

**JAVA EE COURSE**

# **EXERCISE**

**HELLO WORLD WITH JPA**

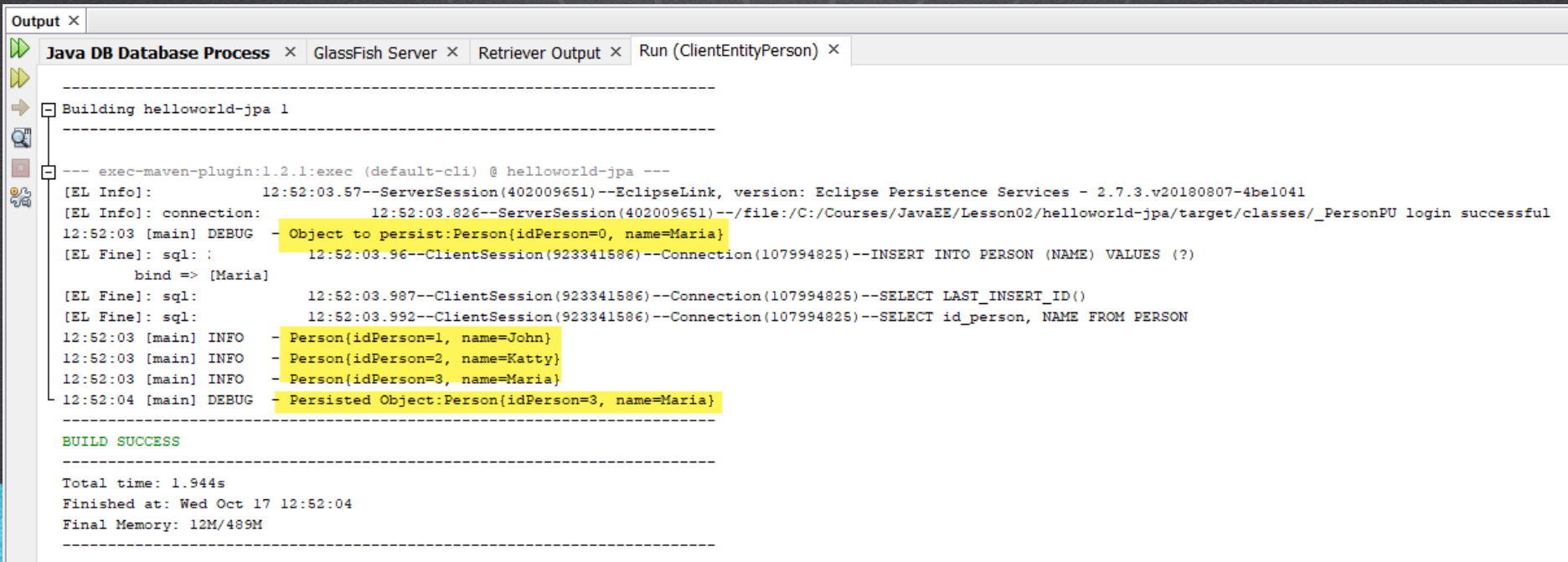


**JAVA EE COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

# EXERCISE OBJECTIVE

- The objective of the exercise is to create a Hello World with JPA. At the end we must observe the following result:



```
Output x
Java DB Database Process x GlassFish Server x Retriever Output x Run (ClientEntityPerson) x

-----
Building helloworld-jpa 1
-----

--- exec-maven-plugin:1.2.1:exec (default-cli) @ helloworld-jpa ---
[EL Info]: 12:52:03.57--ServerSession(402009651)--EclipseLink, version: Eclipse Persistence Services - 2.7.3.v20180807-4be1041
[EL Info]: connection: 12:52:03.826--ServerSession(402009651)--/file:/C:/Courses/JavaEE/Lesson02/helloworld-jpa/target/classes/_PersonPU login successful
12:52:03 [main] DEBUG - Object to persist:Person{idPerson=0, name=Maria}
[EL Fine]: sql: 12:52:03.96--ClientSession(923341586)--Connection(107994825)--INSERT INTO PERSON (NAME) VALUES (?)
      bind => [Maria]
[EL Fine]: sql: 12:52:03.987--ClientSession(923341586)--Connection(107994825)--SELECT LAST_INSERT_ID()
[EL Fine]: sql: 12:52:03.992--ClientSession(923341586)--Connection(107994825)--SELECT id_person, NAME FROM PERSON
12:52:03 [main] INFO - Person{idPerson=1, name=John}
12:52:03 [main] INFO - Person{idPerson=2, name=Katty}
12:52:03 [main] INFO - Person{idPerson=3, name=Maria}
12:52:04 [main] DEBUG - Persisted Object:Person{idPerson=3, name=Maria}

-----
BUILD SUCCESS
-----
Total time: 1.944s
Finished at: Wed Oct 17 12:52:04
Final Memory: 12M/489M
-----
```



# INSTALL MYSQL (OPTIONAL)

Install MySql database in case you don't have it. Omit this step if you already have installed MySql.

<http://icursos.net/en/Installations/CJ-B-Exercise-06-MySqlInstalation.pdf>



**STRUTS AND NEW COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

# CREATE MYSQL DATABASE (OPTIONAL)

Then, we need to create a mysql database. Follow the next guide to create the database, the table and insert some rows (omit this step if you already created this database):

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



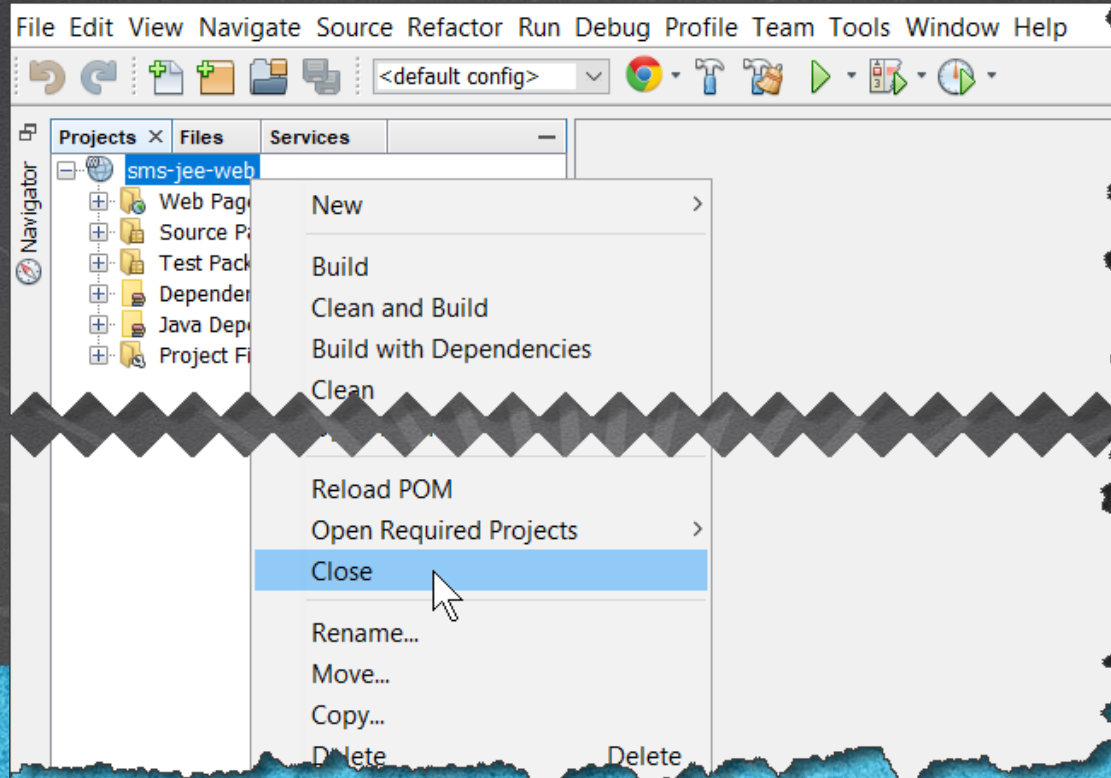
**STRUTSATA NEWORSEOURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)



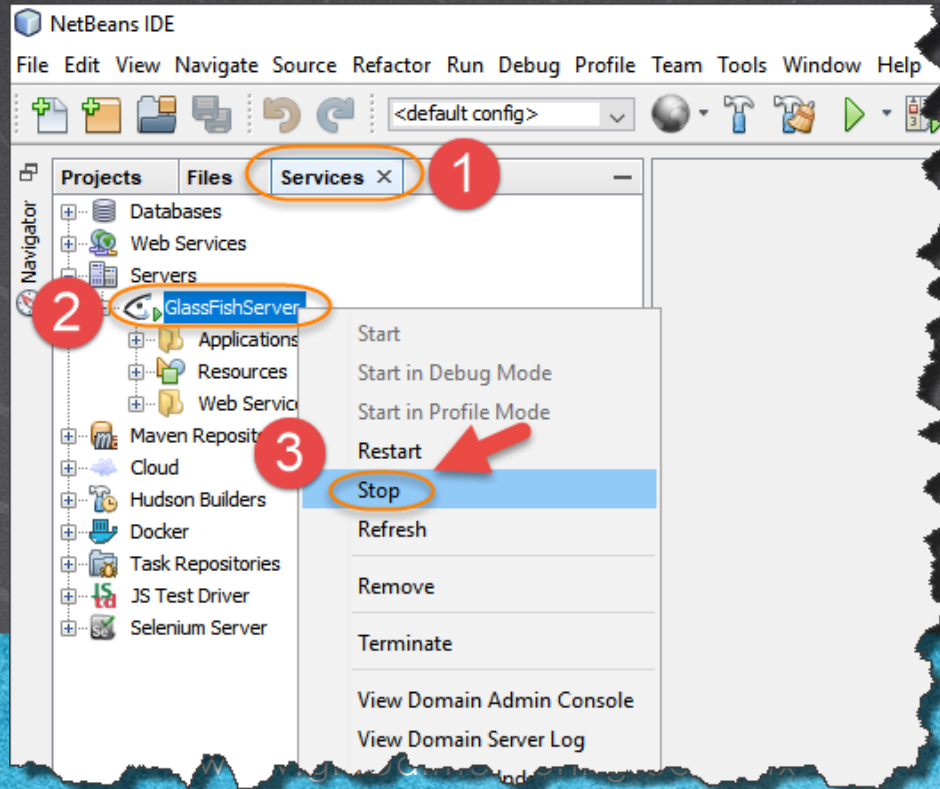
# CLOSE ANY OTHER PROJECT

We close any project that we no longer use:



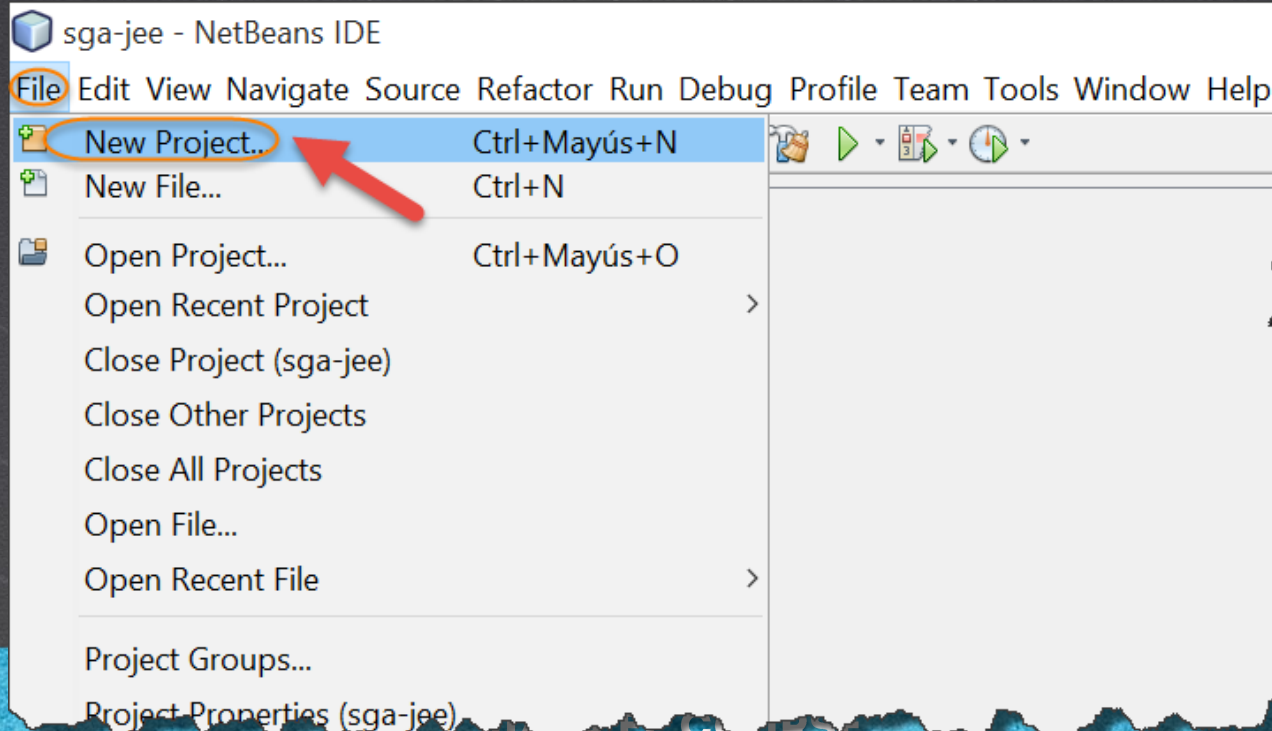
# LET'S STOP GLASSFISH

Stop Glassfish in case it is started :



# 1. CREATE A NEW PROJECT

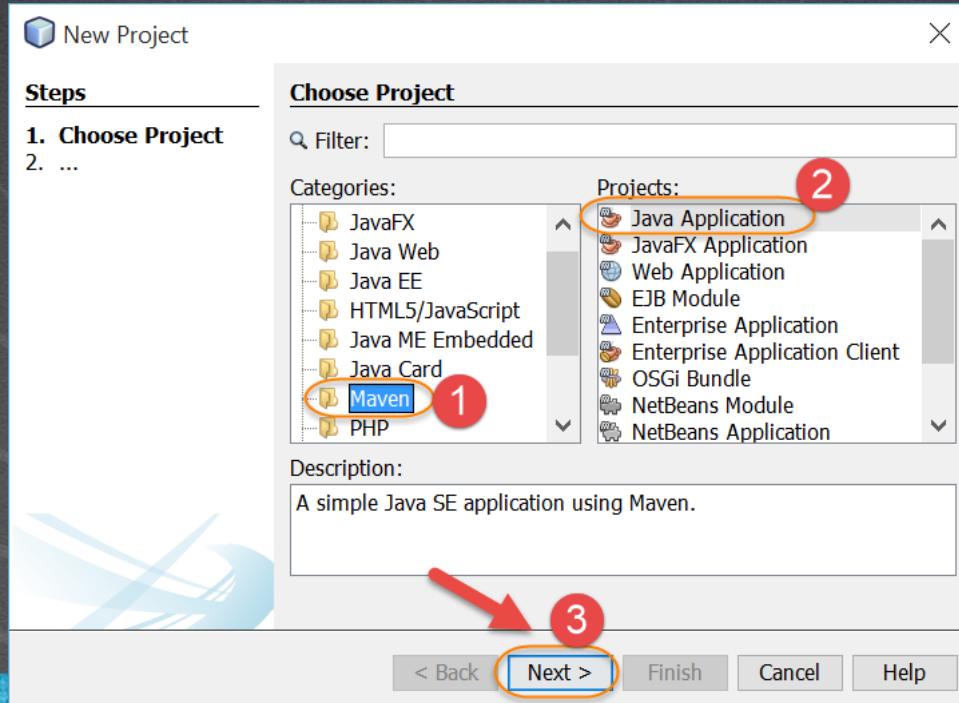
Create a new Project:





# 1. CREATE A NEW PROJECT

Create a new Project:



**JAVA EE COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)



# 1. CREATE A NEW PROJECT

Create a new Project:

**New Java Application**

**Steps**

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

**Name and Location**

Project Name: helloworld-jpa

Project Location: C:\Courses\JavaEE\Lesson02 Browse...

Project Folder: C:\Courses\JavaEE\Lesson02\helloworld-jpa

Artifact Id: helloworld-jpa

Group Id: sms

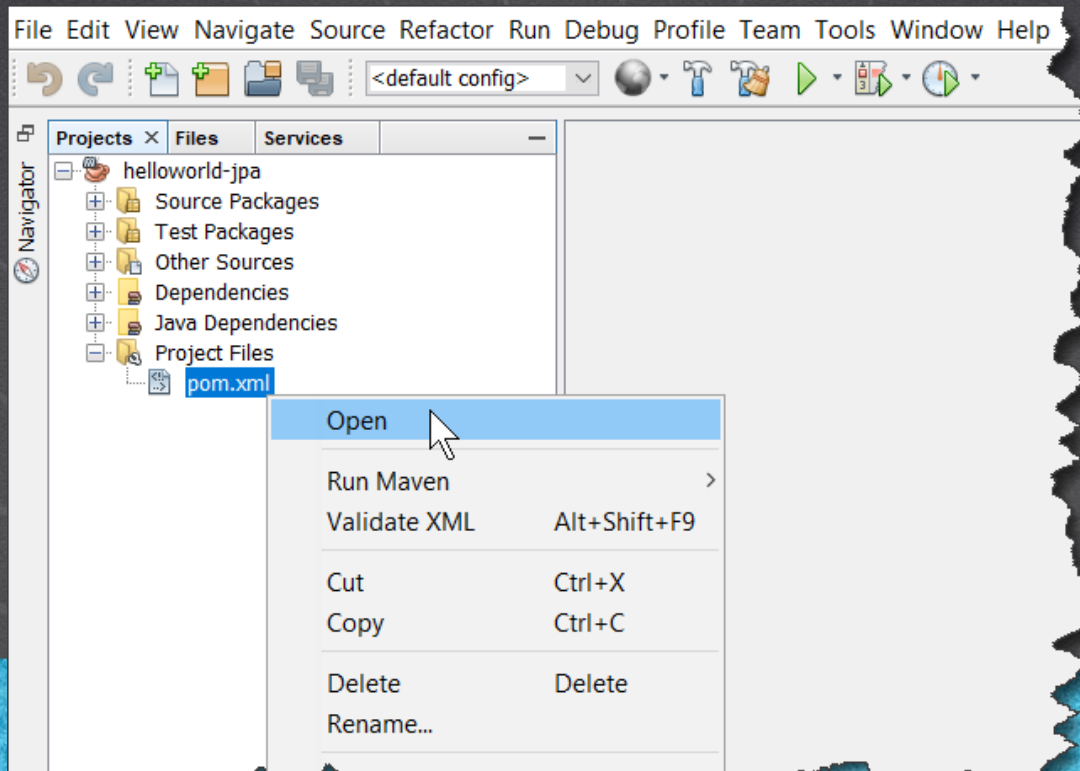
Version: 1

Package: (Optional)

< Back Next > **Finish** Cancel Help

## 2. MODIFY THE POM.XML

Modify the pom.xml file to add the .jar libraries:





## 2. MODIFY THE FILE

### pom.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>sms</groupId>
  <artifactId>helloworld-jpa</artifactId>
  <version>1</version>
  <packaging>jar</packaging>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>
  <dependencies>
    <dependency>
      <groupId>javax</groupId>
      <artifactId>javaee-api</artifactId>
      <version>8.0</version>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>javax.persistence</groupId>
      <artifactId>javax.persistence-api</artifactId>
      <version>2.2</version>
    </dependency>
  </dependencies>
</project>
```

**JAVA EE COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

## 2. MODIFY THE FILE

pom.xml:

```
<dependency>
  <groupId>org.eclipse.persistence</groupId>
  <artifactId>eclipselink</artifactId>
  <version>2.7.3</version>
</dependency>
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>5.1.47</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>
</project>
```

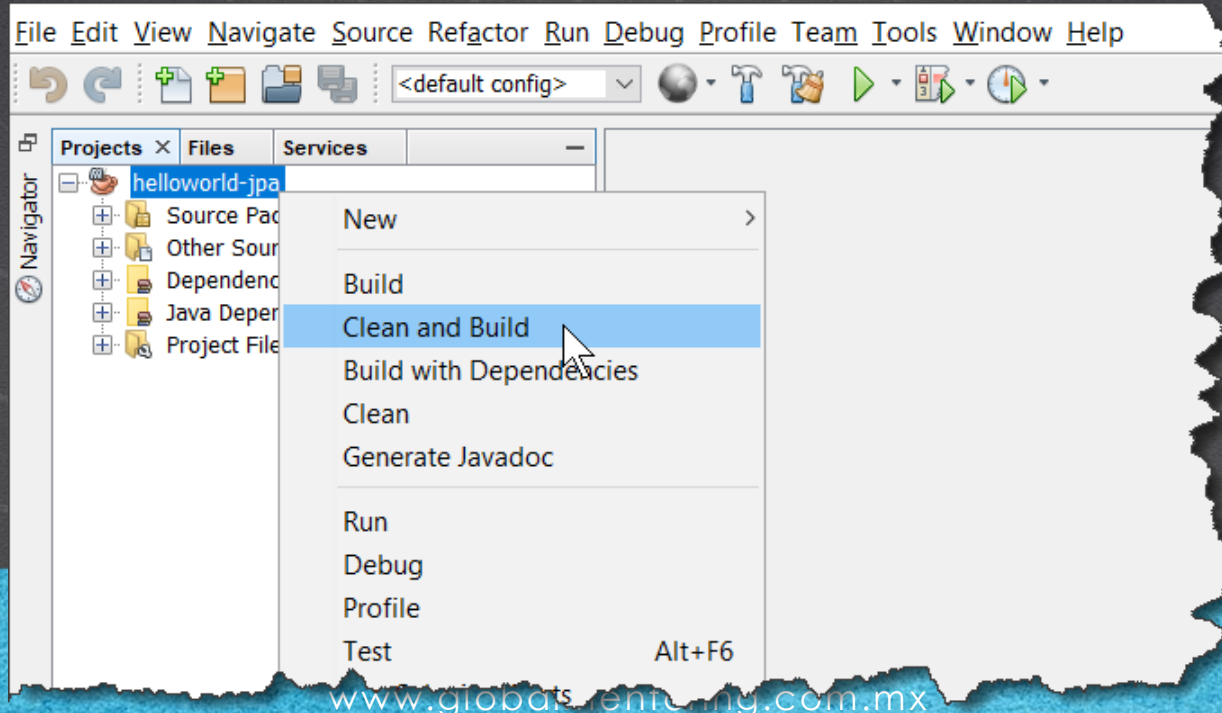
**JAVA EE COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)



### 3. EXECUTE CLEAN & BUILD

We do a clean & build for the new libraries to be downloaded if necessary. We must do this every time we modify the pom.xml file:



### 3. EXECUTE CLEAN & BUILD

We must observe that the libraries are downloaded and that our project is successfully compiled:



```
Output X
Java DB Database Process x GlassFish Server x Retriever Output x Build (helloworld-jpa) x

TESTS
-----
Results :
Tests run: 0, Failures: 0, Errors: 0, Skipped: 0

--- maven-jar-plugin:2.3.2:jar (default-jar) @ helloworld-jpa ---
Building jar: C:\Courses\JavaEE\Lesson02\helloworld-jpa\target\helloworld-jpa-1.jar

--- maven-install-plugin:2.3.1:install (default-install) @ helloworld-jpa ---
Installing C:\Courses\JavaEE\Lesson02\helloworld-jpa\target\helloworld-jpa-1.jar to C:\Users\user\.m2\repository\helloworld-jpa-1.jar
Installing C:\Courses\JavaEE\Lesson02\helloworld-jpa\pom.xml to C:\Users\user\.m2\repository\helloworld-jpa-1.jar

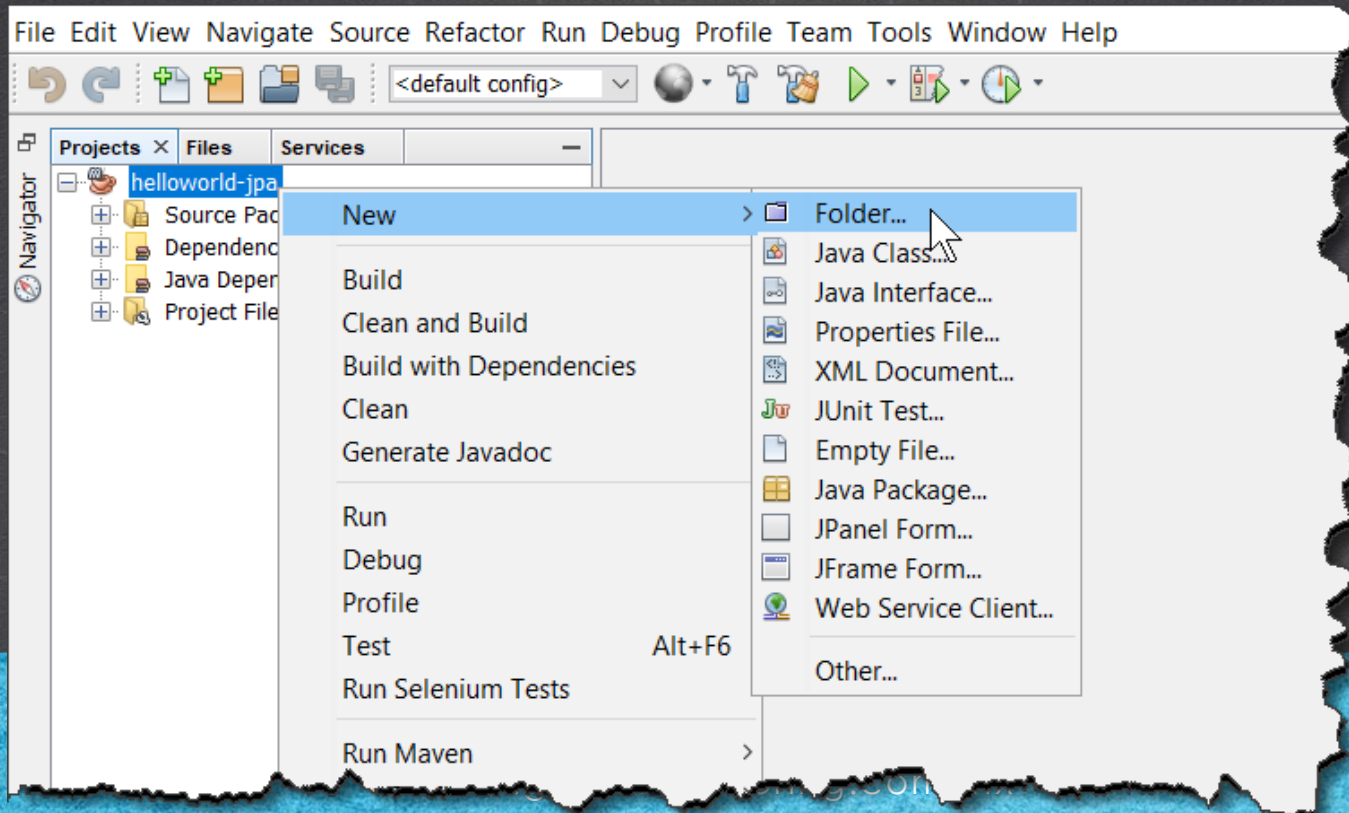
BUILD SUCCESS
-----

Total time: 4.954s
Finished at: Wed Oct 17 12:05:15
Final Memory: 16M/489M
```



## 4. CREATE A NEW FOLDER

We create a folder called: resources



## 4. CREATE A NEW FOLDER

We created a folder called resources. It must be created in the path shown:

**New Folder**

**Steps**

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

**Name and Location**

Folder Name:

Project:

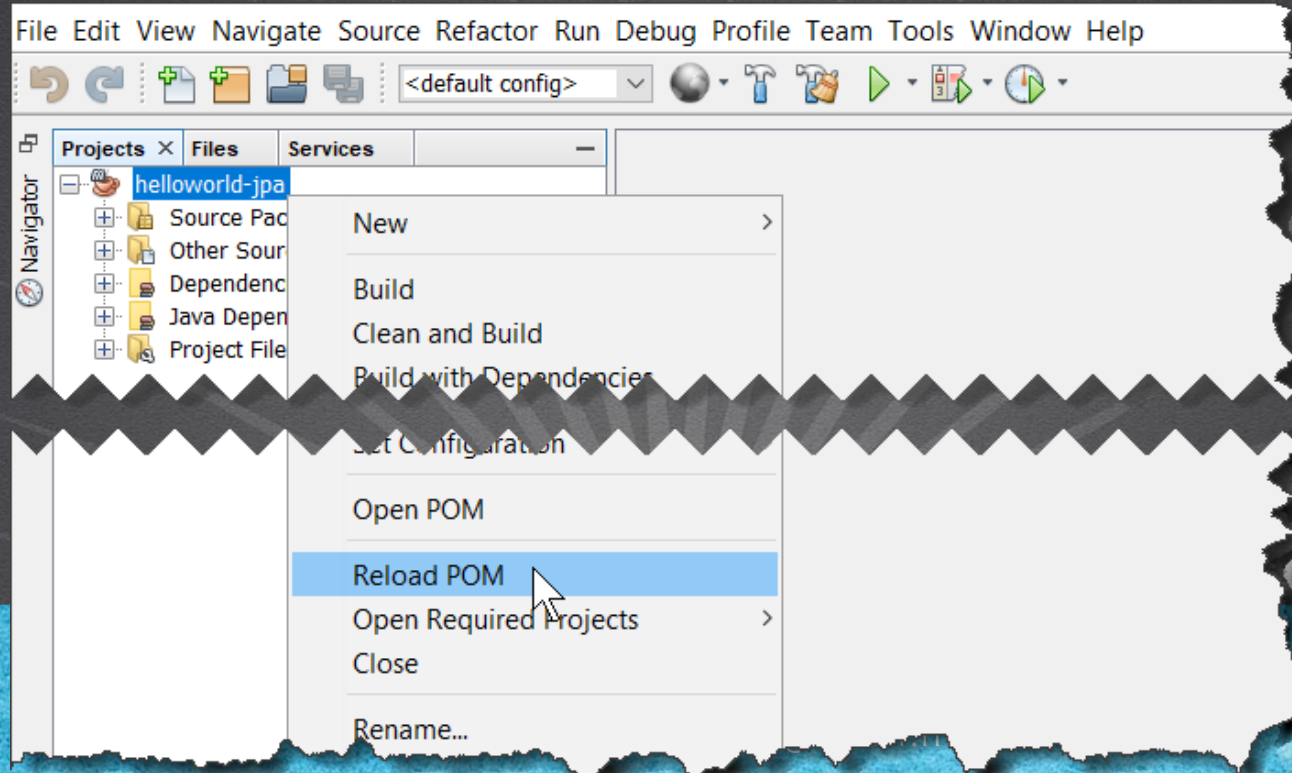
Parent Folder:

Created Folder:

< Back   Next >   **Finish**   Cancel   Help

## 5. RELOAD THE PROJECT

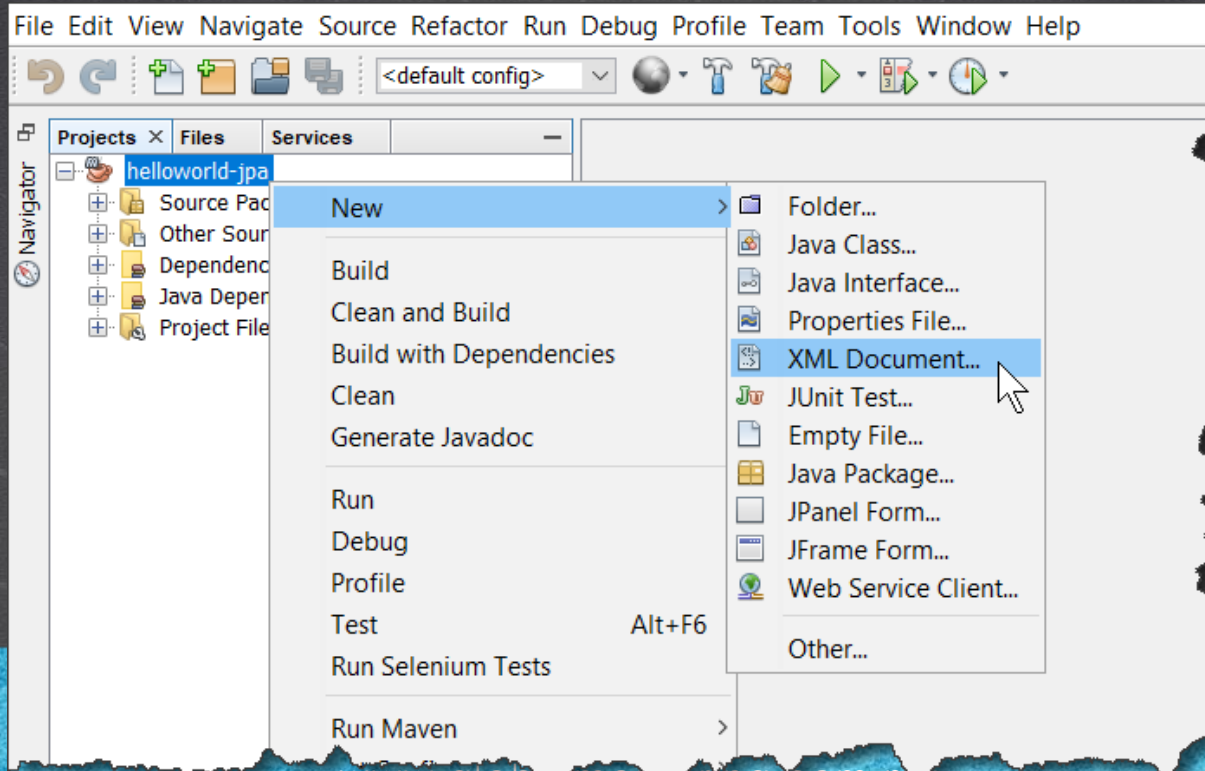
We reload the project to see the new folder in case it is not displayed:





## 6. CREATE AN XML FILE

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



## 6. CREATE AN XML FILE

We 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:

Project:

Folder:

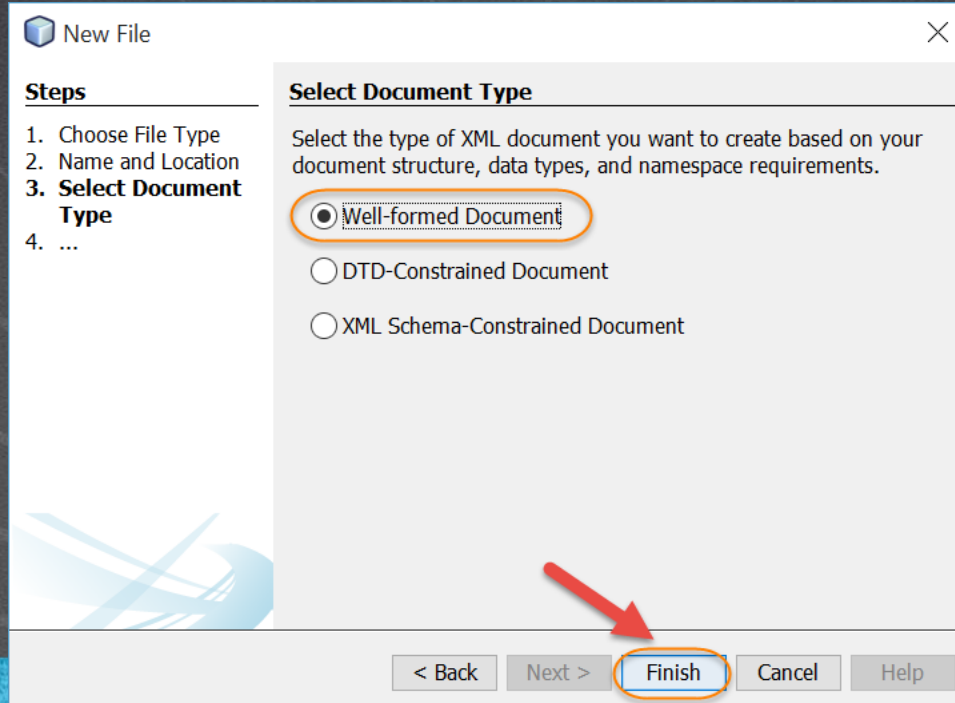
Created File:

**JAVA EE COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

## 6. CREATE AN XML FILE

We create the log4j2.xml file:



**JAVA EE COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)



# 7. MODIFY THE FILE

## log4j2.xml:

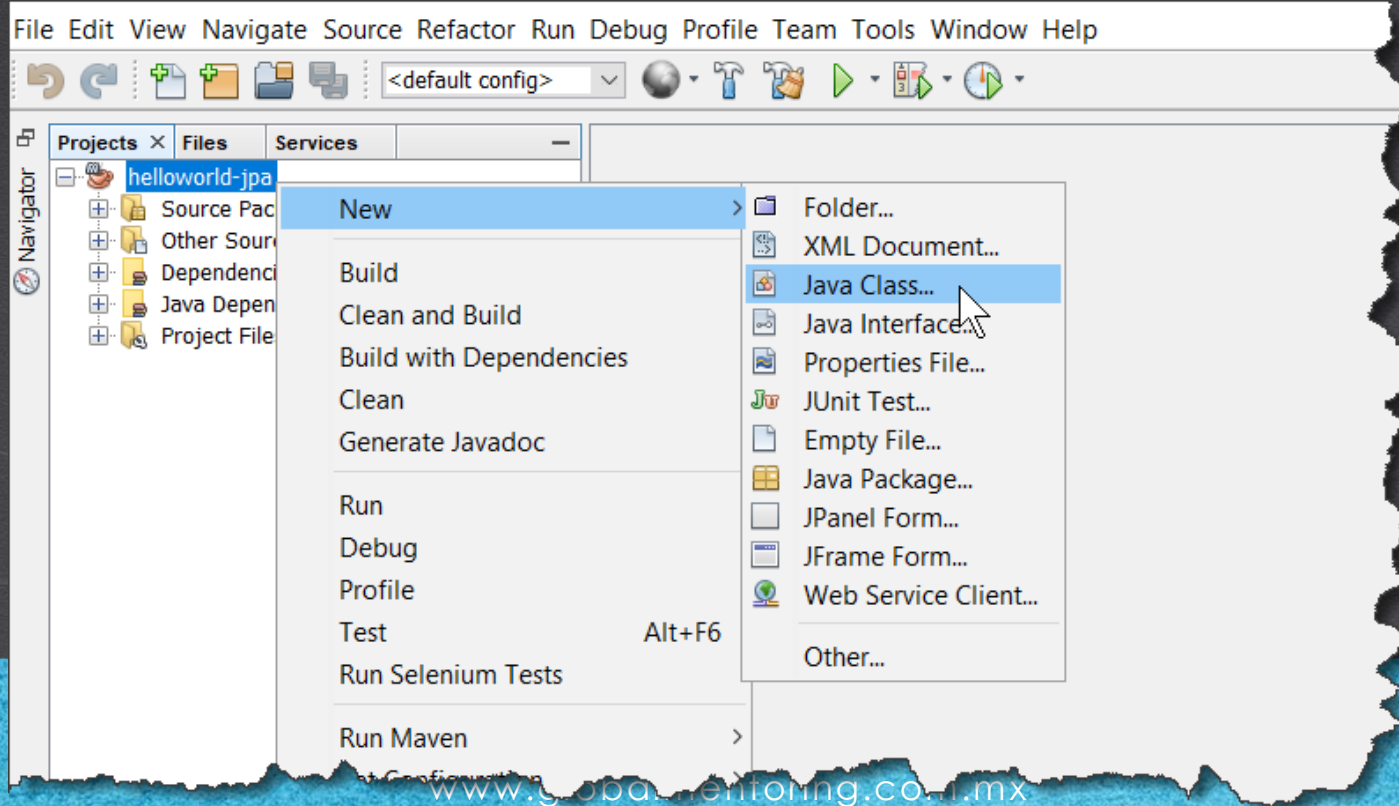
```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="INFO">
  <Appenders>
    <Console name="Console" target="SYSTEM_OUT">
      <PatternLayout pattern="%d{HH:mm:ss} [%t] %-5level %logger{36} - %msg%n" />
    </Console>
  </Appenders>
  <Loggers>
    <Root level="debug">
      <AppenderRef ref="Console" />
    </Root>
  </Loggers>
</Configuration>
```

**JAVA EE COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

## 8. CREATE A JAVA CLASS

We create the Person.java class using JPA annotations:



## 8. CREATE A NEW CLASS

We 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



# 9. MODIFY THE FILE

## Persona.java:

```
package sms.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 int idPerson;

    private String name;

    public Person() {
    }

    public int getIdPerson() {
        return this.idPerson;
    }
}
```

# 9. MODIFY THE FILE

## Persona.java:

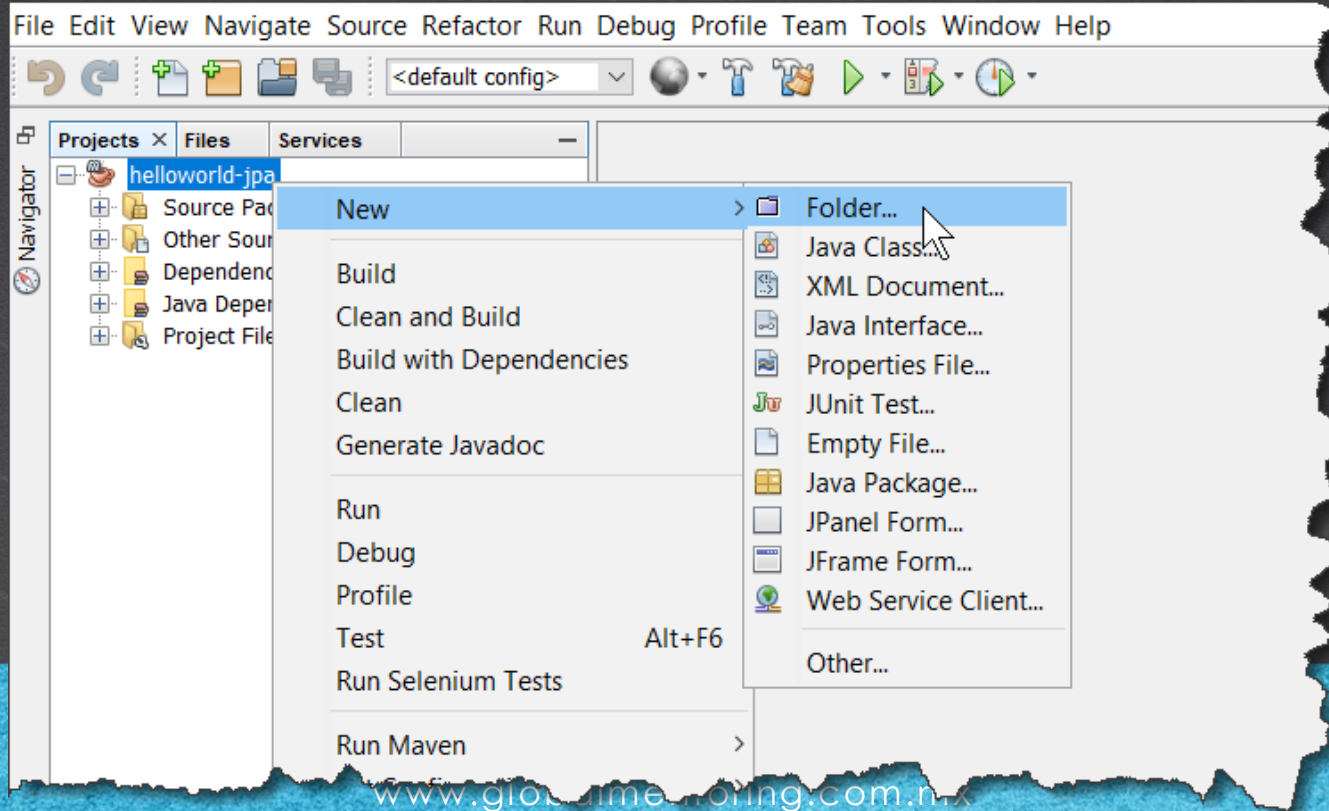
```
public void setIdPerson(int idPerson) {  
    this.idPerson = idPerson;  
}  
  
public String getName() {  
    return this.name;  
}  
  
public void setName(String name) {  
    this.name = name;  
}  
  
@Override  
public String toString() {  
    return "Person{" + "idPerson=" + idPerson +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

**JAVA EE COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

# 10. CREATE A FOLDER

We create the META-INF folder within the src of the project:





# 10. CREATE A FOLDER

We create the META-INF folder within the src of the project:

**New Folder**

**Steps**

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

**Name and Location**

Folder Name:

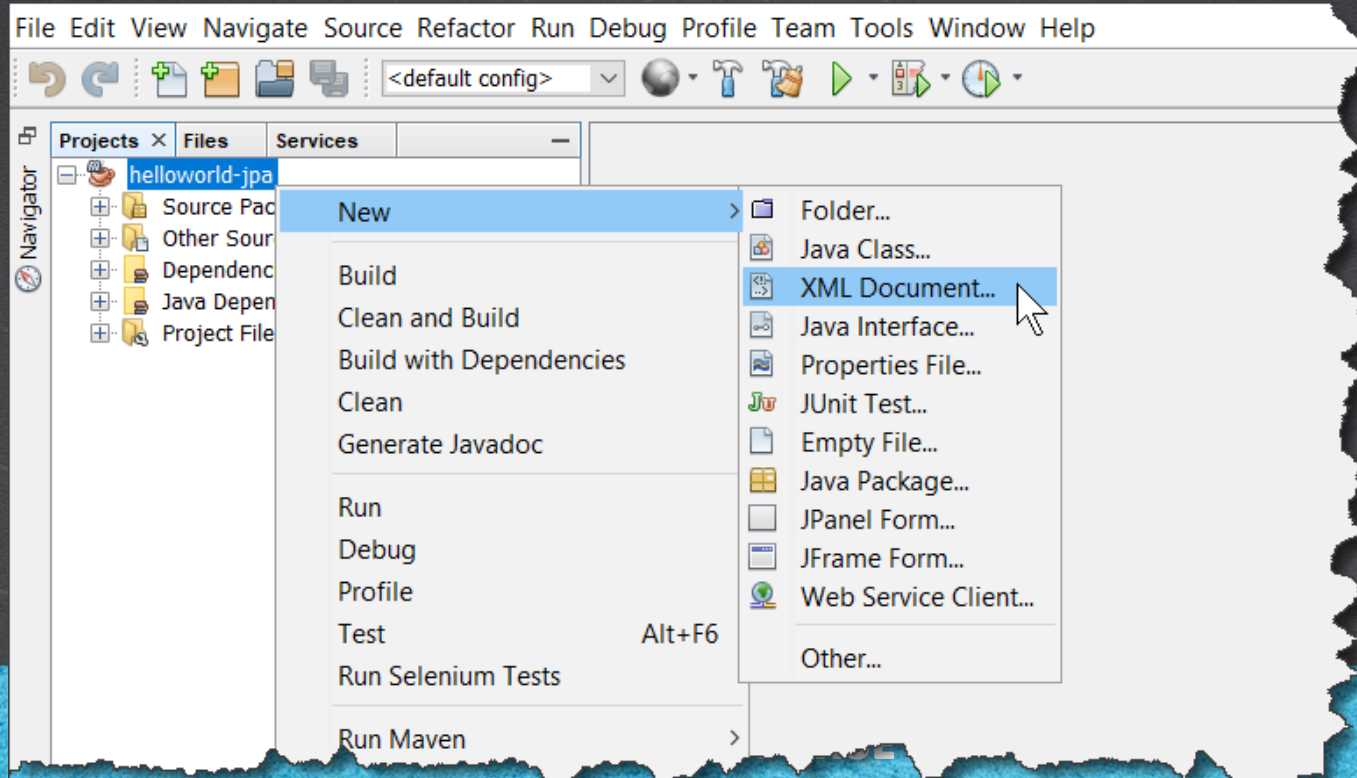
Project:

Parent Folder:

Created Folder:

# 11. CREATE AN XML FILE

We create the persistence.xml file:



# 11. 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:

Project:

Folder:

Created File:

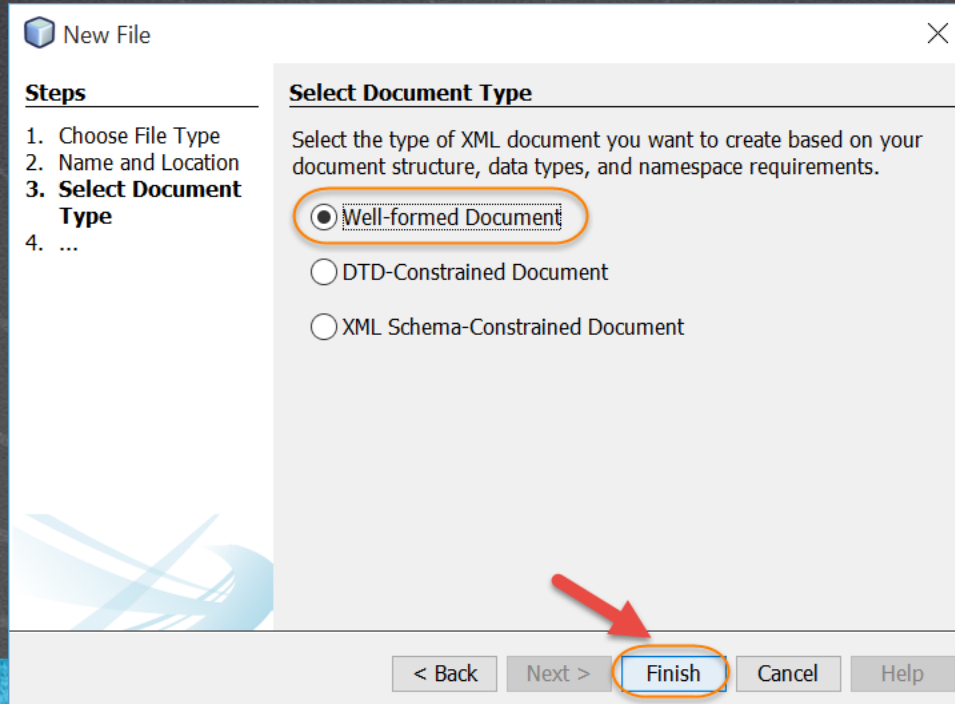
**JAVA EE COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)



# 11. CREATE AN XML FILE

We create the persistence.xml file:



**JAVA EE COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

# 12. MODIFY THE FILE

## persistence.xml:

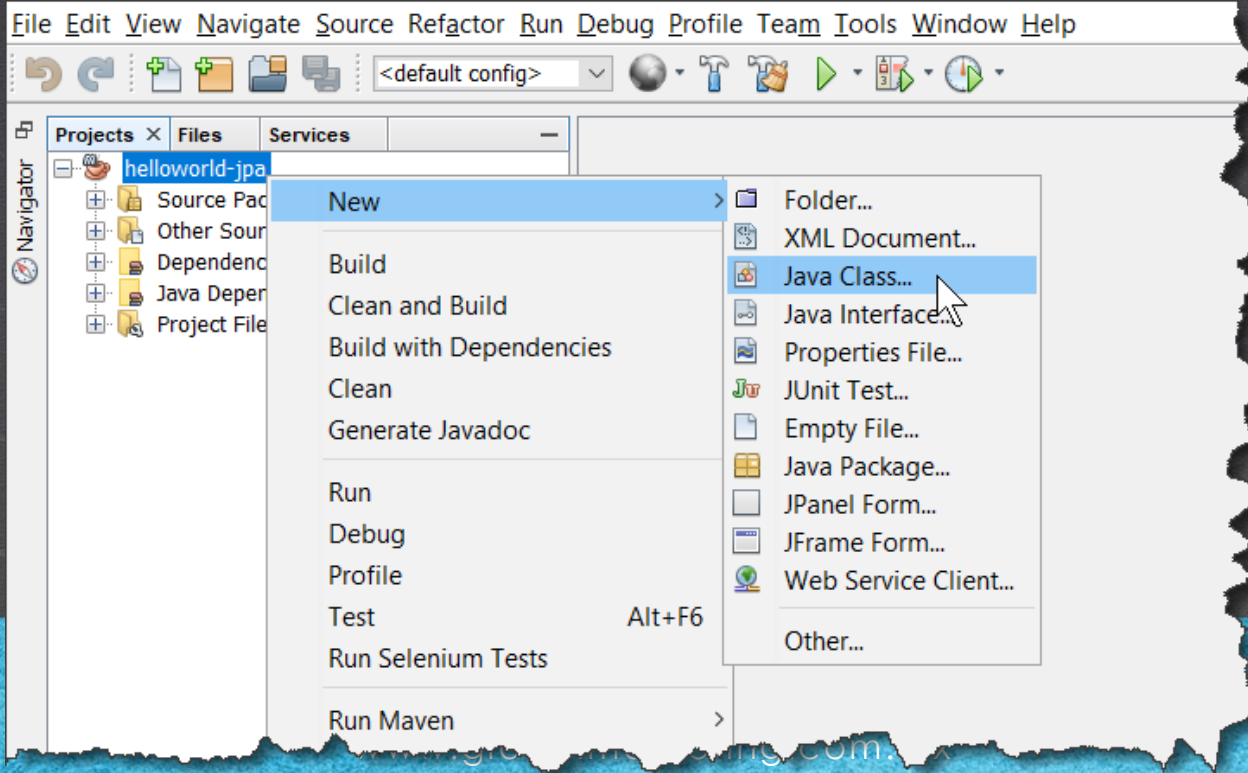
```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence\_2\_2.xsd" version="2.2">
  <persistence-unit name="PersonPU" transaction-type="RESOURCE_LOCAL">
    <class>sms.domain.Person</class>
    <properties>
      <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/test?useSSL=false"/>
      <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="eclipselink.logging.level.sql" value="FINE"/>
      <property name="eclipselink.logging.parameters" value="true"/>
    </properties>
  </persistence-unit>
</persistence>
```

**JAVA EE COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

# 13. CREATE A JAVA CLASS

We are going to create a Java class with a main method and thus create the respective test. We create the class ClientEntityPerson.java:





# 13. CREATE A JAVA CLASS

We create the class ClientEntityPerson.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

# 14. MODIFY THE CODE

## ClientEntityPerson.java:

```
package beans.cliente;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;
import beans.dominio.Persona;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class ClienteEntidadPersona {

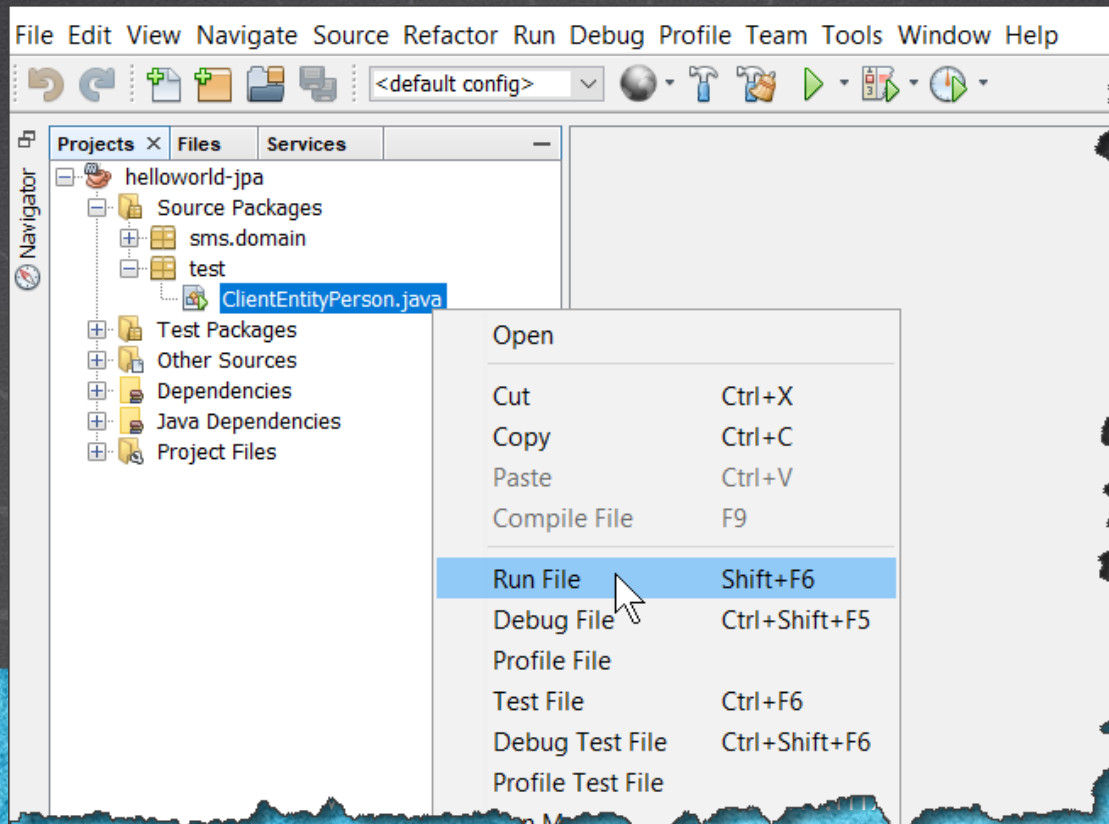
    static Logger log = LogManager.getRootLogger();

    public static void main(String[] args) {

        EntityManagerFactory emf = Persistence.createEntityManagerFactory("PersonaPU");
        EntityManager em = emf.createEntityManager();
        EntityTransaction tx = em.getTransaction();
        //Inicia la transaccion
        tx.begin();
        //No se debe especificar el ID, ya que se genera en automático
        Persona personal = new Persona("Maria", "Gutierrez", "Esparza", "maria@mail.com.mx", "11113333");
        log.debug("Objeto a persistir:" + personal);
        //Persistimos el objeto
        em.persist(personal);
        //Terminamos la transaccion
        tx.commit();
        log.debug("Objeto persistido:" + personal);
        em.close();
    }
}
```

# 15. EXECUTE THE PROJECT

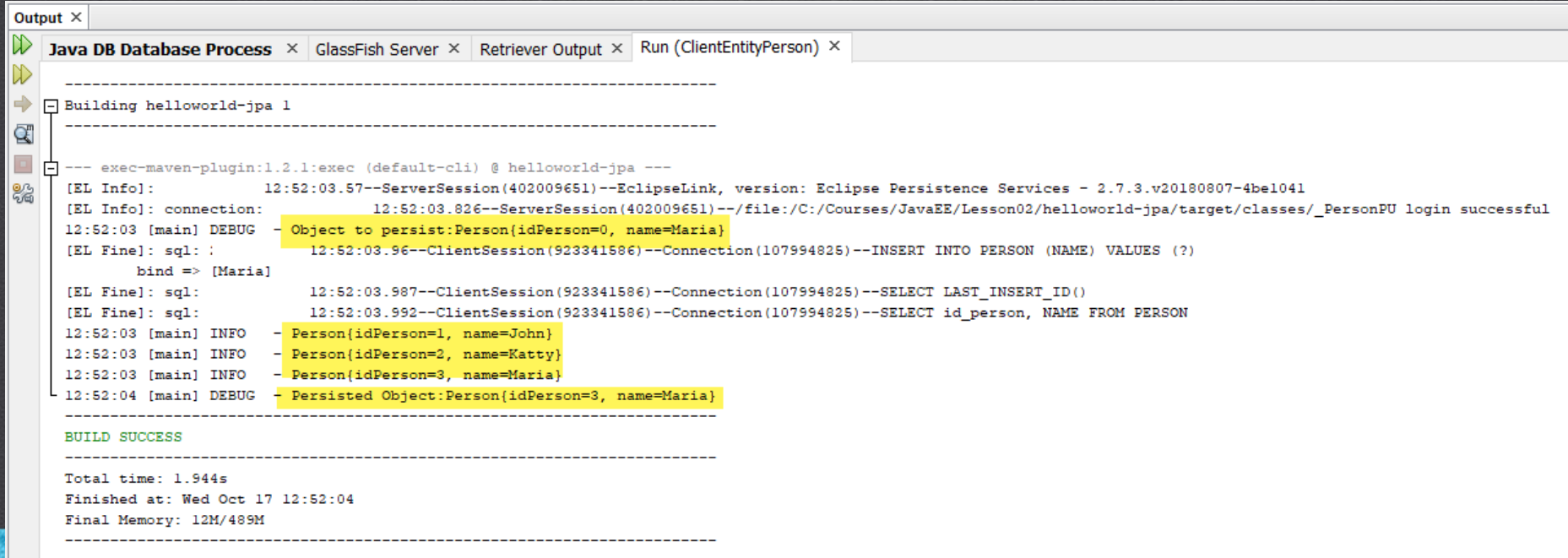
We execute the test class:





# 15. EXECUTE THE PROJECT

We execute the test class:



```
Output x
Java DB Database Process x GlassFish Server x Retriever Output x Run (ClientEntityPerson) x

-----
Building helloworld-jpa 1
-----

--- exec-maven-plugin:1.2.1:exec (default-cli) @ helloworld-jpa ---
[EL Info]:      12:52:03.57--ServerSession(402009651)--EclipseLink, version: Eclipse Persistence Services - 2.7.3.v20180807-4bel041
[EL Info]: connection:      12:52:03.826--ServerSession(402009651)--/file:/C:/Courses/JavaEE/Lesson02/helloworld-jpa/target/classes/_PersonPU login successful
12:52:03 [main] DEBUG - Object to persist:Person{idPerson=0, name=Maria}
[EL Fine]: sql: :      12:52:03.96--ClientSession(923341586)--Connection(107994825)--INSERT INTO PERSON (NAME) VALUES (?)
      bind => [Maria]
[EL Fine]: sql:      12:52:03.987--ClientSession(923341586)--Connection(107994825)--SELECT LAST_INSERT_ID()
[EL Fine]: sql:      12:52:03.992--ClientSession(923341586)--Connection(107994825)--SELECT id_person, NAME FROM PERSON
12:52:03 [main] INFO - Person{idPerson=1, name=John}
12:52:03 [main] INFO - Person{idPerson=2, name=Katty}
12:52:03 [main] INFO - Person{idPerson=3, name=Maria}
12:52:04 [main] DEBUG - Persisted Object:Person{idPerson=3, name=Maria}

-----
BUILD SUCCESS

-----
Total time: 1.944s
Finished at: Wed Oct 17 12:52:04
Final Memory: 12M/489M
-----
```

# OBSERVATIONS IN CASE OF PROBLEMS

If for some reason the execution of the project does not work, we recommend you perform the following actions:

- 1) If there are problems in the execution of the project, it is necessary to execute a Clean & Build to obtain the latest version of each file.
- 2) It is necessary to check that the database is working so that this project can be executed correctly, and each value of the connection to the database must be correct.
- 3) If none of this works, try loading the resolved project, which is 100% functional.



# EXERCISE CONCLUSION

- With this exercise we have created our HelloWorld with JPA project.
- We add the persistence unit by configuring it in the persistence.xml file
- We also added the logging management by configuring it in the log4j2.xml file
- We add an entity class Person.java using JPA annotations
- Finally, we created a ClientEntityPerson.java client, which we executed to check the operation of the persistence unit from a standard Java class and do a query to select all entities from Person table using JPA.

**JAVA EE COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)



**ONLINE COURSE**

# **JAVA EE JAKARTA EE**

By: Ing. Ubaldo Acosta



**JAVA EE COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)