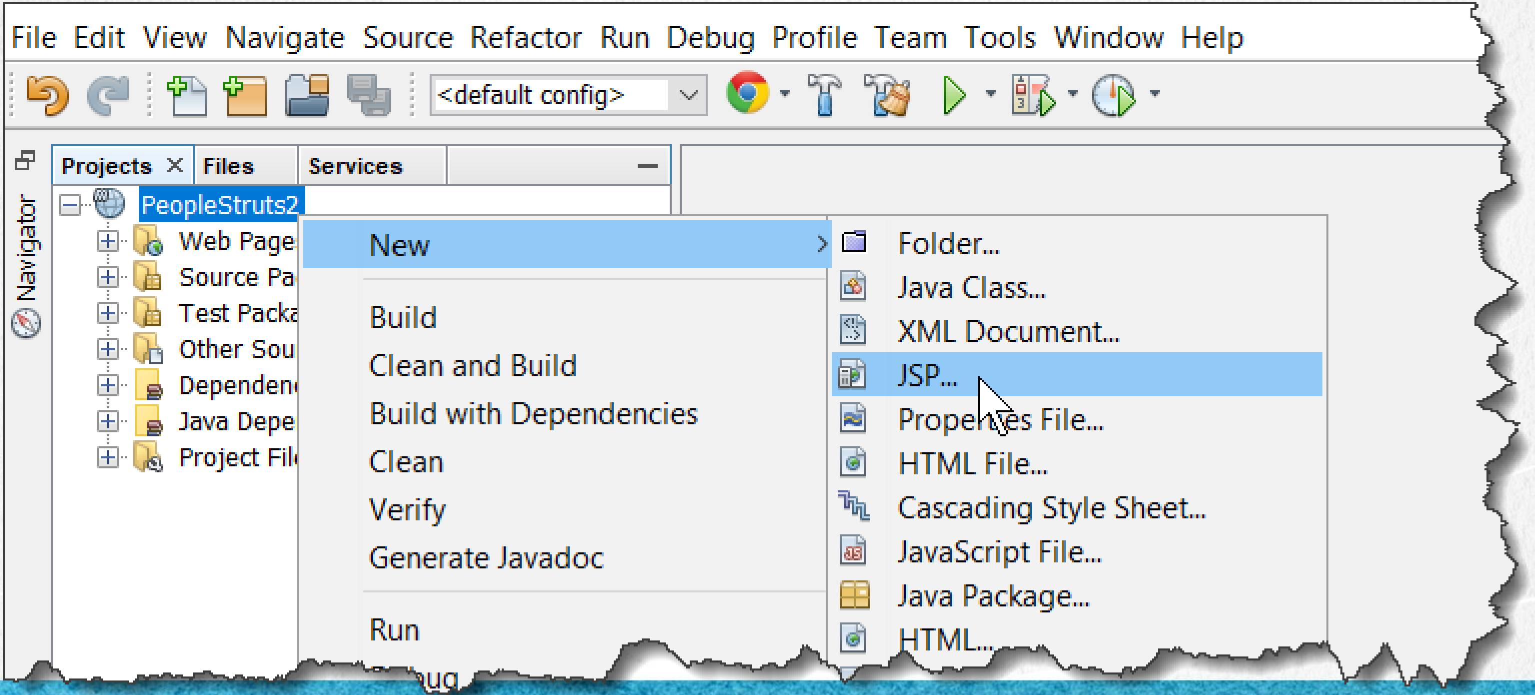
We must also deposit this JSP in the folder /WEB-INF/content as we have seen in the Struts 2 conventions topic.

Global Mentoring

Experiencia y Conocimiento para tu vida

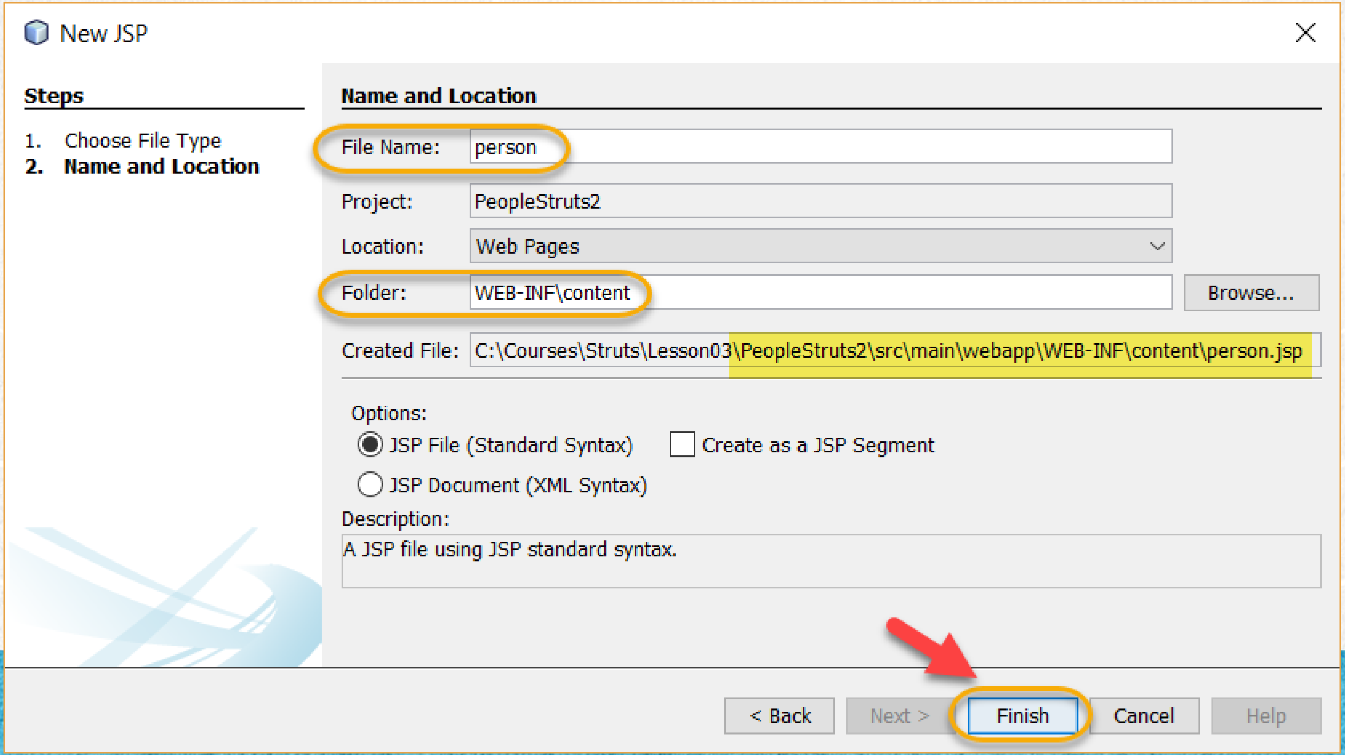
14. CREATE A NEW JSP

- Create a new person.jsp file:



14. CREATE A NEW JAVA CLASS

- Create a new person.jsp in the path shown below:



New JSP

Steps

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

Name and Location

File Name: person

Project: PeopleStruts2

Location: Web Pages

Folder: WEB-INF\content

Created File: C:\Courses\Struts\Lesson03\PeopleStruts2\src\main\webapp\WEB-INF\content\person.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

person.jsp:

Click to download

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib  prefix="s" uri="/struts-tags" %>
<!DOCTYPE html>
<html>
    <head>
        <title>Person Form</title>
        <%--We add the basic styles of Struts 2 --%>
        <s:head />
    </head>
    <body>
        <h1>Person Form with OGNL</h1>
        <s:form>
            <s:textfield label="Name" name="person.name" />
            <s:textfield label="Street Name (Address)" name="person.address.streetName" />
            <s:textfield label="Street Number (Address)" name="person.address.streetNumber" />
             <s:textfield label="Country (Address)" name="person.address.country" />
             <s:submit value="Send"/>
        </s:form>
```

14. MODIFY THE CODE

person.jsp:

Click to download

```
        <h2>Given values</h2>
        Name: <s:property value="person.name"/></br>
        Street Name <s:property value="person.address.streetName"/></br>
        Street Number: <s:property value="person.address.streetNumber"/></br>
        Country: <s:property value="person.address.country"/></br>

    </body>
</html>
```

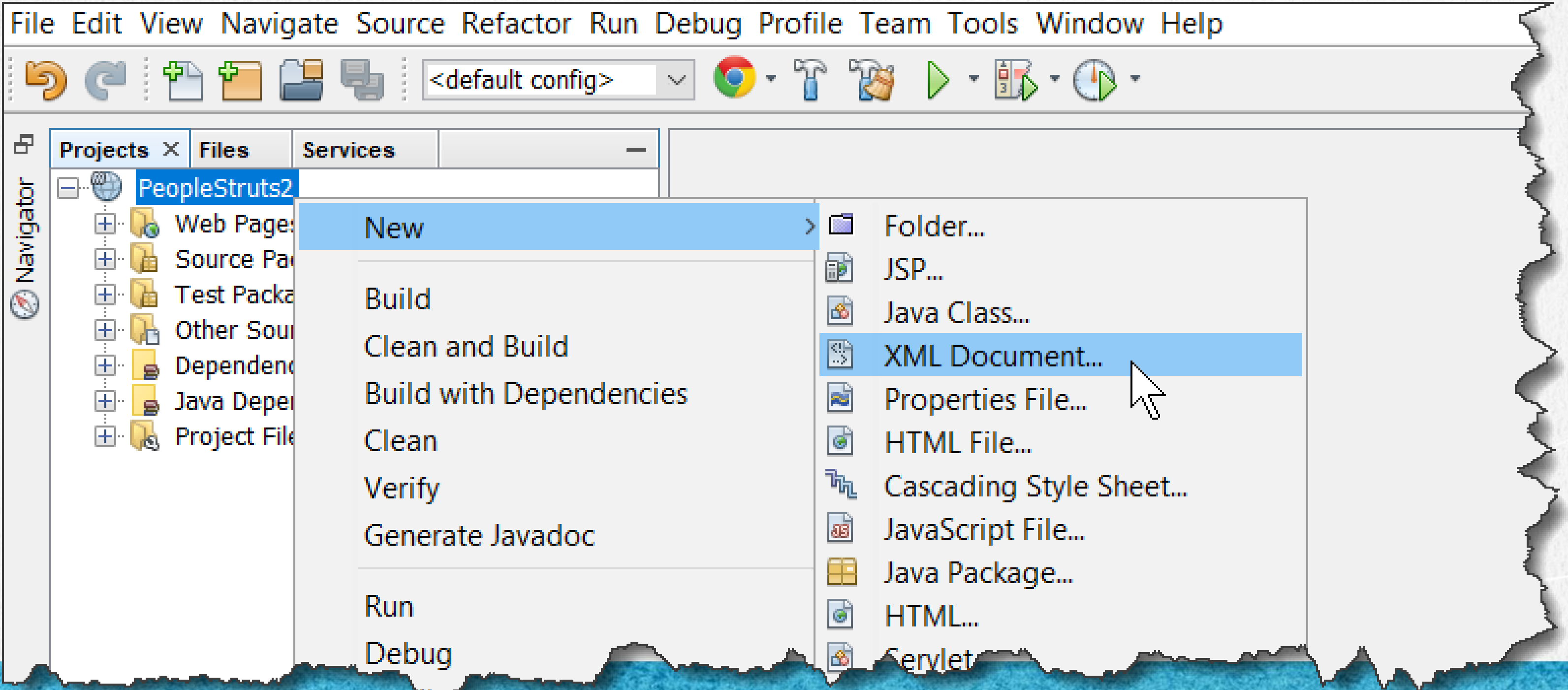
15. CREATE THE LOG4J2.XML FILE

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.

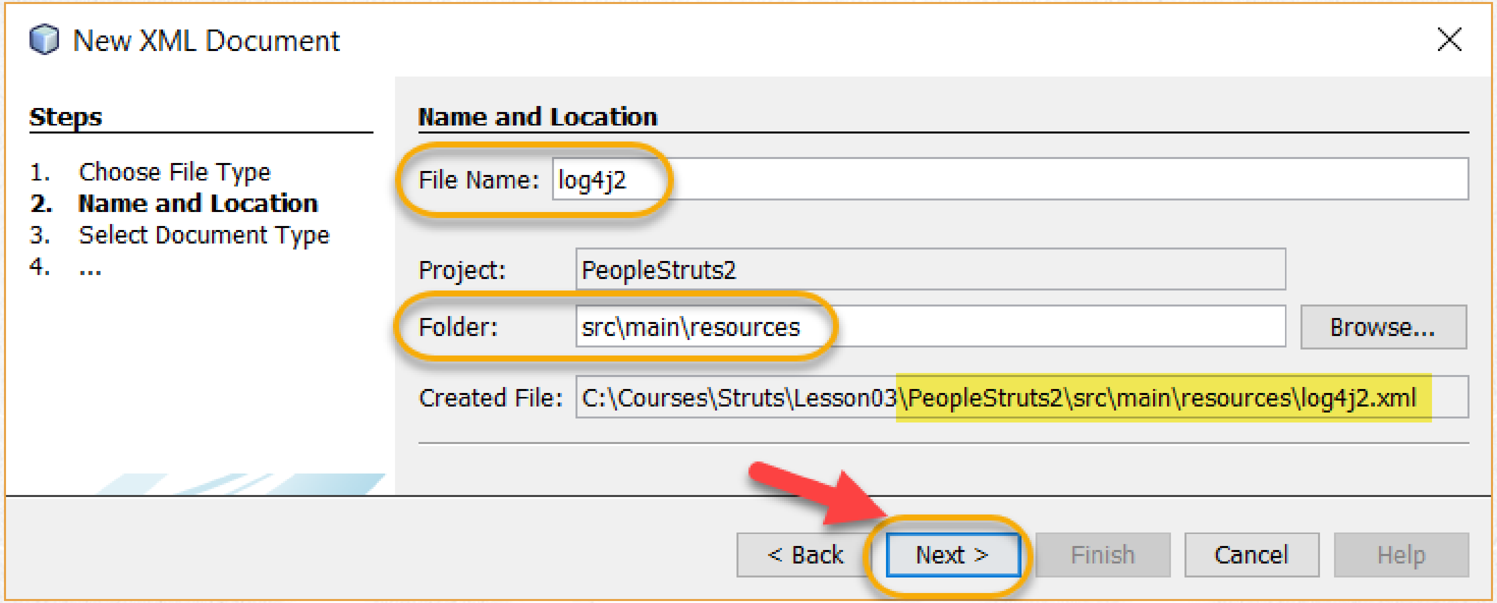
15. CRETE THE LOG4J2.XML FILE

- Create the log4j2.xml as follows:



15. 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

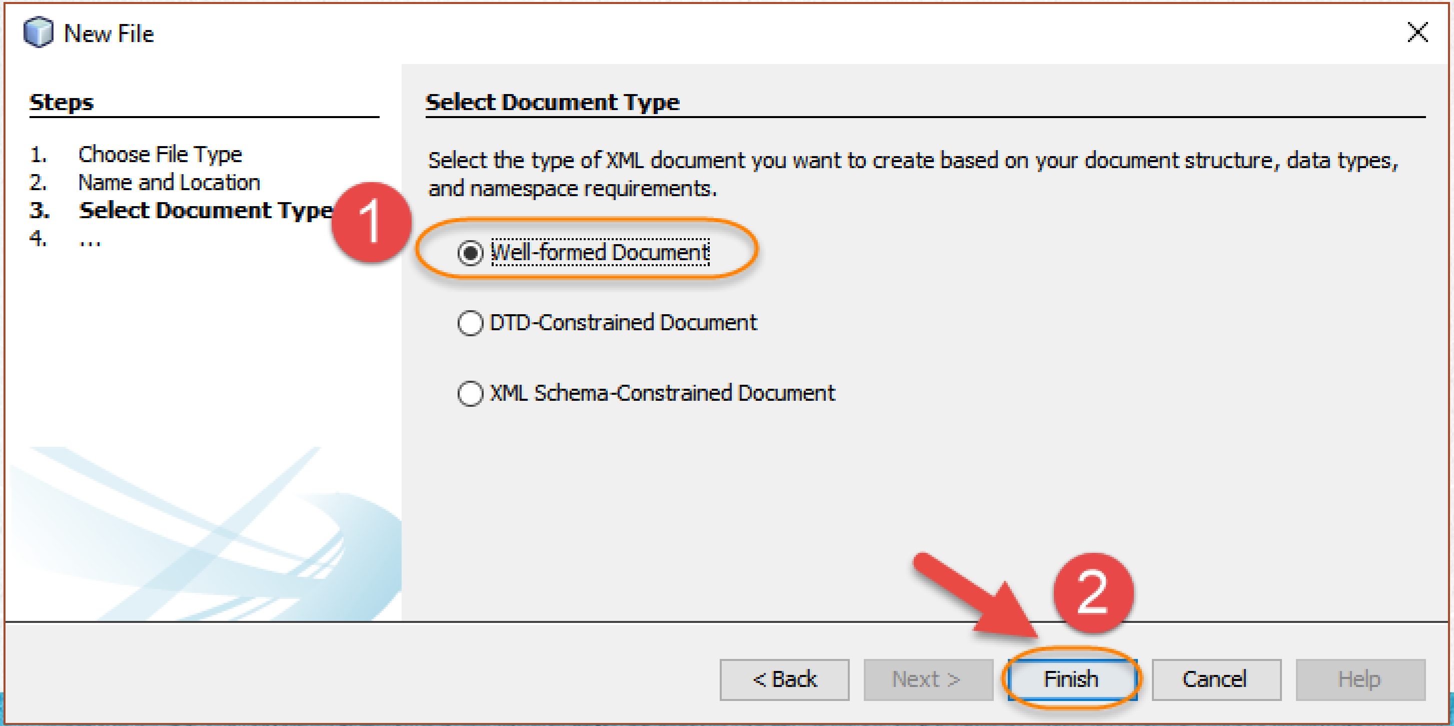
Project: PeopleStruts2

Folder: src\main\resources

Created File: C:\Courses\Struts\Lesson03\PeopleStruts2\src\main\resources\log4j2.xml

15. 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

15. MODIFY THE CODE

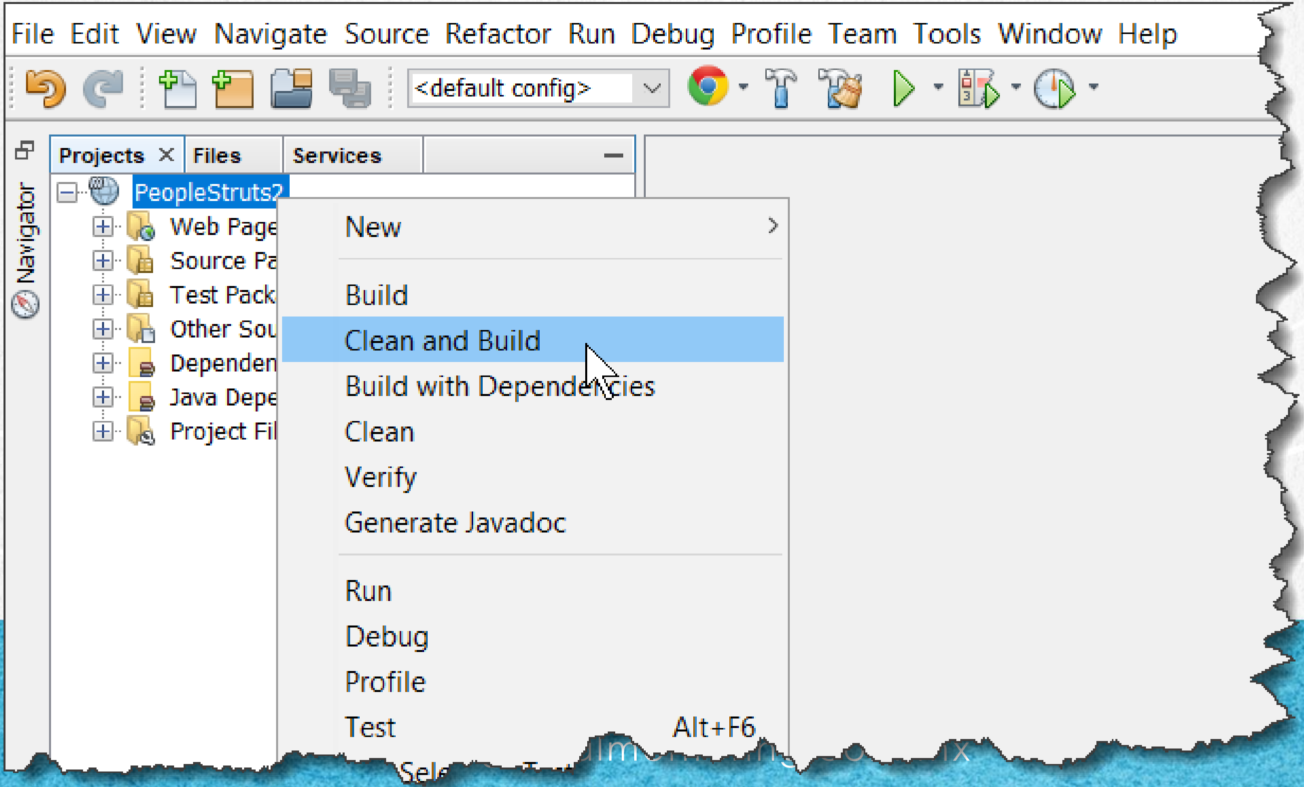
log4j2.xml:

Click to download

```xml
<?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>
```

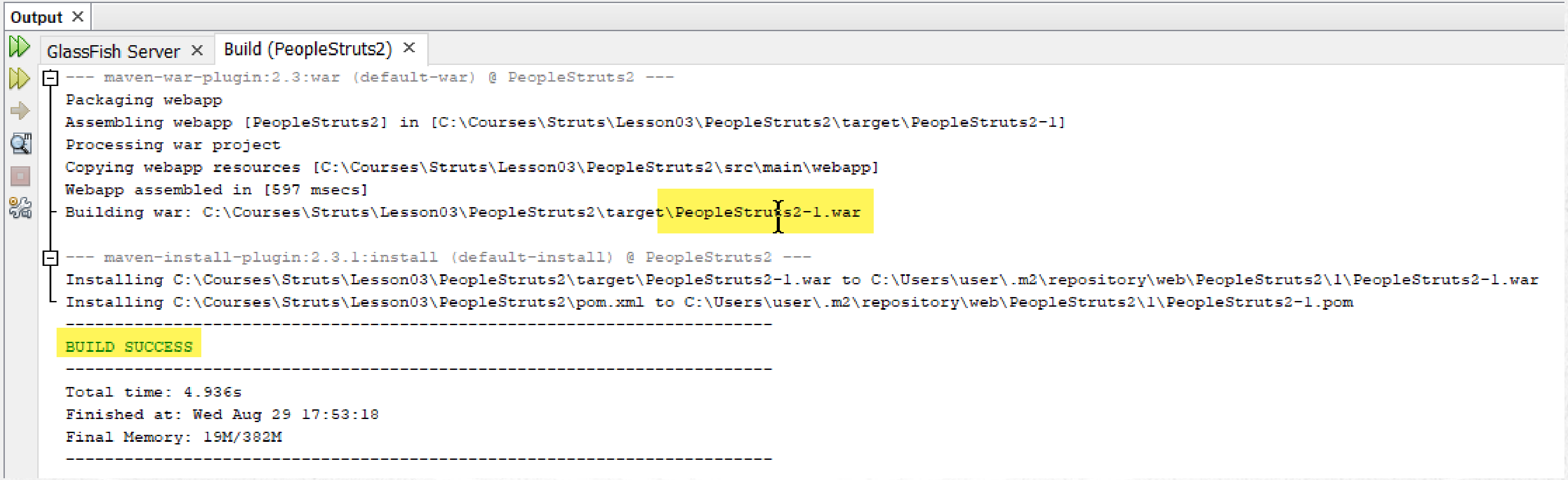
16. EXECUTE CLEAN & BUILD

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



16. EXECUTE CLEAN & BUILD

- We must observe a result similar to the following:

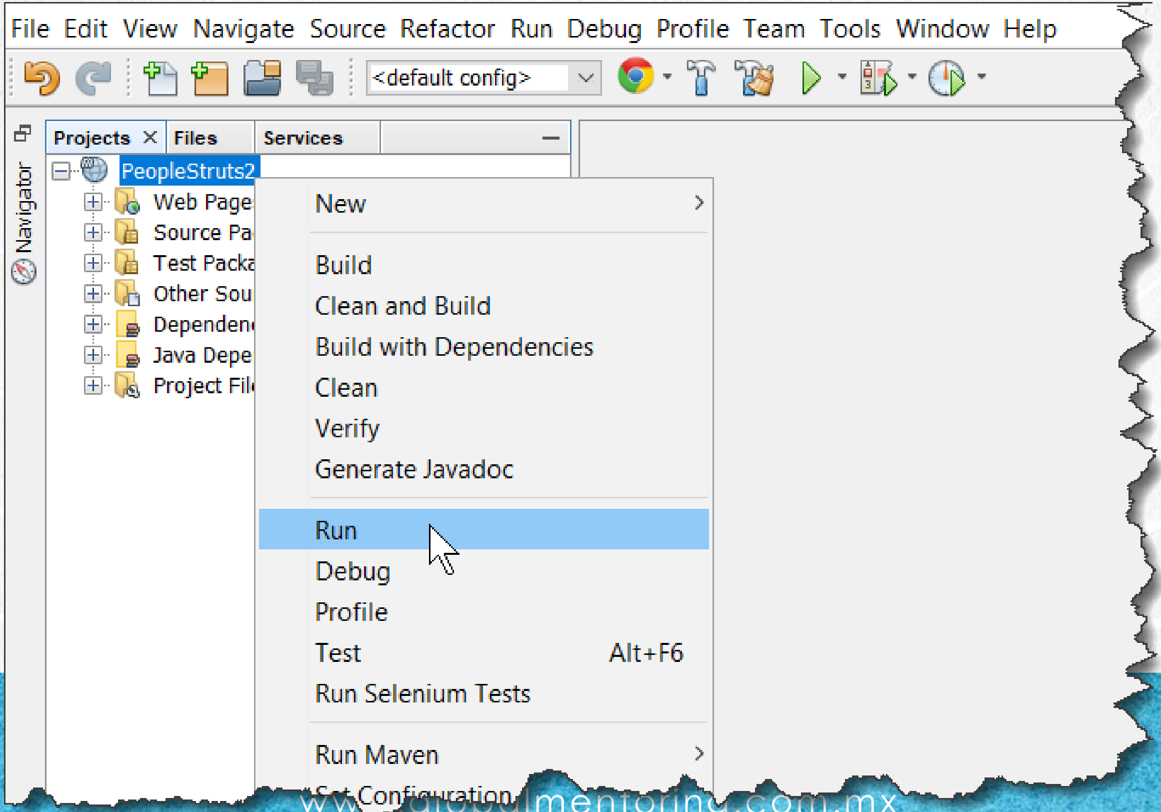


```
--- maven-war-plugin:2.3:war (default-war) @ PeopleStruts2 ---
Packaging webapp
Assembling webapp [PeopleStruts2] in [C:\Courses\Struts\Lesson03\PeopleStruts2\target\PeopleStruts2-1]
Processing war project
Copying webapp resources [C:\Courses\Struts\Lesson03\PeopleStruts2\src\main\webapp]
Webapp assembled in [597 msecs]
Building war: C:\Courses\Struts\Lesson03\PeopleStruts2\target\PeopleStruts2-1.war

--- maven-install-plugin:2.3.1:install (default-install) @ PeopleStruts2 ---
Installing C:\Courses\Struts\Lesson03\PeopleStruts2\target\PeopleStruts2-1.war to C:\Users\user\.m2\repository\web\PeopleStruts2\1\PeopleStruts2-1.war
Installing C:\Courses\Struts\Lesson03\PeopleStruts2\pom.xml to C:\Users\user\.m2\repository\web\PeopleStruts2\1\PeopleStruts2-1.pom
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 4.936s
Finished at: Wed Aug 29 17:53:18
Final Memory: 19M/382M
------------------------------------------------------------------------
```

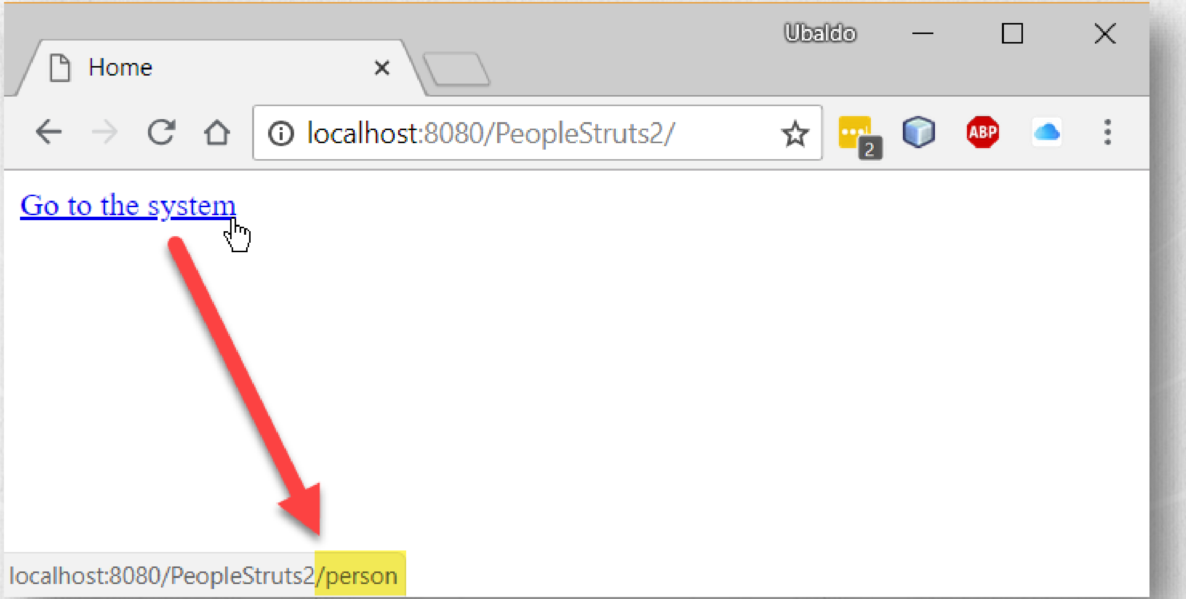
17. EXECUTE THE APPLICATION

- Execute the Project as follows:



17. EXECUTE THE APPLICATION

- Execute the application as follows:

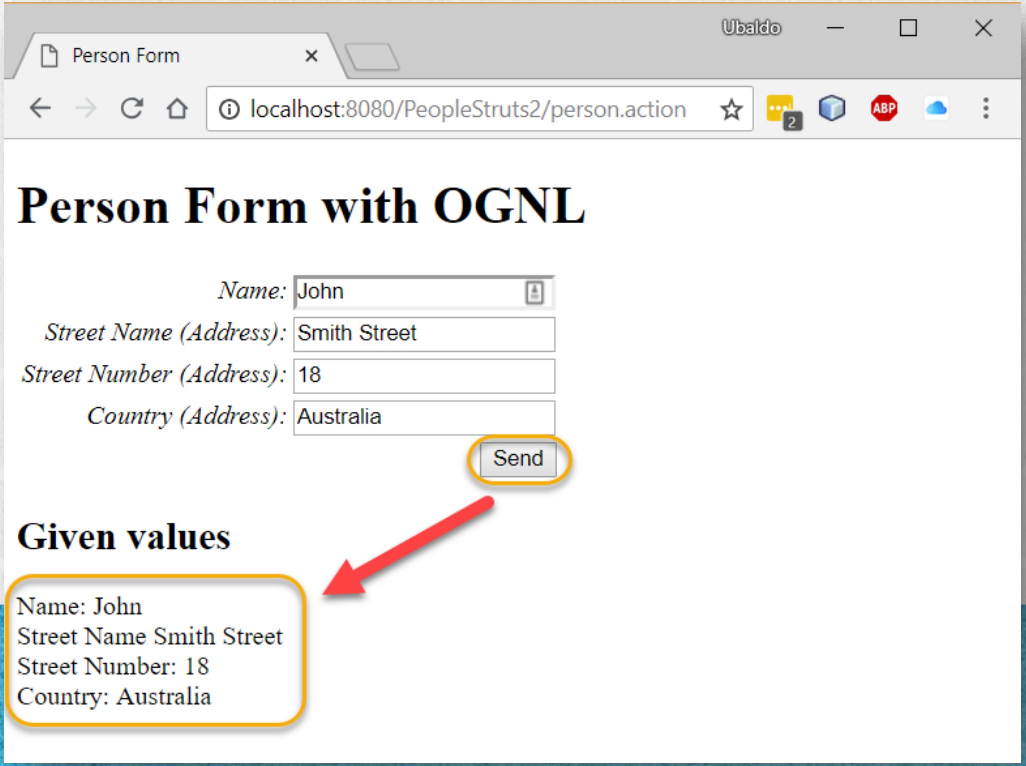


Go to the system

localhost:8080/PeopleStruts2/person

17. EXECUTE THE APPLICATION

- Fill out the form and click on Send. We will observe each of the values provided in the form and with it we will have applied the concept of OGNL:



Person Form with OGNL

Name:	John
Street Name (Address):	Smith Street
Street Number (Address):	18
Country (Address):	Australia

Given values

Name: John
Street Name Smith Street
Street Number: 18
Country: Australia

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 CONLUSION

With this exercise we put into practice the management of OGNL in 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 OGNL is clear, since it is one of the most important topic of Struts 2.

We saw how there was no need to instantiate objects of Address or Person type, everything in automatic through OGNL each of the properties of these objects was filled and we could access each of its properties.

Even the conversions to the most used types are done automatically, so it was not necessary to make any conversion for the types of data we use and the formats that we apply. However, it is possible to perform conversions and apply different formats to more personalized types depending on the type that we want to handle, such as the Date type, or any other type of data.

COURSE COURSE

STRUTS 2 FRAMEWORK

By: Eng. Ubaldo Acosta