

STRUTS FRAMEWORK COURSE

INTEGRATION OF STRUTS + SPRING + JPA CRUD



By the expert: Ubaldo Acosta

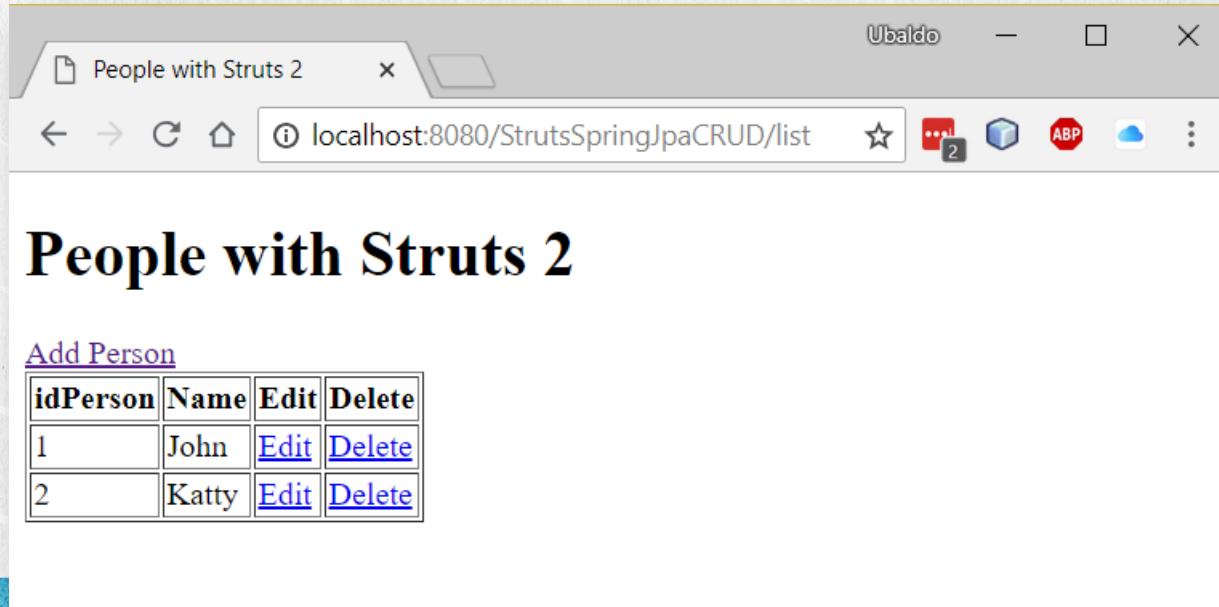


STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

EXERCISE OBJECTIVE

Create an application to implement the integration of Struts + Spring + JPA frameworks by applying the CRUD operations (Cread-Read-Update-Delete) on the person table. At the end we should observe the following:

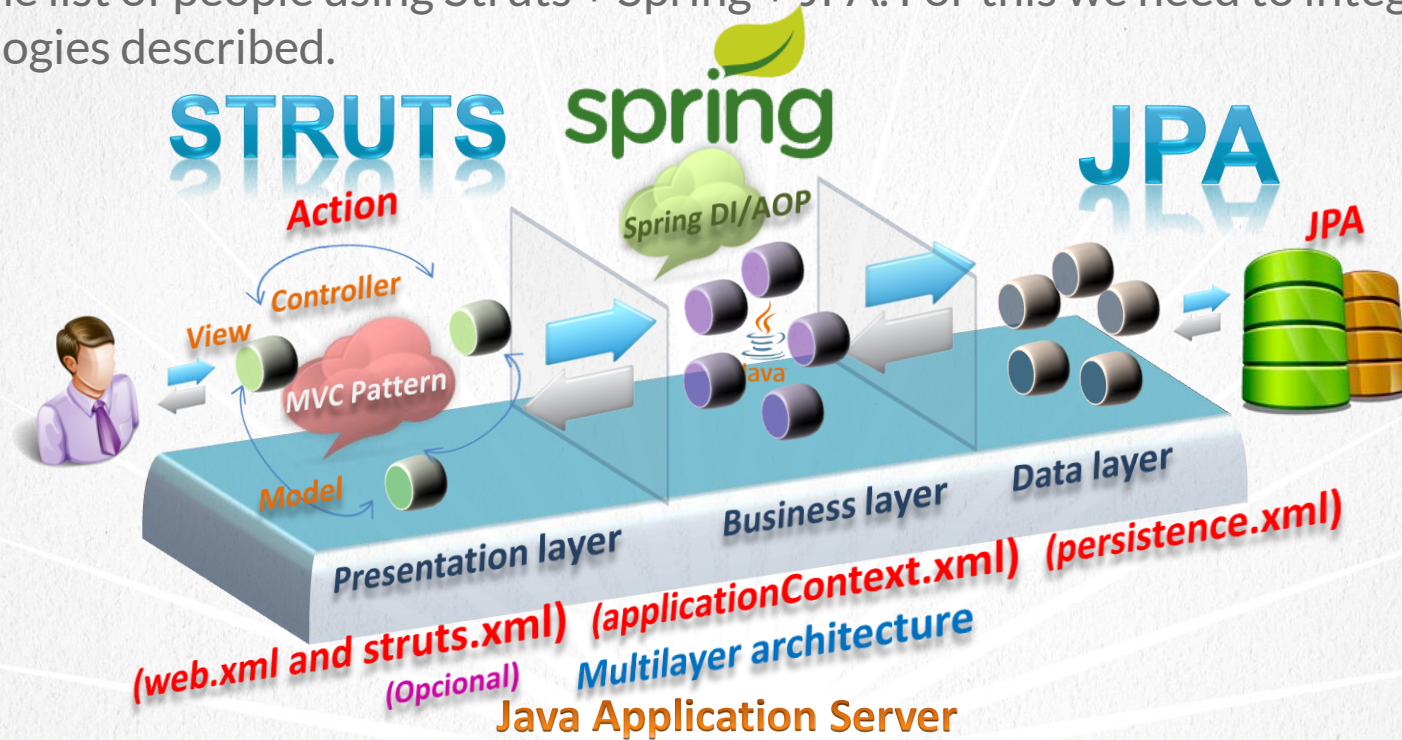


CURSO STRUTS FRAMEWORK

www.globalmentoring.com.mx

EXERCISE REQUIREMENT

Show the list of people using Struts + Spring + JPA. For this we need to integrate the 3 technologies described.



STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

1. CREATE A NEW PROJECT

We are going to use Maven to create the Java Web project. The project will be called StrutsSpringJpaCRUD. This project will integrate the 3 technologies: Struts + Spring + JPA.

Let's start with our exercise:

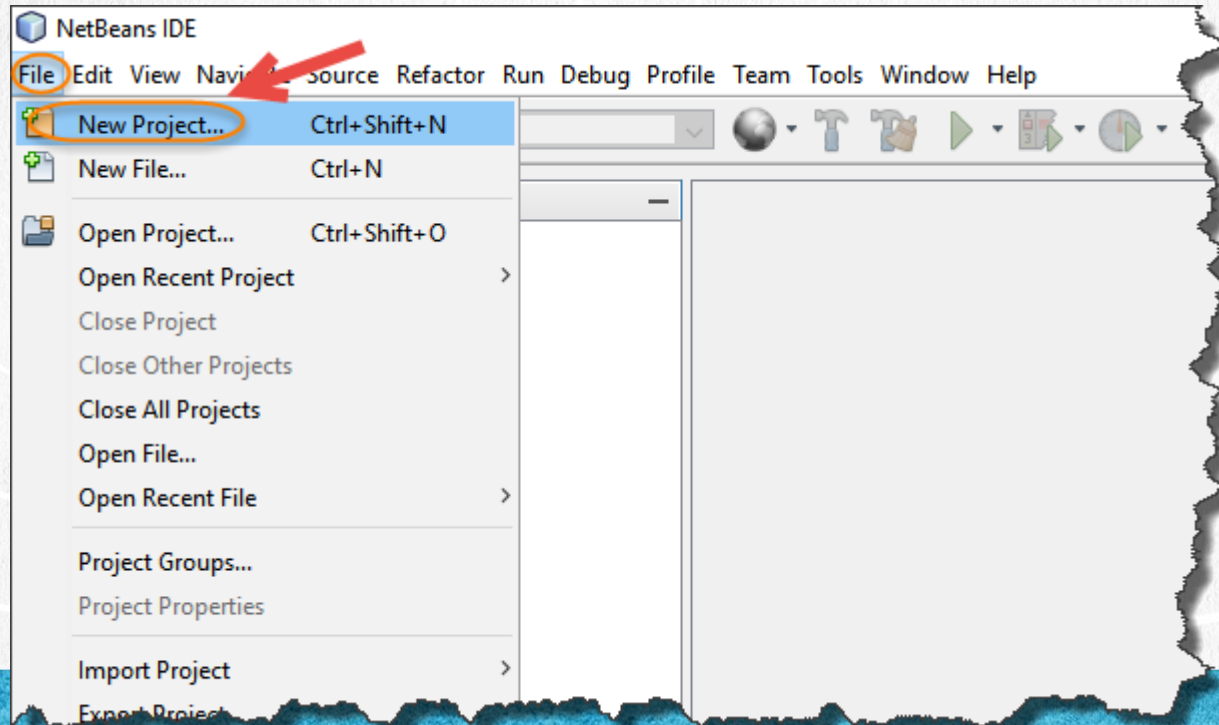


CURSO STRUTS FRAMEWORK

www.globalmentoring.com.mx

1. CREATE A NEW PROJECT

We created our exercise called StrutsSpringJpaCRUD:

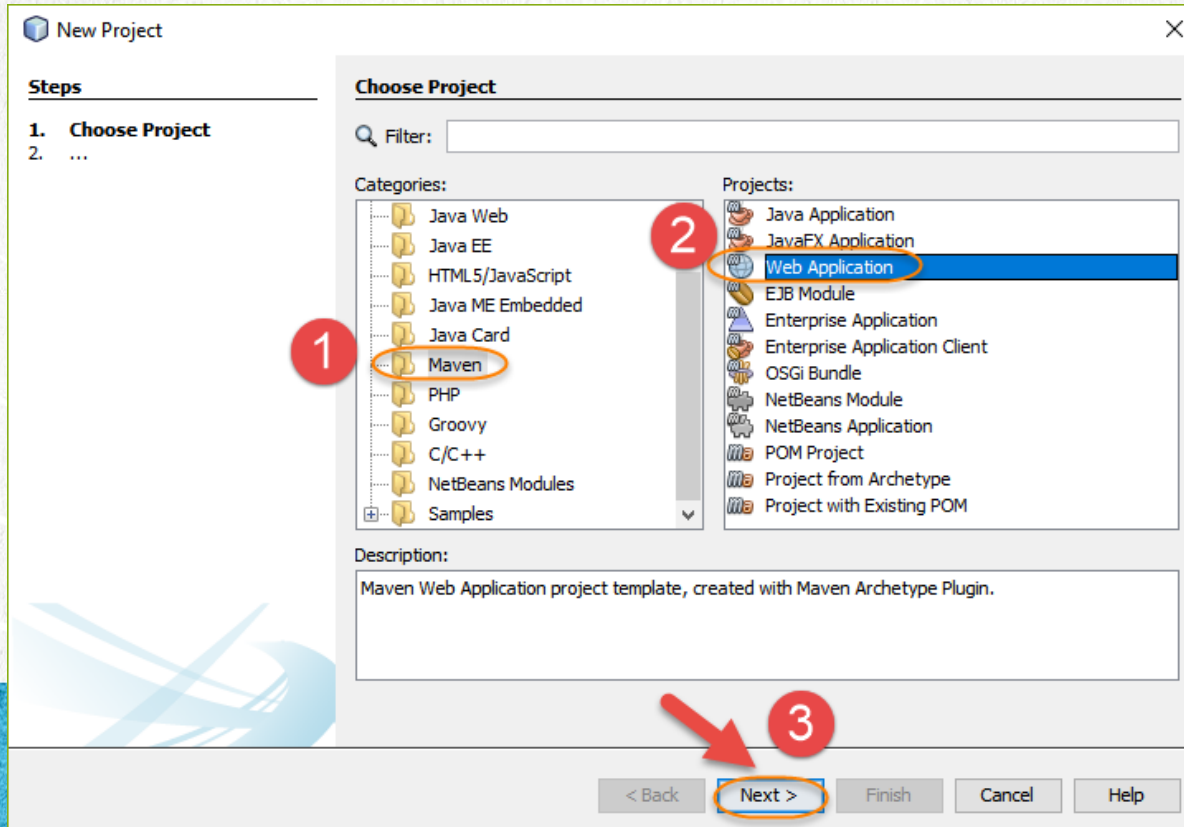


CURSO STRUTS FRAMEWORK

www.globalmentoring.com.mx

1. CREATE A NEW PROJECT

- We create a new Java Maven project of type Web Application:



1. CREATE A NEW PROJECT

- We create a new Java Maven project:

New Web Application [X]

Steps

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

Name and Location

Project Name: StrutsSpringJpaCRUD

Project Location: C:\Courses\Struts\02-StrutsSpringJpaCRUD [Browse...]

Project Folder: C:\Courses\Struts\02-StrutsSpringJpaCRUD\StrutsSpringJpaCRUD

Artifact Id: StrutsSpringJpaCRUD

Group Id: web

Version: 1

Package: (Optional)

< Back **Next >** Finish Cancel Help

CURSO STRUTS FRAMEWORK

www.globalmentoring.com.mx

1. CREATE A NEW PROJECT

- We create a new Java Maven project:

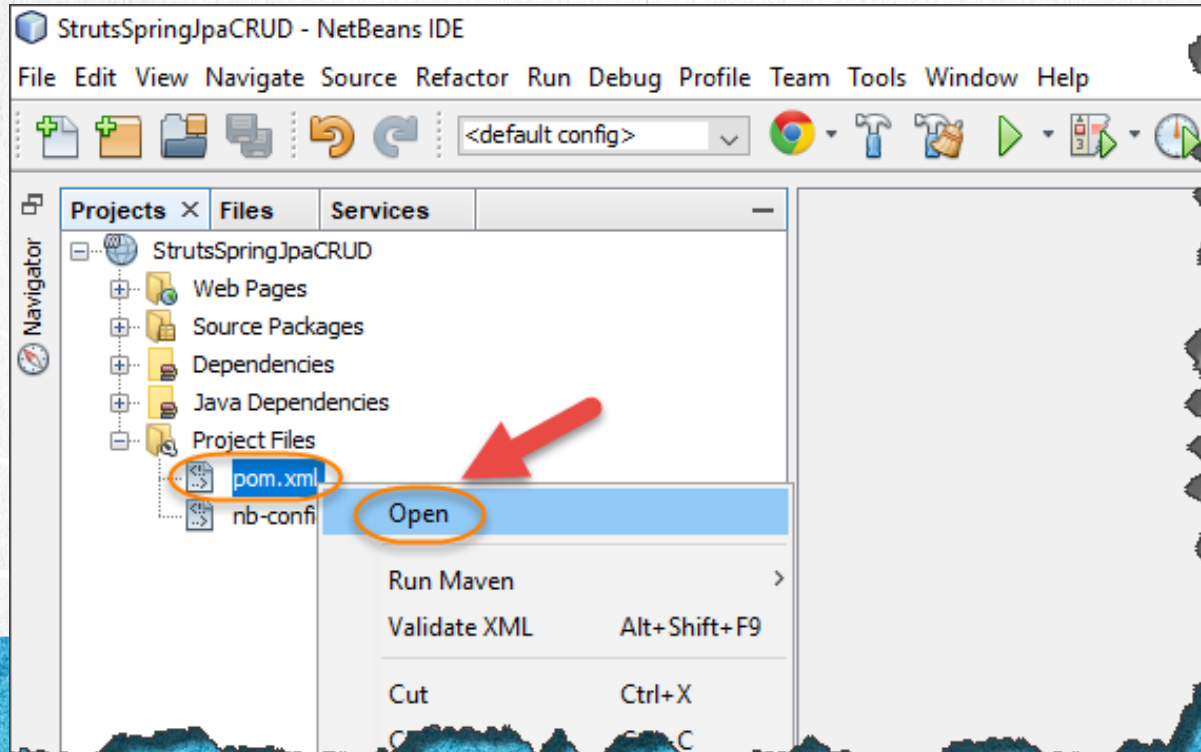
The screenshot shows the 'New Web Application' dialog box. On the left, the 'Steps' list includes: 1. Choose Project, 2. Name and Location, and 3. **Settings**. The 'Settings' section on the right contains two dropdown menus: 'Server' set to 'GlassFishServer' and 'Java EE Version' set to 'Java EE Web'. Both dropdowns are circled in orange. A red circle with the number '1' is next to the 'Server' dropdown, and a red circle with the number '2' is next to the 'Java EE Version' dropdown. At the bottom, the 'Finish' button is circled in orange, with a red circle containing the number '3' and a red arrow pointing to it. Other buttons at the bottom include '< Back', 'Next >', 'Cancel', and 'Help'.

STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

2. OPEN MAVEN'S POM.XML FILE

- The maven pom.xml file manages the Java libraries we are going to use. We will add all the necessary libraries to integrate the technologies described:



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

    <name>StrutsSpringJpaCRUD</name>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <struts.version>2.5.17</struts.version>
        <spring.version>5.0.8.RELEASE</spring.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>javax</groupId>
            <artifactId>javaee-web-api</artifactId>
            <version>8.0</version>
            <scope>provided</scope>
        </dependency>
```


3. MODIFY THE FILE

[pom.xml:](#)

[Click to download](#)

```
<dependency>
  <groupId>org.apache.struts</groupId>
  <artifactId>struts2-core</artifactId>
  <version>${struts.version}</version>
</dependency>
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-api</artifactId>
  <version>2.10.0</version>
</dependency>
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-core</artifactId>
  <version>2.10.0</version>
</dependency>
<dependency>
  <groupId>org.apache.struts</groupId>
  <artifactId>struts2-convention-plugin</artifactId>
  <version>${struts.version}</version>
</dependency>
```

3. MODIFY THE FILE

[pom.xml:](#)

[Click to download](#)

```
<!--Spring-->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-core</artifactId>
  <version>${spring.version}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-context</artifactId>
  <version>${spring.version}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-test</artifactId>
  <version>${spring.version}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-orm</artifactId>
  <version>${spring.version}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-web</artifactId>
  <version>${spring.version}</version>
</dependency>
```


3. MODIFY THE FILE

[pom.xml:](#)

[Click to download](#)

```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-webmvc</artifactId>
  <version>${spring.version}</version>
</dependency>
<!--Struts 2 y Spring integracion-->
<dependency>
  <groupId>org.apache.struts</groupId>
  <artifactId>struts2-spring-plugin</artifactId>
  <version>${struts.version}</version>
</dependency>
<!-- MySql -->
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>5.1.42</version>
</dependency>
</dependencies>
```

CURSO DE JAVA CON JDBC

www.globalmentoring.com.mx

3. MODIFY THE FILE

[pom.xml:](#)

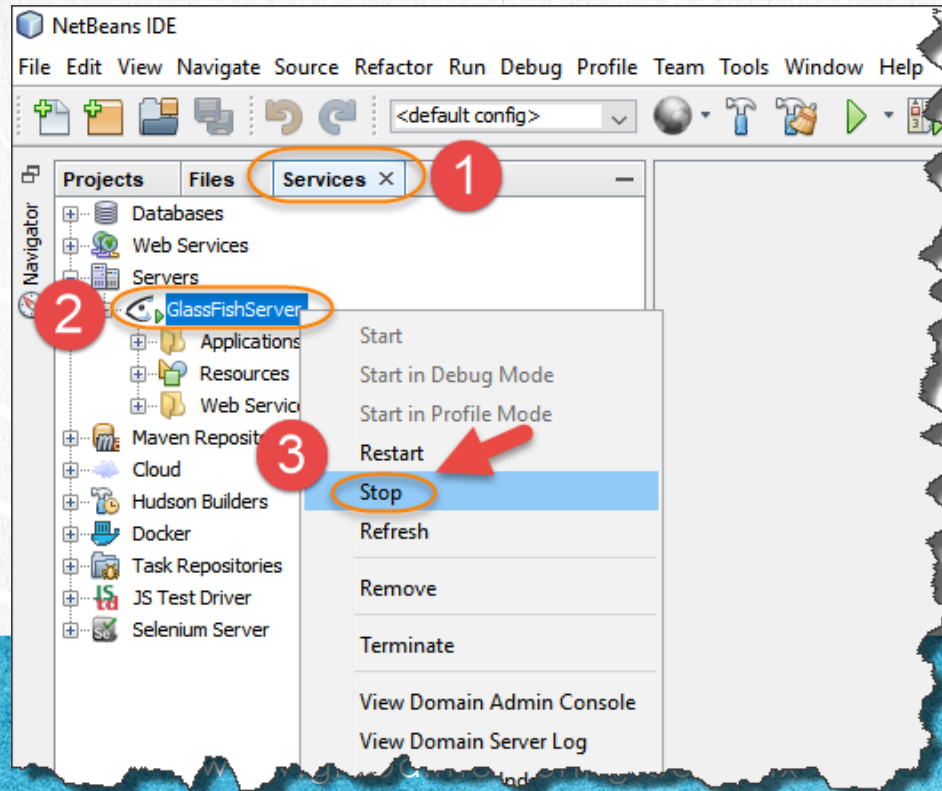
[Click to download](#)

```
<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>
```

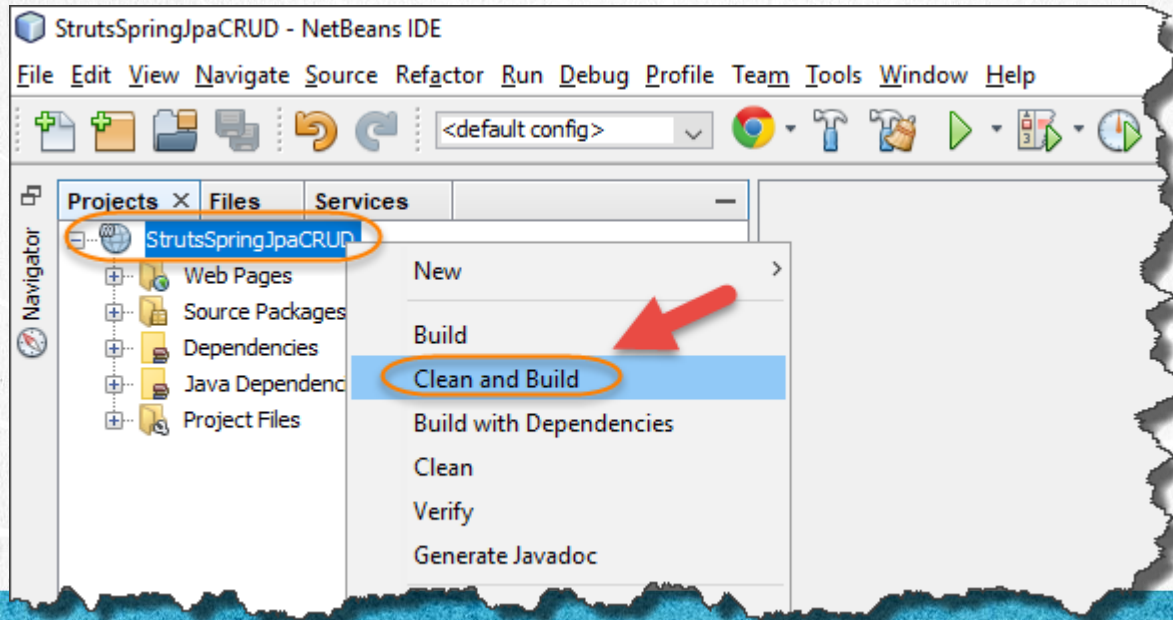

4. STOP GLASSFISH IF IT WAS STARTED

- Before doing Clean & Build of the project to download the libraries if necessary, 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.



5. EXECUTE CLEAN & BUILD

- To download the new libraries if necessary, 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:

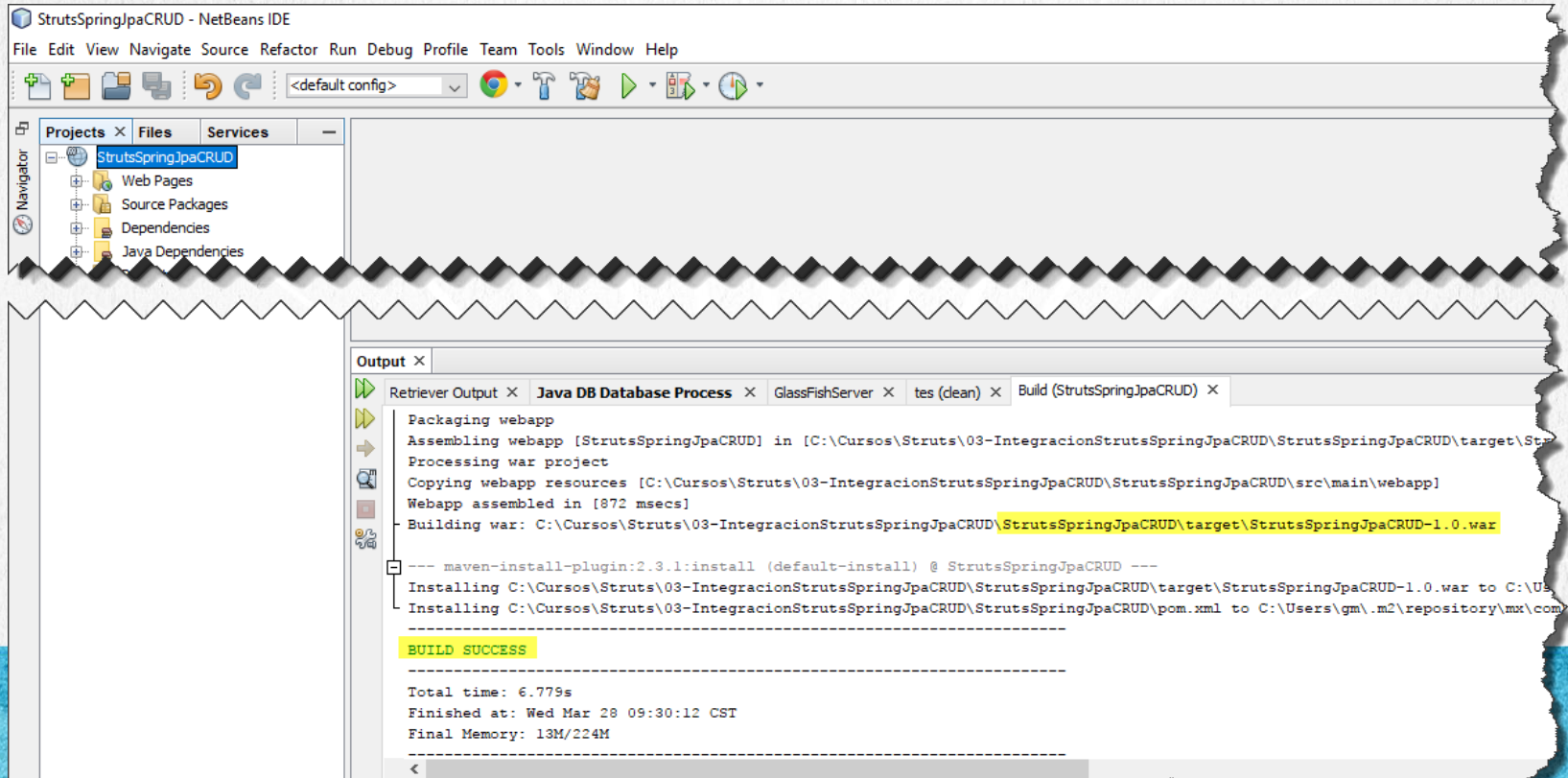


STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

5. EXECUTE CLEAN & BUILD

- If it was no longer necessary to download any library because it 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 persistence.xml file

This file is what allows us to configure JPA technology (Java Persistence API).

Let's see how our persistence.xml file is.

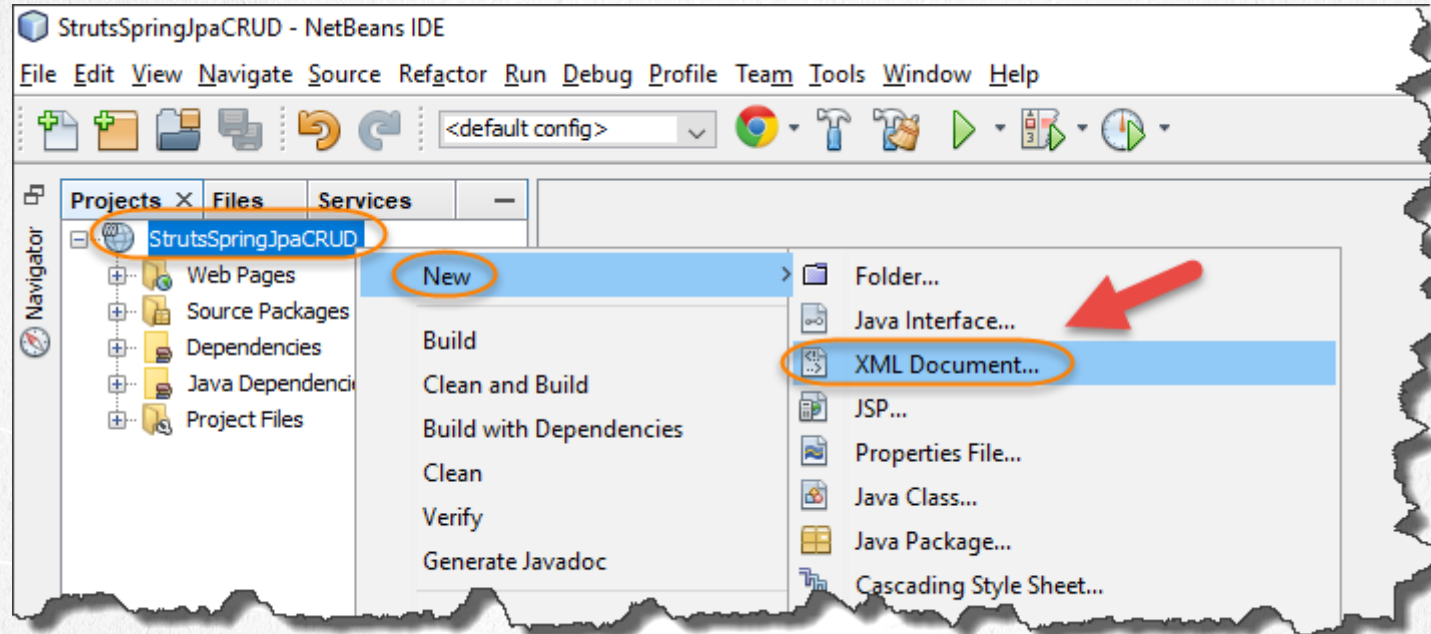


STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

6. CREATE AN XML FILE

- We create the persistence.xml file and add it to the following folder as shown:



6. CREATE AN XML FILE

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

New XML Document

Steps

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

Name and Location

File Name:

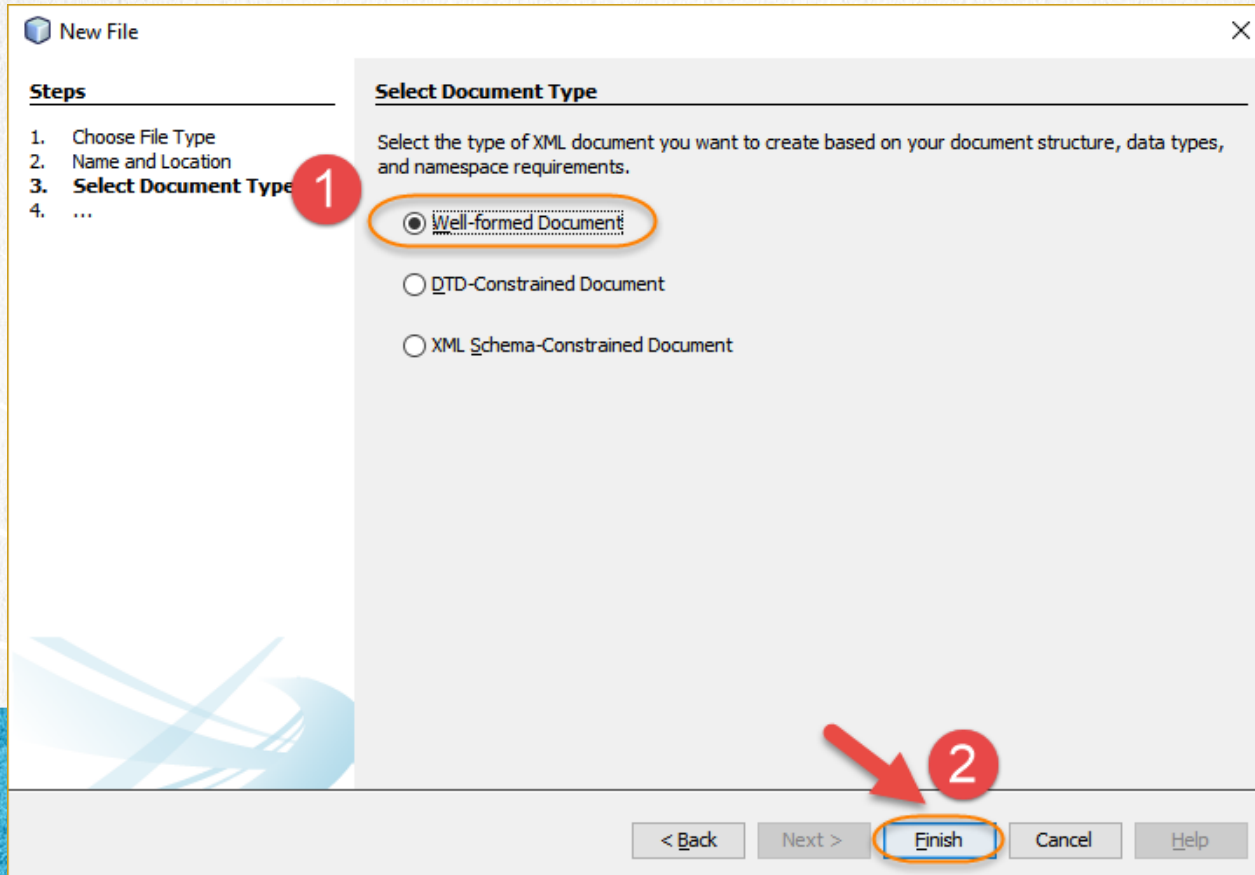
Project:

Folder:

Created File:

6. CREATE AN XML FILE

- We select the indicated type and click on finish.



7. MODIFY THE FILE

[persistence.xml:](#)

[Click to download](#)

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
    http://xmlns.jcp.org/xml/ns/persistence/persistence_2_2.xsd"
  version="2.2">
  <persistence-unit name="PersistenceUnit" transaction-type="JTA">
    <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
    <jta-data-source>jdbc/PersonDb</jta-data-source>
    <properties>
      <property name="eclipselink.logging.level" value="FINE"/>
      <property name="eclipselink.logging.parameters" value="true"/>
    </properties>
  </persistence-unit>
</persistence>
```


8. CREATE AN XML FILE

We are going to create the applicationContext.xml file below

This file is what allows us to configure the Spring framework.

Let's see how our applicationContext.xml file looks

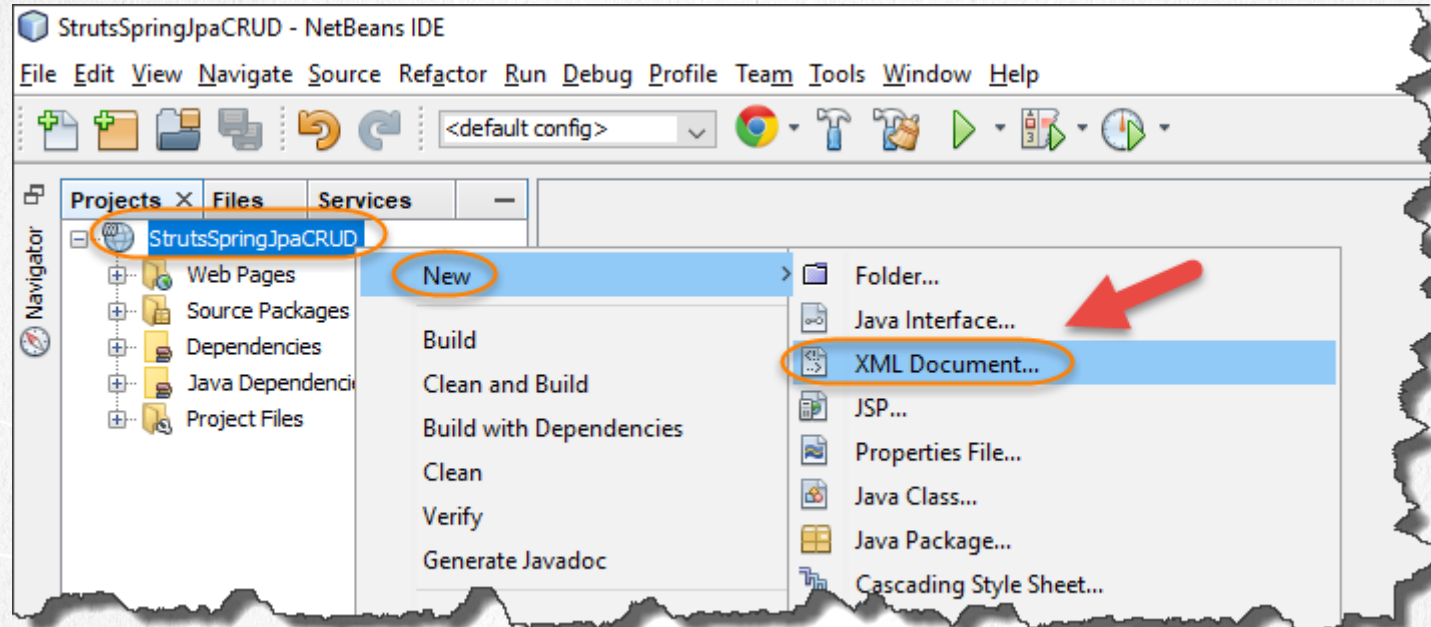


STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

8. CREATE AN XML FILE

- We create the applicationContext.xml file and add it to the following folder as shown:



8. CREATE AN XML FILE

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

New XML Document

Steps

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

Name and Location

File Name:

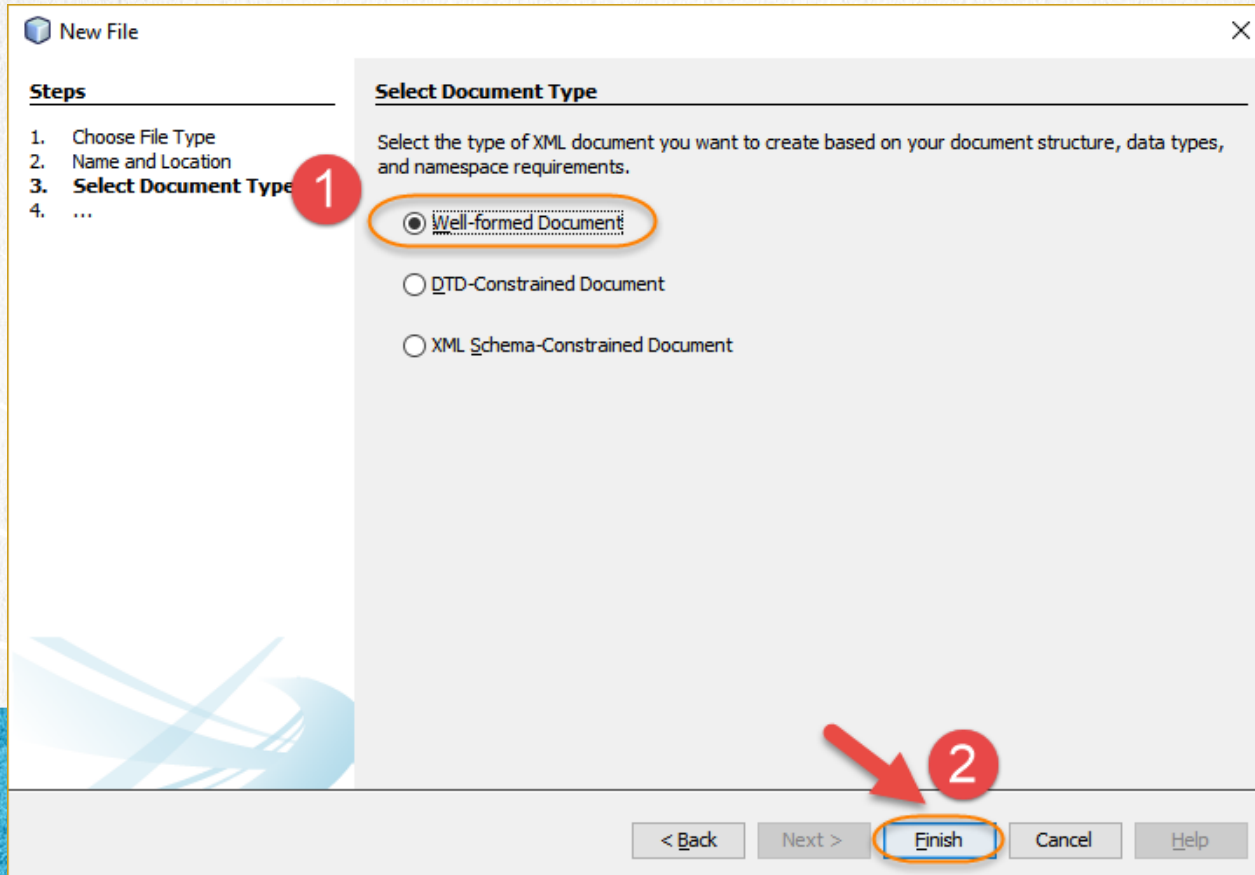
Project:

Folder:

Created File:

8. CREATE AN XML FILE

- We select the indicated type and click on finish.



9. MODIFY THE CODE

applicationContext.xml:

Click to download

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xmlns:jee="http://www.springframework.org/schema/jee"
       xsi:schemaLocation="
           http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans.xsd
           http://www.springframework.org/schema/context
           http://www.springframework.org/schema/context/spring-context.xsd
           http://www.springframework.org/schema/jee
           http://www.springframework.org/schema/jee/spring-jee.xsd
           http://www.springframework.org/schema/tx
           http://www.springframework.org/schema/tx/spring-tx.xsd">

    <context:component-scan base-package="mx.com.gm.businesslayer" />
    <context:component-scan base-package="mx.com.gm.datalayer" />

    <!-- Get the injected entity manager at the Spring factory -->
    <bean class="org.springframework.orm.jpa.support.PersistenceAnnotationBeanPostProcessor" />
```

9. MODIFY THE CODE

[applicationContext.xml:](#)

[Click to download](#)

```
<bean id="transactionManager" class="org.springframework.transaction.jta.JtaTransactionManager" />

<!--Name that maps with the Persistence Unit in the web.xml file-->
<jee:jndi-lookup id="entityManagerFactory" jndi-name="persistence/PersistenceUnit" />

<!-- Detect @Transactional -->
<tx:annotation-driven transaction-manager="transactionManager" />
</beans>
```


10. 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.

In addition, it also allows us to integrate the Spring framework with our web application through the configuration of a Spring listener.

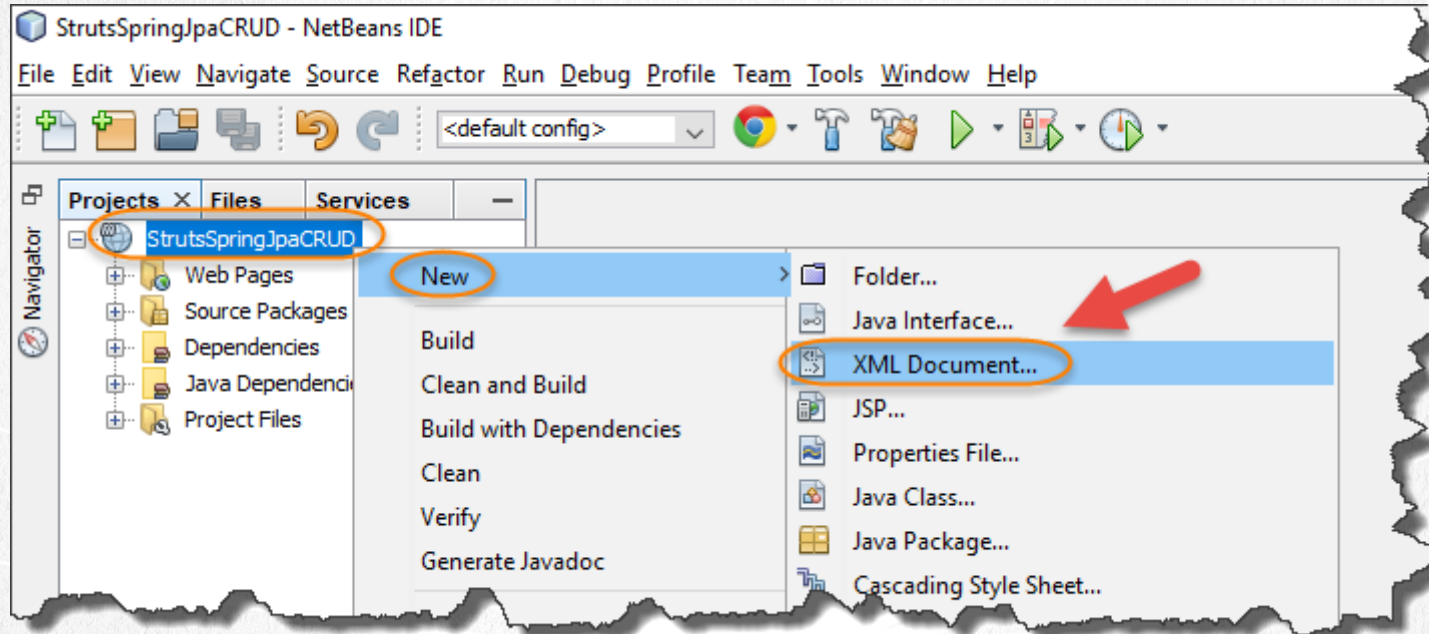
Our web.xml file also allows us to configure the JNDI name for the database connection that we will use with JPA via JTA.

Normally we should use the latest version of the JavaEE namespace, but due to problems with Spring compatibility, we will use version 3.1 of the namespace.

Let's see how our web.xml file is.

10. CREATE AN XML FILE

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



10. CREATE AN XML FILE

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

New XML Document

Steps

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

Name and Location

File Name:

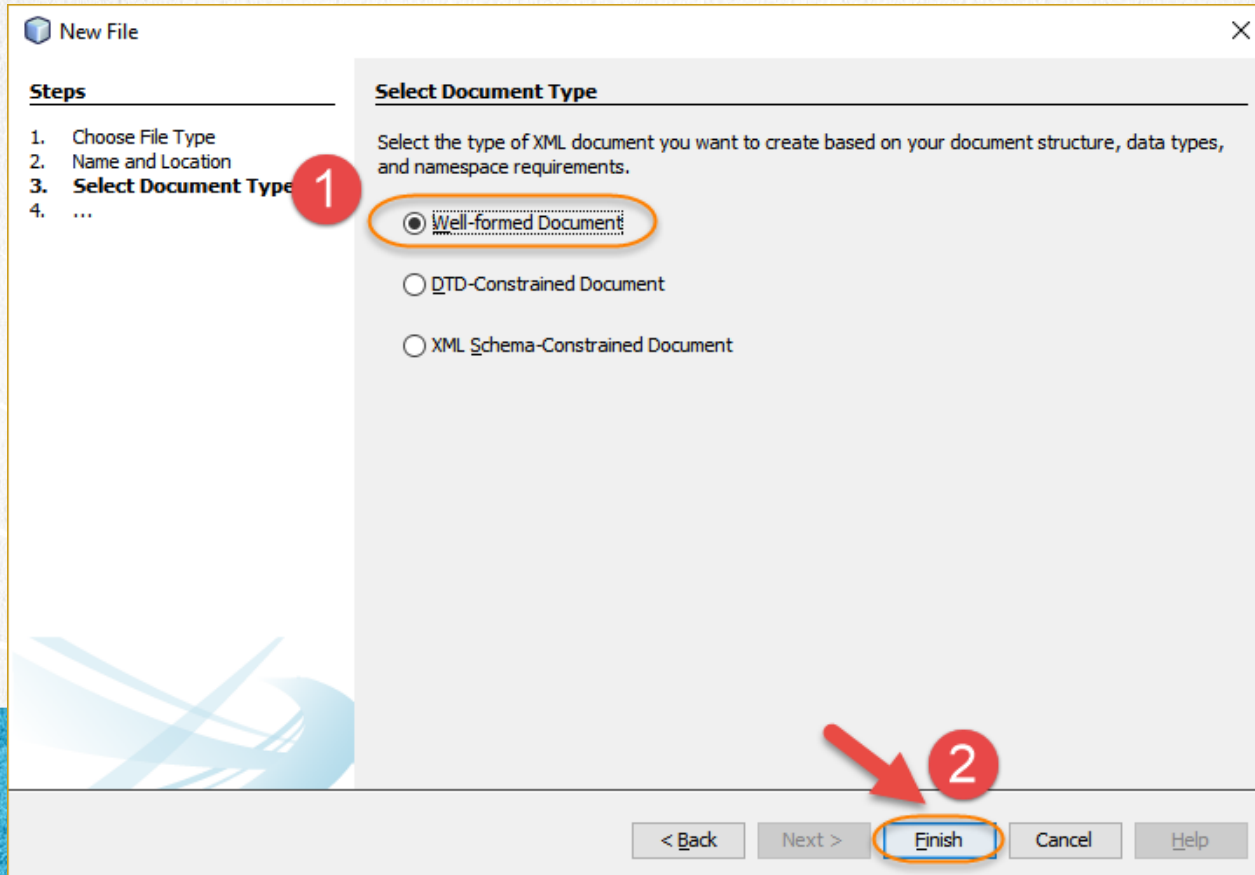
Project:

Folder:

Created File:

10. CREATE AN XML FILE

- We select the indicated type and click on finish.



11. MODIFY THE CODE

web.xml:

Click to download

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app 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_3_1.xsd"
        version="3.1">
    <!-- Integration with Struts Framework-->
    <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>

    <!-- Integration with Spring Framework-->
    <listener>
        <listener-class>
            org.springframework.web.context.ContextLoaderListener
        </listener-class>
    </listener>

    <!-- Name used in the applicationContext.xml file of Spring and JPA-->
    <persistence-unit-ref>
        <persistence-unit-ref-name>persistence/PersistenceUnit</persistence-unit-ref-name>
        <persistence-unit-name>PersistenceUnit</persistence-unit-name>
    </persistence-unit-ref>
</web-app>
```

12. CREATE A JAVA CLASS

The entity class `Persona.java` that we are going to create next is the class that will be used by the JPA technology to represent a record of the person table in the database.

We will use JPA annotations where necessary to customize the `Person.java` class and thus be able to represent exactly the records of the person table in the database.

This type of class is also known as a domain class.

Let's see how our class `Persona.java` is

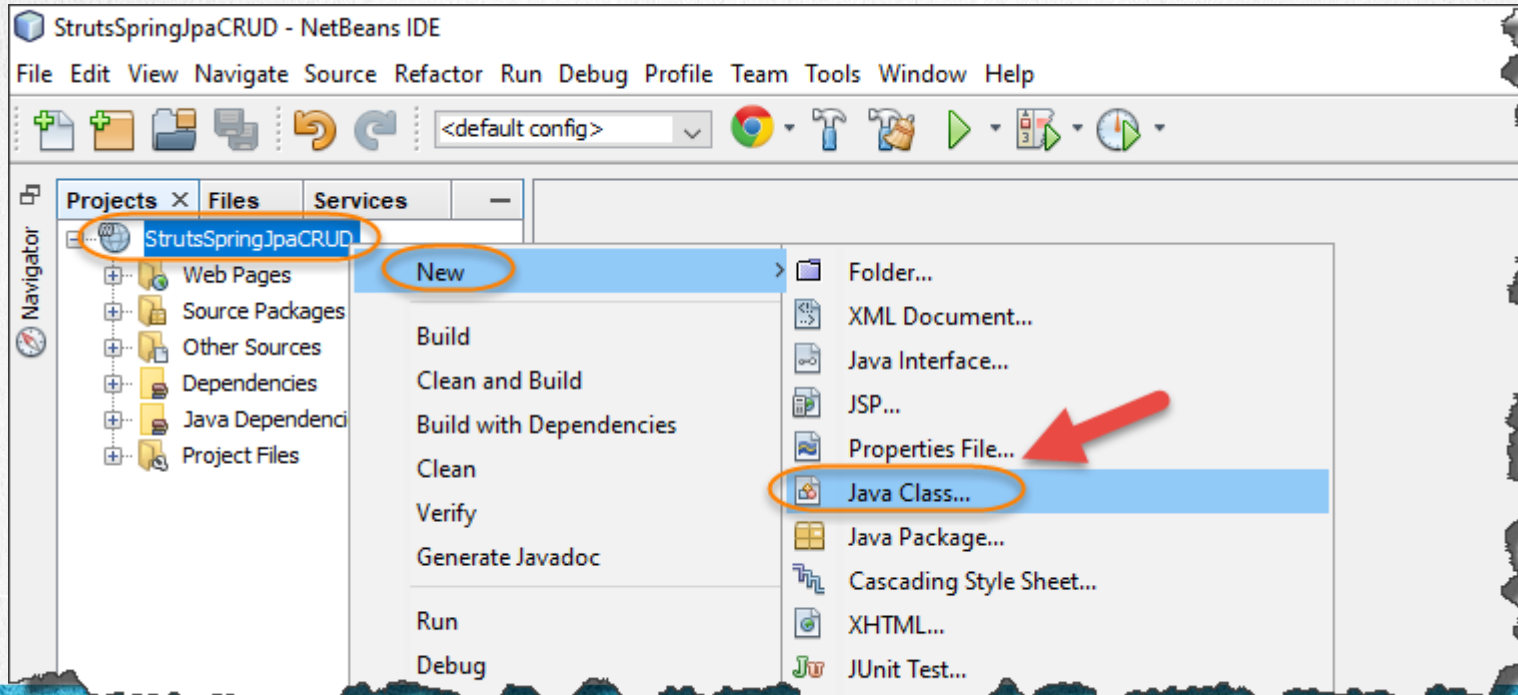


STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

12. CREATE A JAVA CLASS

- We create the Person.java class:



STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

12. CREATE A JAVA CLASS

- We create the Person.java class:

New Java Class

Steps

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

Name and Location

Class Name: Person

Project: StrutsSpringJpaCRUD

Location: Source Packages

Package: mx.com.gm.datalayer.domain

Created File: C:\Courses\Struts\02-StrutsSpringJpaCRUD\StrutsSpringJpaCRUD\src\main\java\mx\com\gm\datalayer\domain\Person.java

< Back Next > **Finish** Cancel Help

13. MODIFY THE CODE

Person.java:

[Click to download](#)

```
package mx.com.gm.datalayer.domain;

import java.io.Serializable;
import javax.persistence.*;

@Entity
public class Person implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_person")
    private Long idPerson;

    private String name;

    public Person() {
    }

    public Person(Long idPerson) {
        this.idPerson = idPerson;
    }
}
```


13. MODIFY THE CODE

Person.java:

[Click to download](#)

```
public Long getIdPerson() {  
    return idPerson;  
}  
  
public void setIdPerson(Long idPerson) {  
    this.idPerson = idPerson;  
}  
  
public String getName() {  
    return name;  
}  
  
public void setName(String name) {  
    this.name = name;  
}  
  
@Override  
public String toString() {  
    return "Person{" + "idPerson=" + idPerson + ", name=" + name + '}';  
}  
  
}
```

14. CREATE A JAVA INTERFACE

We are going to create a Java interface. Remember that it is a good practice to program using interfaces to separate the layers of our Java application. So we will create an interface and then its implementation.

In this interface we will apply the DAO (Data Access Object) design pattern, since it is the interface that will allow us to apply the operations on the Person entity class, methods such as listing, adding, modifying, eliminating Person objects.

The name of the interface is PersonDao.java, let's see how is this interface :

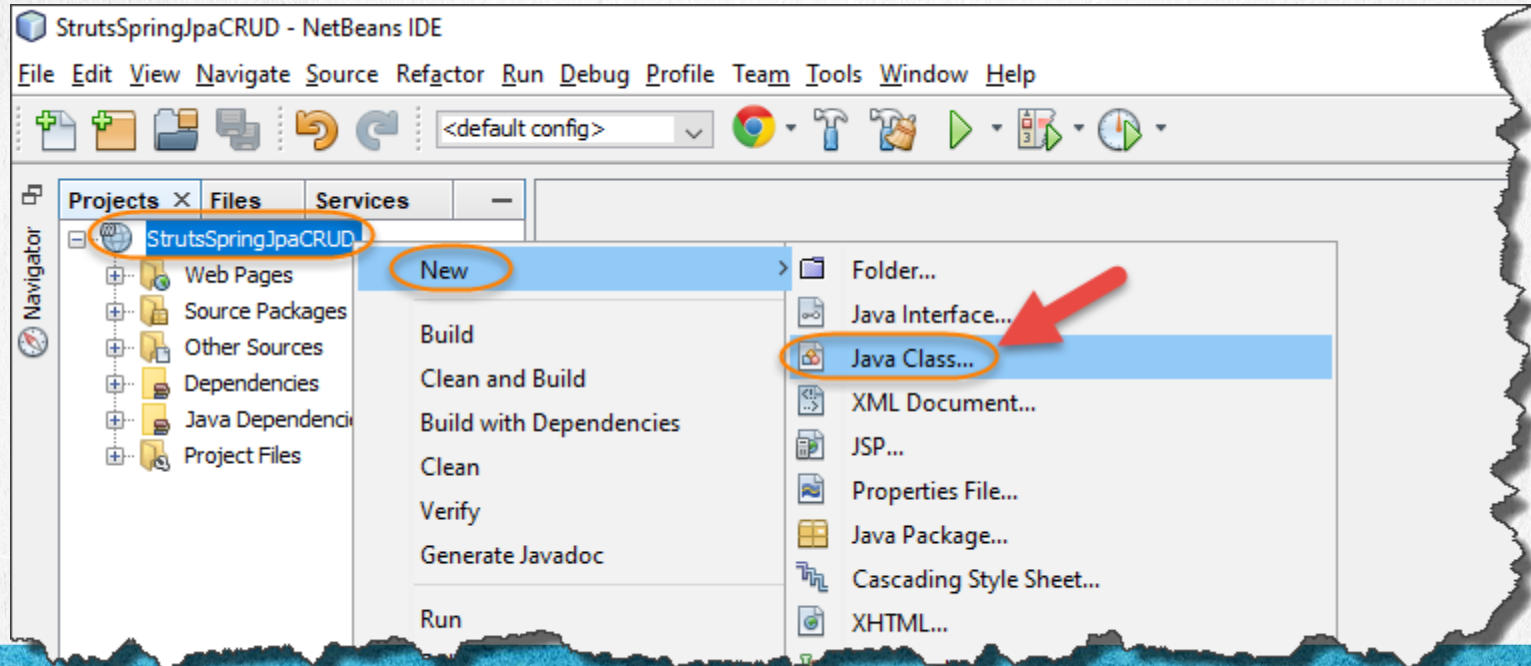


STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

14. CREATE A JAVA INTERFACE

- We create the PersonDao.java interface :

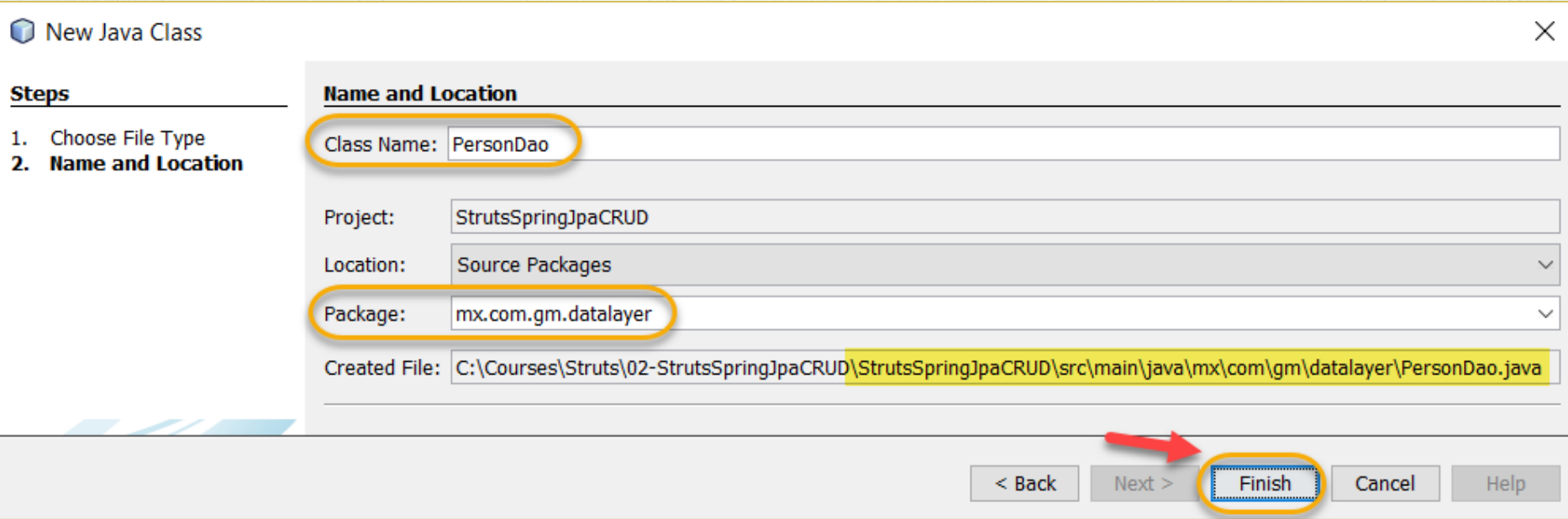


STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

14. CREATE A JAVA INTERFACE

- We create the PersonDao.java interface :



The image shows a 'New Java Class' dialog box with a sidebar on the left and a main form area. The sidebar has a 'Steps' section with two items: '1. Choose File Type' and '2. Name and Location'. The main form area is titled 'Name and Location' and contains several input fields. The 'Class Name' field is set to 'PersonDao'. The 'Project' field is set to 'StrutsSpringJpaCRUD'. The 'Location' field is set to 'Source Packages'. The 'Package' field is set to 'mx.com.gm.datalayer'. The 'Created File' field shows the full path: 'C:\Courses\Struts\02-StrutsSpringJpaCRUD\StrutsSpringJpaCRUD\src\main\java\mx\com\gm\datalayer\PersonDao.java'. At the bottom right, there are four buttons: '< Back', 'Next >', 'Finish', and 'Cancel'. A red arrow points to the 'Finish' button, which is also highlighted with a blue dashed border.

New Java Class

Steps

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

Name and Location

Class Name: PersonDao

Project: StrutsSpringJpaCRUD

Location: Source Packages

Package: mx.com.gm.datalayer

Created File: C:\Courses\Struts\02-StrutsSpringJpaCRUD\StrutsSpringJpaCRUD\src\main\java\mx\com\gm\datalayer\PersonDao.java

< Back Next > **Finish** Cancel Help

15. MODIFY THE FILE

PersonDao.java:

[Click to download](#)

```
package mx.com.gm.datalayer;

import java.util.List;
import mx.com.gm.datalayer.domain.Person;

public interface PersonDao {

    void insertPerson(Person person);

    void updatePerson(Person person);

    void deletePerson(Person person);

    Person findPersonById(long idPerson);

    List<Person> findAllPeople();

    long peopleCounter();

}
```

16. CREATE A JAVA CLASS

We are going to create a Java class called `PersonaDaoImpl.java` that implements the newly created `PersonaDao.java` interface. This class will use the Spring and JPA technology to perform the operations on the database and obtain the connection to the database, as well as the `Persona.java` entity class to be able to communicate with the database and perform the operations described. through the interface.

Let's see how the `PersonaDaoImpl.java` class is:

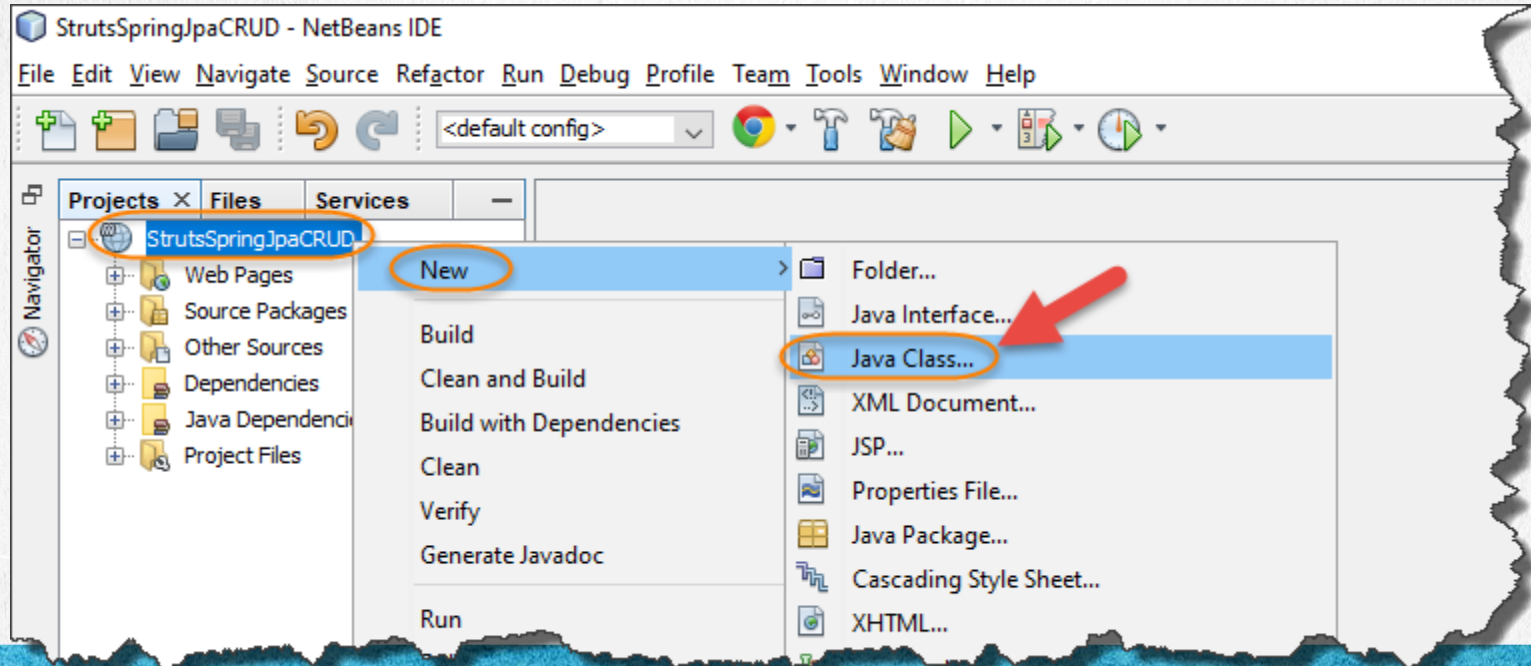


STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

16. CREATE A JAVA CLASS

- We create the PersonaDaoImpl.java class :

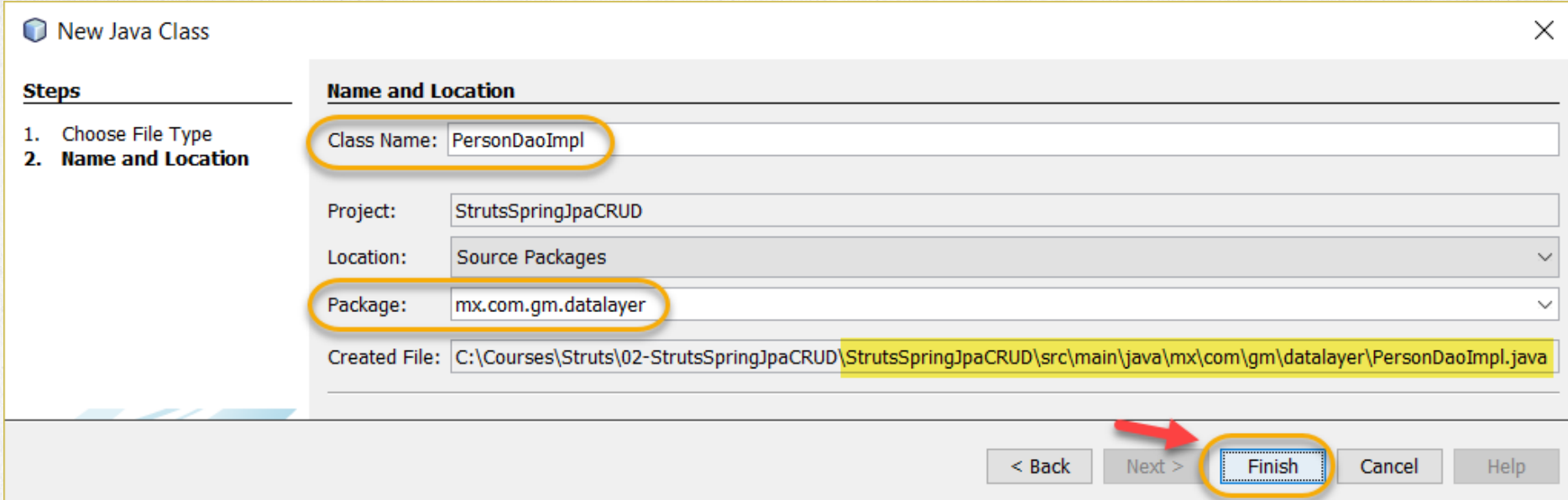


STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

16. CREATE A JAVA CLASS

- We create the PersonaDaoImpl.java class:

A screenshot of the 'New Java Class' dialog box in an IDE. The dialog has a title bar with a blue cube icon and the text 'New Java Class'. On the left, under the 'Steps' section, there are two steps: '1. Choose File Type' and '2. Name and Location'. The 'Name and Location' section contains several input fields: 'Class Name' with the value 'PersonDaoImpl', 'Project' with 'StrutsSpringJpaCRUD', 'Location' with 'Source Packages', and 'Package' with 'mx.com.gm.datalayer'. Below these is a 'Created File' field showing the full path: 'C:\Courses\Struts\02-StrutsSpringJpaCRUD\StrutsSpringJpaCRUD\src\main\java\mx\com\gm\datalayer\PersonDaoImpl.java'. At the bottom right, there are five buttons: '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'. A red arrow points to the 'Finish' button, which is also highlighted with an orange circle. The 'Class Name' and 'Package' fields are also highlighted with orange circles. The 'Created File' field has a yellow background.

Steps

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

Name and Location

Class Name: PersonDaoImpl

Project: StrutsSpringJpaCRUD

Location: Source Packages

Package: mx.com.gm.datalayer

Created File: C:\Courses\Struts\02-StrutsSpringJpaCRUD\StrutsSpringJpaCRUD\src\main\java\mx\com\gm\datalayer\PersonDaoImpl.java

< Back Next > **Finish** Cancel Help

17. MODIFY THE CODE

PersonDaoImpl.java:

[Click to download](#)

```
package mx.com.gm.datalayer;

import java.util.List;
import javax.persistence.*;
import mx.com.gm.datalayer.domain.Person;
import org.apache.logging.log4j.*;
import org.springframework.stereotype.Repository;

@Repository
public class PersonDaoImpl implements PersonDao {

    Logger log = LogManager.getRootLogger();

    @PersistenceContext
    private EntityManager em;

    @Override
    public void insertPerson(Person person) {
        // Insert new object
        em.persist(person);
    }

    @Override
    public void updatePerson(Person person) {
        // Update the object
        em.merge(person);
    }
}
```


17. MODIFY THE CODE

PersonDaoImpl.java:

[Click to download](#)

```
@Override
public void deletePerson(Person person) {
    em.remove(em.merge(person));
}

@Override
public Person findPersonById(long idPerson) {
    return em.find(Person.class, idPerson);
}

@Override
public List<Person> findAllPeople() {
    String jpql = "SELECT p FROM Person p";
    Query query = em.createQuery(jpql);
    //Force to go directly to the database to refresh data
    query.setHint("javax.persistence.cache.storeMode", CacheStoreMode.REFRESH);
    List<Person> people = query.getResultList();
    log.info("list of people:" + people);
    return people;
}
```

17. MODIFY THE CODE

PersonDaoImpl.java:

[Click to download](#)

```
@Override
public long peopleCounter() {
    String query = "select count(p) from Person p";
    Query q = em.createQuery(query);
    long counter = (long) q.getSingleResult();
    log.info("people Counter: " + counter);
    return counter;
}
}
```

18. CREATE A JAVA INTERFACE

We are going to create a Java `PersonaService.java` interface. Remember that it is a good practice to program using interfaces to separate the layers of our Java application. So we will create an interface and then its implementation.

Let's see how this `PersonaService.java` interface is:



STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

18. CREATE A JAVA INTERFACE

- We create the PersonaService.java interface:

New Java Class

Steps

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

Name and Location

Class Name:

Project:

Location:

Package:

Created File:

< Back Next > **Finish** Cancel Help

19. MODIFY THE FILE

PersonService.java:

[Click to download](#)

```
package mx.com.gm.businesslayer;

import java.util.List;
import mx.com.gm.datalayer.domain.Person;

public interface PersonService {

    public List<Person> listPeople();

    public Person findPeople(Person person);

    public void addPerson(Person person);

    public void modifyPerson(Person person);

    public void deletePerson(Person person);

    public long countPeople();
}
```

20. CREATE A JAVA CLASS

We are going to create a Java class called PersonServiceImpl.java that implements the newly created PersonService.java interface.

This class will use Spring's technology to automatically handle the concept of transactions.

Let's see how the PersonServiceImpl.java class is

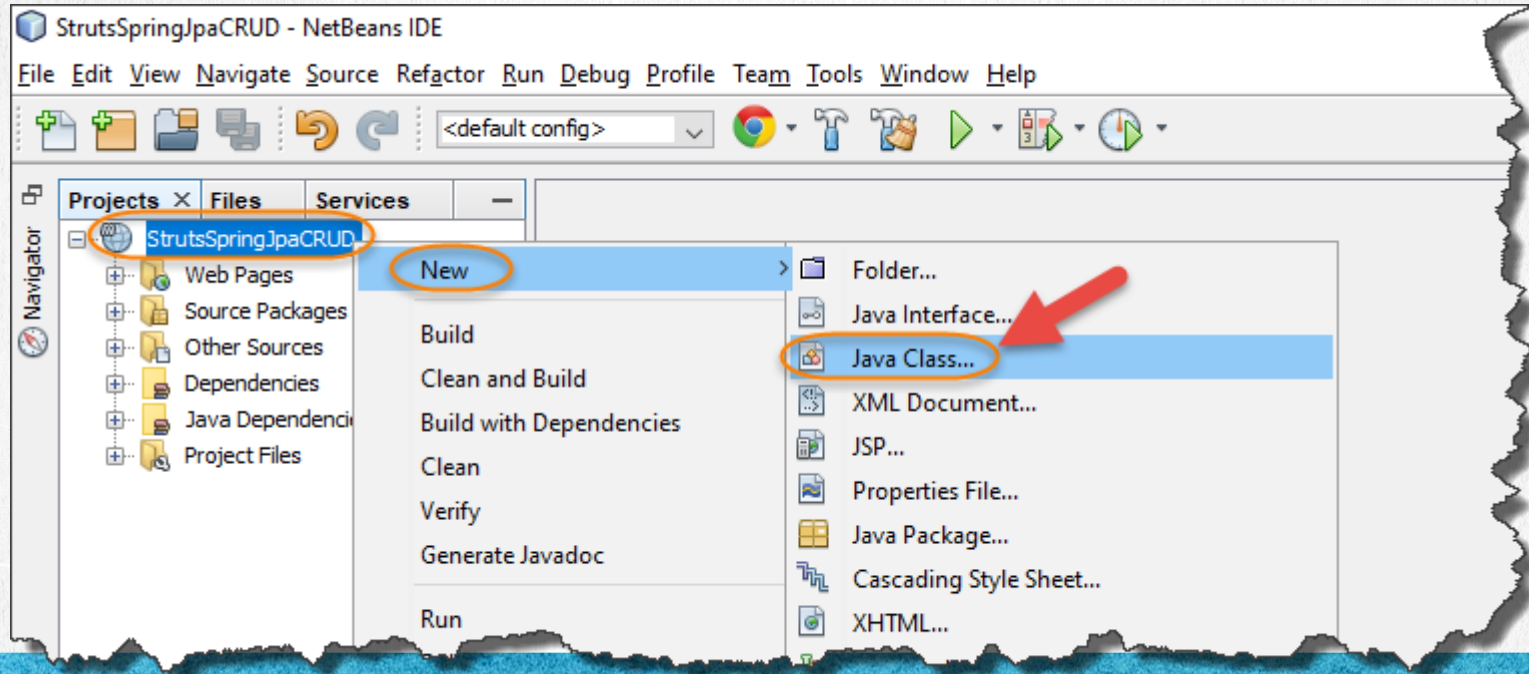


STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

20. CREATE A JAVA CLASS

- We create the PersonServiceImpl.java class :



STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

20. CREATE A JAVA CLASS

- We create the PersonServiceImpl.java class:

New Java Class

Steps

1. Choose File Type

2. Name and Location

Name and Location

Class Name: PersonServiceImpl

Project: StrutsSpringJpaCRUD

Location: Source Packages

Package: mx.com.gm.businesslayer

Created File: C:\Courses\Struts\02-StrutsSpringJpaCRUD\StrutsSpringJpaCRUD\src\main\java\mx\com\gm\businesslayer\PersonServiceImpl.java

< Back

Next >

Finish

Cancel

Help

STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

21. MODIFY THE CODE

PersonServiceImpl.java:

[Click to download](#)

```
package mx.com.gm.businesslayer;

import java.util.List;
import mx.com.gm.datalayer.PersonDao;
import mx.com.gm.datalayer.domain.Person;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

@Service("personService")
@Transactional
public class PersonServiceImpl implements PersonService {

    @Autowired
    private PersonDao personDao;

    @Override
    public List<Person> listPeople() {
        return personDao.findAllPeople();
    }

    @Override
    public Person findPeople(Person person) {
        return personDao.findPersonById(person.getIdPerson());
    }
}
```


21. MODIFY THE CODE

[PersonServiceImpl.java:](#)

[Click to download](#)

```
@Override
public void addPerson(Person person) {
    personDao.insertPerson(person);
}

@Override
public void modifyPerson(Person person) {
    personDao.updatePerson(person);
}

@Override
public void deletePerson(Person person) {
    personDao.deletePerson(person);
}

@Override
public long countPeople() {
    return personDao.peopleCounter();
}
}
```

22. CREAR UNA CLASE JAVA

The PersonAction.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.

We will obtain the model with the help of the service interface, which we will inject with the help of Spring and the integration plug-in between Struts and Spring that we add to the pom.xml file

Recall that we must respect the conventions of Struts2, so this class must be within a package that contains the word: struts, struts2, action or actions, also must end with the word Action.

This class does have changes with respect to the basic exercise, since this Action class is precisely the one that will support all the changes in the view that we want to apply. From changes in the names of our classes, to new functions that we will use to be able to apply the CRUD (Create-Read-Update-Delete) operations, that is, the operations of: listing, adding, modifying and deleting records.

PASO 22. CREAR UNA CLASE JAVA

In the class `PersonAction.java` we have added several methods of type `execute` and not just one. In such a way that for each action that we execute from the navigator, this class of type `Action` will be in charge of processing said actions.

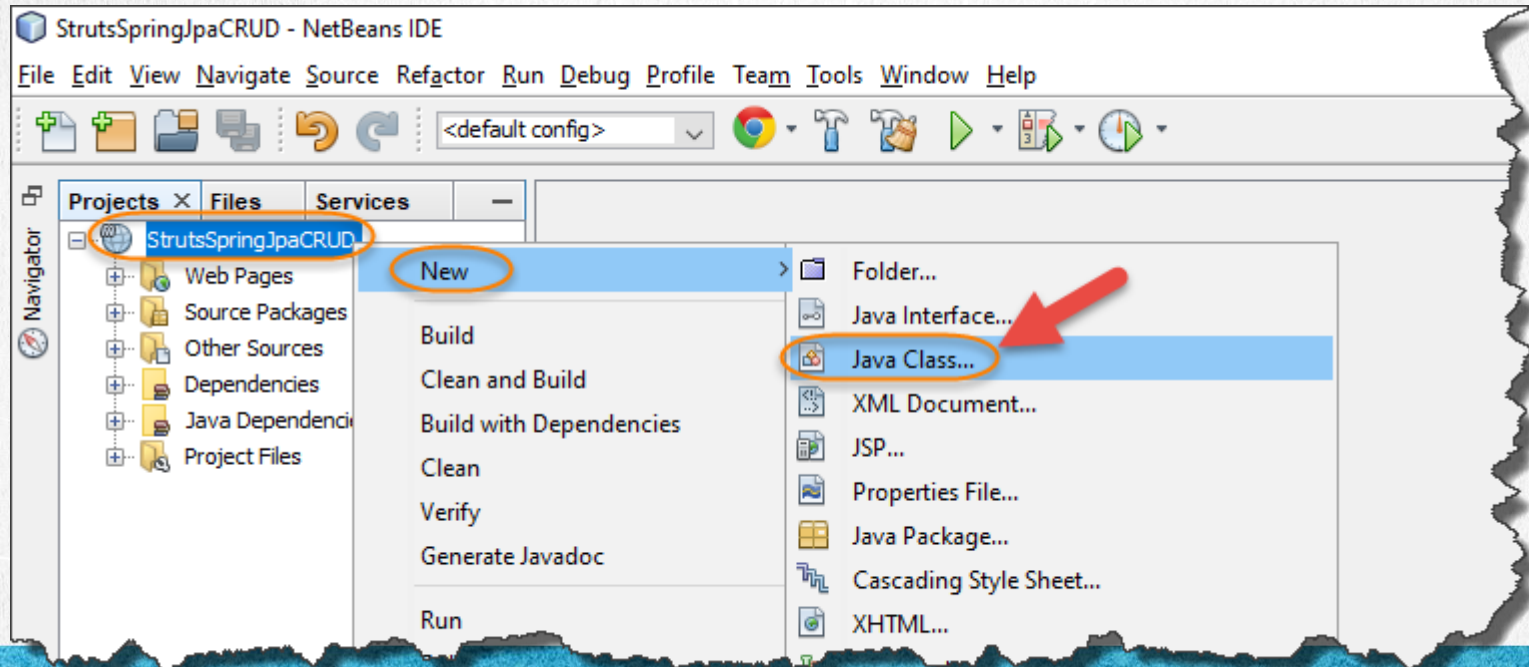
In this way we are demonstrating that an `Action` type class can have multiple methods of type `execute`, but that they can have any name, and with the help of the `@Action` annotation is that we define the name of the path that will process the method indicated with this annotation. So it is not necessary that the name of the path and the name of the method match.

Another characteristic that we see in these methods is that in some cases it was necessary to specify the view that is executed after having finished executing said method. To do this we made the `@Result` annotation and specified some attributes depending on the type of view selected, such as the name of the result, the location of the view and in some cases the type of result such as `redirect`, so that it can be executed again a path the `Action` method and not only the JSP is executed, since in the cases of adding, modifying or deleting a listing from the list, we need to redeploy the new updated list, and for this it is not enough to directly call the JSP for a list, but we must go through the `list` method of the `PersonAction` class and thus refresh the list of people.

Let's see how our `PersonAction` class is.

22. CREATE A JAVA CLASS

- We create the PersonAction.java class:

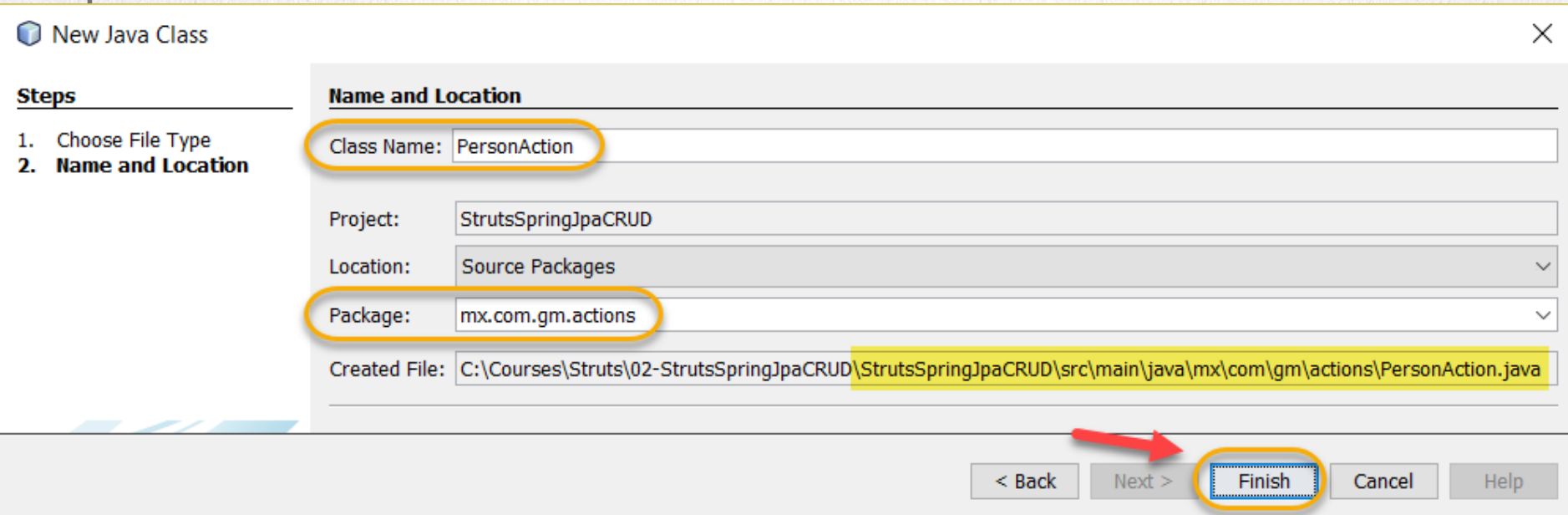


STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

22. CREATE A JAVA CLASS

- We create the PersonAction.java class:



The image shows a 'New Java Class' dialog box with a title bar containing a blue cube icon and the text 'New Java Class'. On the left, a 'Steps' section lists '1. Choose File Type' and '2. Name and Location'. The main area is titled 'Name and Location' and contains several input fields: 'Class Name' with the value 'PersonAction', 'Project' with 'StrutsSpringJpaCRUD', 'Location' with 'Source Packages', and 'Package' with 'mx.com.gm.actions'. The 'Created File' field shows the full path: 'C:\Courses\Struts\02-StrutsSpringJpaCRUD\StrutsSpringJpaCRUD\src\main\java\mx\com\gm\actions\PersonAction.java'. At the bottom right, there are buttons for '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'. A red arrow points to the 'Finish' button, which is also highlighted with a blue dashed border.

Steps

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

Name and Location

Class Name: PersonAction

Project: StrutsSpringJpaCRUD

Location: Source Packages

Package: mx.com.gm.actions

Created File: C:\Courses\Struts\02-StrutsSpringJpaCRUD\StrutsSpringJpaCRUD\src\main\java\mx\com\gm\actions\PersonAction.java

< Back Next > **Finish** Cancel Help

23. MODIFY THE CODE

PersonAction.java:

[Click to download](#)

```
package mx.com.gm.actions;

import com.opensymphony.xwork2.ActionSupport;
import java.util.List;
import mx.com.gm.businesslayer.PersonService;
import mx.com.gm.datalayer.domain.Person;
import org.apache.logging.log4j.*;
import org.apache.struts2.convention.annotation.*;
import org.springframework.beans.factory.annotation.Autowired;

public class PersonAction extends ActionSupport {

    private Person person;

    private List<Person> people;

    @Autowired
    private PersonService personService;

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

    @Action(value = "/list", results = {
        @Result(name = "people", location = "/WEB-INF/content/people.jsp") })
    public String list() {
        this.people = personService.listPeople();
        return "people";
    }
}
```


23. MODIFY THE CODE

PersonAction.java:

[Click to download](#)

```
@Action(value = "/addPerson", results = {
    @Result(name = "person", location = "/WEB-INF/content/person.jsp")})
public String add() {
    //We create a new object of type person
    person = new Person();
    return "person";
}

@Action(value = "/editPerson", results = {
    @Result(name = "person", location = "/WEB-INF/content/person.jsp")})
public String edit() {
    person = personService.findPeople(person);
    return "person";
}

@Action(value = "/deletePerson", results = {
    @Result(name = "success", location = "list", type = "redirect")})
public String delete() {
    //We retrieve the person object, since we only have the idPerson
    log.info("Method to eliminate person before recovering:" + person);
    person = personService.findPeople(person);
    log.info("Method to eliminate person after recovering:" + person);
    personService.deletePerson(person);
    return SUCCESS;
}
```

23. MODIFY THE CODE

PersonAction.java:

Click to download

```
//It is not enough to send to the JSP, but to the action of list
//for that reason we redirected to the action of list
@Action(value = "/savePerson", results = {
    @Result(name = "success", location = "list", type = "redirect")})
public String save() {
    //We differentiate the action of adding or editing with the idPerson
    if (person.getIdPerson() == null) {
        personService.addPerson(person);
    } else {
        personService.modifyPerson(person);
    }
    return SUCCESS;
}

public Person getPerson() {
    return person;
}

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

public List<Person> getPeople() {
    return people;
}

public void setPeople(List<Person> people) {
    this.people = people;
}
}
```

24. CREATE THE PROPERTIES FILE

We create a file `PersonAction.properties`. This file has the messages that we will use in the Struts JSP pages.

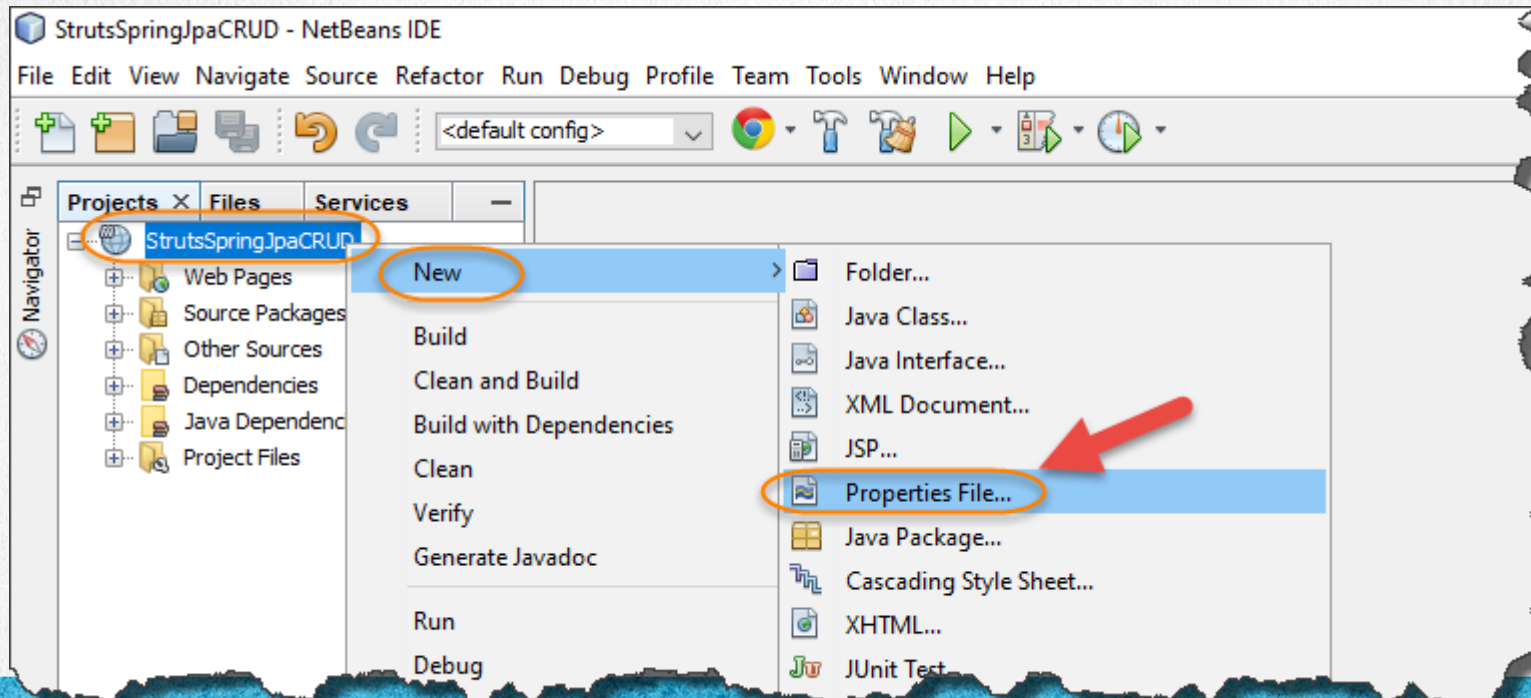
Let's see how is this file `PersonAction.properties`

This file was also modified according to the previous basic exercise, to support the new list and the operations to add and modify.



24. CREATE THE PROPERTIES FILE

- We create the file PersonAction.properties as follows:



CURSO STRUTS FRAMEWORK

www.globalmentoring.com.mx

24. CREATE THE PROPERTIES FILE

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

New Properties File

Steps

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

Name and Location

File Name:

Project:

Folder:

Created File:

25. MODIFY THE FILE

PersonAction.properties:

[Click to download](#)

```
pform.title: People with Struts 2
pform.counter: Records Found
pform.send: Send
pform.detail: Person Detail
pform.add: Add Person
pform.list: Return to the list of people
pform.edit: Edit
pform.delete: Delete
p.idPerson: idPerson
p.name: Name
```


26. 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: [list](#)

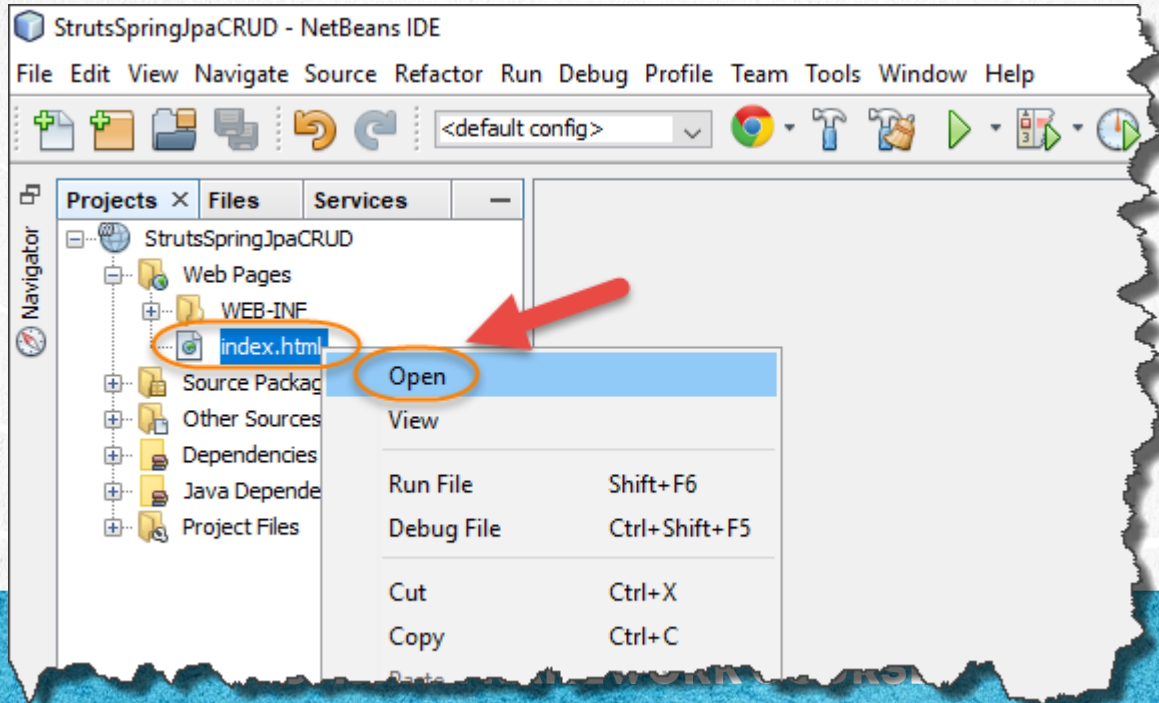


STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

26. 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, as shown:



26. MODIFY THE FILE

[index.html:](#)

[Click to download](#)

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

STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

27. CREATE A JSP FILE

Now we create the file: people.jsp.

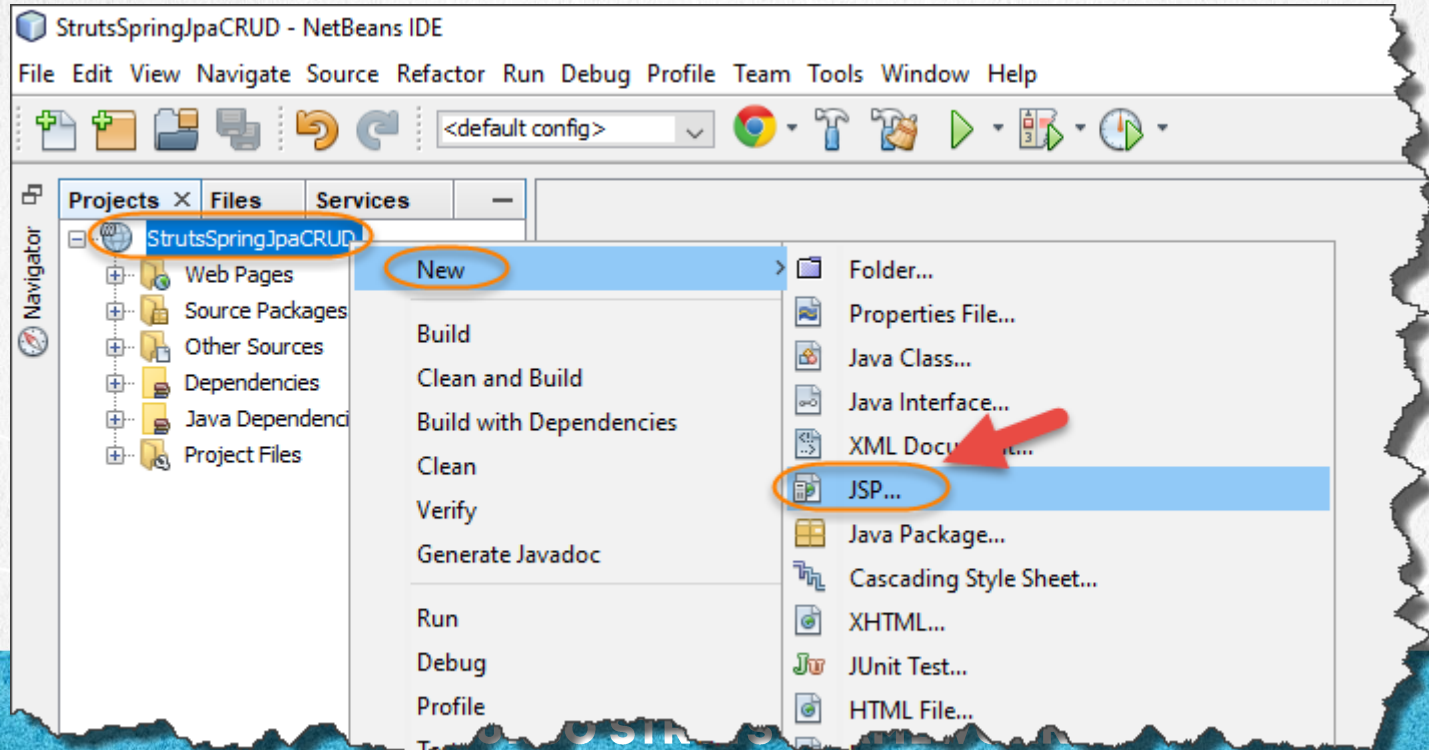
This is the first JSP that we are going to create, which will serve us to show the list of person type objects, and will have the links to add, modify and eliminate a record of type person.

We deposit this JSP in the folder /WEB-INF/content, but it's not necessary as we are not following the conventions.

This file also changed according to the basic previous exercise, since we have added the Add link, as well as the Edit and Delete columns in each of the records in the list. Let's see how was our file people.jsp

27. CREATE A JSP FILE

- Create the people.jsp file:



27. CREATE A JSP FILE

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

New JSP

Steps

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

Name and Location

File Name:

Project:

Location:

Folder:

Created File:

Options:

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

☐ JSP Document (XML Syntax)

Description:

28. MODIFY THE CODE

people.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="pform.title" /></title>
  </head>
  <body>
    <h1><s:text name="pform.title" /></h1>
    <a href="<s:url action="addPerson"/>"><s:text name="pform.add" /></a>
    <s:if test="people.size() > 0">
      <div>
        <table border="1">
          <tr>
            <th><s:text name="p.idPerson" /></th>
            <th><s:text name="p.name" /></th>
            <th><s:text name="pform.edit" /></th>
            <th><s:text name="pform.delete" /></th>
          </tr>
```

CURSO STRUTS FRAMEWORK

www.globalmentoring.com.mx

28. MODIFY THE CODE

people.jsp:

Click to download

```
<s:iterator value="people">
  <tr>
    <td><s:property value="idPerson" /></td>
    <td><s:property value="name" /></td>
    <td>
      <s:url action="editPerson" var="editURL">
        <s:param name="person.idPerson" value="%{idPerson}"></s:param>
      </s:url>
      <s:a href="%{editURL}"><s:text name="pform.edit" /></s:a>
    </td>
    <td>
      <s:url action="deletePerson" var="deleteURL">
        <s:param name="person.idPerson" value="%{idPerson}"></s:param>
      </s:url>
      <s:a href="%{deleteURL}"><s:text name="pform.delete" /></s:a>
    </td>
  </tr>
</s:iterator>
</table>
</div>
</s:if>
</body>
</html>
```

29. CREATE A JSP FILE

Now we create the file: person.jsp.

This is the second JSP that we will create, and we will use it basically to show the detail of a person-type object. And we will use it both to add or modify a person type object.

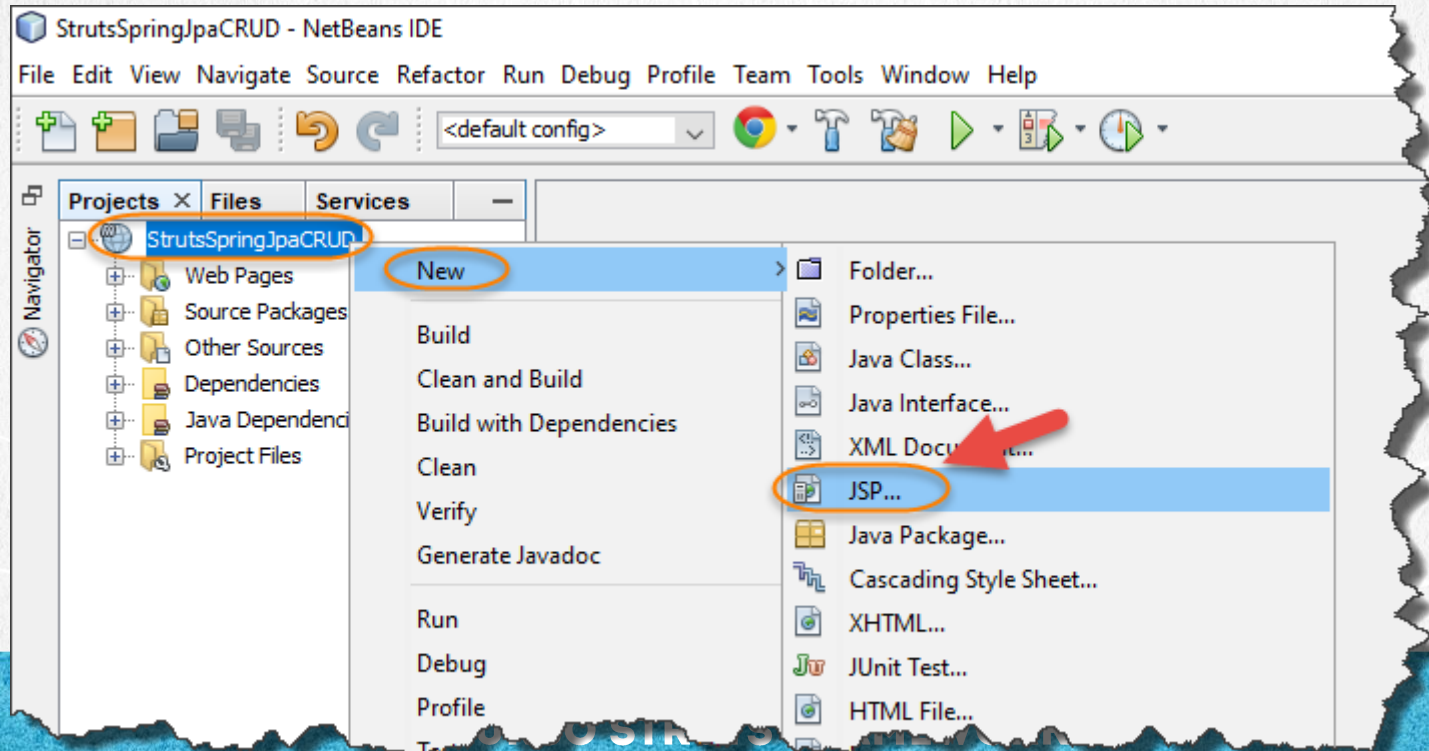
The only difference between adding and modifying a person is the values of idPerson, if it is equal to null it means that it does not yet have a representation in the database and therefore an insert is made, that is, it is added to the database. On the other hand, if the value is different from null, it means that there is already a database representation of that Person type object, then we are modifying the object (update).

It is important to note the use of <s: hidden> to include the idPerson field. So even if we do not observe this value in the form, it is being sent to the browser as a hidden field, being able to have a null value if it is a new record, or a value other than null if it is a record that is being modified.

This file is not necessary to deposit it in the folder [/WEB-INF/content](#), however we will deposit it there to continue maintaining an order in the organization of JSPs files.

29. CREATE A JSP FILE

- Create a person.jsp file:



29. CREATE A JSP FILE

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

New JSP

Steps

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

Name and Location

File Name:

Project:

Location:

Folder:

Created File:

Options:

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

☐ JSP Document (XML Syntax)

Description:

A JSP file using JSP standard syntax.

30. 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><s:text name="pform.detail" /></title>
  </head>
  <body>
    <h1><s:text name="pform.detail" /></h1>
    <a href="<s:url action="list"/>"><s:text name="pform.list" /></a>

    <s:form action="savePerson">
      <s:hidden name="person.idPerson" />

      <s:textfield name="person.name" key="p.name" />
      <s:submit action="savePerson" key="pform.send"/>
    </s:form>
  </body>
</html>
```

CURSO STRUTS FRAMEWORK

www.globalmentoring.com.mx

31. 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.

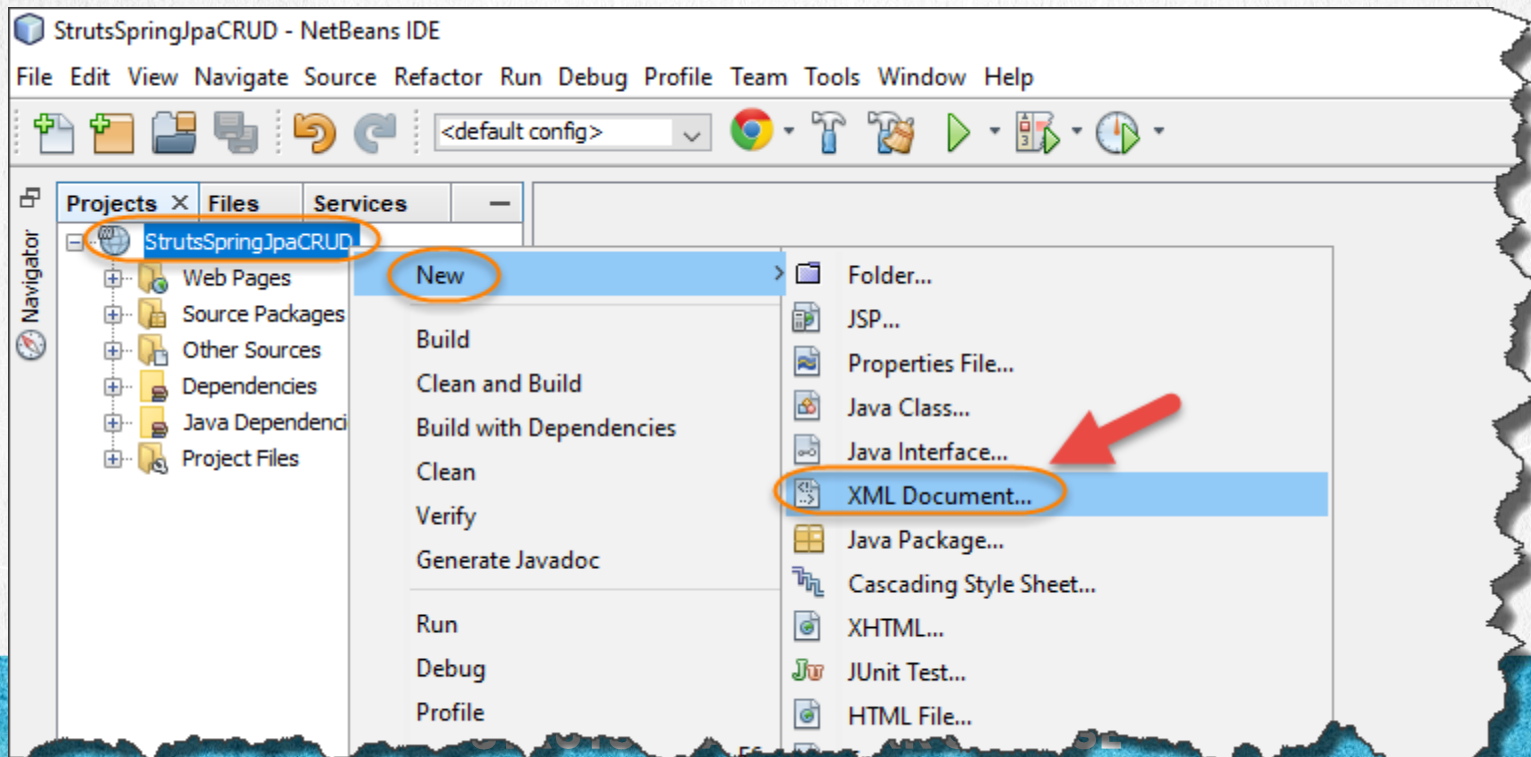


STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

31. CREATE THE LOG4J2.XML FILE

- We create the log4j2.xml file as follows:



31. 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:

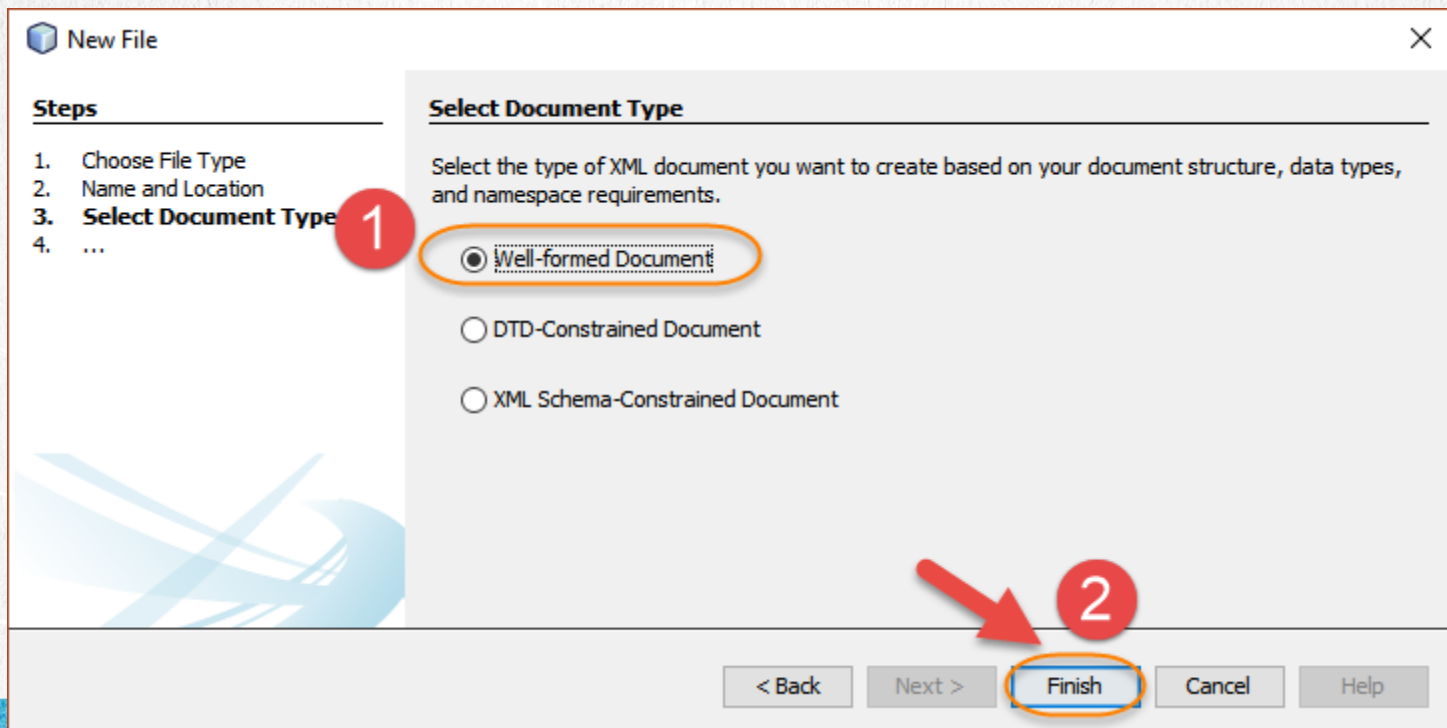
Project:

Folder:

Created File:

31. CREATE THE LOG4J2.XML FILE

- We select the option shown :



STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

32. MODIFY THE FILE

log4j2.xml:

[Click to download](#)

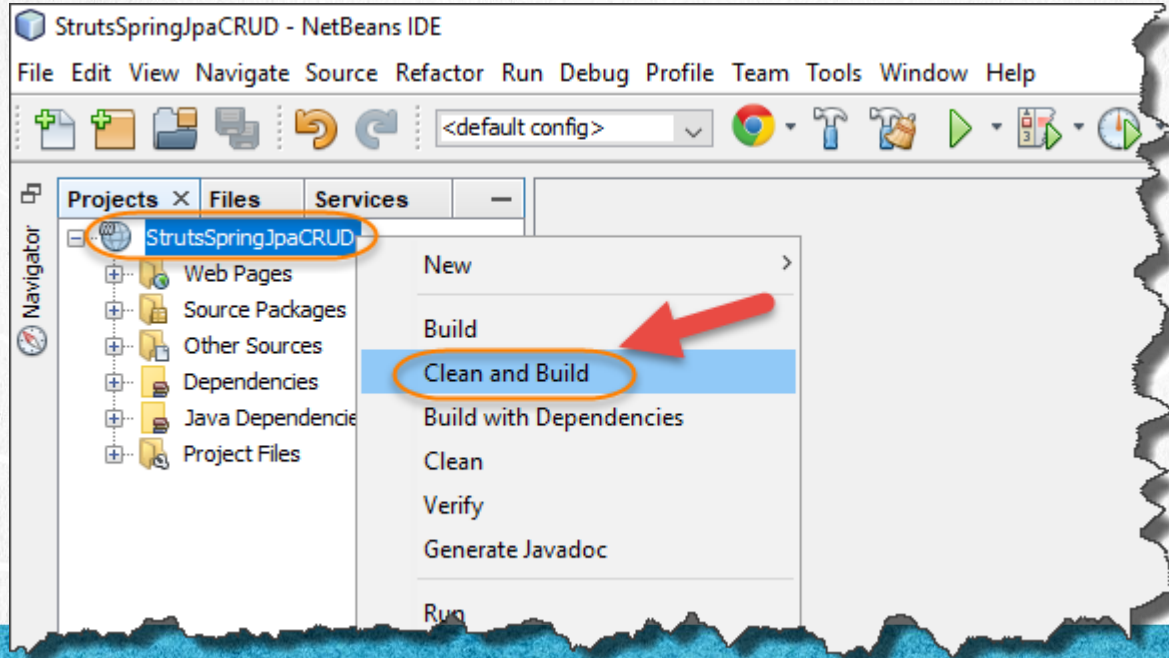
```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration>
  <Appenders>
    <Console name="STDOUT" target="SYSTEM_OUT">
      <PatternLayout pattern="%F:%L) - %m%n"/>
    </Console>
  </Appenders>
  <Loggers>
    <Logger name="com.opensymphony.xwork2" level="info"/>
    <Logger name="org.apache.struts2"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

STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

33. EXECUTE CLEAN & BUILD

- Before running the application, we make Clean & Build to make sure we have everything ready:

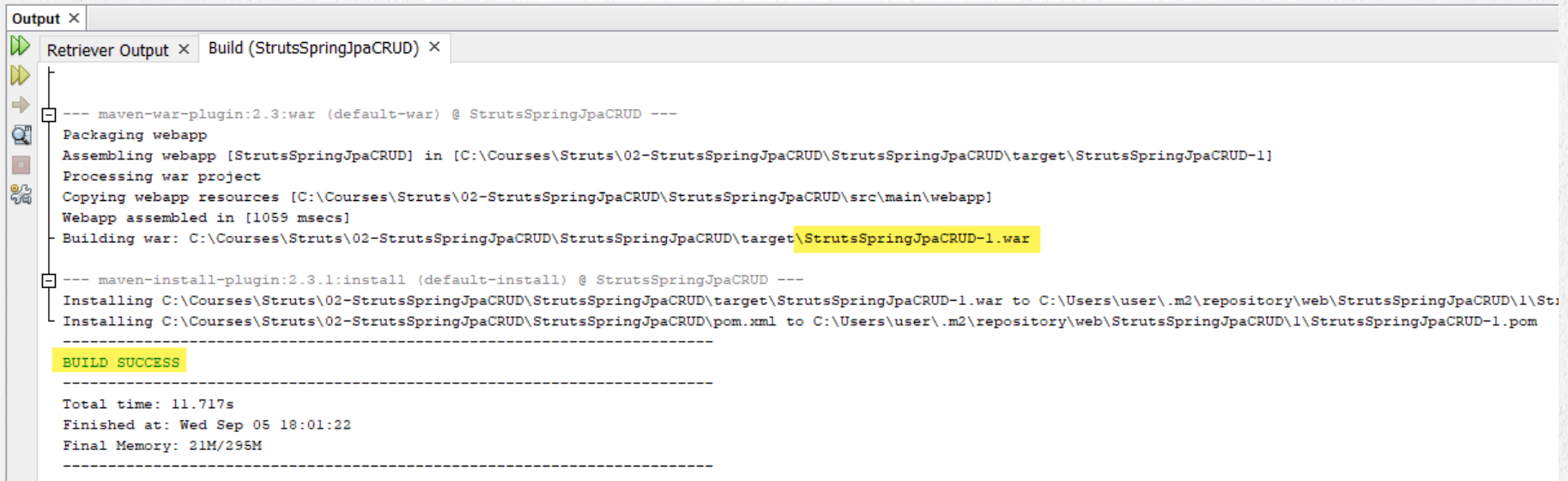


STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

33. EXECUTE CLEAN & BUILD

- Before running the application, we do Clean & Build. We observe that the process has been executed successfully (Build Success):

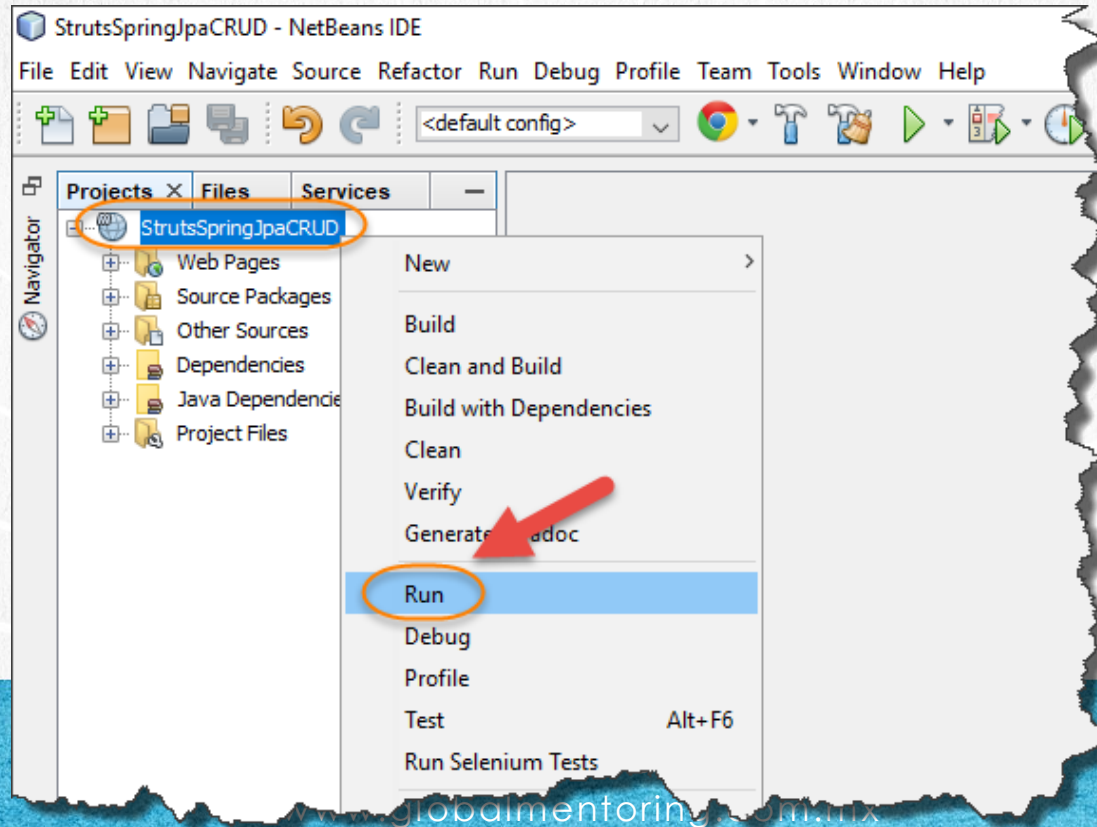


The screenshot shows the 'Output' window of an IDE with two tabs: 'Retriever Output' and 'Build (StrutsSpringJpaCRUD)'. The 'Build' tab is active, displaying the output of a Maven build. The output shows the execution of the 'maven-war-plugin' and 'maven-install-plugin'. The 'maven-war-plugin' output includes steps like 'Packaging webapp', 'Assembling webapp', 'Processing war project', 'Copying webapp resources', and 'Building war: C:\Courses\Struts\02-StrutsSpringJpaCRUD\StrutsSpringJpaCRUD\target\StrutsSpringJpaCRUD-1.war'. The 'maven-install-plugin' output shows the installation of the war file and the pom.xml to the local repository. The build concludes with 'BUILD SUCCESS', 'Total time: 11.717s', 'Finished at: Wed Sep 05 18:01:22', and 'Final Memory: 21M/295M'.

```
Output X
Retriever Output X Build (StrutsSpringJpaCRUD) X
--- maven-war-plugin:2.3:war (default-war) @ StrutsSpringJpaCRUD ---
Packaging webapp
Assembling webapp [StrutsSpringJpaCRUD] in [C:\Courses\Struts\02-StrutsSpringJpaCRUD\StrutsSpringJpaCRUD\target\StrutsSpringJpaCRUD-1]
Processing war project
Copying webapp resources [C:\Courses\Struts\02-StrutsSpringJpaCRUD\StrutsSpringJpaCRUD\src\main\webapp]
Webapp assembled in [1059 msecs]
Building war: C:\Courses\Struts\02-StrutsSpringJpaCRUD\StrutsSpringJpaCRUD\target\StrutsSpringJpaCRUD-1.war
--- maven-install-plugin:2.3.1:install (default-install) @ StrutsSpringJpaCRUD ---
Installing C:\Courses\Struts\02-StrutsSpringJpaCRUD\StrutsSpringJpaCRUD\target\StrutsSpringJpaCRUD-1.war to C:\Users\user\.m2\repository\web\StrutsSpringJpaCRUD\1\StrutsSpringJpaCRUD-1.war
Installing C:\Courses\Struts\02-StrutsSpringJpaCRUD\StrutsSpringJpaCRUD\pom.xml to C:\Users\user\.m2\repository\web\StrutsSpringJpaCRUD\1\StrutsSpringJpaCRUD-1.pom
-----
BUILD SUCCESS
-----
Total time: 11.717s
Finished at: Wed Sep 05 18:01:22
Final Memory: 21M/295M
-----
```

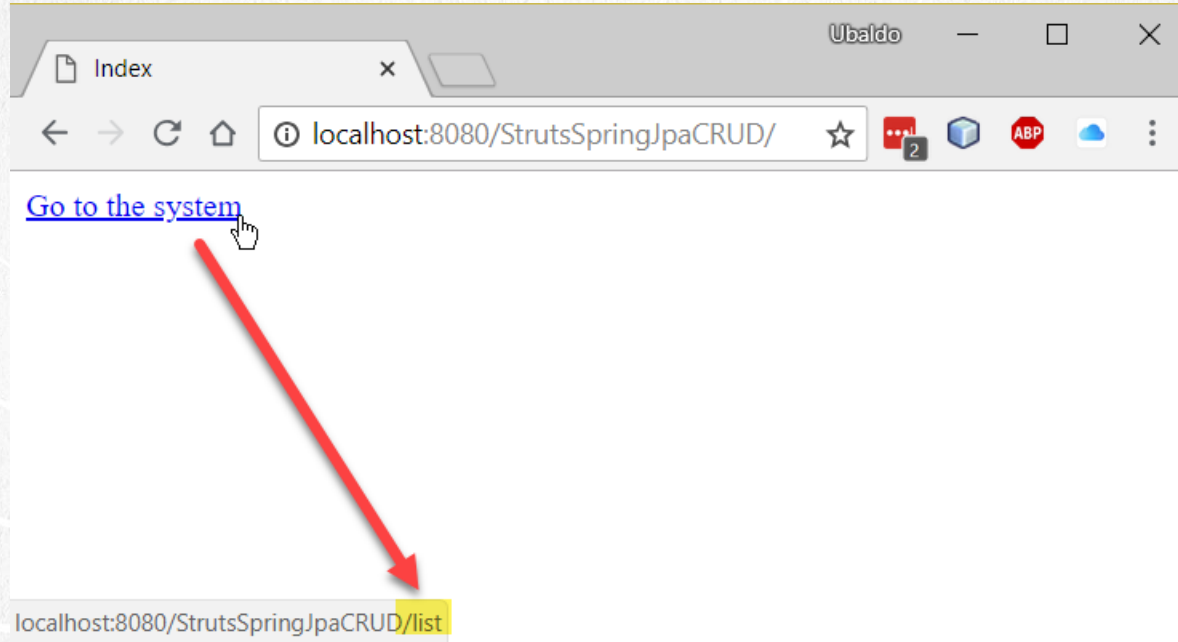
34. EXECUTE THE APPLICATION

- We execute the StrutsSpringJpaCRUD application as follows:



34. EXECUTE THE APPLICATION

- We run the application as follows:



34. EXECUTE THE APPLICATION

- We will have to observe the list of people as follows. The data may vary depending on the information we have in the person table in the database. If we click on the Add Person link, it will take us to another view:

People with Struts 2

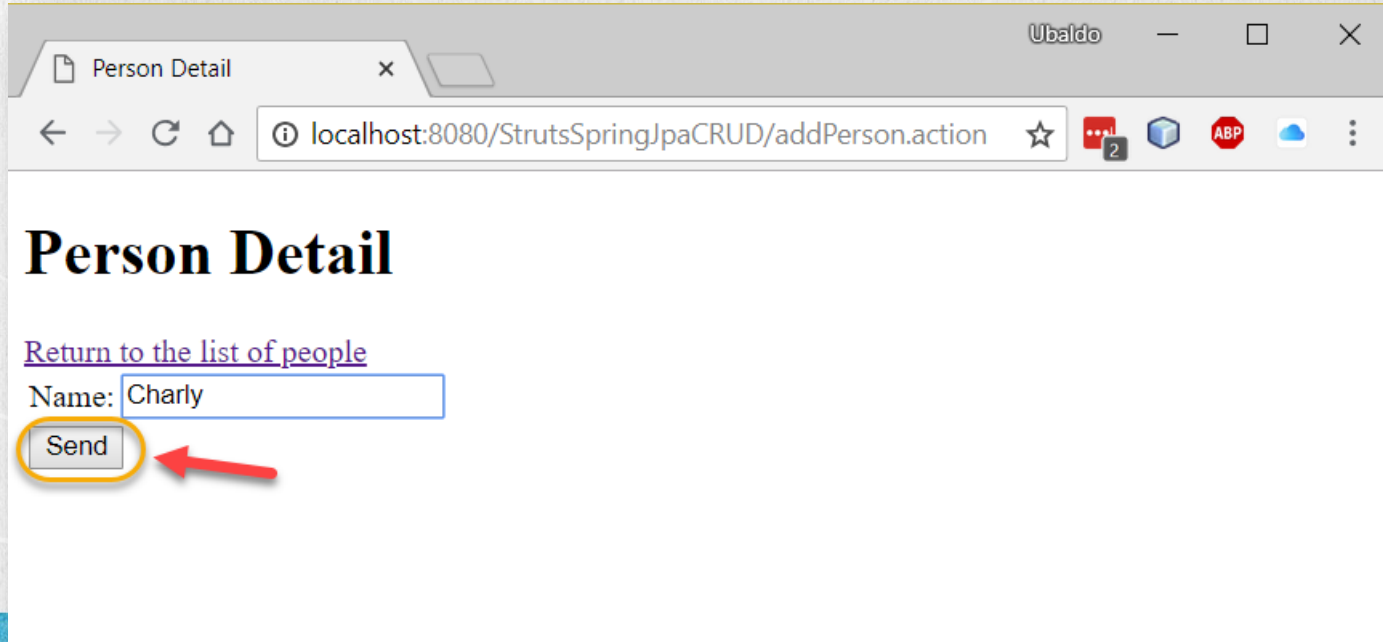
[Add Person](#)

idPerson	Name	Edit	Delete
1	John	Edit	Delete
2	Katty	Edit	Delete

localhost:8080/StrutsSpringJpaCRUD/addPersona.action

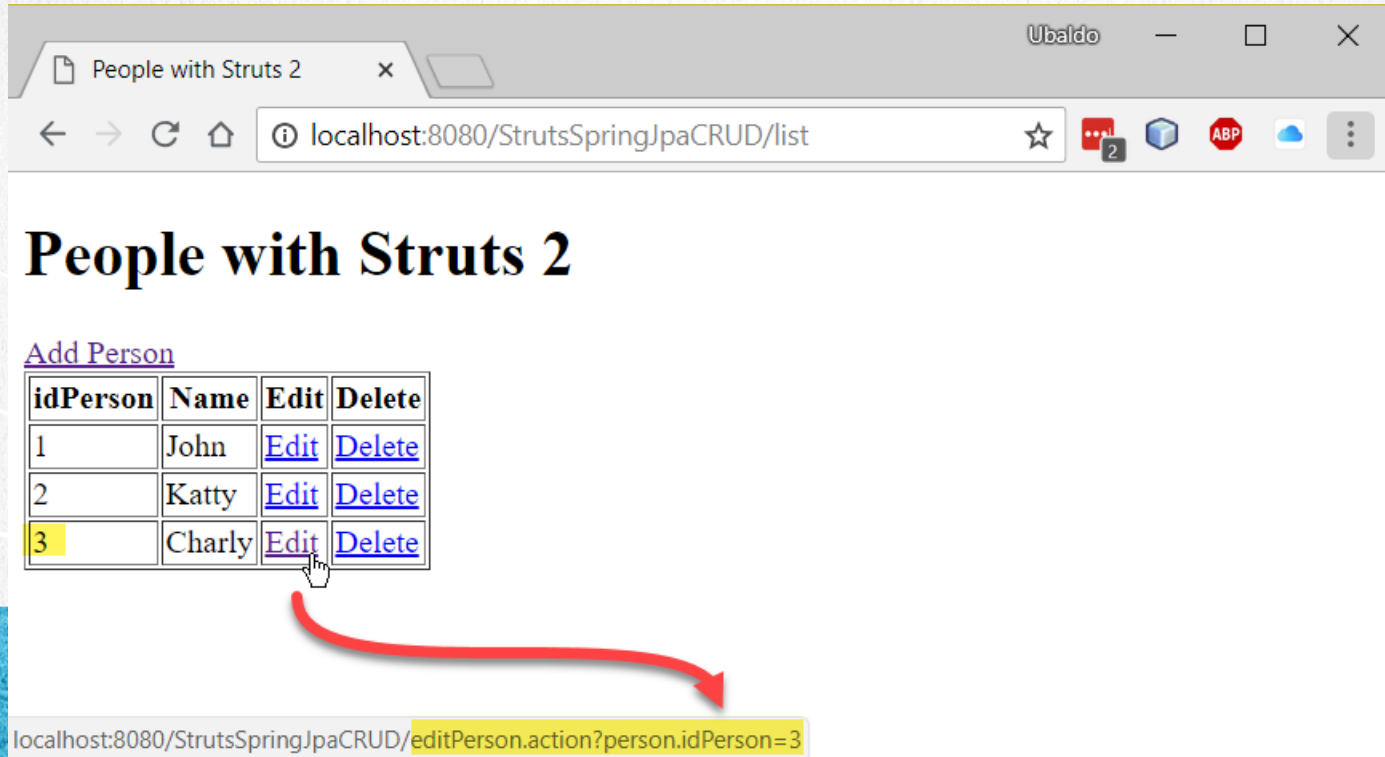
34. EXECUTE THE APPLICATION

- In this screen we can return to the list of people, or send the new data to add a new object of Person type as we can see:



34. EXECUTE THE APPLICATION

- After adding a new record, we should see it reflected in our list. We can also edit or delete a listing from the list. We can see that in case of clicking on Edit on any record, we select which is the idPerson that we want to modify:



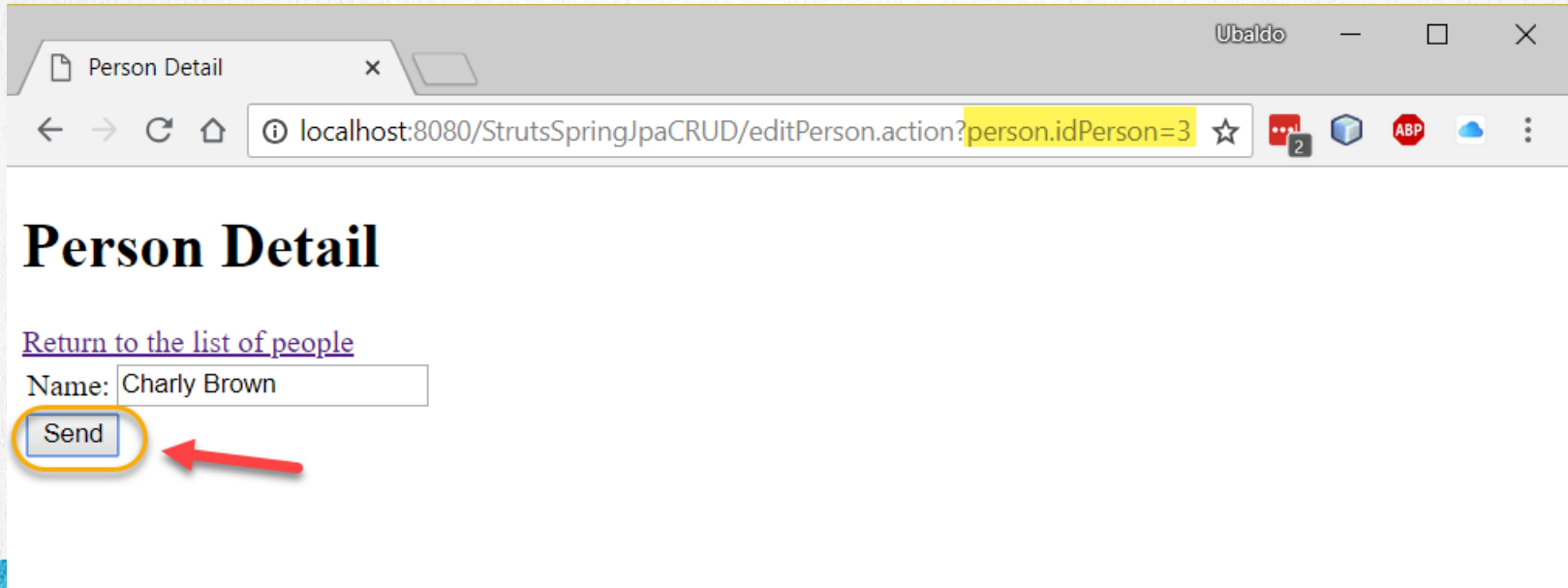
The screenshot shows a web browser window with the title "People with Struts 2". The address bar displays "localhost:8080/StrutsSpringJpaCRUD/list". The page content includes a link "Add Person" and a table with the following data:

idPerson	Name	Edit	Delete
1	John	Edit	Delete
2	Katty	Edit	Delete
3	Charly	Edit	Delete

A red arrow points from the "Edit" link for the record with idPerson 3 to the address bar, which now shows "localhost:8080/StrutsSpringJpaCRUD/editPerson.action?person.idPerson=3".

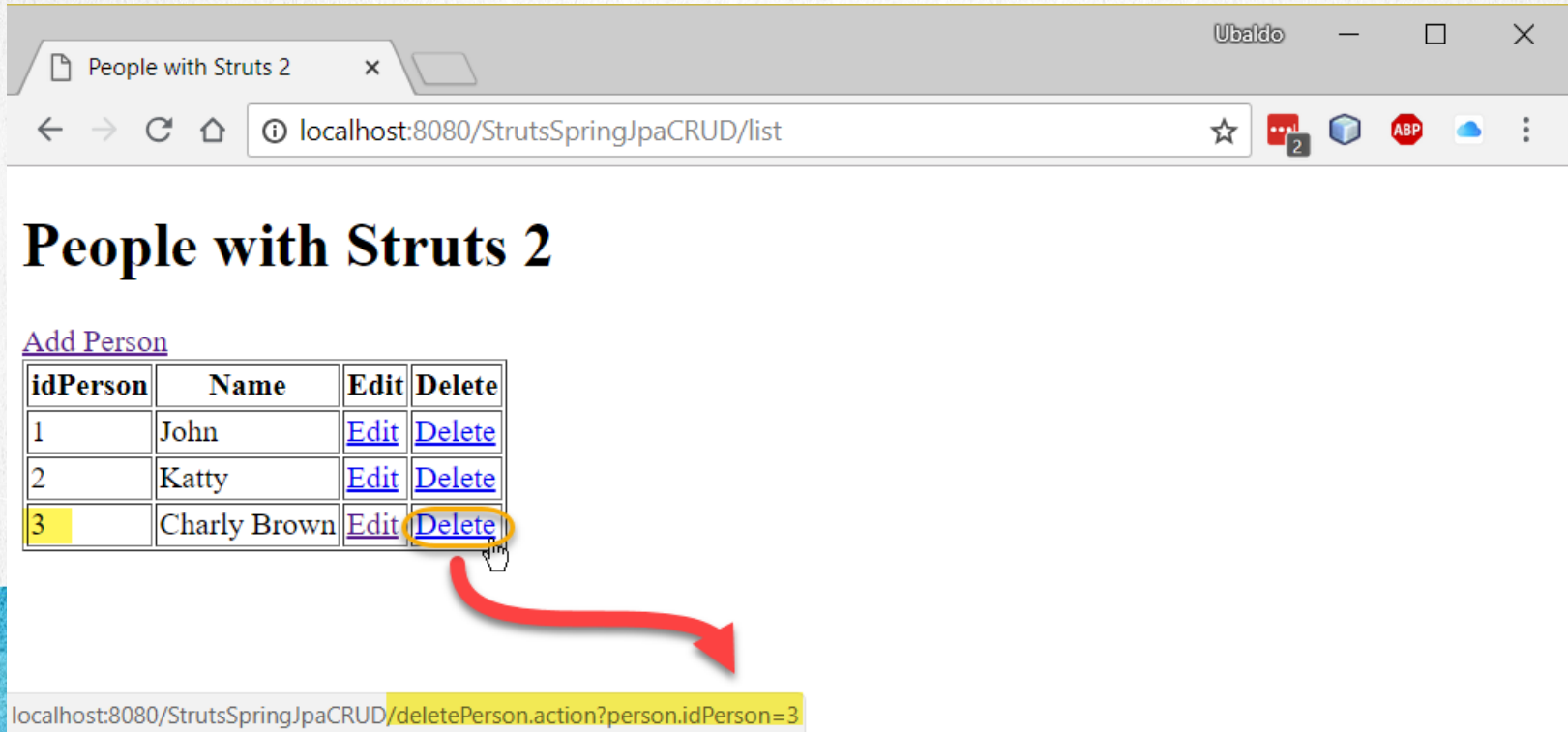
34. EXECUTE THE APPLICATION

- In this screen we see again the detail of the selected object, however since it already has an idPerson, instead of making an insert, an update will be made. This logic is inside the PersonAction object. We keep the indicated changes:



34. EXECUTE THE APPLICATION

- After modifying the selected record, we should see the changes reflected. We can also delete a record from the list. If we click on Delete on any record, we select which is the idPerson that we want to eliminate:



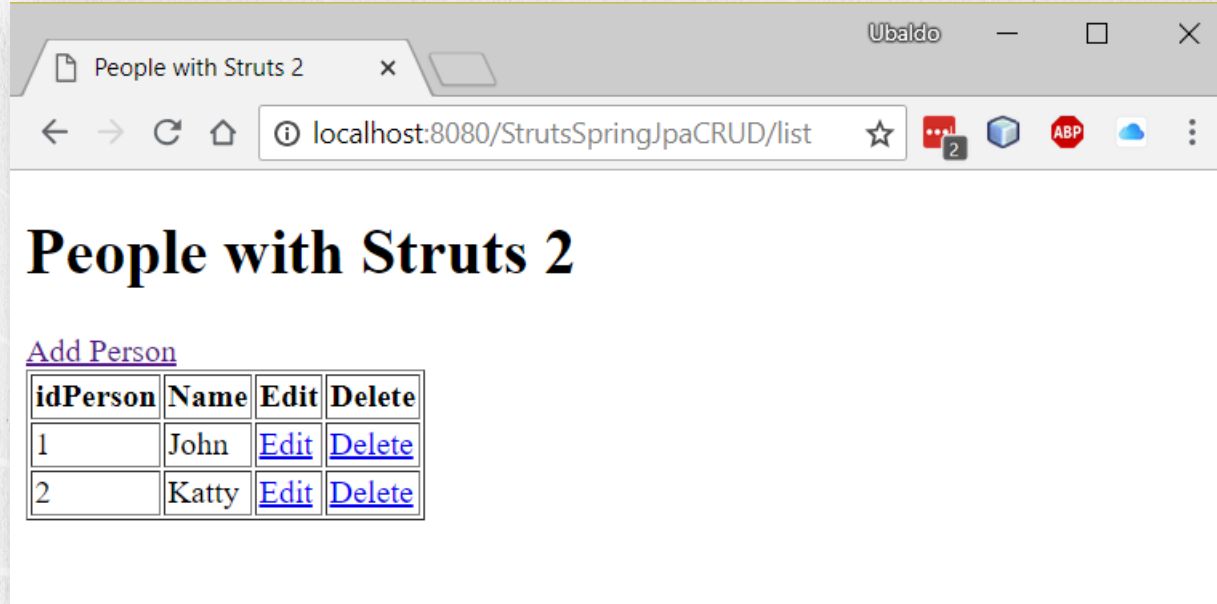
The screenshot shows a web browser window titled "People with Struts 2" with the address bar displaying "localhost:8080/StrutsSpringJpaCRUD/list". The page content includes a link "Add Person" and a table with the following data:

idPerson	Name	Edit	Delete
1	John	Edit	Delete
2	Katty	Edit	Delete
3	Charly Brown	Edit	Delete

The "Delete" link for the third record (Charly Brown) is highlighted with a yellow circle. A red arrow points from this link to the address bar, which now shows the URL: "localhost:8080/StrutsSpringJpaCRUD/deletePerson.action?person.idPerson=3".

34. EXECUTE THE APPLICATION

- And with this we have already executed the 4 operations: List, Add, Modify and Delete, or: CRUD (Create-Read-Update-Delete).



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.

The configuration by conventions of Struts 2, is very sensitive, in such a way that everything must be written as it was specified in the exercise, since any change in the names will cause that the exercise is not executed correctly.

The integration with other frameworks and technologies such as Spring and JPA is also very prone to errors, so you can support yourself from the resolved project that we give you, which is 100% functional, and thus support you at any time of this documentation and the projects resolved that we give you

STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

EXERCISE CONCLUSION

With this exercise we have created an application that integrates the 3 technologies such as:

- Struts 2
- Spring Framework
- JPA (Java Persistence API)

In this exercise we have applied the CRUD (Create-Read-Update-Delete) operations for the person table.

It is important to mention that as indicated throughout the exercise, the major change was made in the presentation layer, that is, in the Action class of Struts and in the view of the JSPs. So everything that was done in the previous exercise regarding the service layer (business) and the data layer was completely reused, in this way we have observed how to apply the best practices and architectures in our Java application.

ONLINE COURSE

STRUTS 2 FRAMEWORK

By: Eng. Ubaldo Acosta



CURSO STRUTS FRAMEWORK

www.globalmentoring.com.mx