

HIBERNATE AND JPA COURSE

EXERCISE

WEB APPLICATION WITH HIBERNATE & JPA

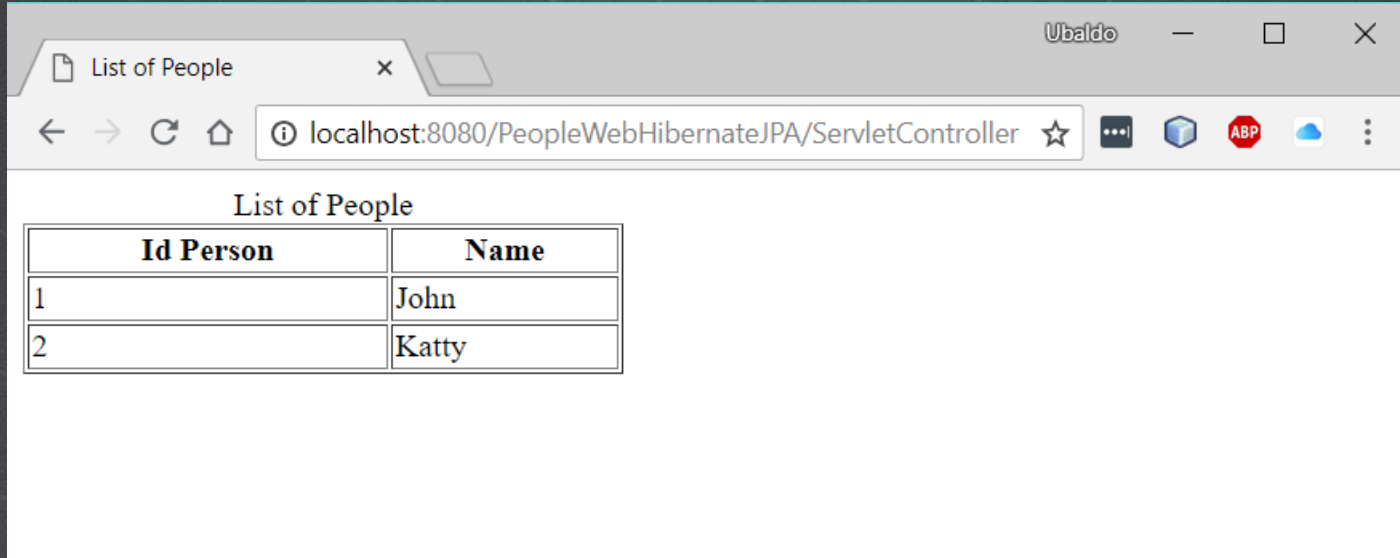


HIBERNATE AND JPA COURSE

www.globalmentoring.com.mx

EXERCISE OBJECTIVE

Create a Web application and integrate it with Hibernate and JPA. The final result is as follows:

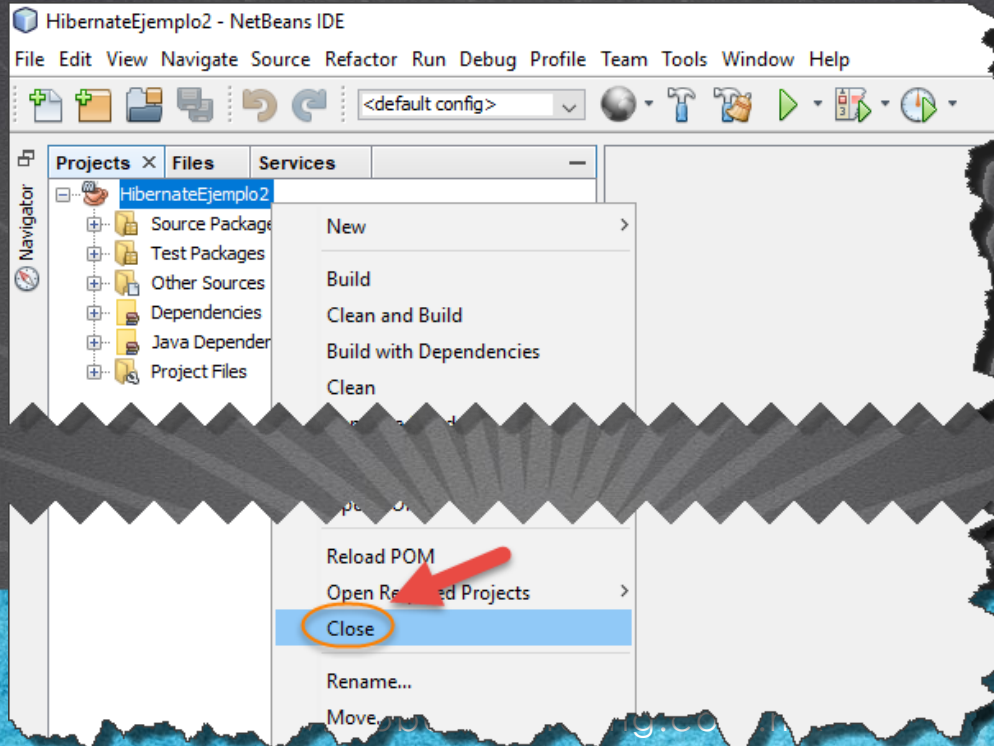


HIBERNATE AND JPA COURSE

www.globalmentoring.com.mx

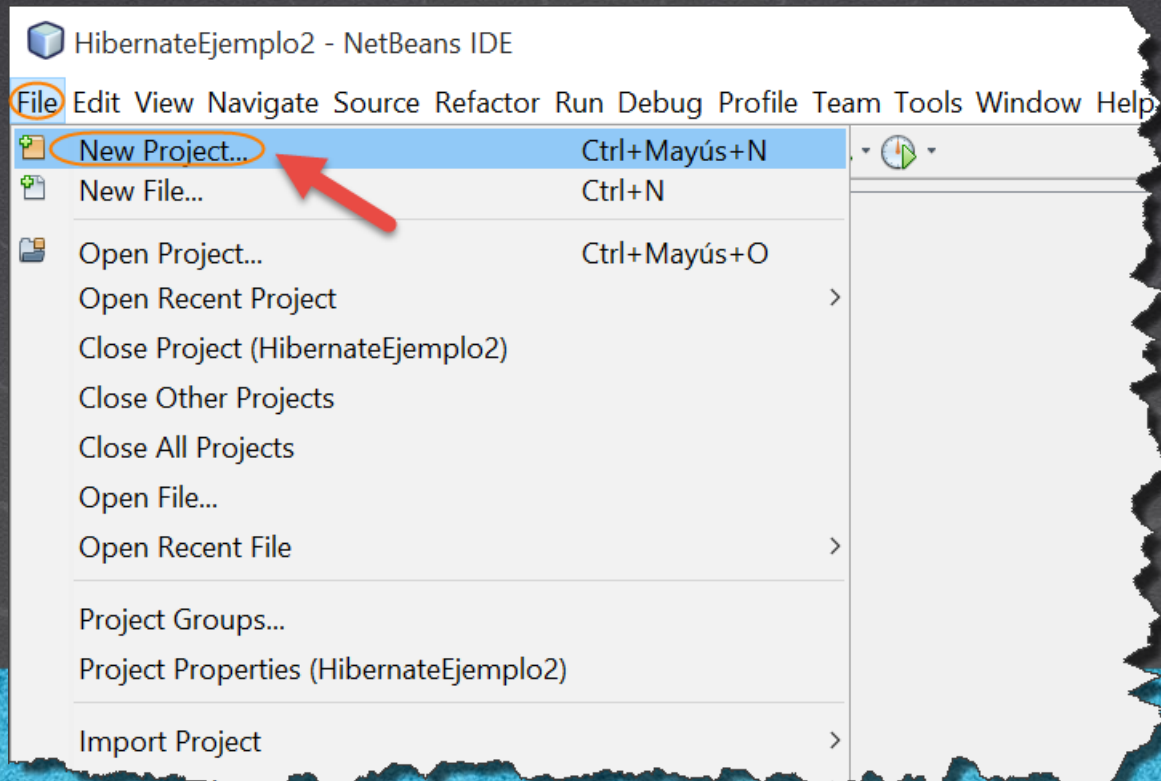
CLOSE ANY OTHER PROJECT

We close any other open project, in order to work only with the project we are going to create:



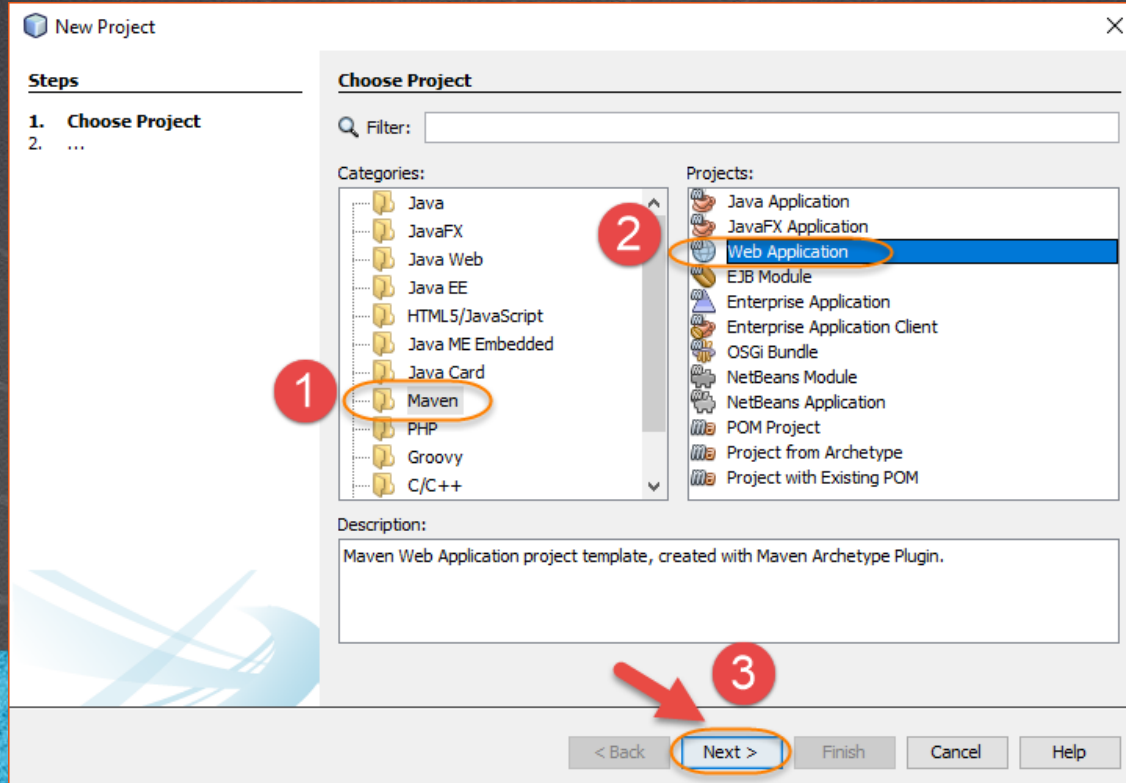
1. CREATE A NEW PROJECT

Create a new Web project using Maven:



1. CREATE A NEW PROJECT

Create a new Web project using Maven:



1. CREATE A NEW PROJECT

Create a new Web project using Maven:

New Web Application [X]

Steps

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

Name and Location

Project Name: PeopleWebHibernateJPA

Project Location: C:\Courses\Hibernate\Lesson03 [Browse...]

Project Folder: \Courses\Hibernate\Lesson03\PeopleWebHibernateJPA

Artifact Id: PeopleWebHibernateJPA

Group Id: web

Version: 1

Package: (Optional)

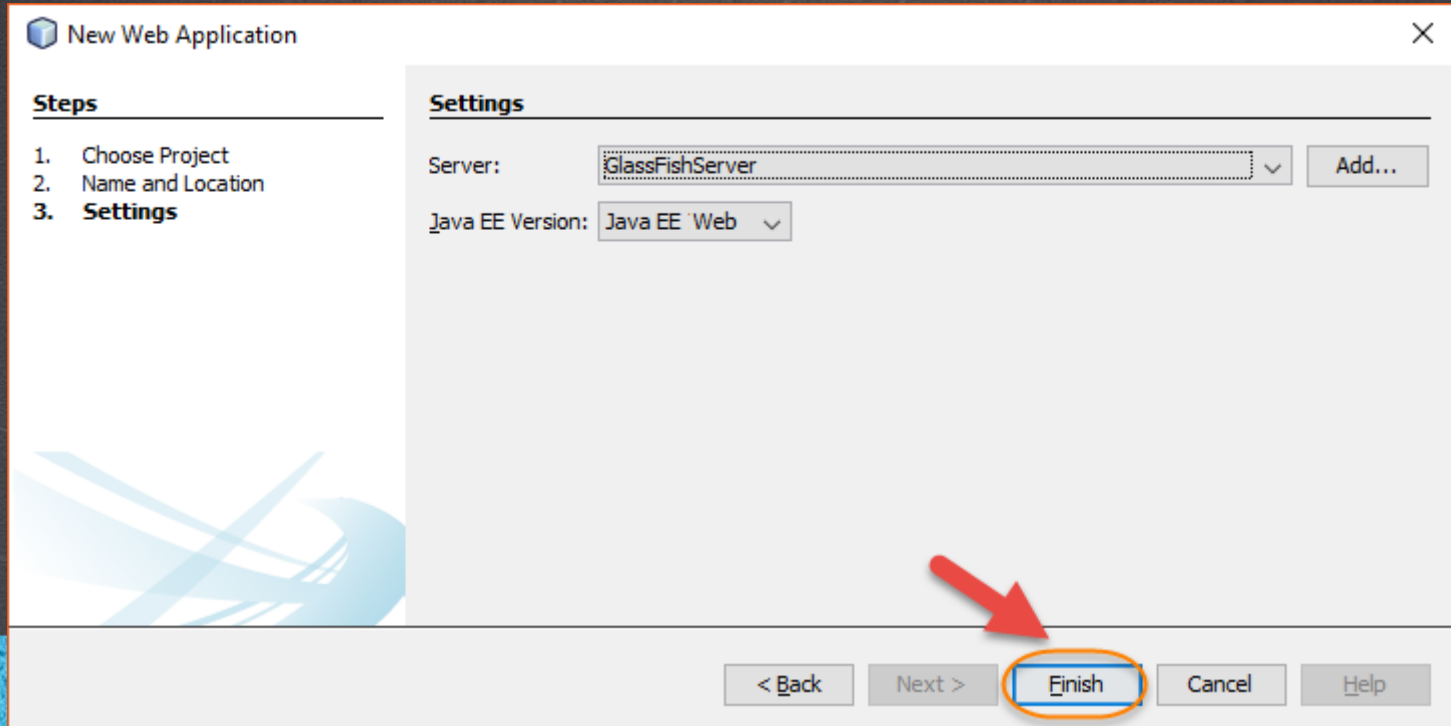
< Back **Next >** Finish Cancel Help

HIBERNATE AND JPA COURSE

www.globalmentoring.com.mx

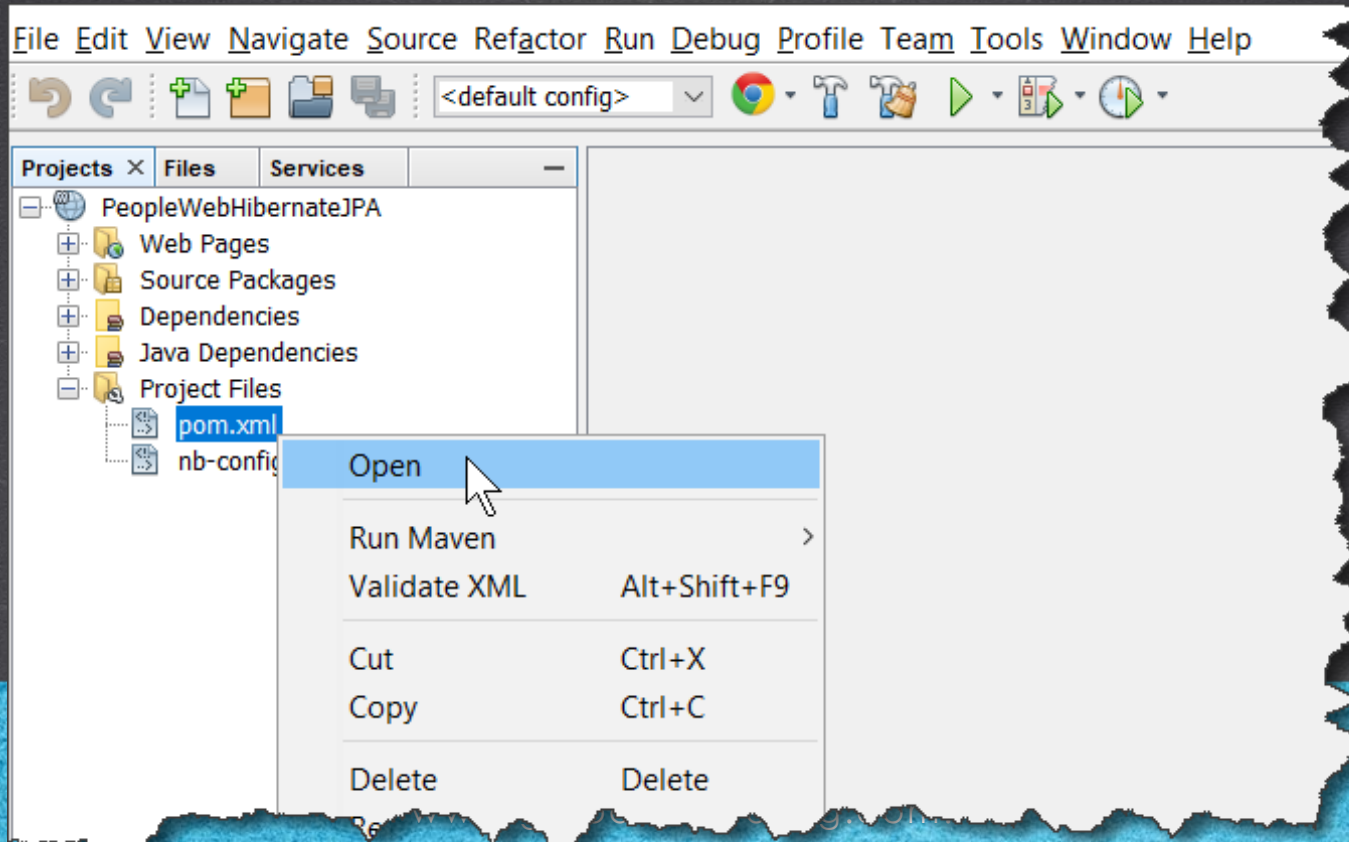
1. CREATE A NEW PROJECT

Create a new Web project using Maven:



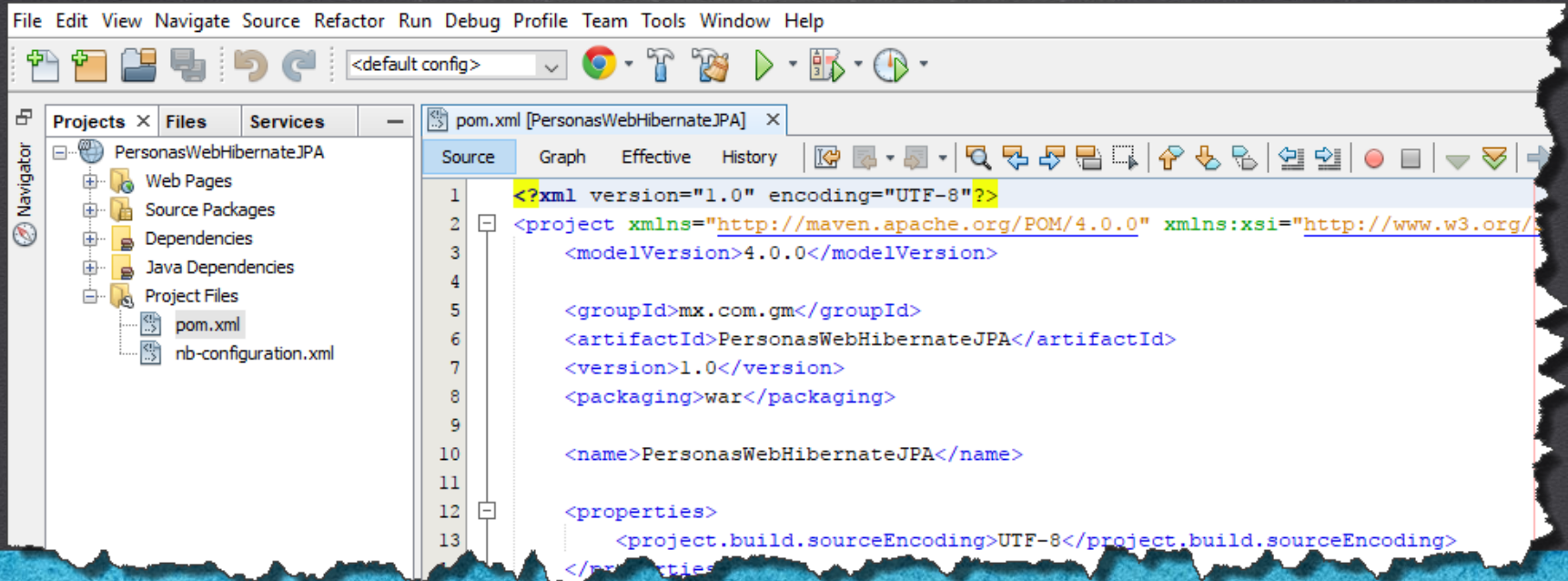
2. OPEN MAVEN'S POM.XML FILE

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



2. OPEN MAVEN'S POM.XML FILE

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



CURSO STRUTS FRAMEWORK

www.globalmentoring.com.mx

3. MODIFY THE CODE

[pom.xml:](#)

Click to download

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

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

    <name>PeopleWebHibernateJPA</name>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>
    <dependencies>
        <dependency>
            <groupId>javax</groupId>
            <artifactId>javaee-web-api</artifactId>
            <version>8.0</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-core</artifactId>
            <version>5.3.6.Final</version>
        </dependency>
```

3. MODIFY THE CODE

[pom.xml:](#)

Click to download

```
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>5.1.46</version>
</dependency>
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-api</artifactId>
  <version>2.11.1</version>
</dependency>
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-core</artifactId>
  <version>2.11.1</version>
</dependency>
</dependencies>
```


3. MODIFY THE CODE

[pom.xml:](#)

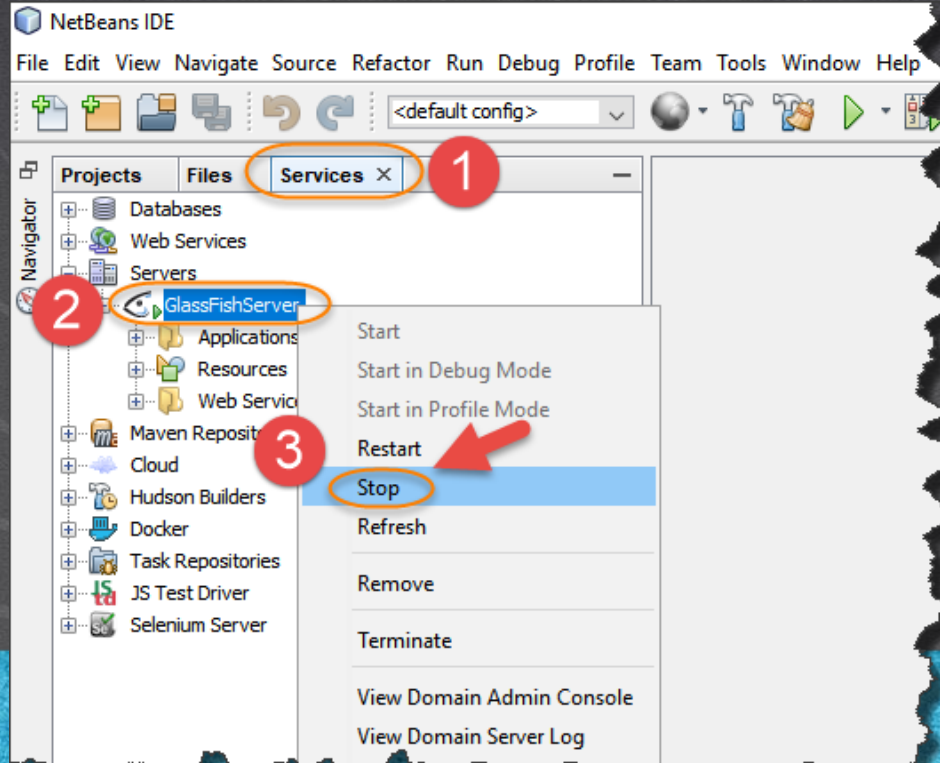
[Click to download](#)

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-war-plugin</artifactId>
      <version>2.3</version>
      <configuration>
        <failOnMissingWebXml>false</failOnMissingWebXml>
      </configuration>
    </plugin>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.7.0</version>
      <configuration>
        <source>1.8</source>
        <target>1.8</target>
      </configuration>
    </plugin>
  </plugins>
</build>

</project>
```

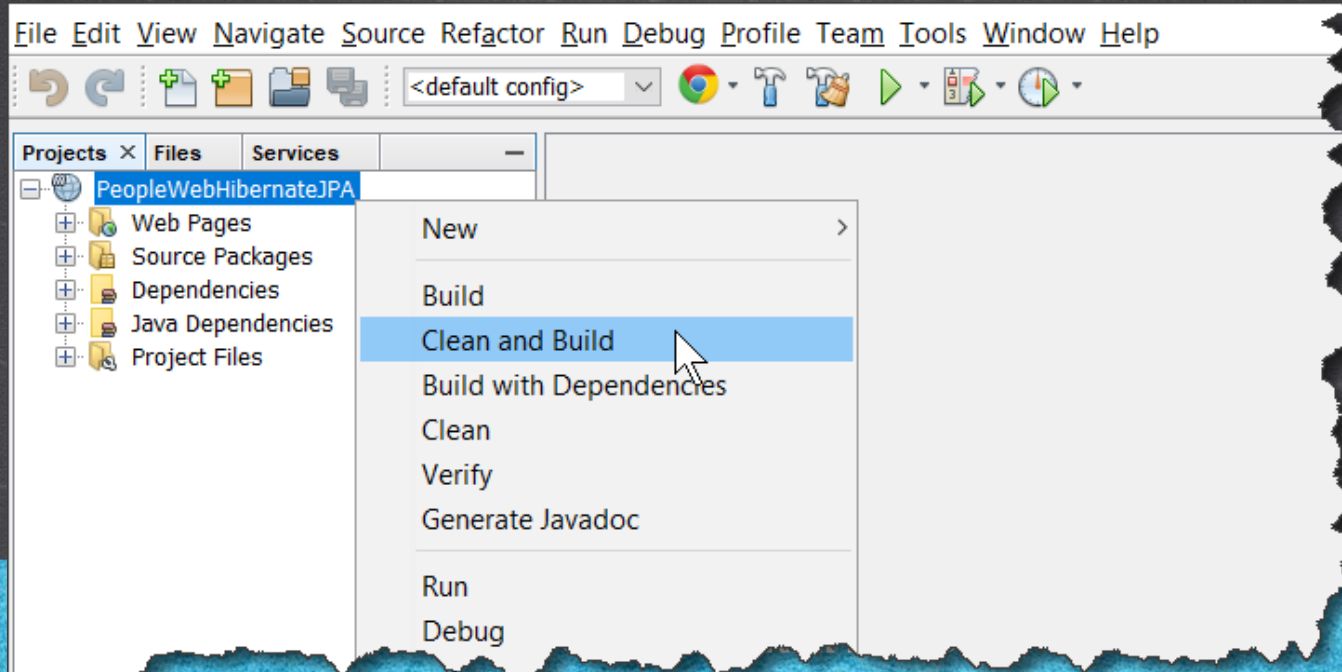
4. STOP GLASSFISH IF IT WAS STARTED

- Before doing Clean & Build of the project to download the new libraries, we verify that the Glassfish server is not started as there may be problems to do the Clean & build process if the server is started. This step is only verification :



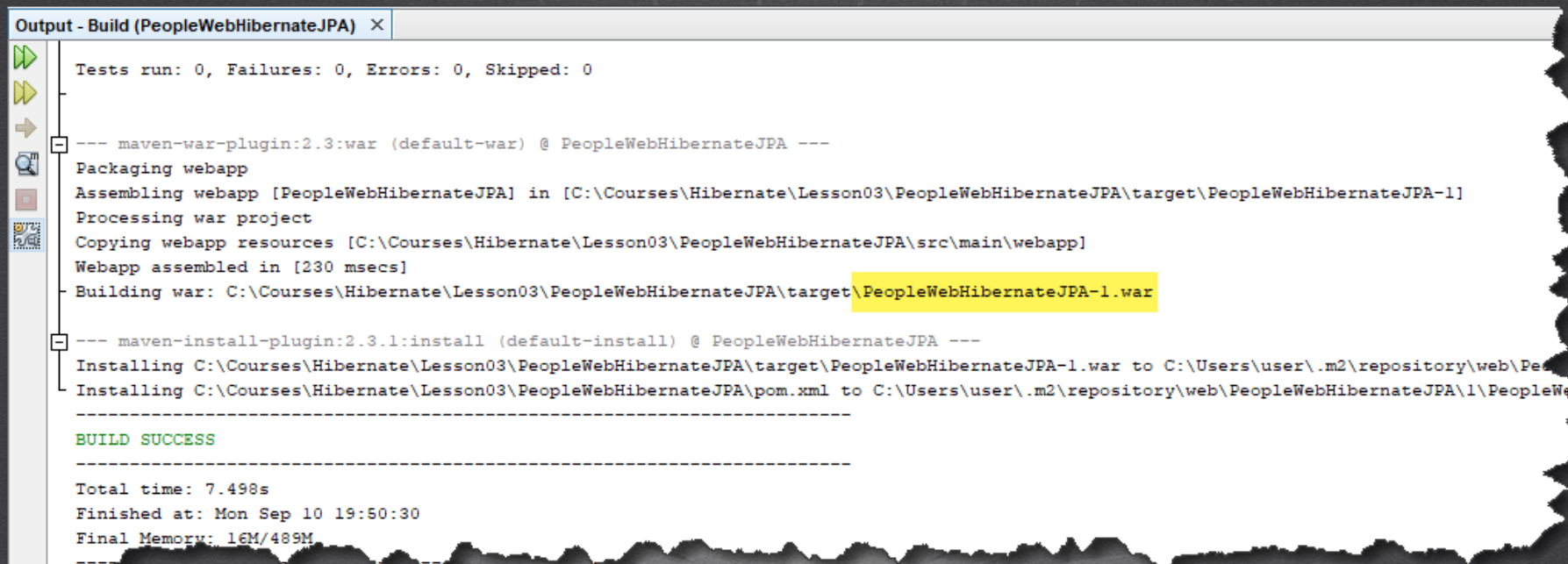
5. EXECUTE CLEAN & BUILD

• We return to the Projects view. In order to download the new libraries, we make Clean & Build the project. If for some reason this process fails, you must disable any software such as antivirus, Windows defender or firewall during this process so that the download of Java .jar files is not prevented. Once finished, these services can be activated again. This process may take several minutes depending on your internet speed:



5. EXECUTE CLEAN & BUILD

- If you no longer had to download any library because you could already have all downloaded, the process is faster. In the end we should observe the following :



The screenshot shows the 'Output - Build (PeopleWebHibernateJPA)' window in an IDE. The output text is as follows:

```
Tests run: 0, Failures: 0, Errors: 0, Skipped: 0

--- maven-war-plugin:2.3:war (default-war) @ PeopleWebHibernateJPA ---
Packaging webapp
Assembling webapp [PeopleWebHibernateJPA] in [C:\Courses\Hibernate\Lesson03\PeopleWebHibernateJPA\target\PeopleWebHibernateJPA-1]
Processing war project
Copying webapp resources [C:\Courses\Hibernate\Lesson03\PeopleWebHibernateJPA\src\main\webapp]
Webapp assembled in [230 msecs]
Building war: C:\Courses\Hibernate\Lesson03\PeopleWebHibernateJPA\target\PeopleWebHibernateJPA-1.war

--- maven-install-plugin:2.3.1:install (default-install) @ PeopleWebHibernateJPA ---
Installing C:\Courses\Hibernate\Lesson03\PeopleWebHibernateJPA\target\PeopleWebHibernateJPA-1.war to C:\Users\user\.m2\repository\web\Pe
Installing C:\Courses\Hibernate\Lesson03\PeopleWebHibernateJPA\pom.xml to C:\Users\user\.m2\repository\web\PeopleWebHibernateJPA\1\PeopleW

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

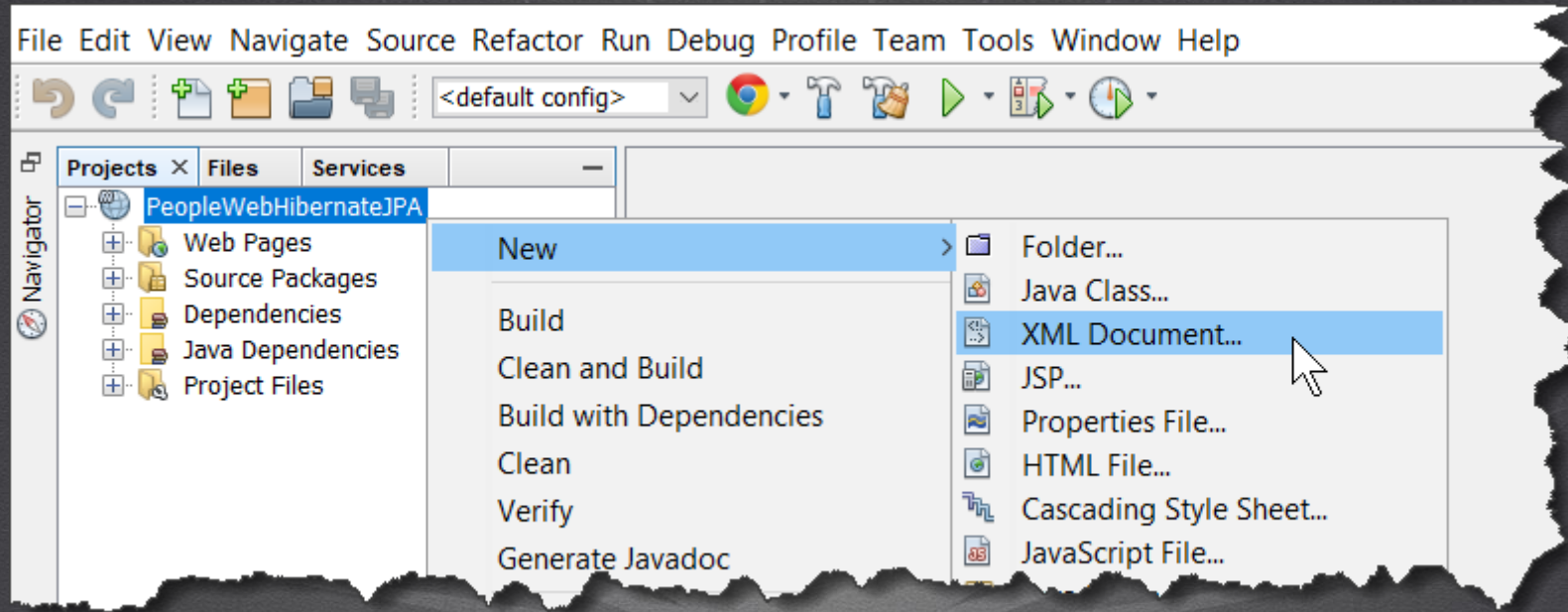
Total time: 7.498s
Finished at: Mon Sep 10 19:50:30
Final Memory: 16M/489M
```

CURSO STRUTS FRAMEWORK

www.globalmentoring.com.mx

6. CREATE AN XML FILE

We create the persistence.xml file



HIBERNATE AND JPA COURSE

www.globalmentoring.com.mx

6. CREATE AN XML FILE

We create the persistence.xml file

New XML Document

Steps

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

Name and Location

File Name: persistence

Project: PeopleWebHibernateJPA

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

Created File: C:\Courses\Hibernate\Lesson03\PeopleWebHibernateJPA\src/main/resources/META-INF\persistence.xml

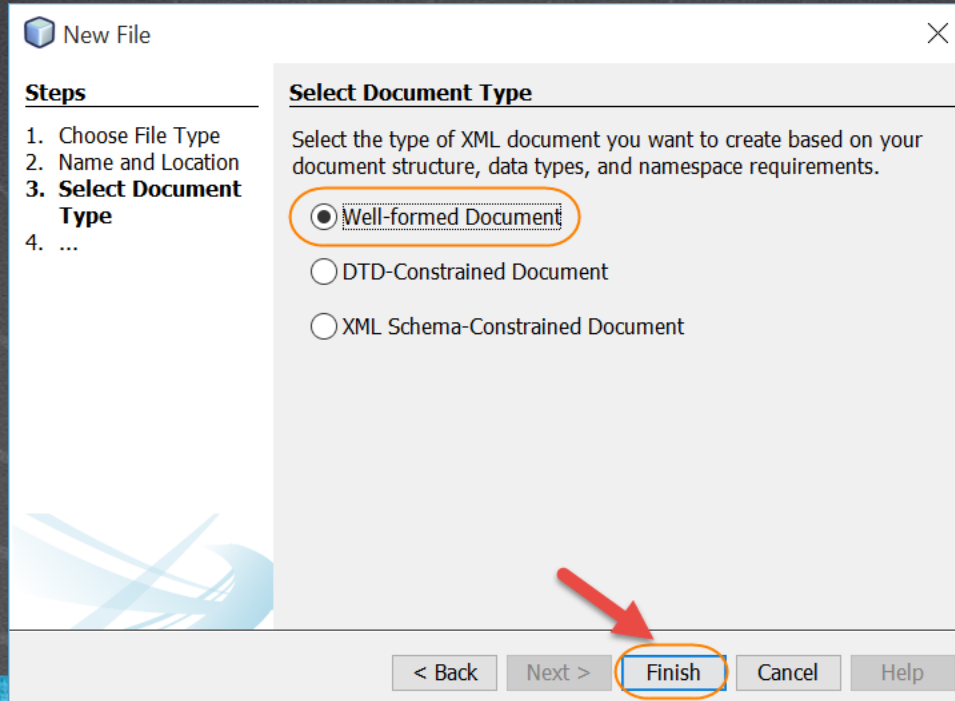
< Back **Next >** Finish Cancel Help

HIBERNATE AND JPA COURSE

www.globalmentoring.com.mx

6. CREATE AN XML FILE

We create the persistence.xml file



HIBERNATE AND JPA COURSE

www.globalmentoring.com.mx

7. MODIFY THE CODE

persistence.xml:

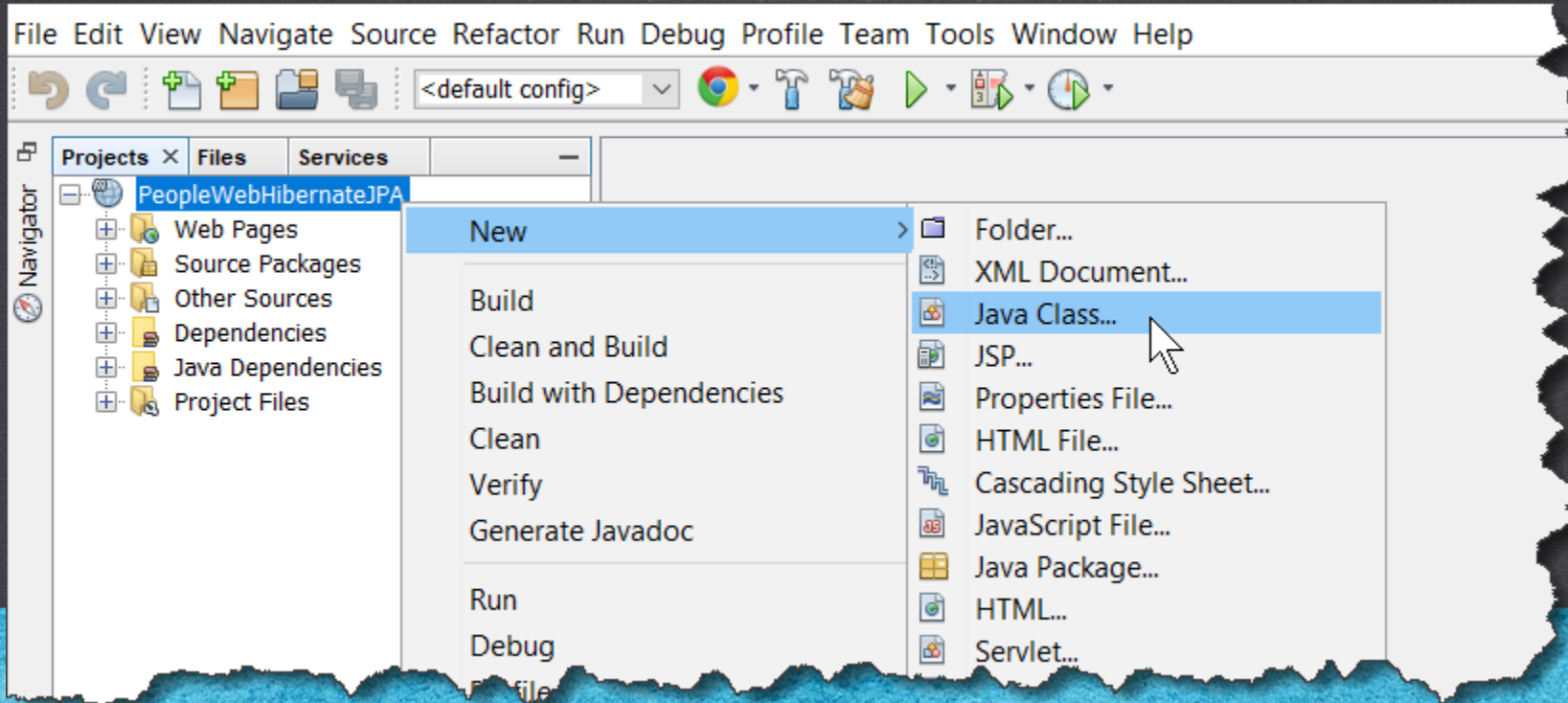
```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
    http://xmlns.jcp.org/xml/ns/persistence/persistence_2_2.xsd"
  version="2.2">
  <persistence-unit name="HibernatePU" transaction-type="RESOURCE_LOCAL">
    <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
    <class>domain.Person</class>
    <properties>
      <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/test?useSSL=true"/>
      <property name="javax.persistence.jdbc.user" value="root"/>
      <property name="javax.persistence.jdbc.password" value="admin"/>
      <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver"/>
      <property name="hibernate.show_sql" value="true"/>
    </properties>
  </persistence-unit>
</persistence>
```

HIBERNATE AND JPA COURSE

www.globalmentoring.com.mx

8. CREATE A NEW CLASS

Create the Person.java class:



8. CREATE A NEW CLASS

Create the Person.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

HIBERNATE AND JPA COURSE

www.globalmentoring.com.mx

9. MODIFY THE CODE

Person.java:

```
package domain;

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

@Entity
public class Person implements Serializable {

    private static final long serialVersionUID = 1L;

    @Column(name = "id_person")
    @Id
    private int idPerson;

    private String name;

    public Person() {
    }
```

9. MODIFY THE CODE

Person.java:

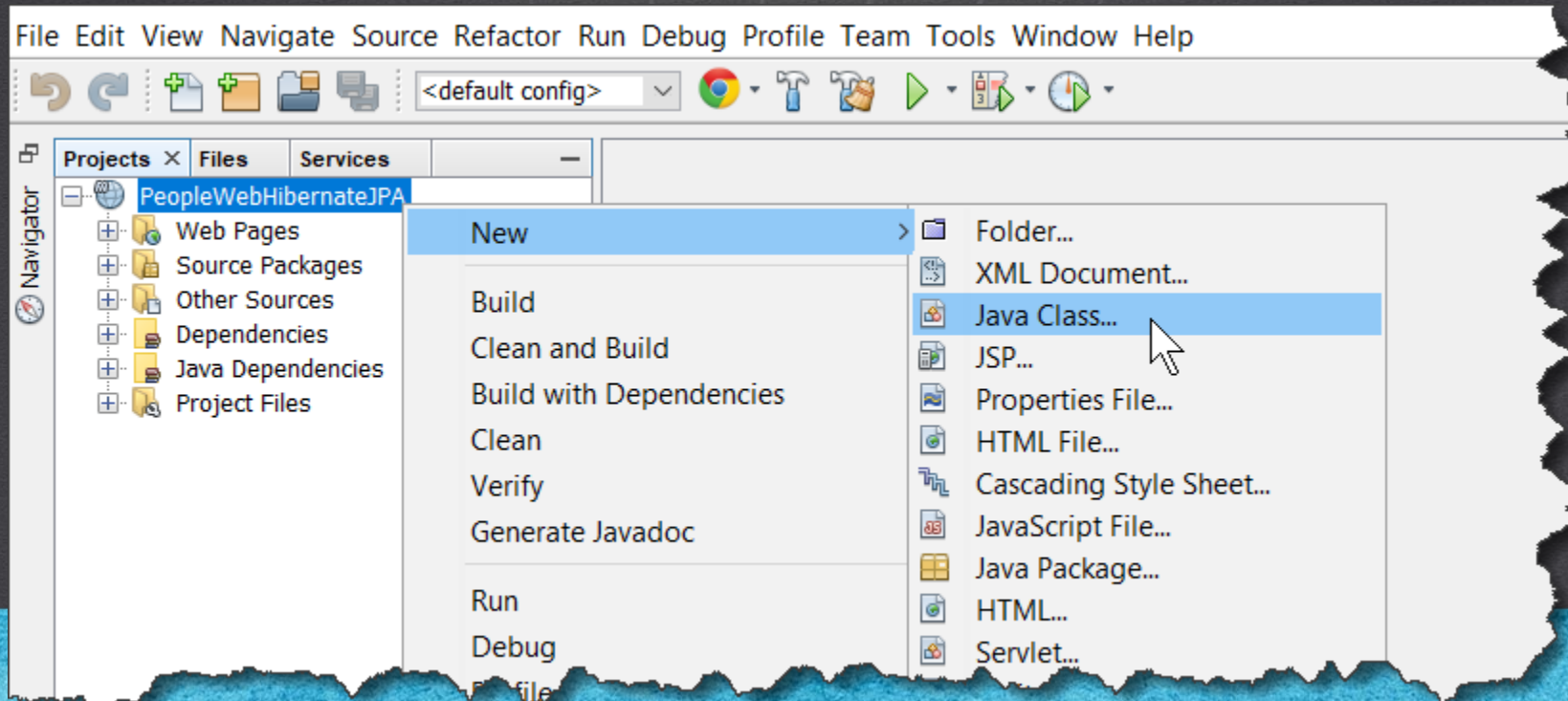
```
public int getIdPerson() {  
    return this.idPerson;  
}  
  
public void setIdPerson(int idPerson) {  
    this.idPerson = idPerson;  
}  
  
public String getName() {  
    return this.name;  
}  
  
public void setName(String name) {  
    this.name = name;  
}  
  
@Override  
public String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

HIBERNATE AND JPA COURSE

www.globalmentoring.com.mx

10. CREATE A NEW CLASS

We create the PersonDAO.java class:



10. CREATE A NEW CLASS

Create the PersonDAO.java class:

New Java Class

Steps

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

Name and Location

Class Name: PersonDao

Project: PeopleWebHibernateJPA

Location: Source Packages

Package: dao

Created File: C:\Courses\Hibernate\Lesson03\PeopleWebHibernateJPA\src\main\java\dao\PersonDao.java

< Back Next > **Finish** Cancel Help

HIBERNATE AND JPA COURSE

www.globalmentoring.com.mx

11. MODIFY THE CODE

PersonDao.java:

```
package dao;

import domain.Person;
import java.util.List;
import javax.persistence.*;

public class PersonDao {

    protected EntityManager em;
    private EntityManagerFactory emf = null;

    public PersonDao() {
        emf = Persistence.createEntityManagerFactory("HibernatePU");
    }

    public List<Person> listPeople() {
        String hql = "SELECT p FROM Person p";
        em = getEntityManager();
        Query query = em.createQuery(hql);
        return query.getResultList();
    }
}
```


11. MODIFY THE CODE

PersonDao.java:

```
public void insert(Person person) {  
    try {  
        em = getEntityManager();  
        em.getTransaction().begin();  
        em.persist(person);  
        em.getTransaction().commit();  
    } catch (Exception ex) {  
        System.out.println("Error inserting object:" + ex.getMessage());  
        ex.printStackTrace(System.out);  
    } finally {  
        if (em != null) {  
            em.close();  
        }  
    }  
}
```

11. MODIFY THE CODE

PersonDao.java:

```
public void update(Person person) {  
    try {  
        em = getEntityManager();  
        em.getTransaction().begin();  
        em.merge(person);  
        em.getTransaction().commit();  
    } catch (Exception ex) {  
        System.out.println("Error updating object:" + ex.getMessage());  
        ex.printStackTrace(System.out);  
    } finally {  
        if (em != null) {  
            em.close();  
        }  
    }  
}
```

11. MODIFY THE CODE

PersonDao.java:

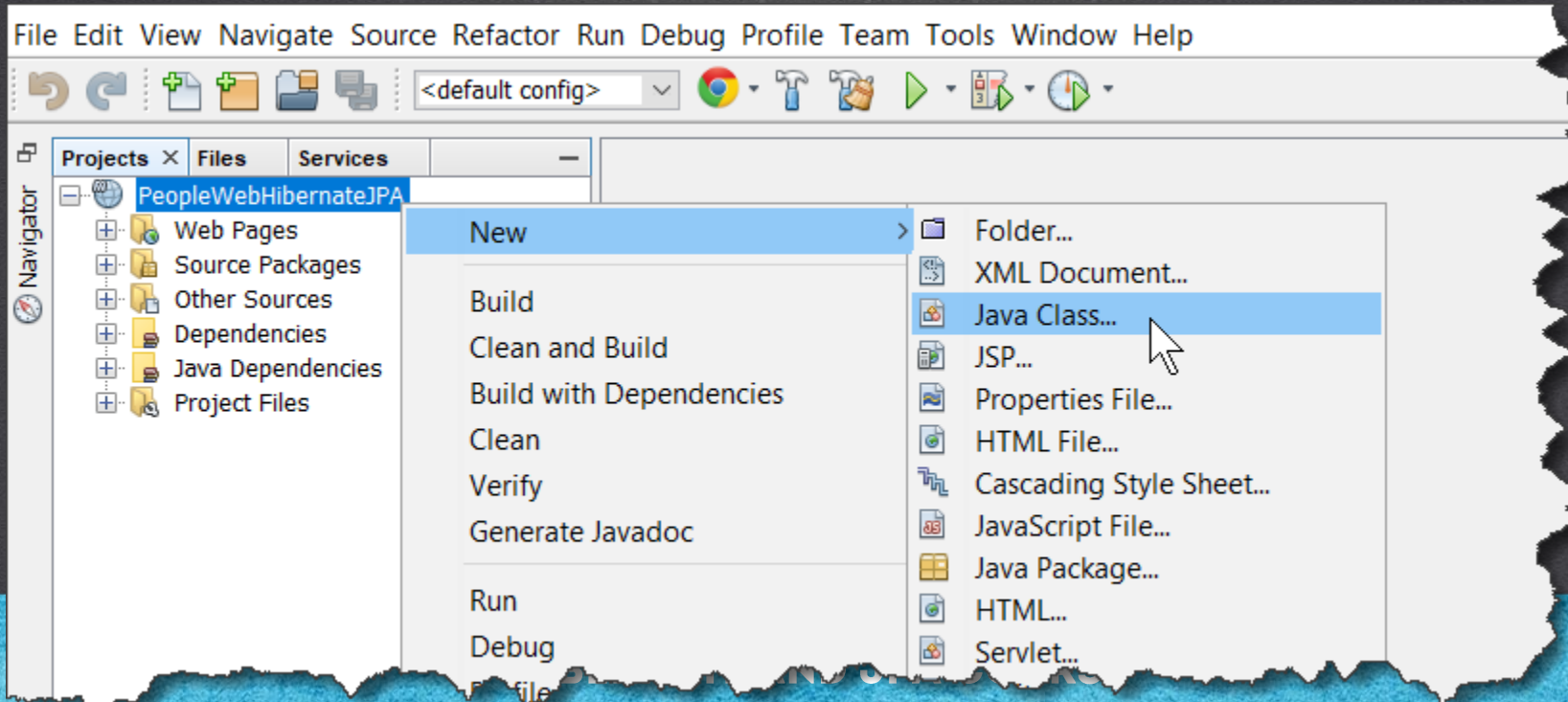
```
public void delete(Person person) {
    try {
        em = getEntityManager();
        em.getTransaction().begin();
        em.remove(em.merge(person));
        em.getTransaction().commit();
    } catch (Exception ex) {
        System.out.println("Error deleting object:" + ex.getMessage());
        ex.printStackTrace(System.out);
    } finally {
        if (em != null) {
            em.close();
        }
    }
}

public Person findById(Person p) {
    em = getEntityManager();
    return em.find(Person.class, p.getIdPerson());
}

private EntityManager getEntityManager() {
    return emf.createEntityManager();
}
}
```


12. CREATE A NEW CLASS

Create the PersonService class:



12. CREATE A NEW CLASS

We create the PersonService.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

HIBERNATE AND JPA COURSE

www.globalmentoring.com.mx

13. MODIFY THE CODE

PersonService.java:

```
package service;

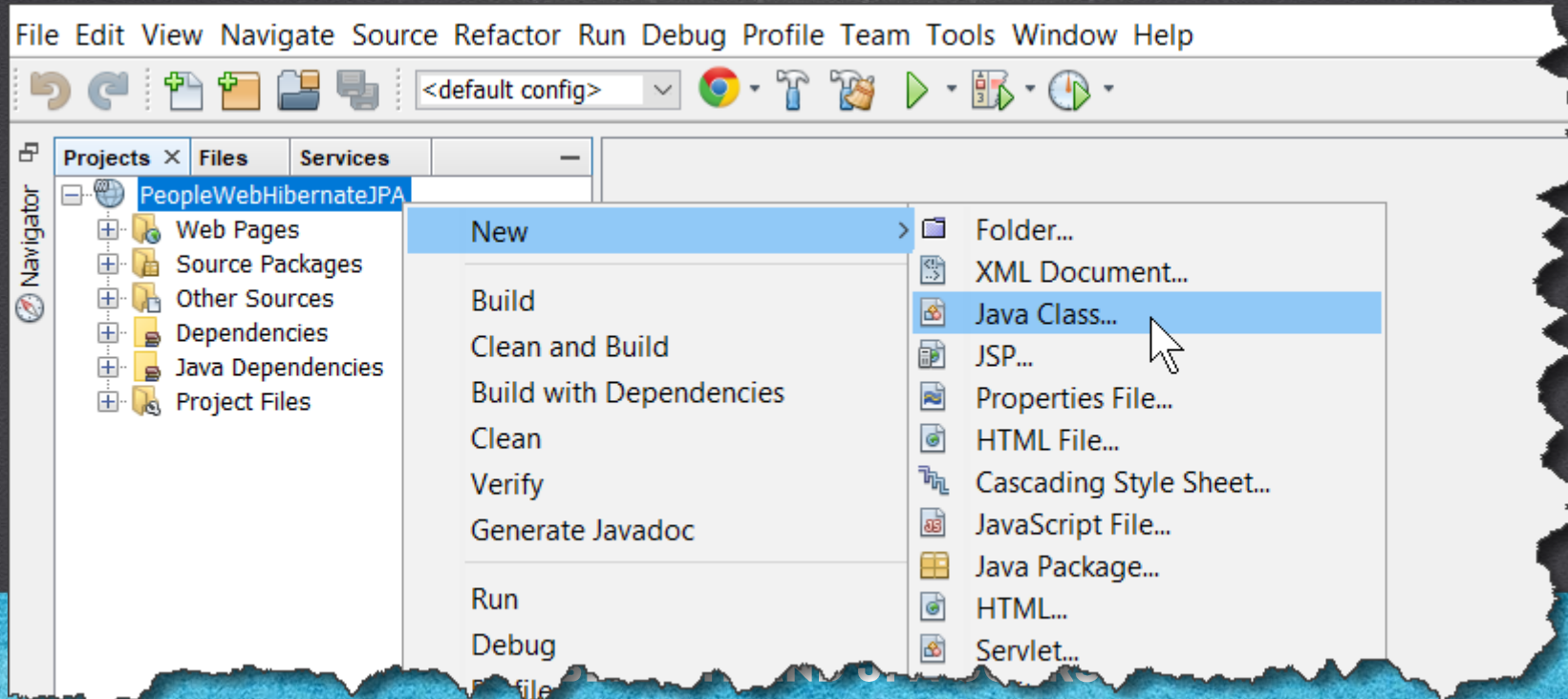
import dao.PersonDao;
import domain.Person;
import java.util.List;

public class PersonService {

    public List<Person> listPeople() {
        PersonDao personaDao = new PersonDao();
        return personaDao.listPeople();
    }
}
```


14. CREATE A NEW CLASS

We create the class ServletController.java:



14. CREATE A NEW CLASS

We create the class ServletController.java:

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

HIBERNATE AND JPA COURSE

www.globalmentoring.com.mx

15. MODIFY THE CODE

ServletController.java:

```
package web;

import domain.Person;
import java.io.IOException;
import java.util.List;
import javax.servlet.*;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import service.PersonService;

@WebServlet("/ServletController")
public class ServletController extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        try {
            // 1.We connect to the service layer
            PersonService personService = new PersonService();

            // 2.We request the list of people (Model)
            List<Person> people = personService.listPeople();
```


15. MODIFY THE CODE

ServletController.java:

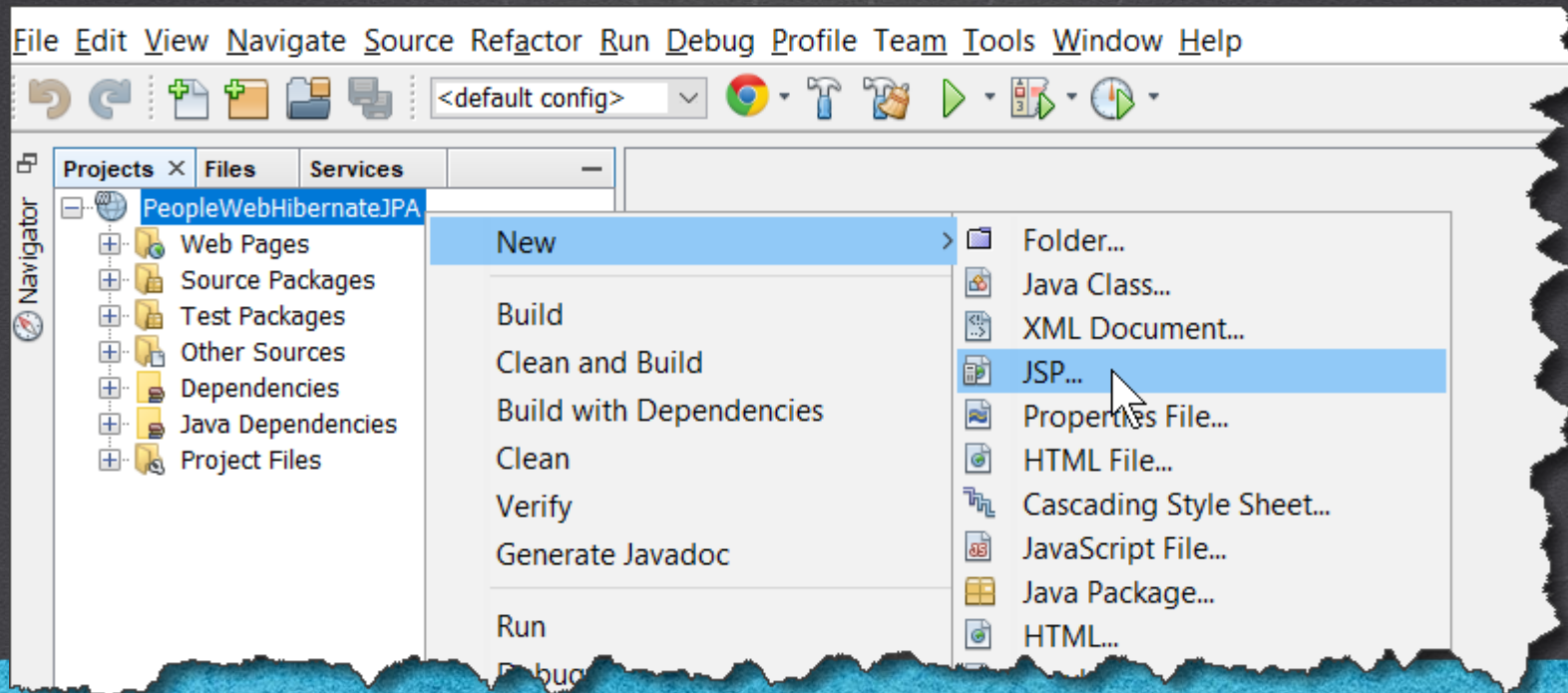
```
// 3.We share the list of people (Model) with the JSP (View)
request.setAttribute("people", people);

// 4.Redireccionamos a la vista
request.getRequestDispatcher("/WEB-INF/people.jsp").forward(request, response);

} catch (Exception e) {
    e.printStackTrace(System.out);
}
}
```

16. CREATE A JSP FILE

We create the file index.jsp



HIBERNATE AND JPA COURSE

www.globalmentoring.com.mx

16. CREATE A JSP FILE

We create the file index.jsp

New JSP

Steps

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

Name and Location

File Name:

Project:

Location:

Folder:

Browse...

Created File:

Options:

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

☐ JSP Document (XML Syntax)

Description:

< Back Next > **Finish** Cancel Help

17. MODIFY THE CODE

index.jsp:

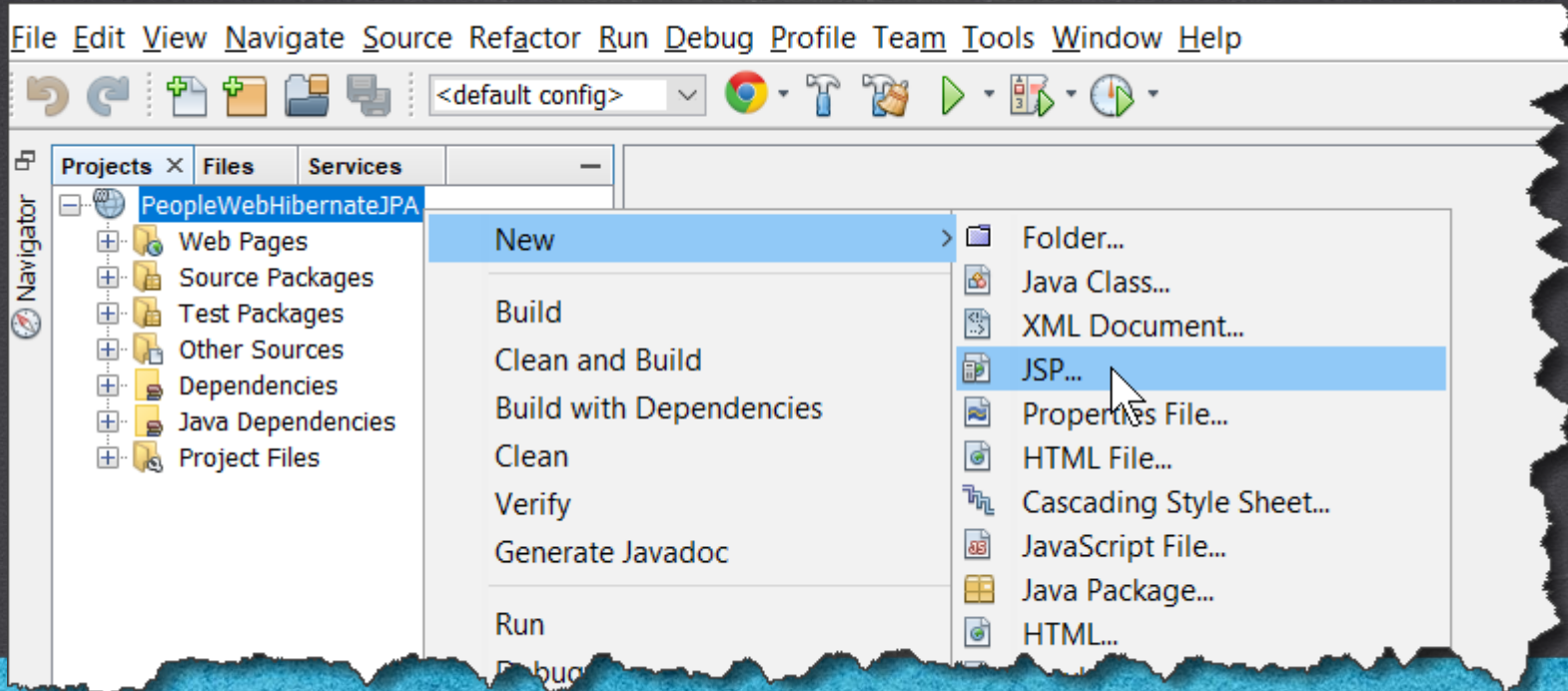
```
<html>
  <head>
    <meta charset="UTF-8">
    <title>Web App with Hibernate/JPA 1</title>
  </head>
  <body>
    <h1>Web App with Hibernate/JPA 1</h1>
    <br>
    <a href="${pageContext.request.contextPath}/ServletController">
      List People with Hibernate y JPA
    </a>
  </body>
</html>
```

HIBERNATE AND JPA COURSE

www.globalmentoring.com.mx

18. CREATE A JSP FILE

Create the people.jsp file:



HIBERNATE AND JPA COURSE

www.globalmentoring.com.mx

18. CREATE A JSP FILE

Create the people.jsp file:

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:

< Back Next > **Finish** Cancel Help

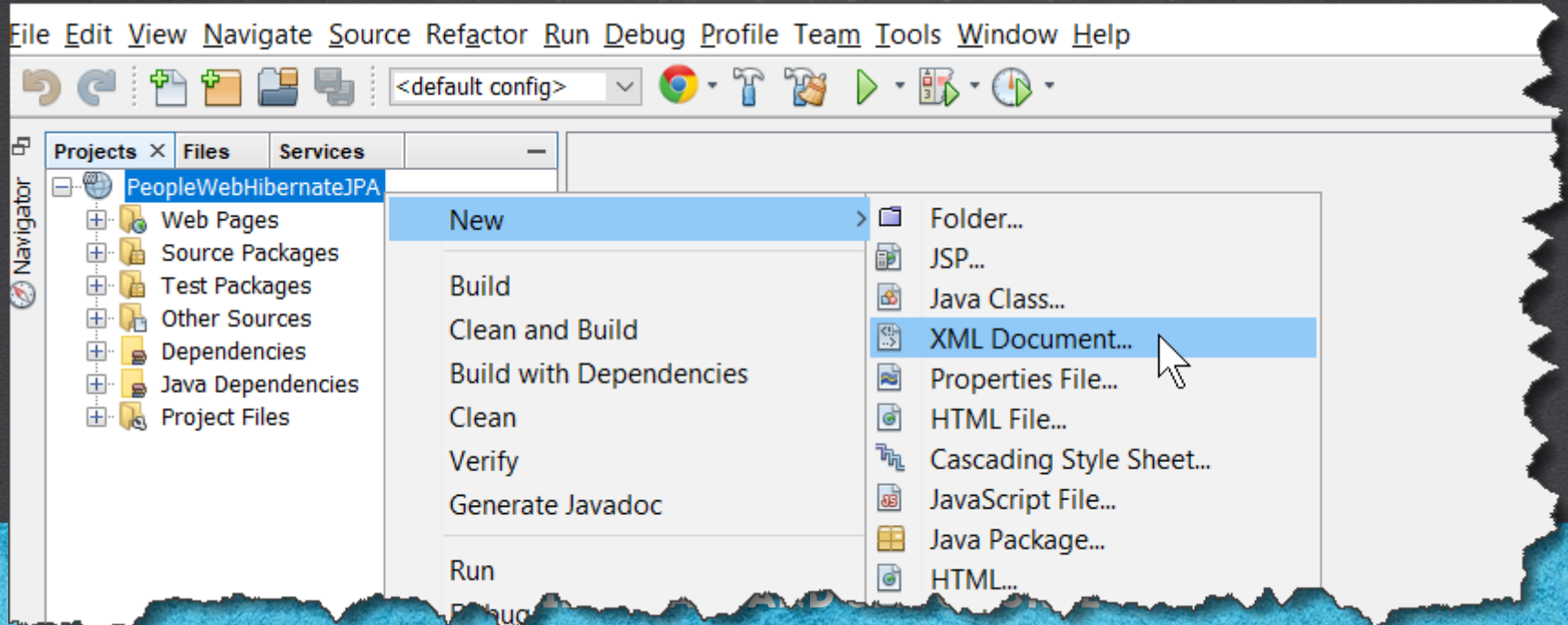
19. MODIFY THE CODE

people.jsp:

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
  <head>
    <meta charset="UTF-8">
    <title>List of People</title>
  </head>
  <body>
    <table width="300" border="1" >
      <tbody>
        <caption>List of People</caption>
        <tr>
          <th>Id Person<br></th>
          <th>Name</th>
        </tr>
        <!--We iterate the elements of the list of people-->
        <c:forEach var="person" items="${people}" >
          <tr>
            <td>${person.idPerson}</td>
            <td>${person.name}</td>
          </tr>
        </c:forEach>
      </tbody>
    </table>
  </body>
</html>
```

20. CREATE AN XML FILE

We create a log4j2.xml file. This file is used to manage Java logging:



20. CREATE AN XML FILE

Create the log4j2.xml file:

New XML Document

Steps

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

Name and Location

File Name: log4j2

Project: PeopleWebHibernateJPA

Folder: src/main/resources Browse...

Created File: C:\Courses\Hibernate\Lesson03\PeopleWebHibernateJPA\src\main\resources\log4j2.xml

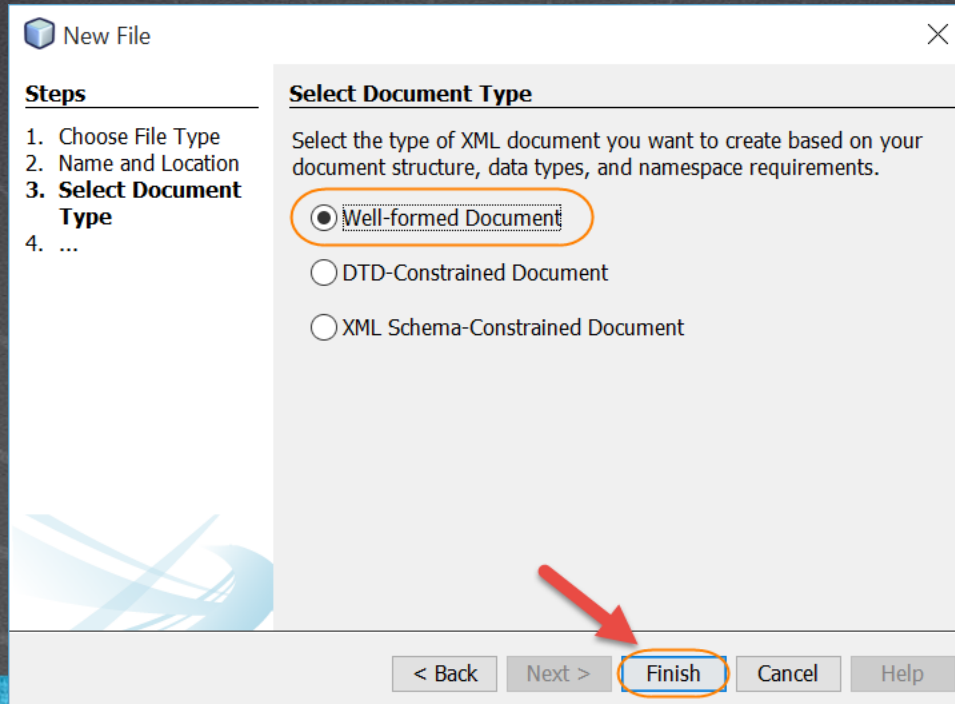
< Back **Next >** Finish Cancel Help

HIBERNATE AND JPA COURSE

www.globalmentoring.com.mx

20. CREATE AN XML FILE

Create the log4j2.xml file:



HIBERNATE AND JPA COURSE

www.globalmentoring.com.mx

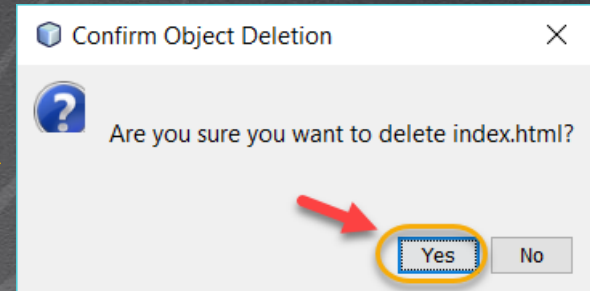
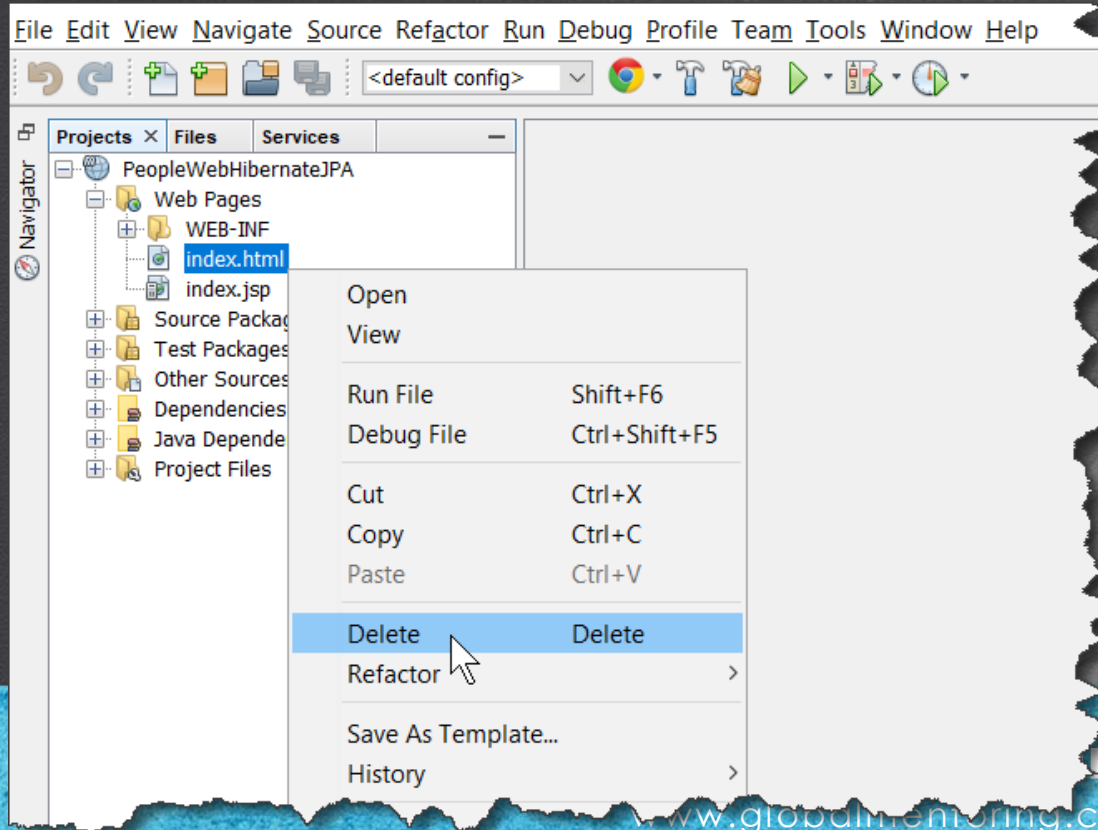
21. MODIFY THE CODE

log4j2.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="INFO">
  <Appenders>
    <Console name="Console" target="SYSTEM_OUT">
      <PatternLayout pattern="%d{HH:mm:ss} [%t] %-5level %logger{36} - %msg%n" />
    </Console>
  </Appenders>
  <Loggers>
    <Logger name="org.hibernate.SQL" level="debug" additivity="false">
      <AppenderRef ref="Console"/>
    </Logger>
    <logger name="org.hibernate.type.descriptor.sql.BasicBinder" level="trace" additivity="false">
      <AppenderRef ref="Console"/>
    </logger>
    <Root level="info">
      <AppenderRef ref="Console" />
    </Root>
  </Loggers>
</Configuration>
```

22. DELETE THE INDEX.HTML FILE

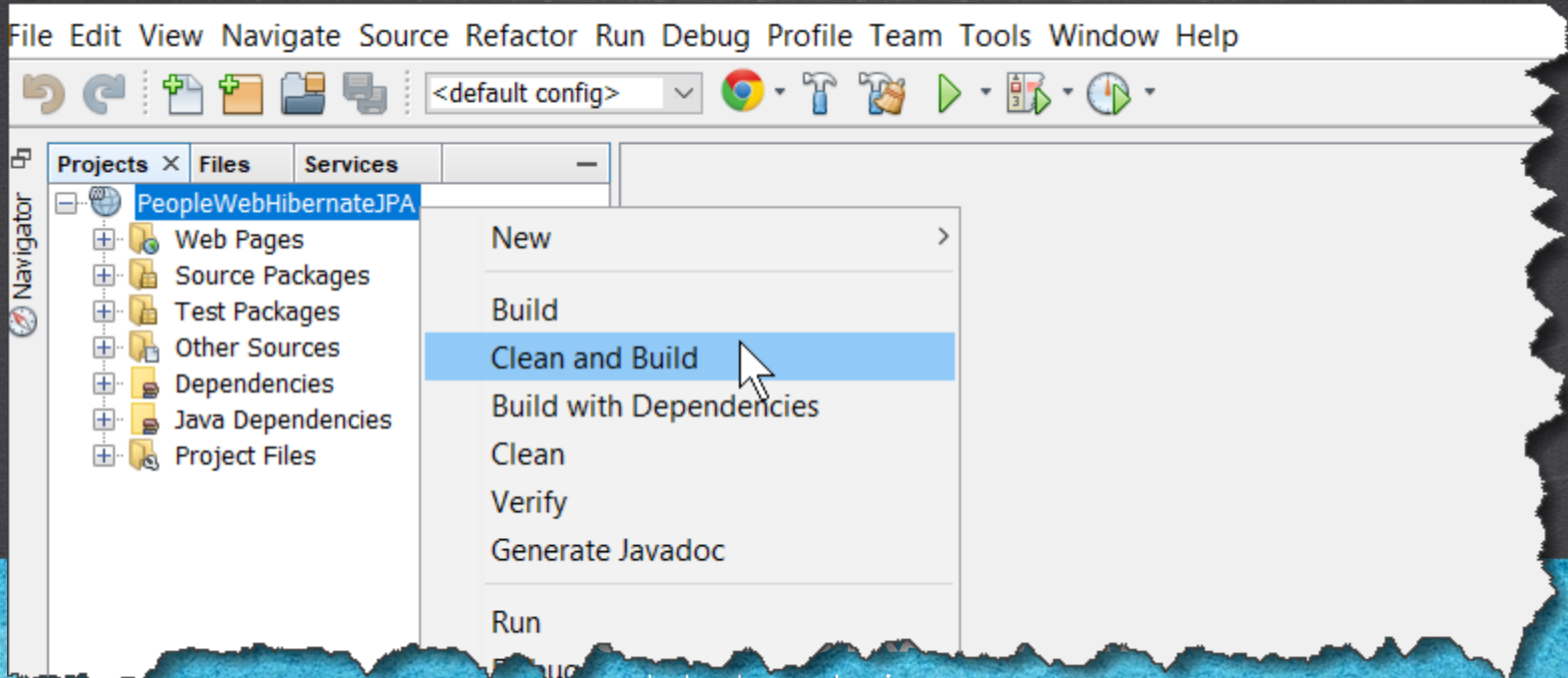
- Remove the index.html file if it exists:



WORK

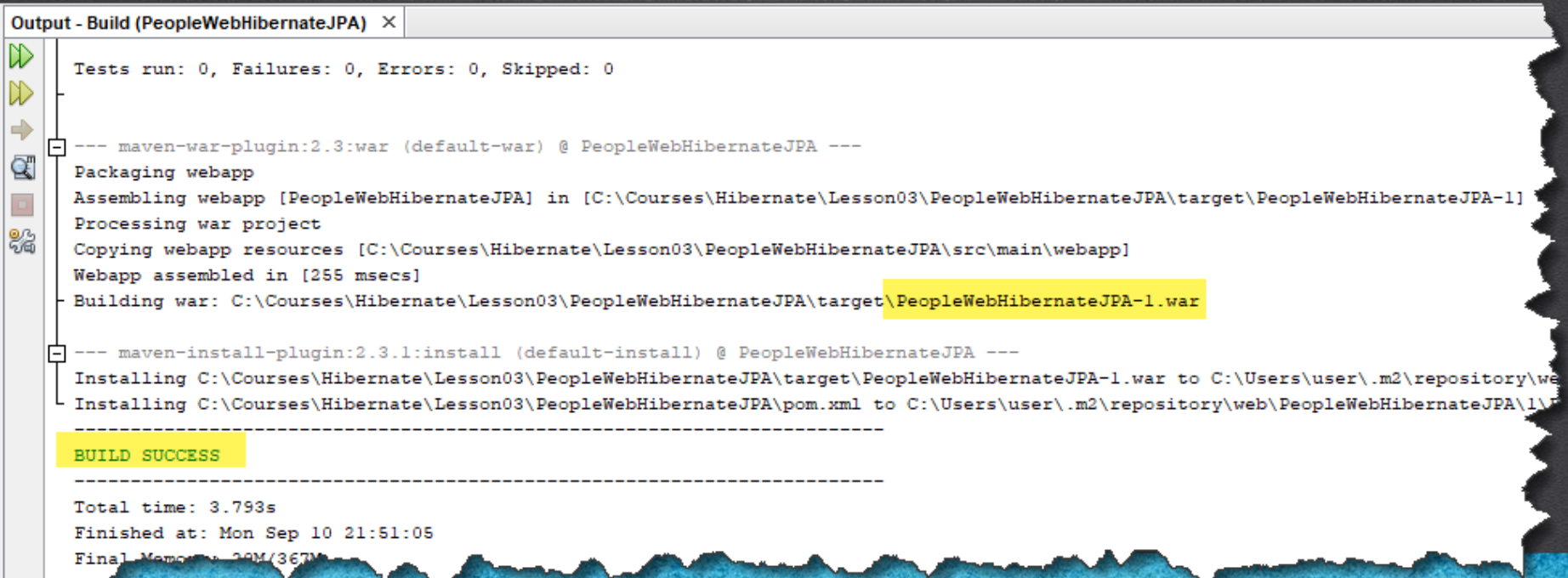
23. EXECUTE CLEAN & BUILD

• We return to the Projects view. In order to download the new libraries, we make Clean & Build the project. If for some reason this process fails, you must disable any software such as antivirus, Windows defender or firewall during this process so that the download of Java .jar files is not prevented. Once finished, these services can be activated again. This process may take several minutes depending on your internet speed:



23. EXECUTE CLEAN & BUILD

- If you no longer had to download any library because you could already have all downloaded, the process is faster. In the end we should observe the following:



The screenshot shows the 'Output - Build (PeopleWebHibernateJPA)' window in an IDE. The output text is as follows:

```
Tests run: 0, Failures: 0, Errors: 0, Skipped: 0

--- maven-war-plugin:2.3:war (default-war) @ PeopleWebHibernateJPA ---
Packaging webapp
Assembling webapp [PeopleWebHibernateJPA] in [C:\Courses\Hibernate\Lesson03\PeopleWebHibernateJPA\target\PeopleWebHibernateJPA-1]
Processing war project
Copying webapp resources [C:\Courses\Hibernate\Lesson03\PeopleWebHibernateJPA\src\main\webapp]
Webapp assembled in [255 msec]
Building war: C:\Courses\Hibernate\Lesson03\PeopleWebHibernateJPA\target\PeopleWebHibernateJPA-1.war

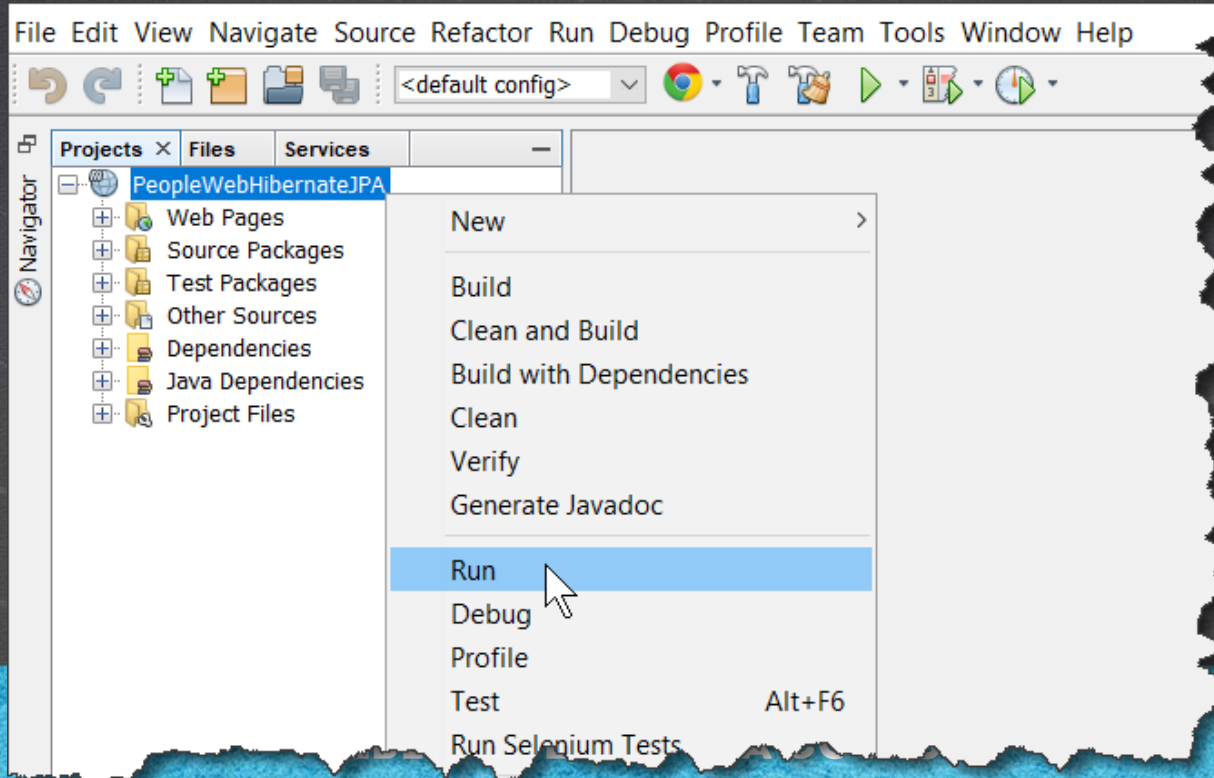
--- maven-install-plugin:2.3.1:install (default-install) @ PeopleWebHibernateJPA ---
Installing C:\Courses\Hibernate\Lesson03\PeopleWebHibernateJPA\target\PeopleWebHibernateJPA-1.war to C:\Users\user\.m2\repository\web\PeopleWebHibernateJPA\1.0-SNAPSHOT
Installing C:\Courses\Hibernate\Lesson03\PeopleWebHibernateJPA\pom.xml to C:\Users\user\.m2\repository\web\PeopleWebHibernateJPA\1.0-SNAPSHOT
-----
BUILD SUCCESS
-----
Total time: 3.793s
Finished at: Mon Sep 10 21:51:05
Final Memory: 29M/367M
```

CURSO STRUTS FRAMEWORK

www.globalmentoring.com.mx

24. EXECUTE THE PROJECT

Execute the project:



24. EXECUTE THE PROJECT

We go to the list of people:



HIBERNATE AND JPA COURSE

www.globalmentoring.com.mx

24. EXECUTE THE PROJECT

We go to the list of people:



HIBERNATE AND JPA COURSE

www.globalmentoring.com.mx

EXERCISE CONCLUSION

- With this exercise we have integrated a Web application with Hibernate and JPA using Glassfish as an application server.
- We apply several design patterns such as MVC, DAO and DTO as a whole to be able to display the list of people, all this with the help of the persistence layer supported by Hibernate and JPA.



HIBERNATE AND JPA COURSE

www.globalmentoring.com.mx

ONLINE COURSE

HIBERNATE & JPA

By: Eng. Ubaldo Acosta



HIBERNATE AND JPA COURSE

www.globalmentoring.com.mx