

STRUTS FRAMEWORK COURSE

BASIC INTEGRATION OF STRUTS + SPRING + JPA



By the expert: Ubaldo Acosta

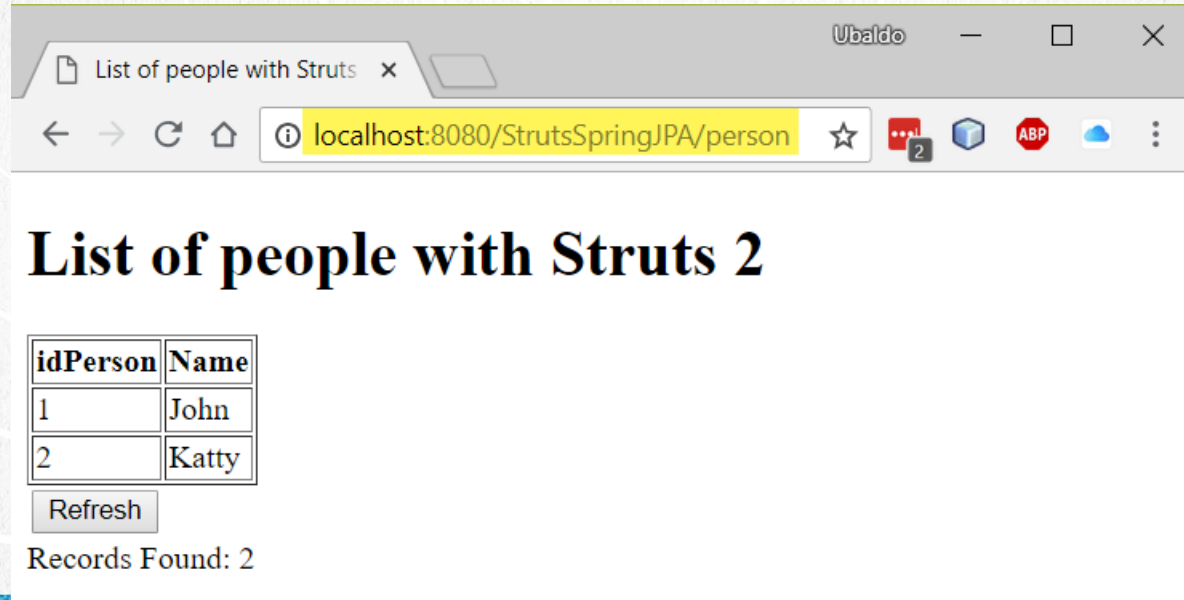


STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

EXERCISE OBJECTIVE

Create an application to implement the integration of Struts + Spring + JPA frameworks. At the end we should observe the following:

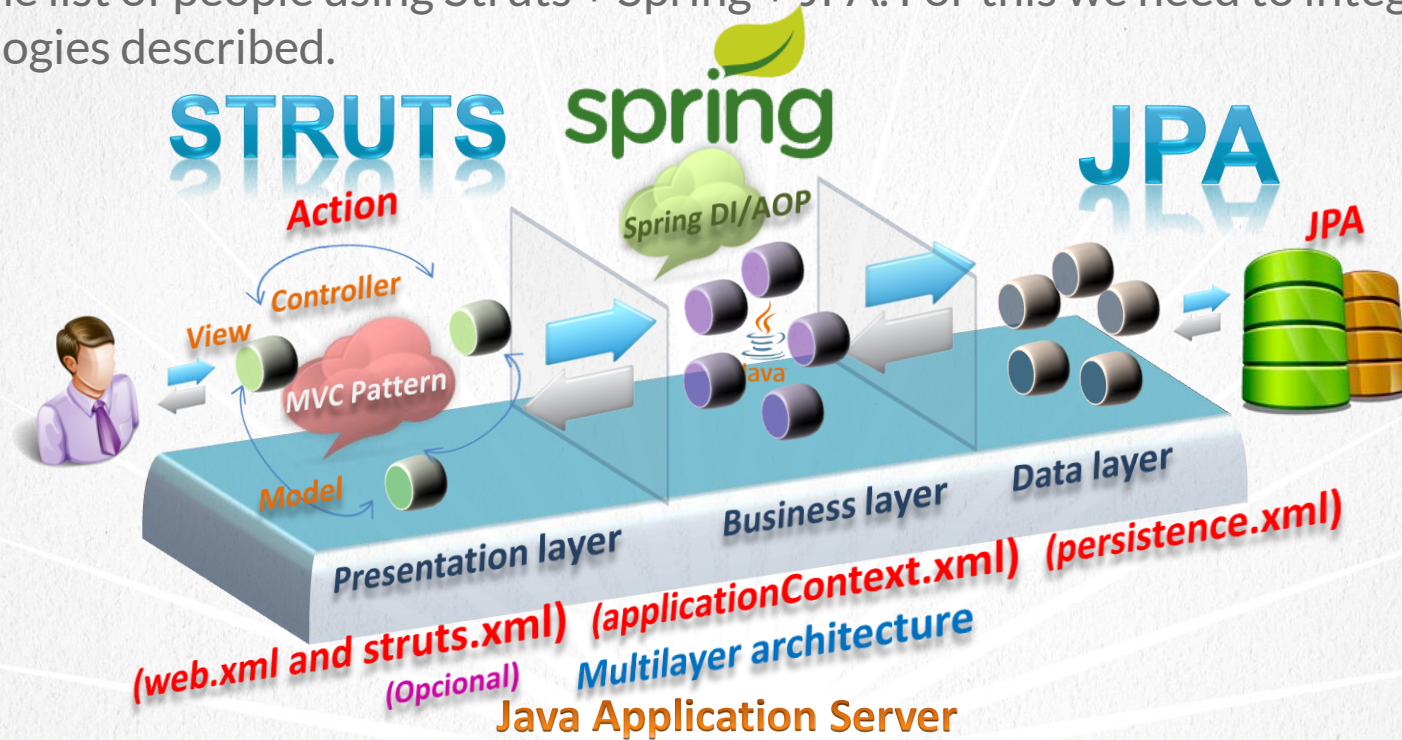


STRUTS FRAMEWORK COURSE

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

CREATE MYSQL DATABASE

First of all, we need to create a new mysql database. Follow the next guide to create the database, the table and insert some rows:

<http://icursos.net/en/Installations/CJ-B-Exercise-03-MySqlDataBase.pdf>



STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

CREATE JTA CONNECTION IN GLASSFISH

We need to create a JTA connection in Glassfish. With this connection we will create a pool of connections to MySql, so the JPA API can connect to the database:

<http://icursos.net/en/Installations/CJ-B-Exercise-04-JTAGlassfish.pdf>



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 StrutsSpringJpa. This project will integrate the 3 technologies: Struts + Spring + JPA.

Let's start with our exercise:

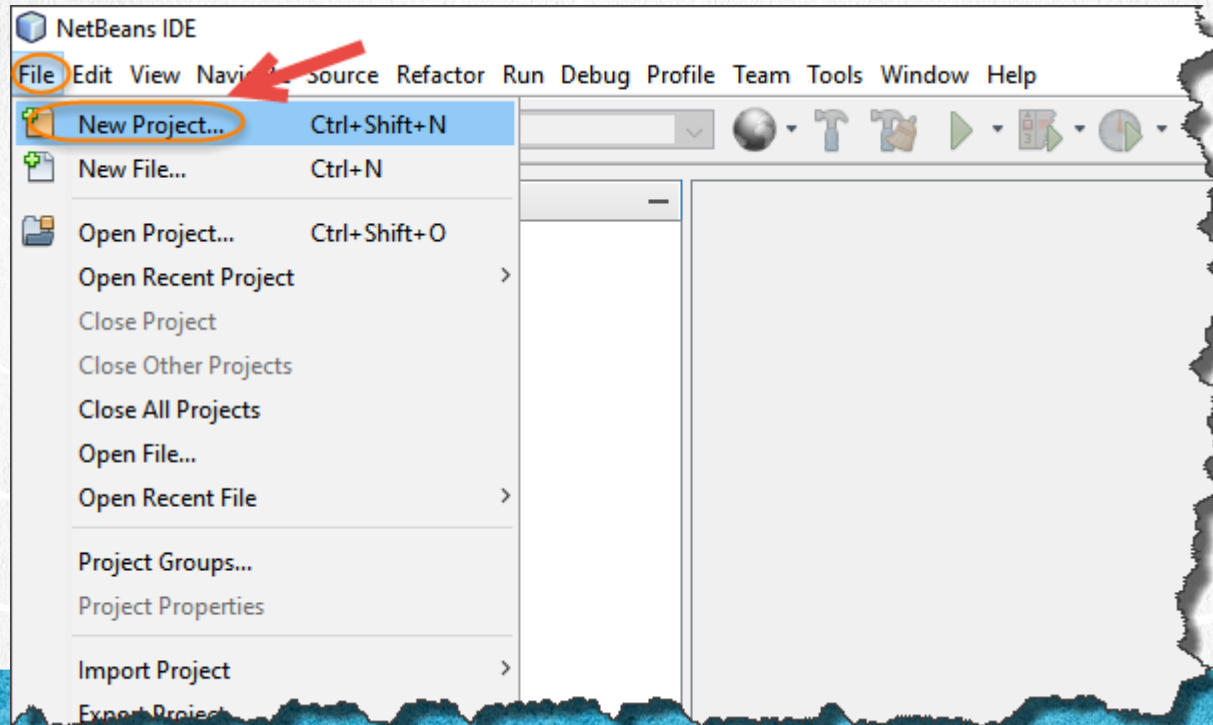


STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

1. CREATE A NEW PROJECT

We created our exercise called StrutsSpringJPA:

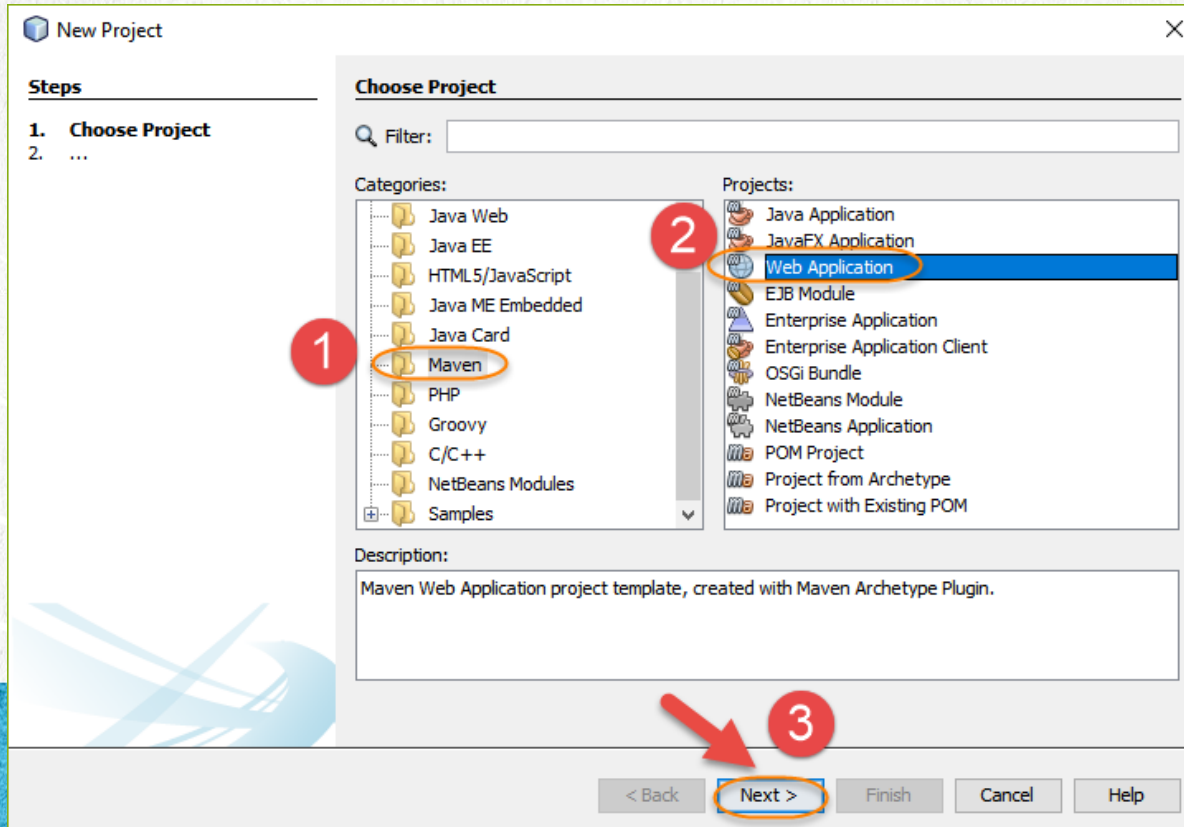


STRUTS FRAMEWORK COURSE

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

Steps

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

Name and Location

Project Name: StrutsSpringJPA

Project Location: C:\Courses\Struts\01-Struts2SpringJpa-Basic Browse...

Project Folder: C:\Courses\Struts\01-Struts2SpringJpa-Basic\StrutsSpringJPA

Artifact Id: StrutsSpringJPA

Group Id: web

Version: 1

Package: (Optional)

< Back **Next >** Finish Cancel Help

STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

1. CREATE A NEW PROJECT

- We create a new Java Maven project:

New Web Application

Steps

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

Settings

Server: Add...

Java EE Version:

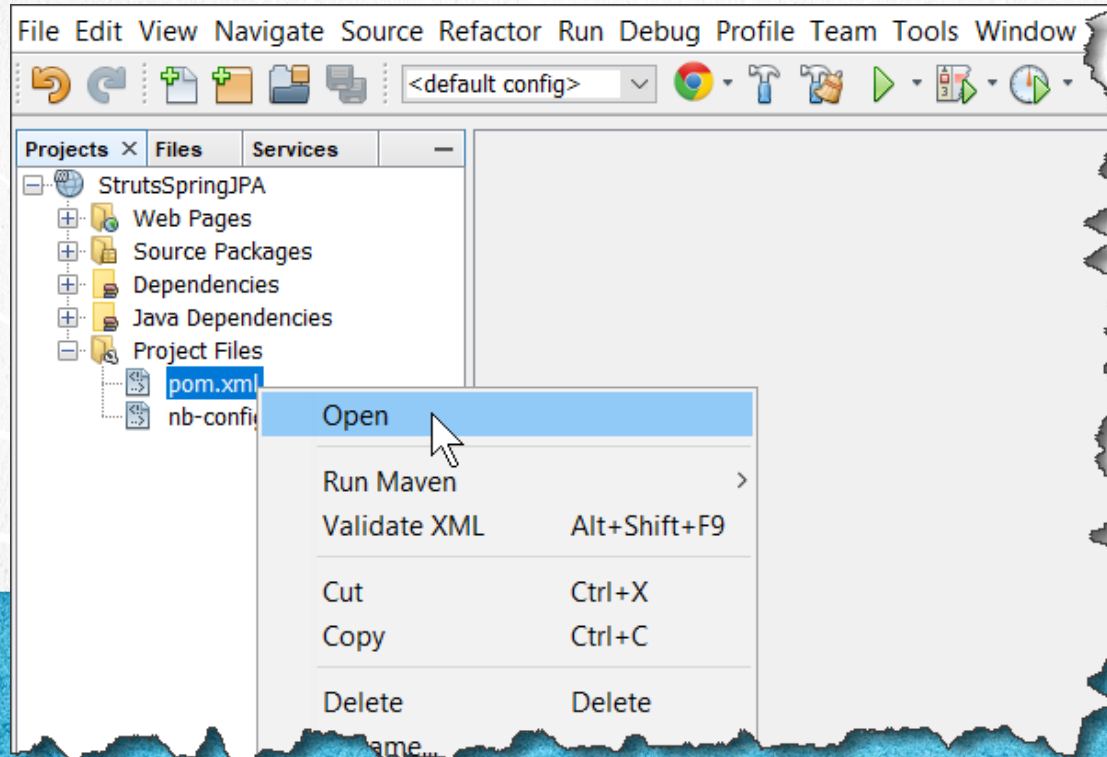
< Back Next > **Finish** Cancel Help

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

    <name>StrutsSpringJPA</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>
```

CURSO DE JAVA CON JDBC

www.globalmentoring.com.mx

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>
```

CURSO DE JAVA CON JDBC

www.globalmentoring.com.mx

3. MODIFY THE FILE

[pom.xml:](#)

[Click to download](#)

```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-web</artifactId>
  <version>${spring.version}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-webmvc</artifactId>
  <version>${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>
```

3. MODIFY THE FILE

[pom.xml:](#)

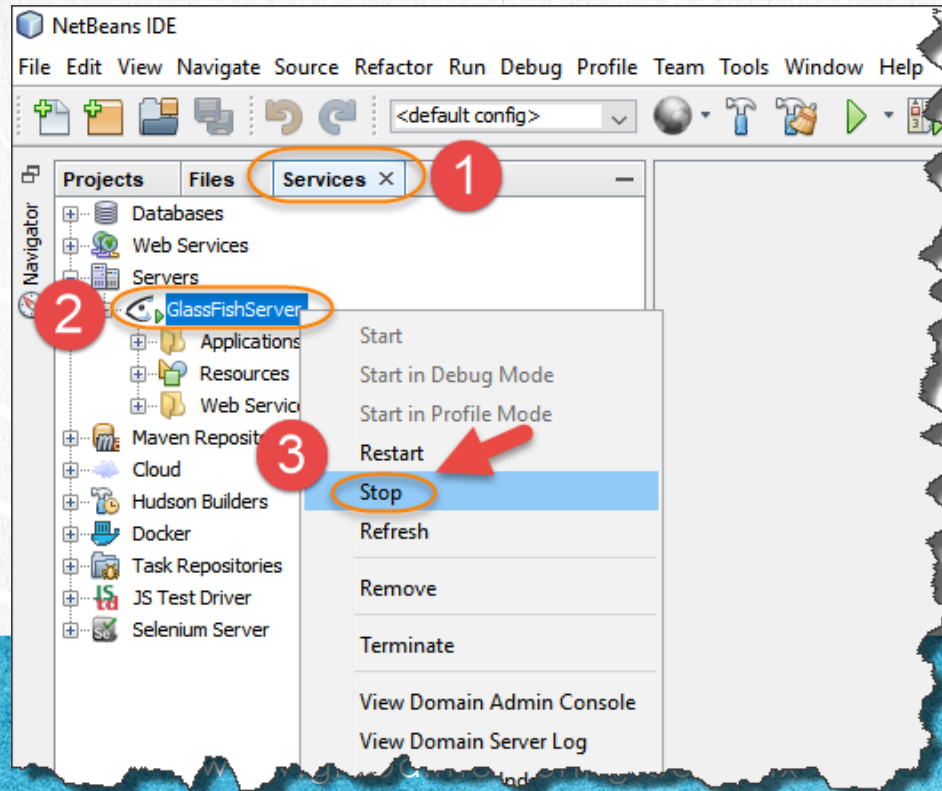
[Click to download](#)

```
<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>
```

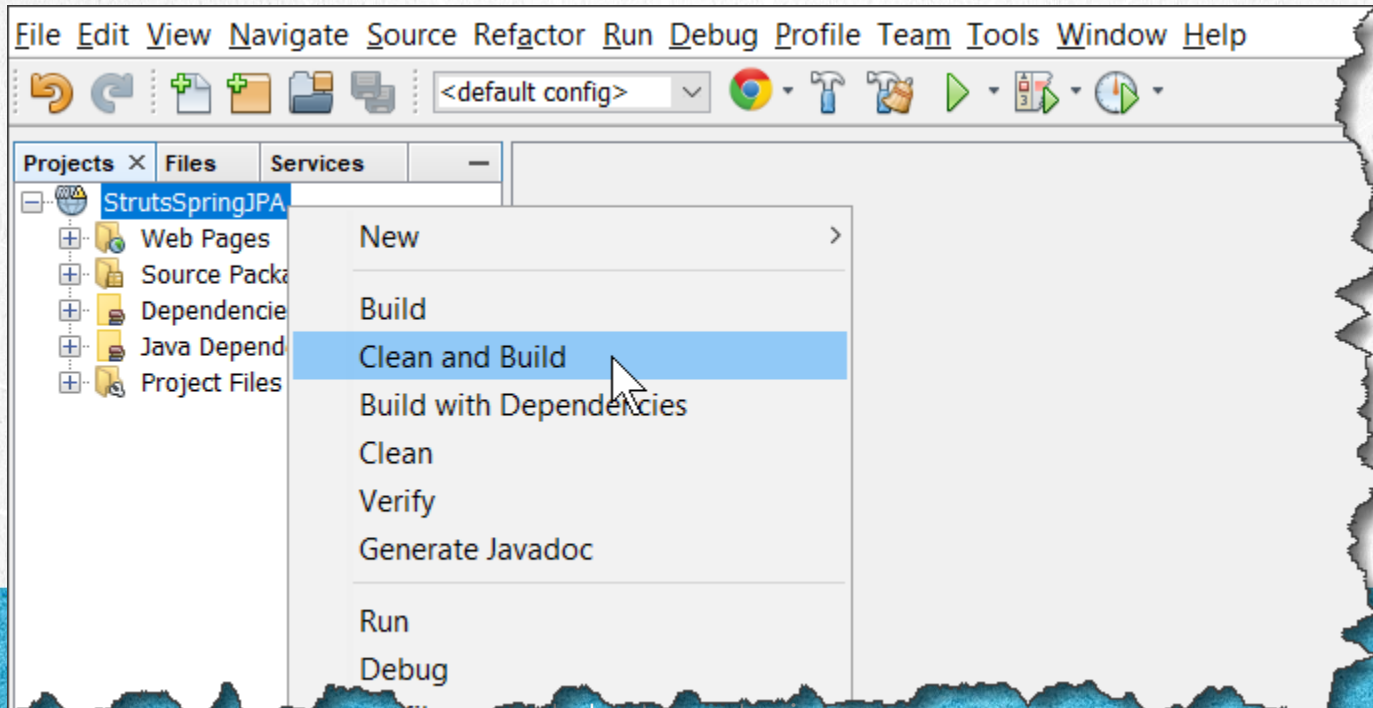

4. STOP GLASSFISH IF IT WAS STARTED

- 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.



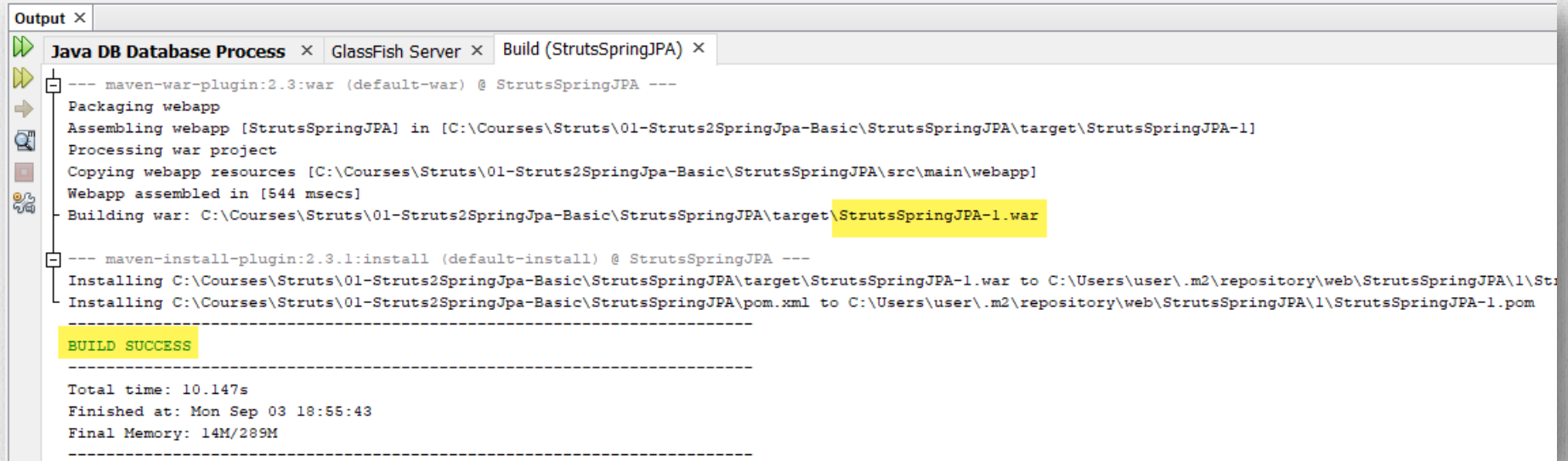
5. EXECUTE CLEAN & BUILD

- 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:



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:



```
Output X
Java DB Database Process X GlassFish Server X Build (StrutsSpringJPA) X
--- maven-war-plugin:2.3:war (default-war) @ StrutsSpringJPA ---
Packaging webapp
Assembling webapp [StrutsSpringJPA] in [C:\Courses\Struts\01-Struts2SpringJpa-Basic\StrutsSpringJPA\target\StrutsSpringJPA-1]
Processing war project
Copying webapp resources [C:\Courses\Struts\01-Struts2SpringJpa-Basic\StrutsSpringJPA\src\main\webapp]
Webapp assembled in [544 msecs]
Building war: C:\Courses\Struts\01-Struts2SpringJpa-Basic\StrutsSpringJPA\target\StrutsSpringJPA-1.war
--- maven-install-plugin:2.3.1:install (default-install) @ StrutsSpringJPA ---
Installing C:\Courses\Struts\01-Struts2SpringJpa-Basic\StrutsSpringJPA\target\StrutsSpringJPA-1.war to C:\Users\user\.m2\repository\web\StrutsSpringJPA\1\StrutsSpringJPA-1.war
Installing C:\Courses\Struts\01-Struts2SpringJpa-Basic\StrutsSpringJPA\pom.xml to C:\Users\user\.m2\repository\web\StrutsSpringJPA\1\StrutsSpringJPA-1.pom
-----
BUILD SUCCESS
-----
Total time: 10.147s
Finished at: Mon Sep 03 18:55:43
Final Memory: 14M/289M
-----
```

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.

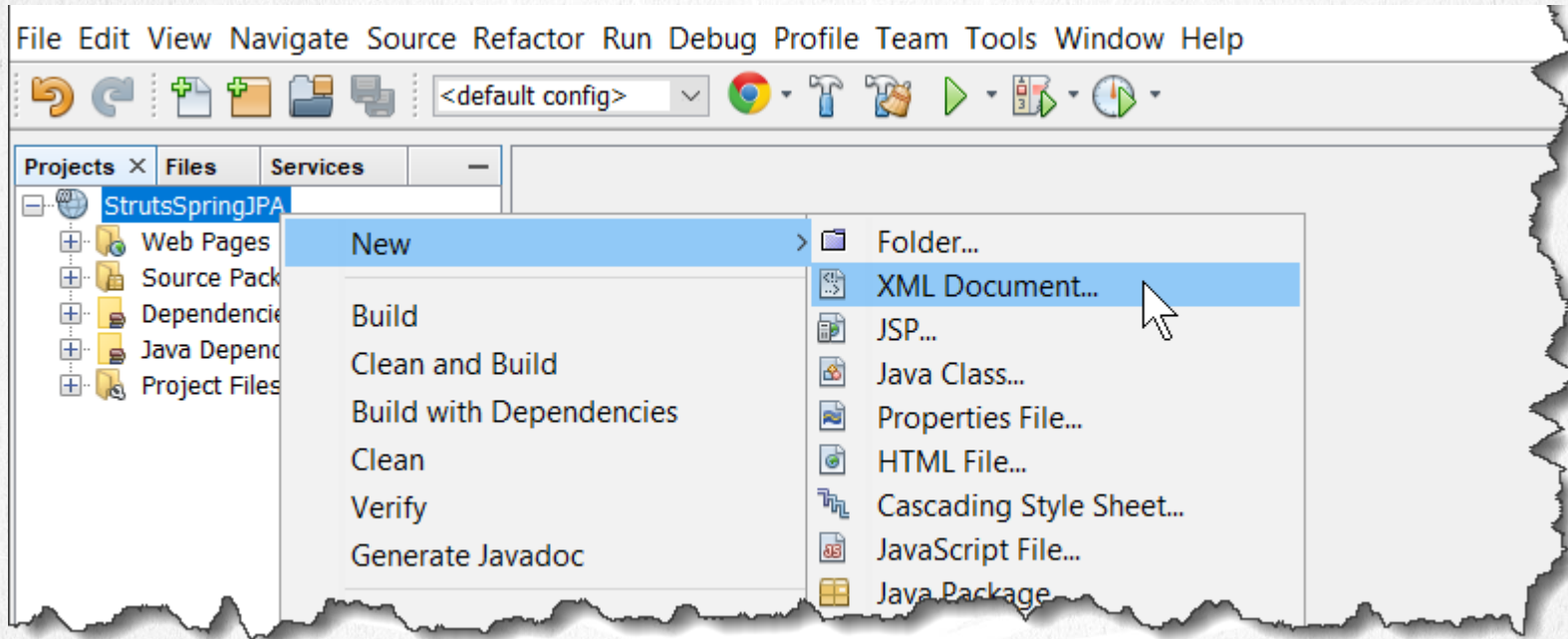


STRUTS FRAMEWORK COURSE

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: persistence

Project: StrutsSpringJPA

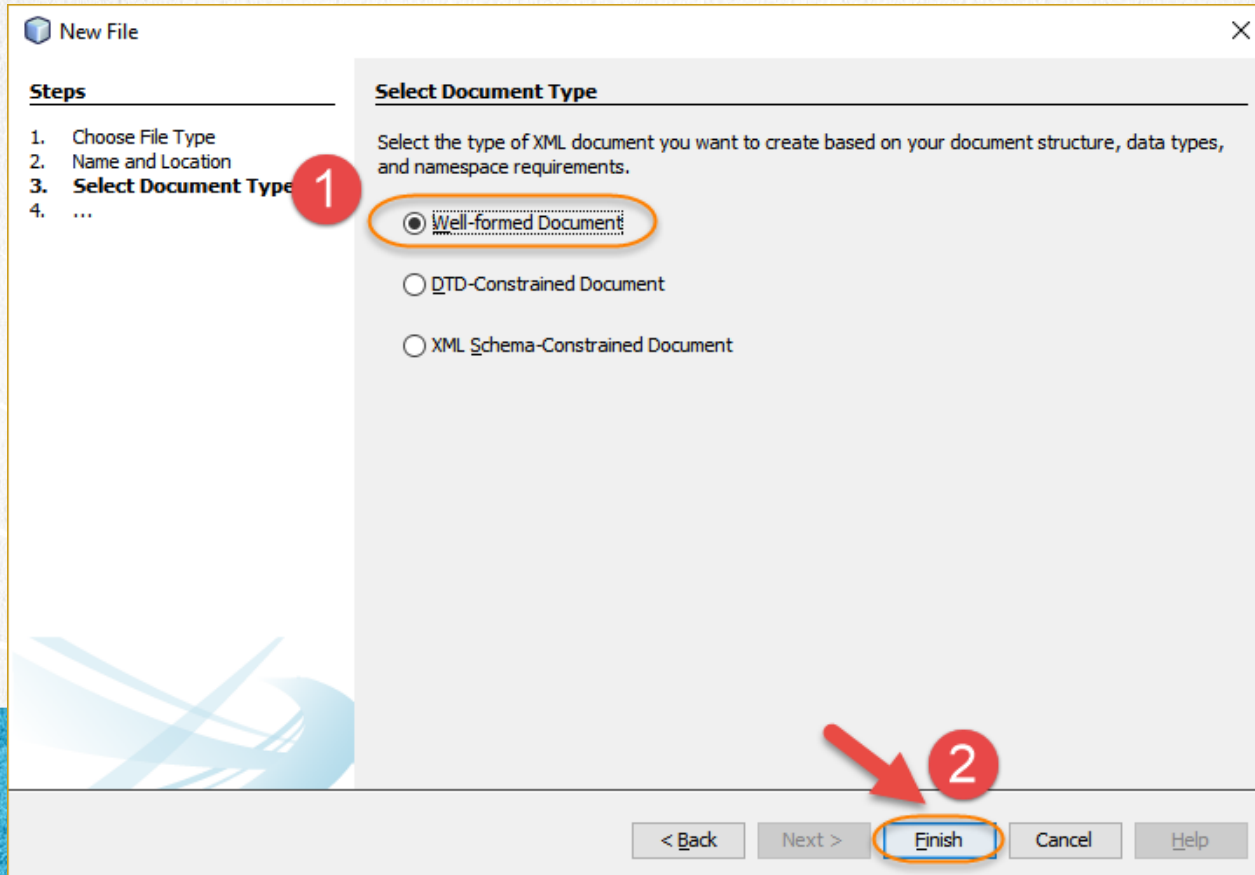
Folder: src/main/resources/META-INF Browse...

Created File: C:\Courses\Struts\01-Struts2SpringJpa-Basic\StrutsSpringJPA\src\main\resources\META-INF\persistence.xml

< Back **Next >** Finish Cancel Help

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

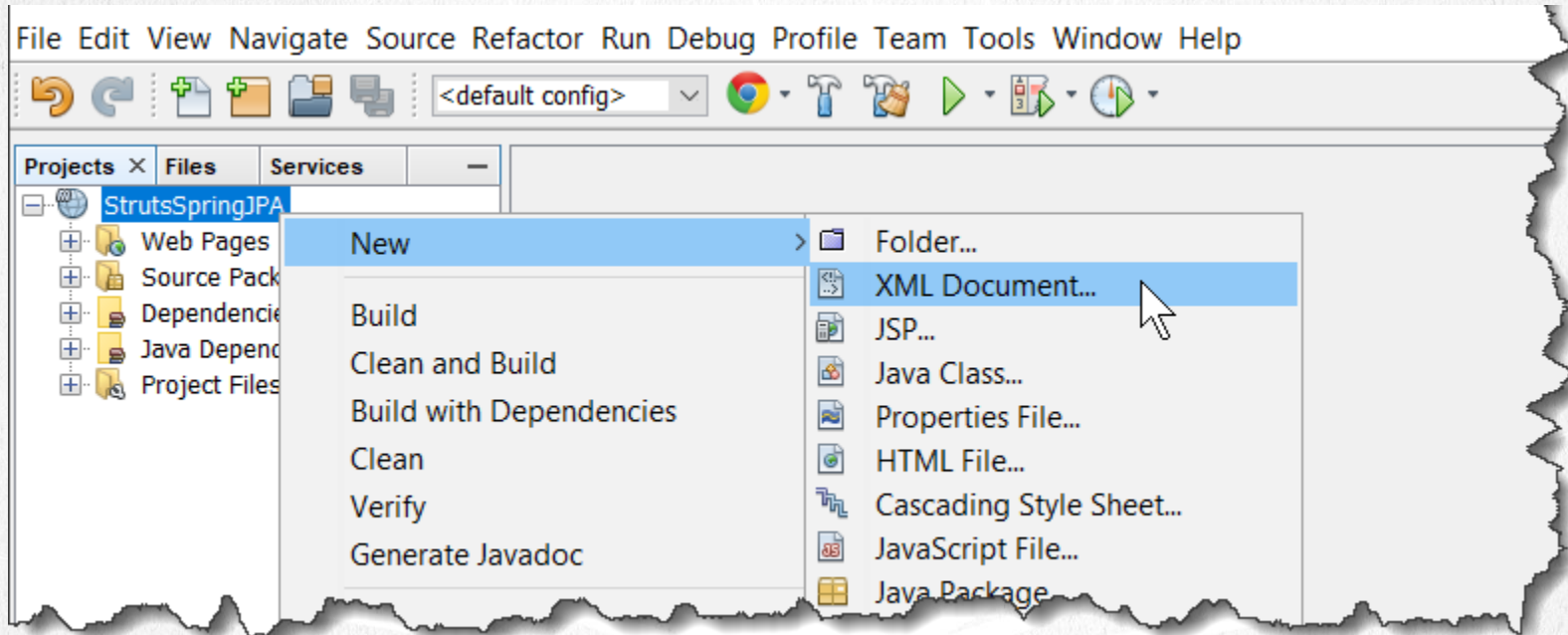


STRUTS FRAMEWORK COURSE

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:

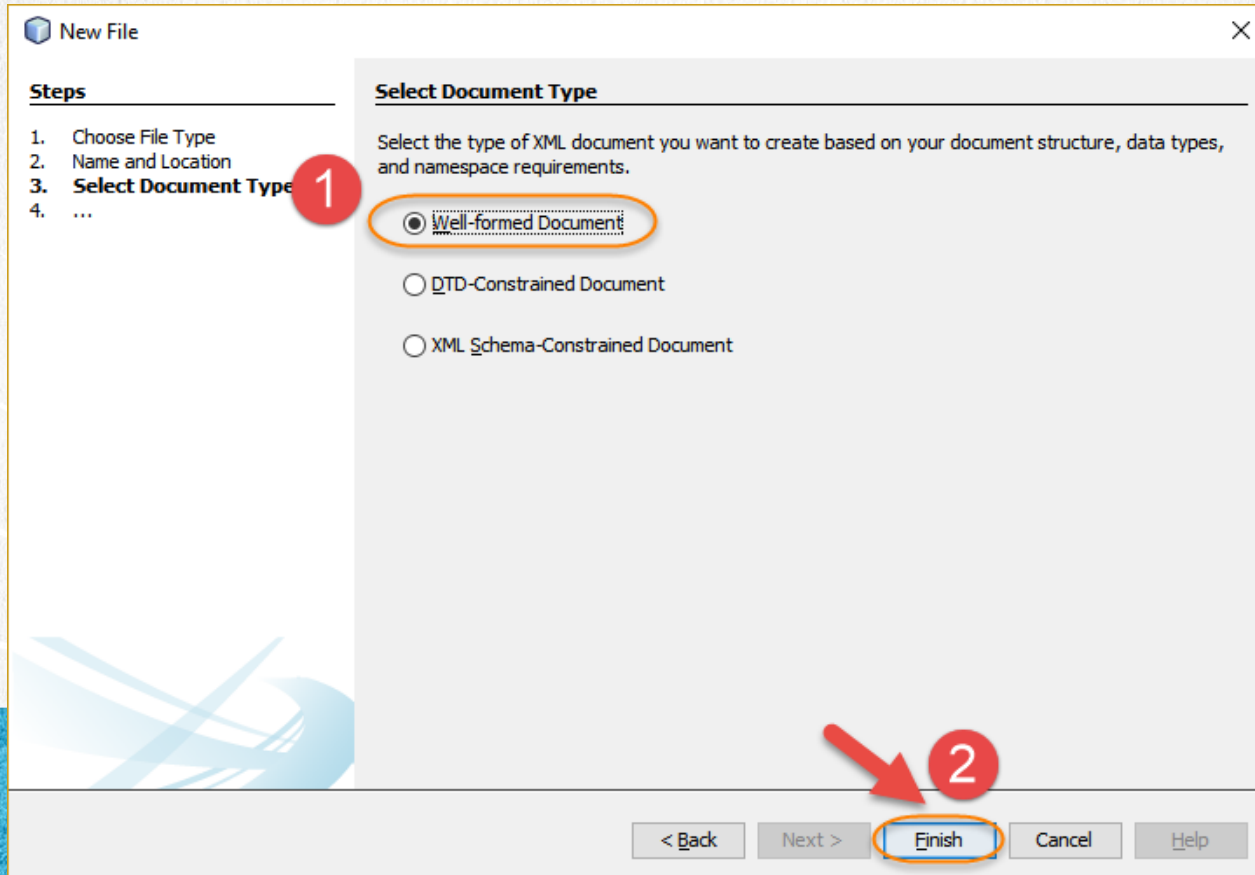
Project:

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.

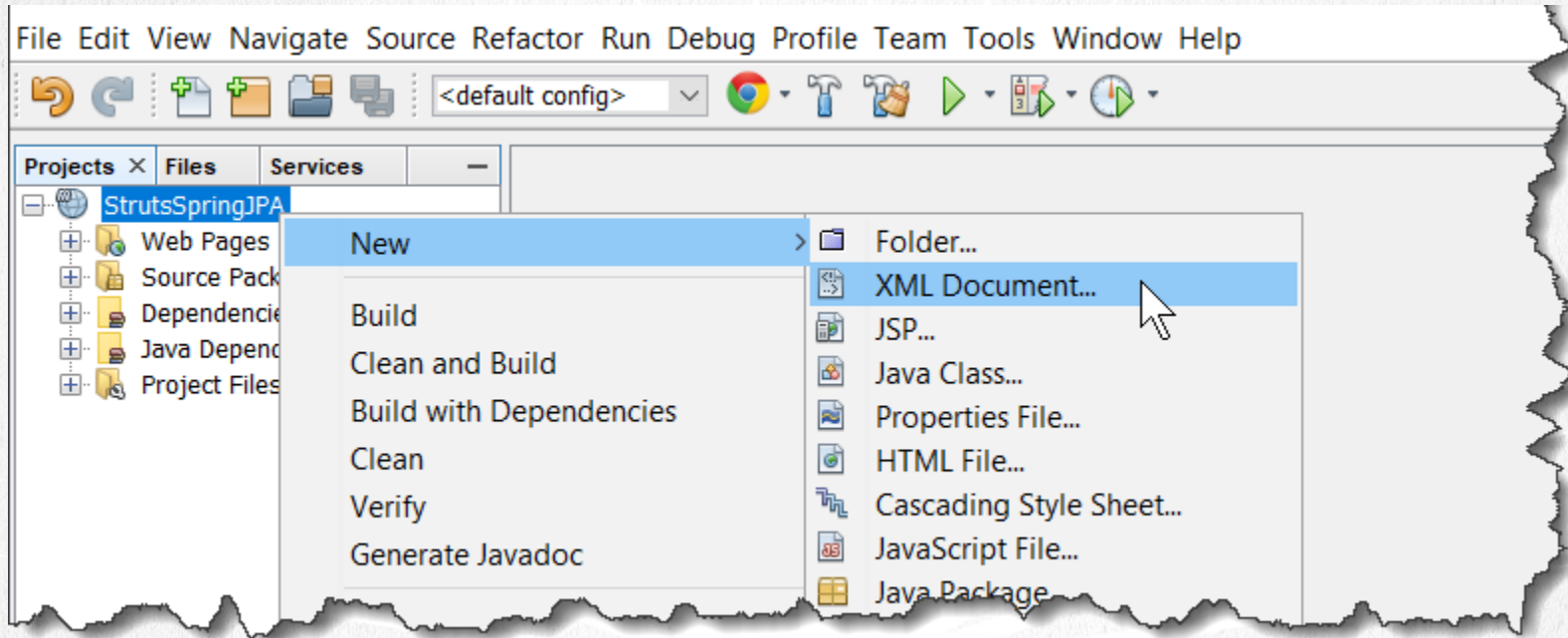
Our web.xml file also allows us to configure the JNDI name for the database connection that we will use with JPA via JTA.

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:



STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

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:

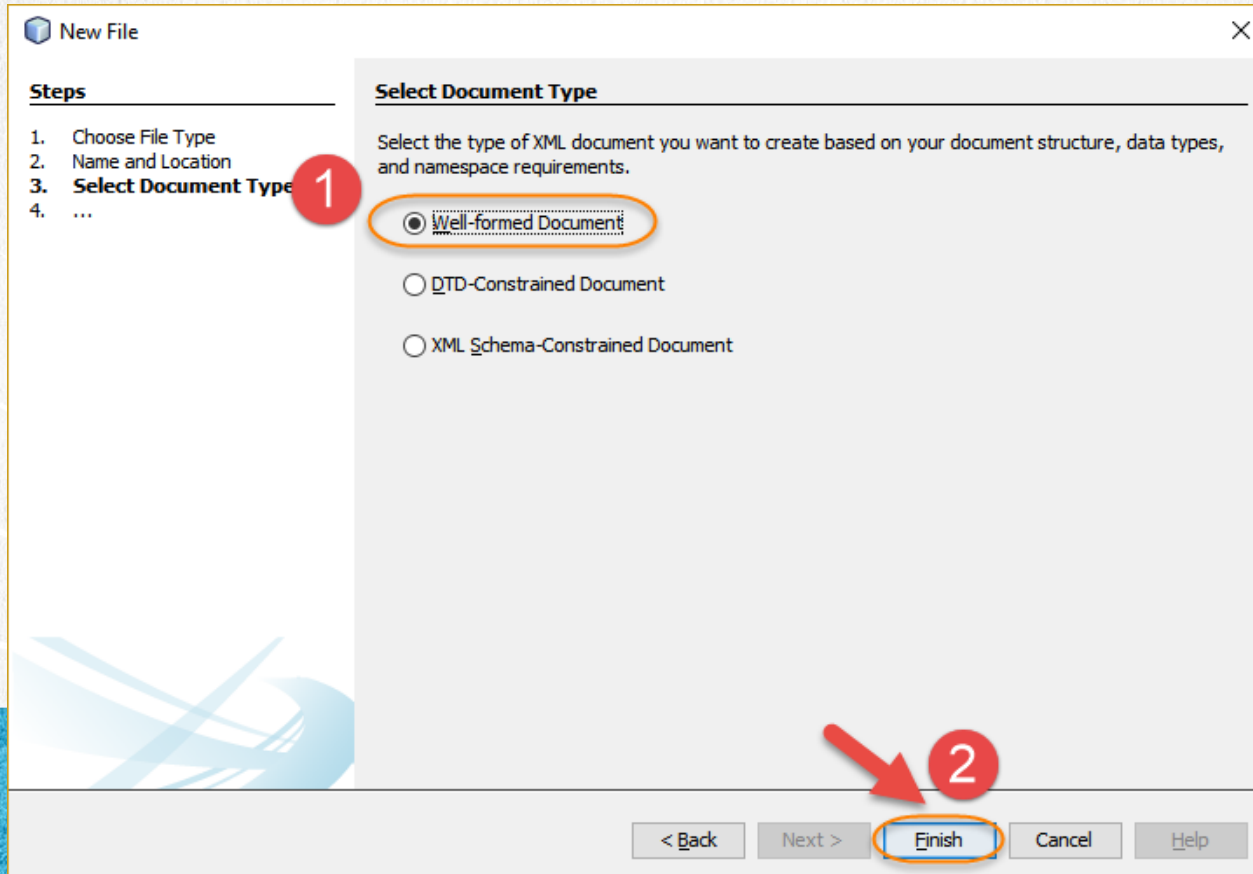
Folder:

Created File:

< Back **Next >** Finish Cancel Help

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

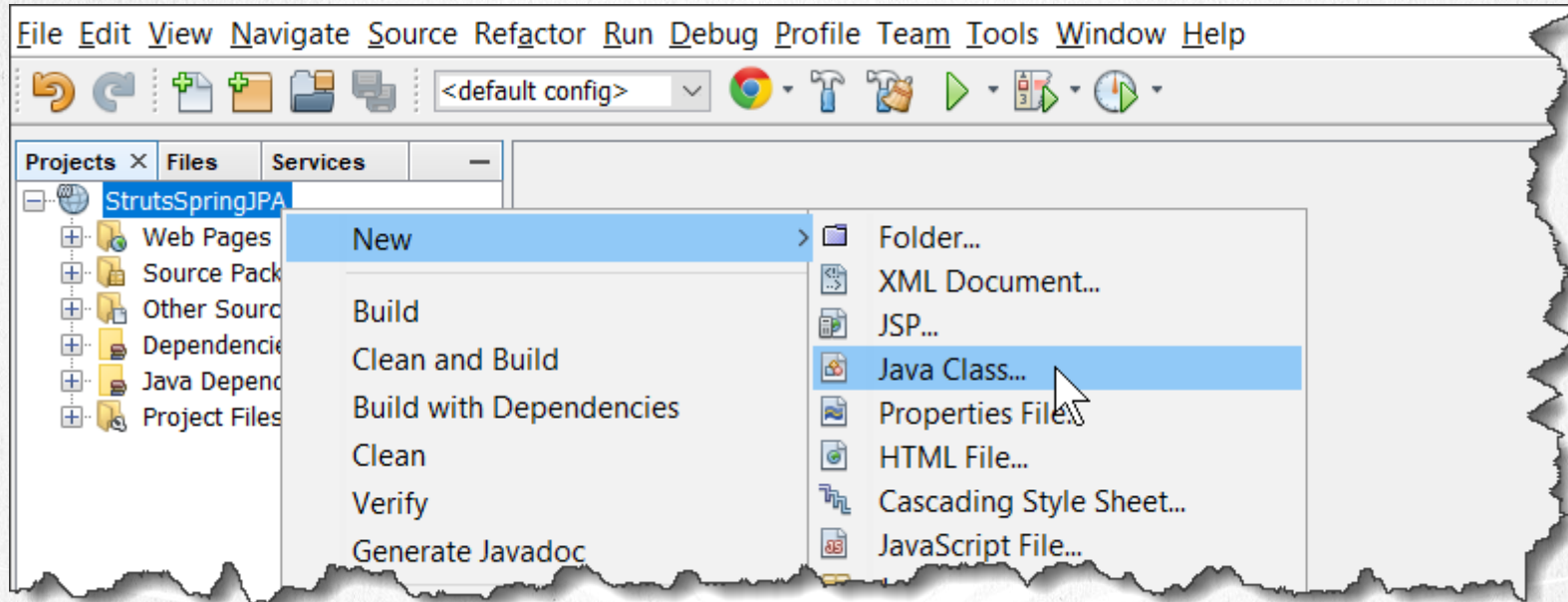


STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

12. CREATE A JAVA CLASS

- We create the Person.java class:



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: StrutsSpringJPA

Location: Source Packages

Package: mx.com.gm.datalayer.domain

Created File: C:\Courses\Struts\01-Struts2SpringJpa-Basic\StrutsSpringJPA\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 :

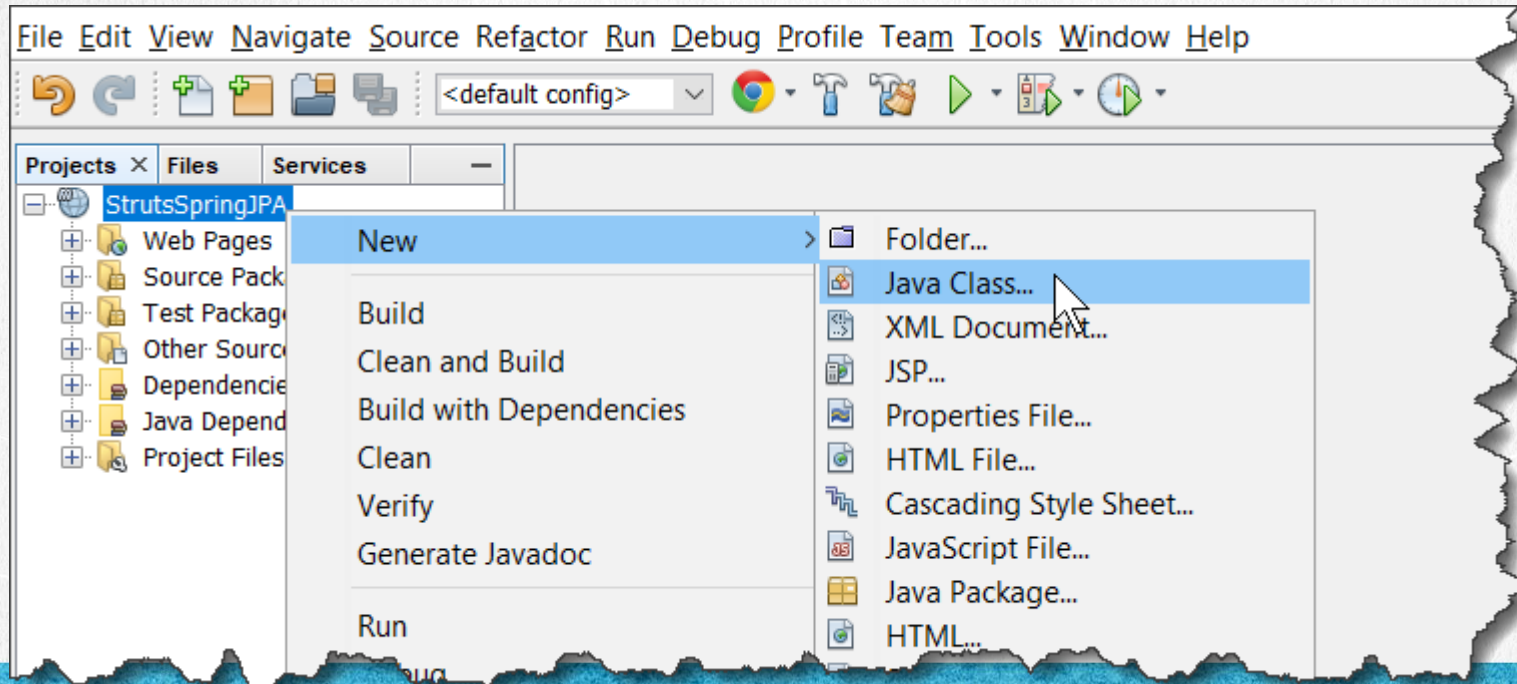


STRUTS FRAMEWORK COURSE

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 :

New Java Class

Steps

1. Choose File Type

2. **Name and Location**

Name and Location

Class Name: PersonDao

Project: StrutsSpringJPA

Location: Source Packages

Package: mx.com.gm.datalayer

Created File: C:\Courses\Struts\01-Struts2SpringJpa-Basic\StrutsSpringJPA\src\main\java\mx\com\gm\datalayer\PersonDao.java

< Back

Next >

Finish

Cancel

Help

STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

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:

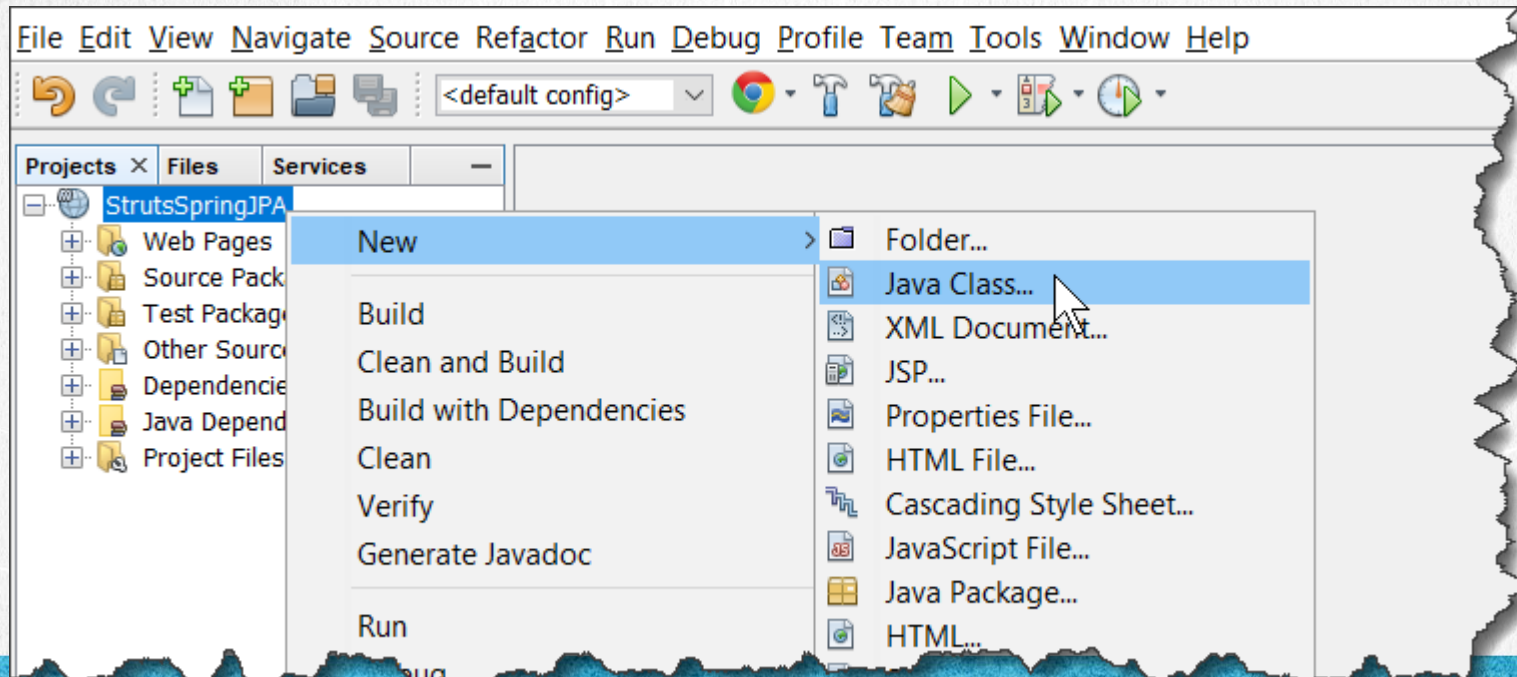


STRUTS FRAMEWORK COURSE

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:

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

STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

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:

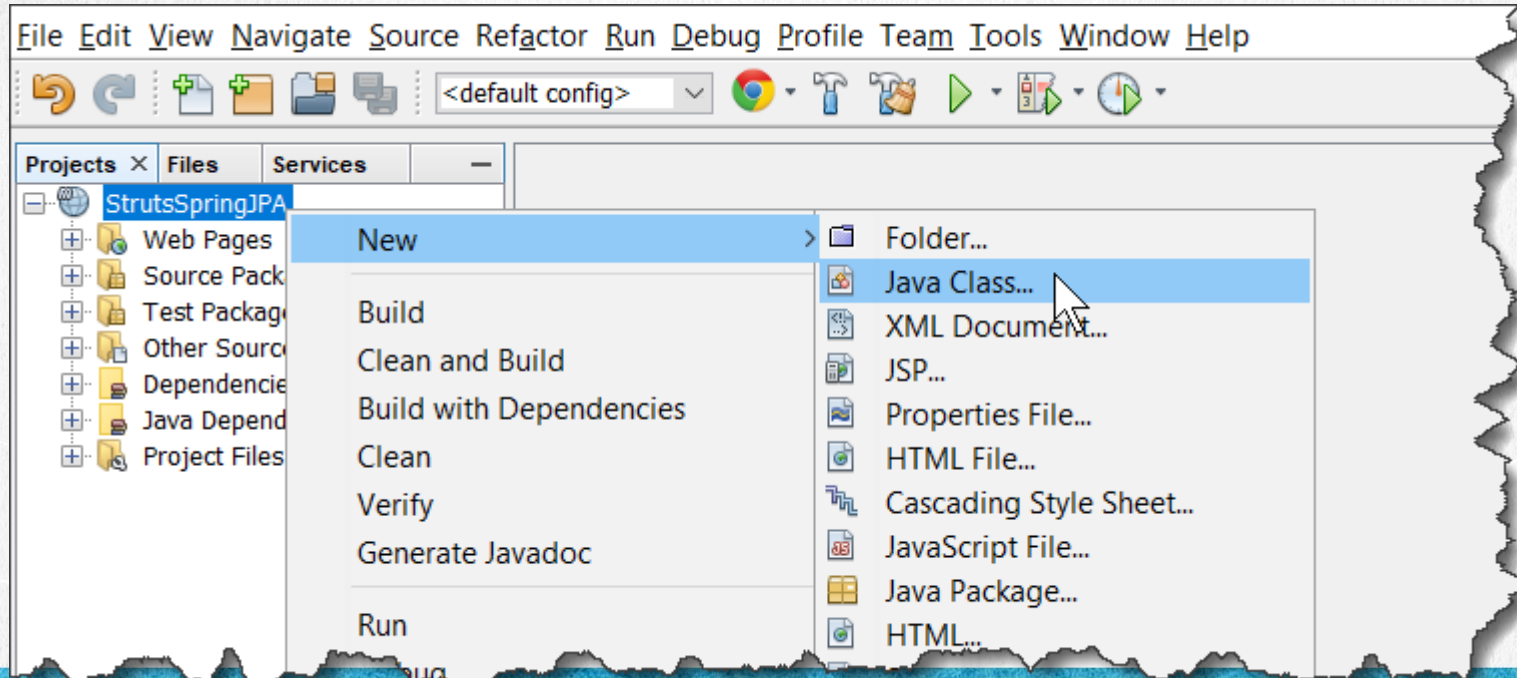


STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

18. CREATE A JAVA INTERFACE

- We create the PersonaService.java interface:



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

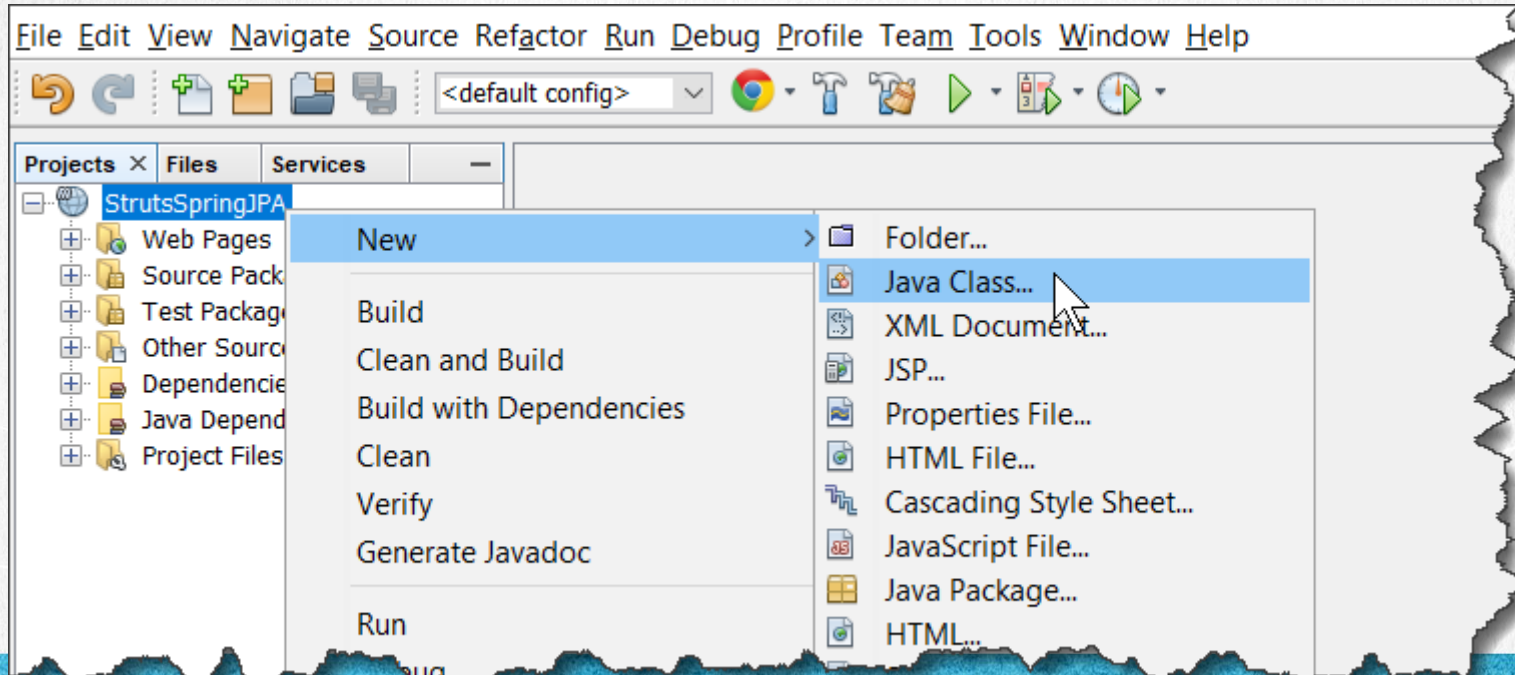


STRUTS FRAMEWORK COURSE

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: StrutsSpringJPA

Location: Source Packages

Package: mx.com.gm.businesslayer

Created File: C:\Courses\Struts\01-Struts2SpringJpa-Basic\StrutsSpringJPA\src\main\java\mx\com\gm\businesslayer\PersonServiceImpl.java

< Back Next > **Finish** Cancel Help

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();
}
}
```

PASO 22. CREATE A JAVA CLASS

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

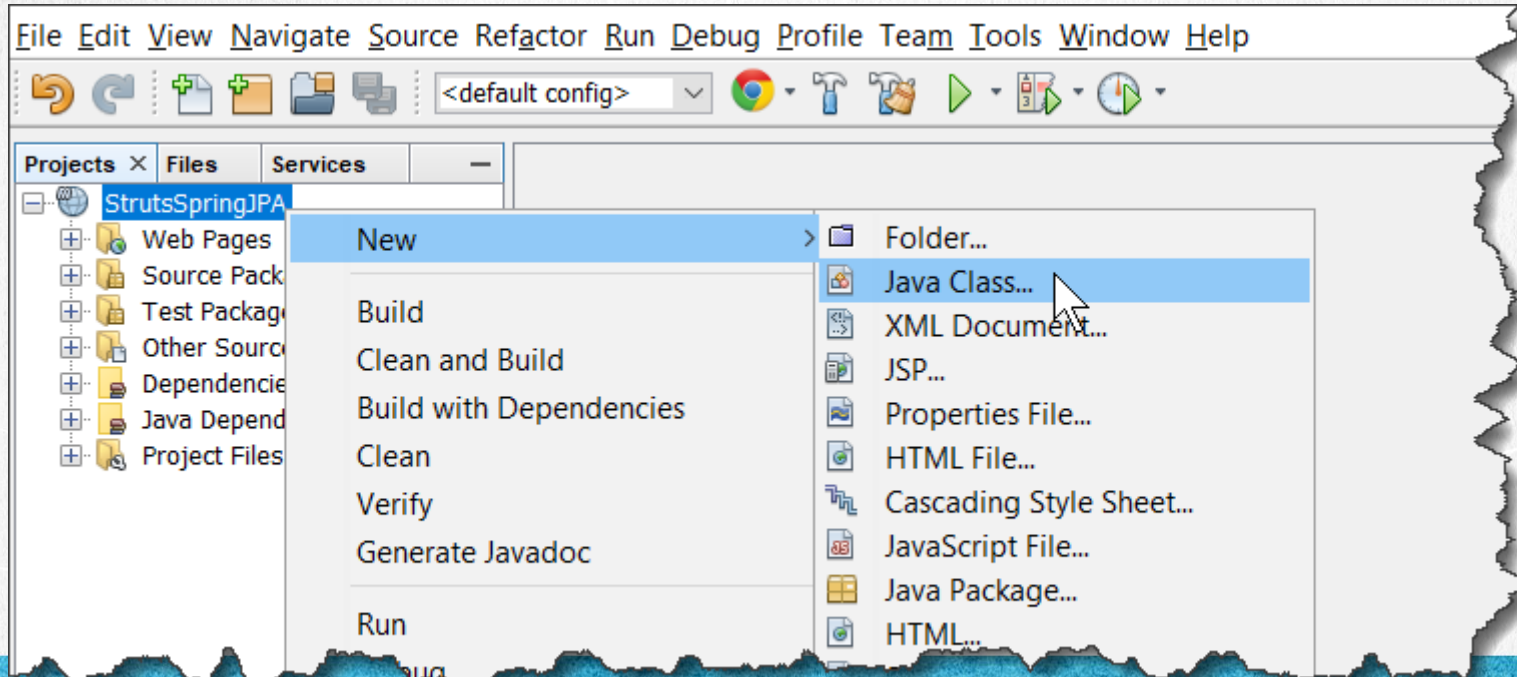
We are going to extend the ActionSupport class and to overwrite the execute method.

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.

22. CREATE A JAVA CLASS

- We create the ShowPersonaAction.java class:



STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

22. CREATE A JAVA CLASS

- We create the PersonaAction.java class:

New Java Class

Steps

1. Choose File Type

2. **Name and Location**

Name and Location

Class Name: PersonAction

Project: StrutsSpringJPA

Location: Source Packages

Package: mx.com.gm.actions

Created File: C:\Courses\Struts\01-Struts2SpringJpa-Basic\StrutsSpringJPA\src\main\java\mx\com\gm\actions\PersonAction.java

< Back

Next >

Finish

Cancel

Help

STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

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.springframework.beans.factory.annotation.Autowired;

public class PersonAction extends ActionSupport {

    private long peopleCounter;

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

    @Autowired
    private PersonService personService;

    private List<Person> people;
```

23. MODIFY THE CODE

PersonAction.java:

Click to download

```
@Override
public String execute() {
    this.people = personService.listPeople();
    this.peopleCounter = personService.countPeople();
    return SUCCESS;
}

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

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

public long getPeopleCounter() {
    return peopleCounter;
}

public void setPeopleCounter(long peopleCounter) {
    this.peopleCounter = peopleCounter;
}
}
```


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 this file `PersonAction.properties`

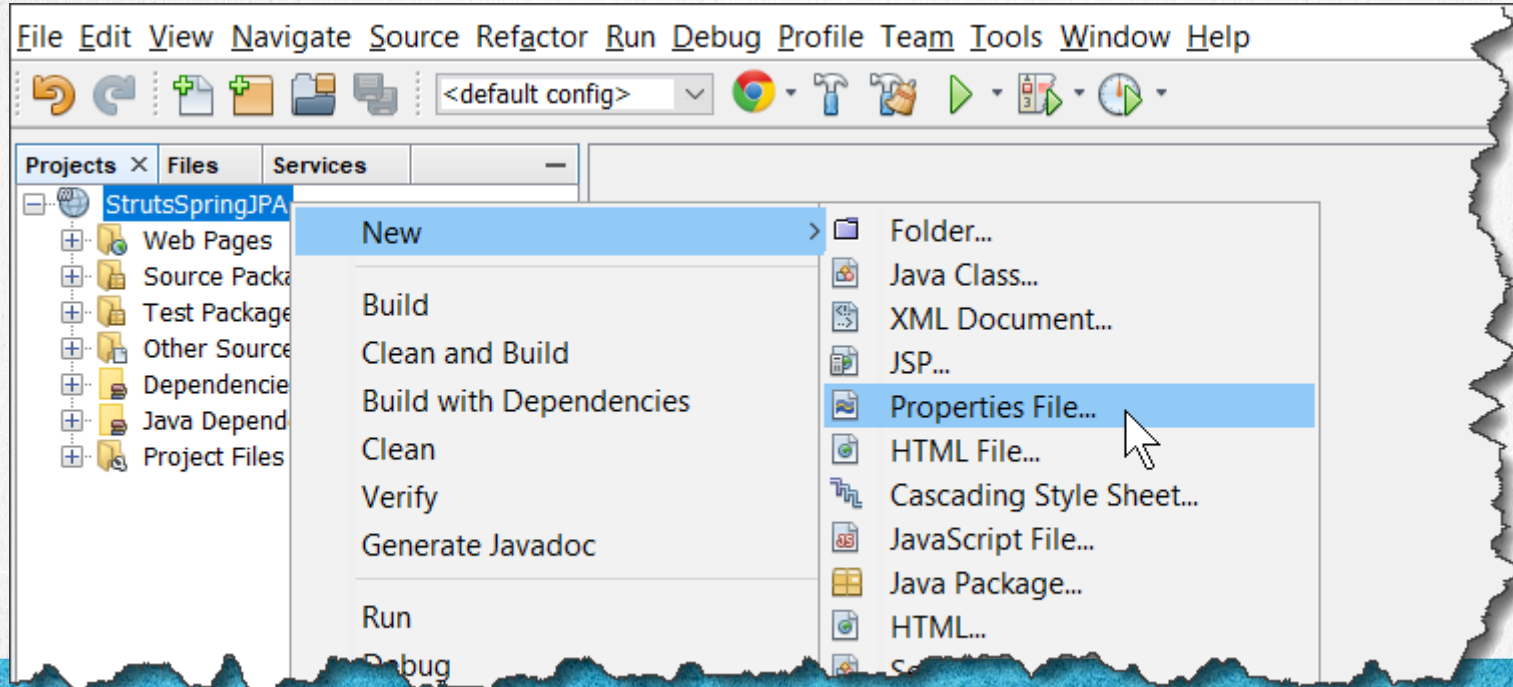


STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

24. CREATE THE PROPERTIES FILE

- We create the file PersonAction.properties as follows :



STRUTS FRAMEWORK COURSE

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:

< Back Next > **Finish** Cancel Help

25. MODIFY THE FILE

PersonAction.properties:

Click to download

```
person.title: List of people with Struts 2  
person.counter: Records Found  
person.button: Refresh  
p.idPerson: idPerson  
p.name: Name
```

STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

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: [person](#)

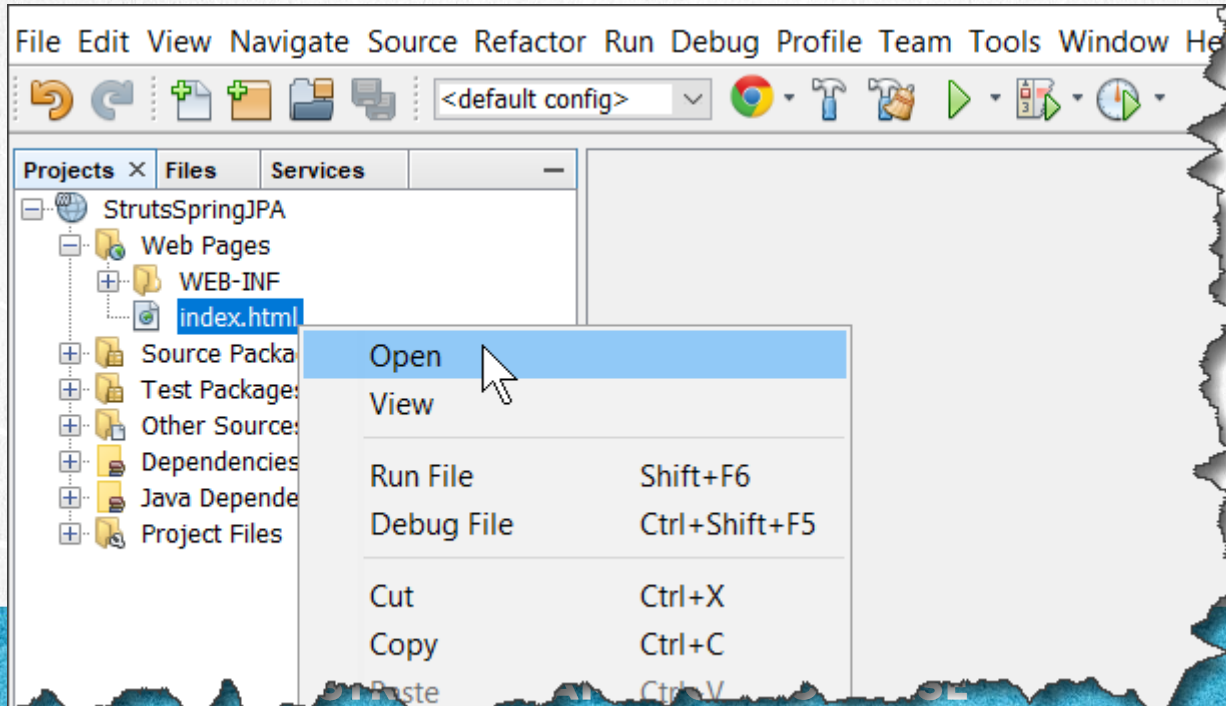


STRUTS FRAMEWORK COURSE

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

STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

27. CREATE THE JSP FILE

Now we create the file: person.jsp. Remember that this name corresponds to the path that is going to be used to call the corresponding action (PersonaAction.java), so we separated by a hyphen each word of the class of type Action.

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



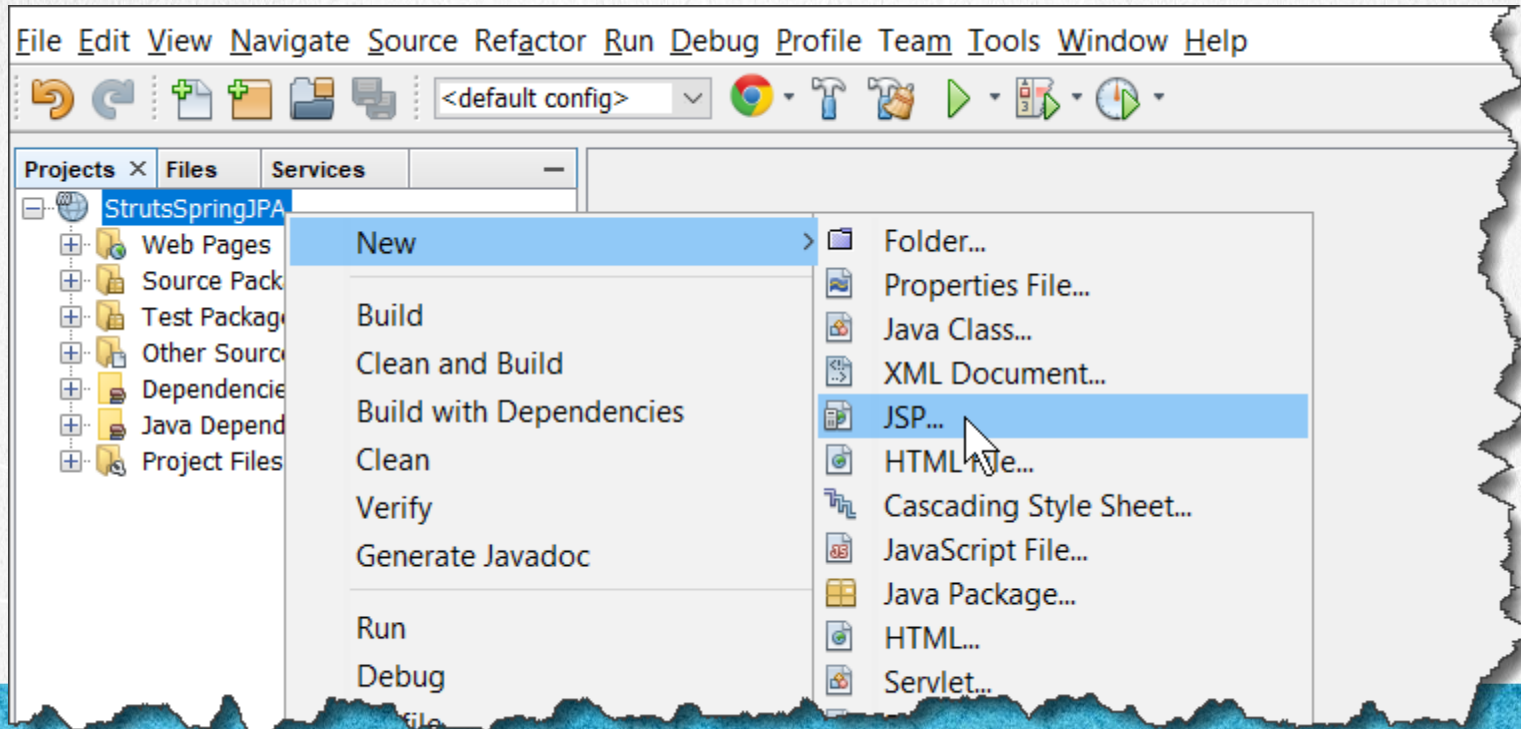
Experiencia y Conocimiento para tu vida

STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

27. CREATE THE JSP FILE

- We create the file person.jsp :



STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

27. CREATE THE JSP FILE

- We create 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:

28. MODIFY THE FILE

[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="person.title" /></title>
  </head>
  <body>
    <h1><s:text name="person.title" /></h1>
```

28. MODIFY THE FILE

person.jsp:

Click to download

```
<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>
      </tr>
      <s:iterator value="people">
        <tr>
          <td><s:property value="idPerson" /></td>
          <td><s:property value="name" /></td>
        </tr>
      </s:iterator>
    </table>
  </div>
</s:if>

<s:form>
  <s:submit key="person.button" name="submit" />
</s:form>

<div><s:text name="person.counter" />: <s:property value="peopleCounter" /></div>
</body>
</html>
```


29. 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.

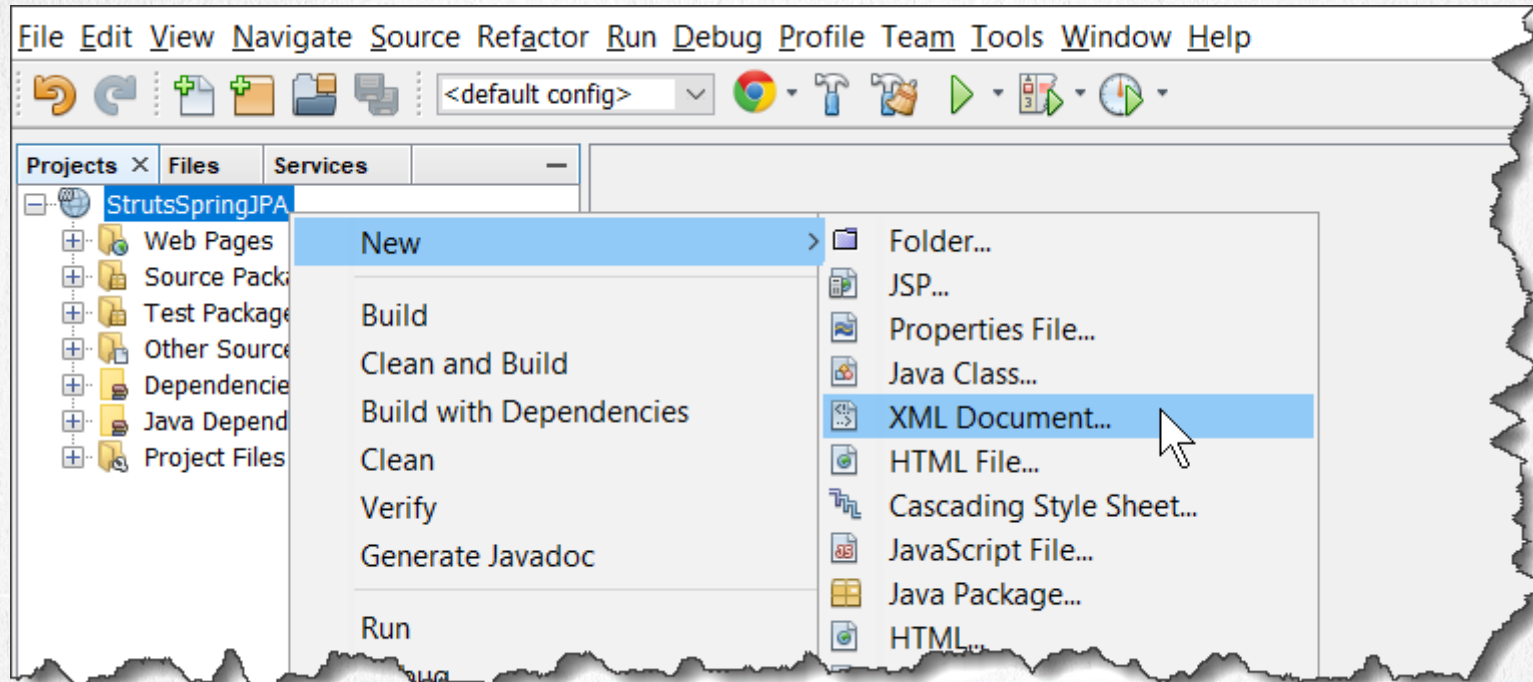


STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

29. CREATE THE LOG4J2.XML FILE

- We create the log4j2.xml file as follows:



29. 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:

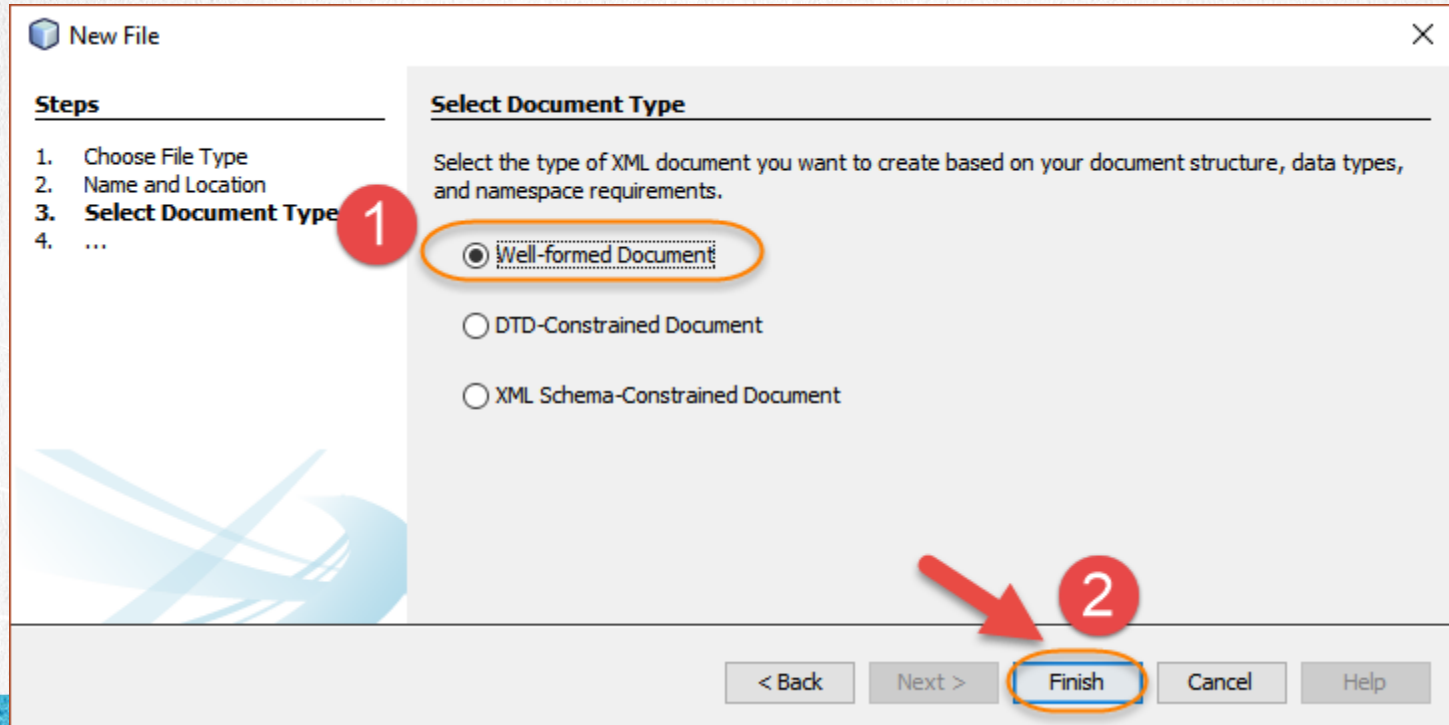
Folder:

Created File:

< Back **Next >** Finish Cancel Help

29. CREATE THE LOG4J2.XML FILE

- We select the option shown :



STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

30. MODIFY THE FILE

log4j2.xml:

[Click to download](#)

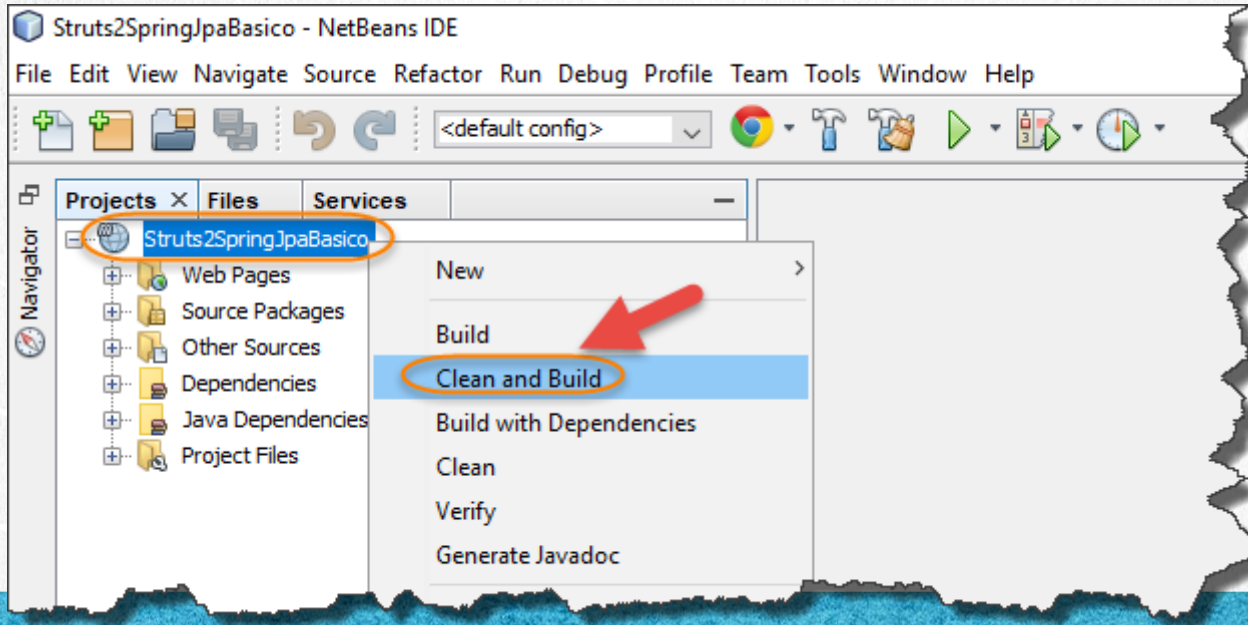
```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration>
  <Appenders>
    <Console name="STDOUT" target="SYSTEM_OUT">
      <PatternLayout pattern="%F:%L) - %m%n"/>
    </Console>
  </Appenders>
  <Loggers>
    <Logger name="com.opensymphony.xwork2" level="info"/>
    <Logger name="org.apache.struts2"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

STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

31. EXECUTE CLEAN & BUILD

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

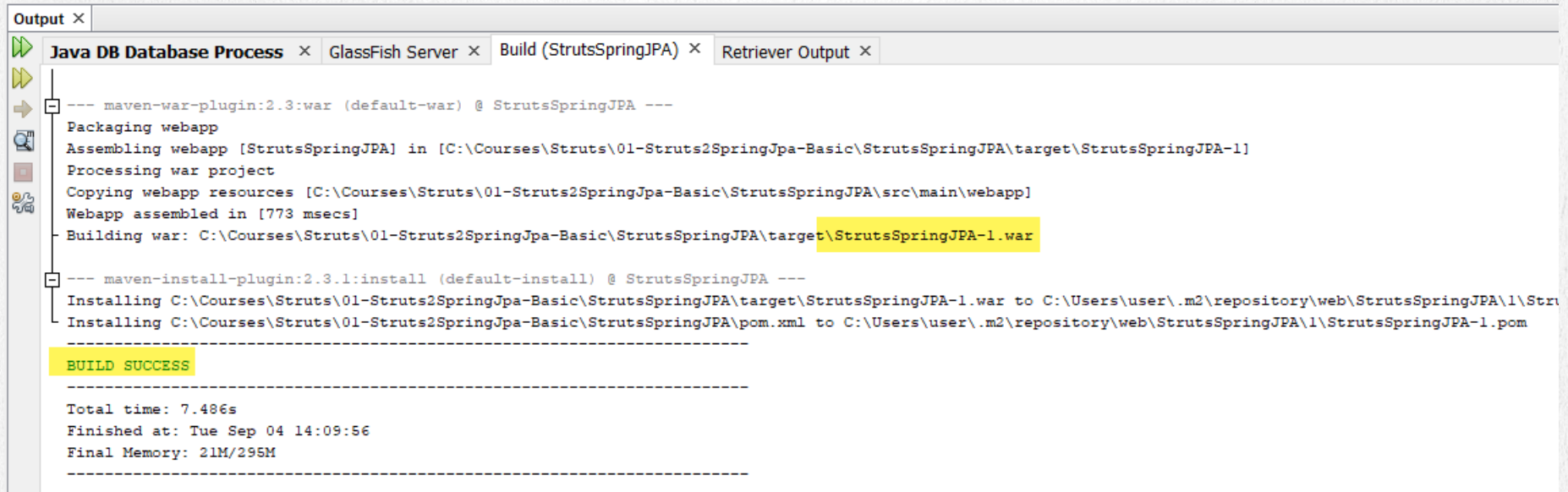


STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

31. 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 several tabs: 'Java DB Database Process', 'GlassFish Server', 'Build (StrutsSpringJPA)', and 'Retriever Output'. The 'Build (StrutsSpringJPA)' tab is active, displaying the following log:

```
--- maven-war-plugin:2.3:war (default-war) @ StrutsSpringJPA ---
Packaging webapp
Assembling webapp [StrutsSpringJPA] in [C:\Courses\Struts\01-Struts2SpringJpa-Basic\StrutsSpringJPA\target\StrutsSpringJPA-1]
Processing war project
Copying webapp resources [C:\Courses\Struts\01-Struts2SpringJpa-Basic\StrutsSpringJPA\src\main\webapp]
Webapp assembled in [773 msec]
Building war: C:\Courses\Struts\01-Struts2SpringJpa-Basic\StrutsSpringJPA\target\StrutsSpringJPA-1.war

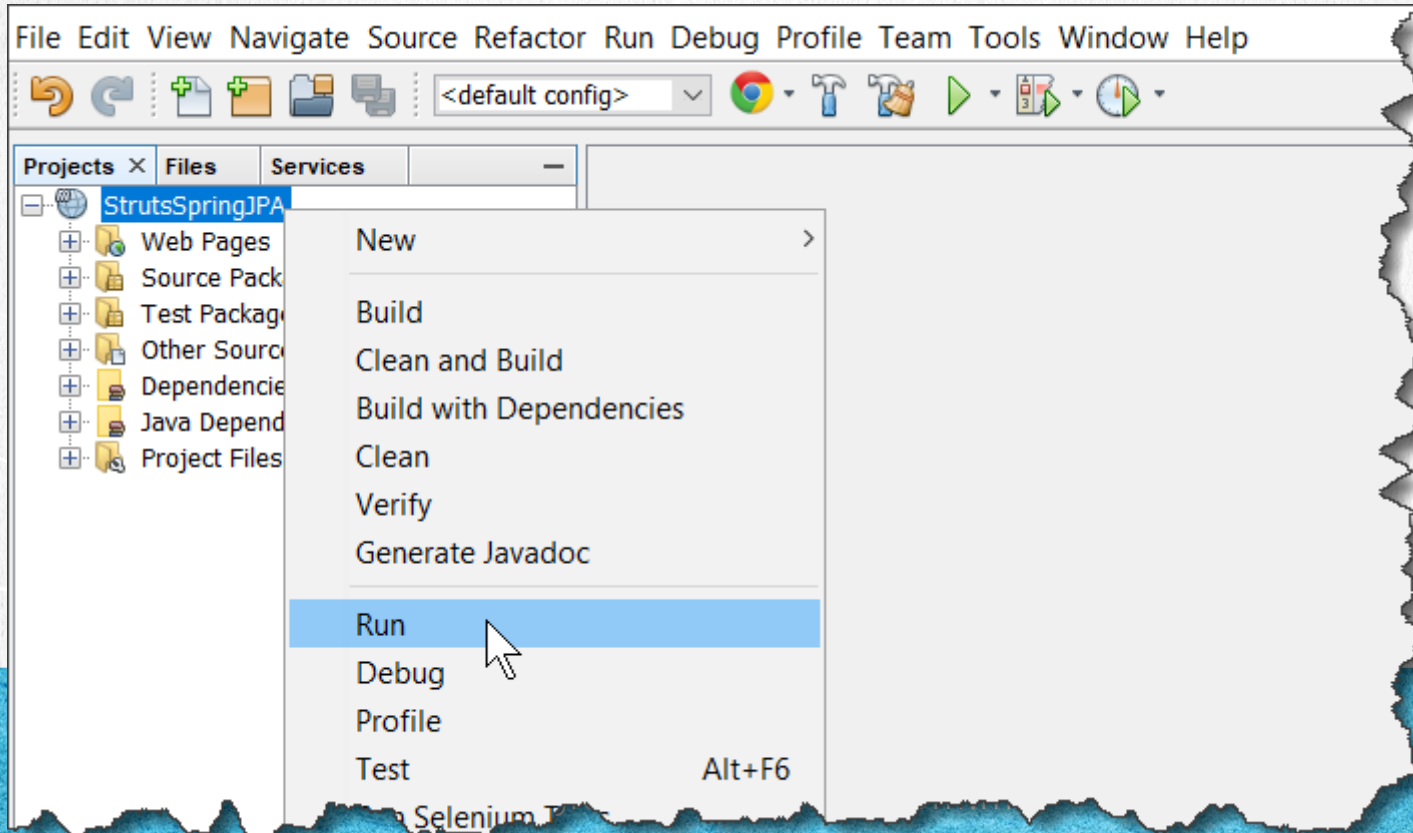
--- maven-install-plugin:2.3.1:install (default-install) @ StrutsSpringJPA ---
Installing C:\Courses\Struts\01-Struts2SpringJpa-Basic\StrutsSpringJPA\target\StrutsSpringJPA-1.war to C:\Users\user\.m2\repository\web\StrutsSpringJPA\1\StrutsSpringJPA-1.war
Installing C:\Courses\Struts\01-Struts2SpringJpa-Basic\StrutsSpringJPA\pom.xml to C:\Users\user\.m2\repository\web\StrutsSpringJPA\1\StrutsSpringJPA-1.pom

-----
BUILD SUCCESS
-----

Total time: 7.486s
Finished at: Tue Sep 04 14:09:56
Final Memory: 21M/295M
-----
```

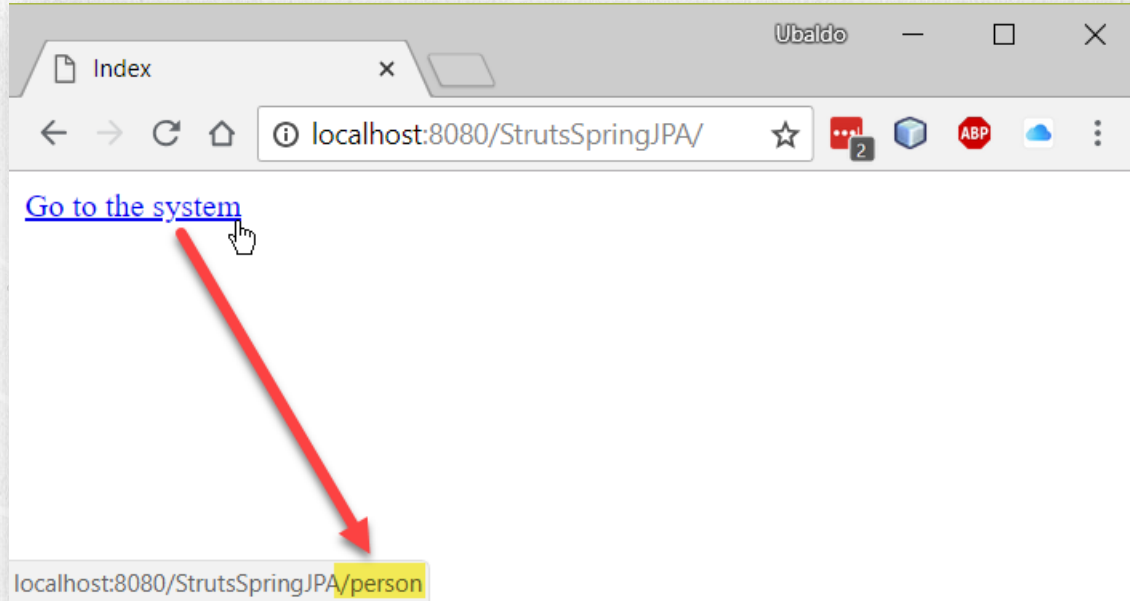
32. EXECUTE THE APPLICATION

- We execute the StrutsSpringJPA application as follows:



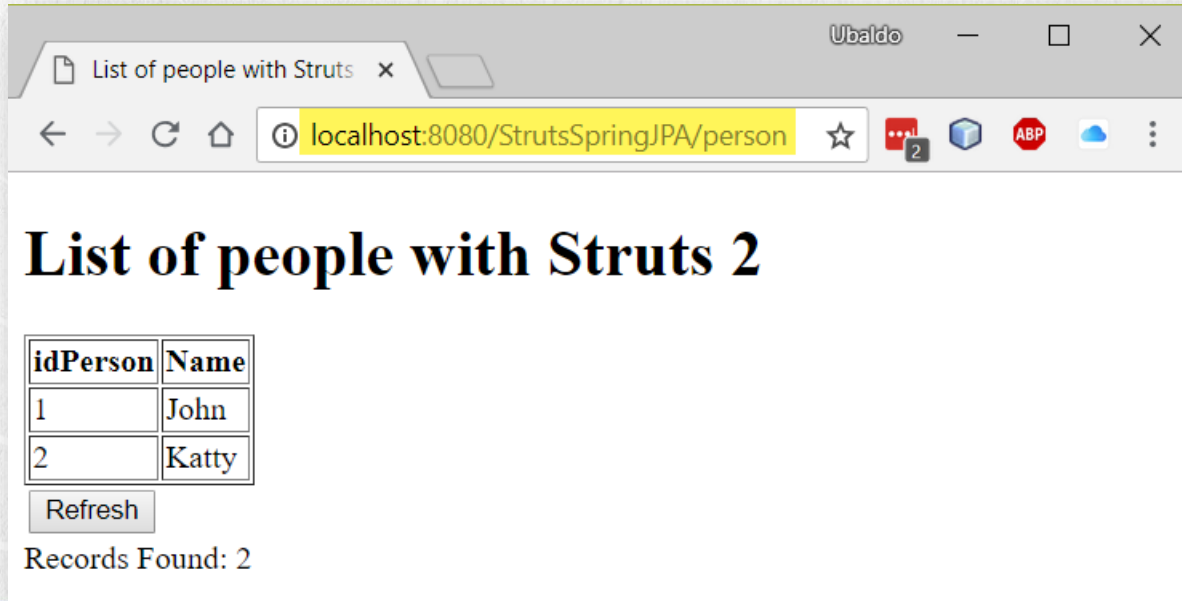
32. EXECUTE THE APPLICATION

- We execute the application as follows:



32. 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:



The screenshot shows a web browser window with the address bar displaying `localhost:8080/StrutsSpringJPA/person`. The page title is "List of people with Struts 2". Below the title, there is a table with the following data:

idPerson	Name
1	John
2	Katty

Below the table, there is a "Refresh" button. At the bottom of the page, it says "Records Found: 2".

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

CONCLUSION OF THE EXERCISE

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

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

The exercise has a list of people, but it is an architecture of the real world, with logical layers, fully functional and to implement the integration of the 3 technologies.



STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

ONLINE COURSE

STRUTS 2 FRAMEWORK

By: Eng. Ubaldo Acosta



STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx