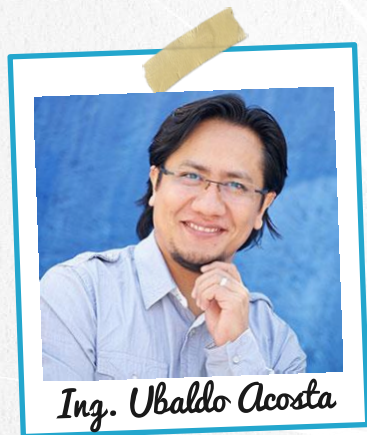


CURSO HIBERNATE Y JPA

HOLA MUNDO CON HIBERNATE/JPA Y ECLIPSE



Por el experto: Ing. Ubaldo Acosta



Experiencia y Conocimiento para tu vida

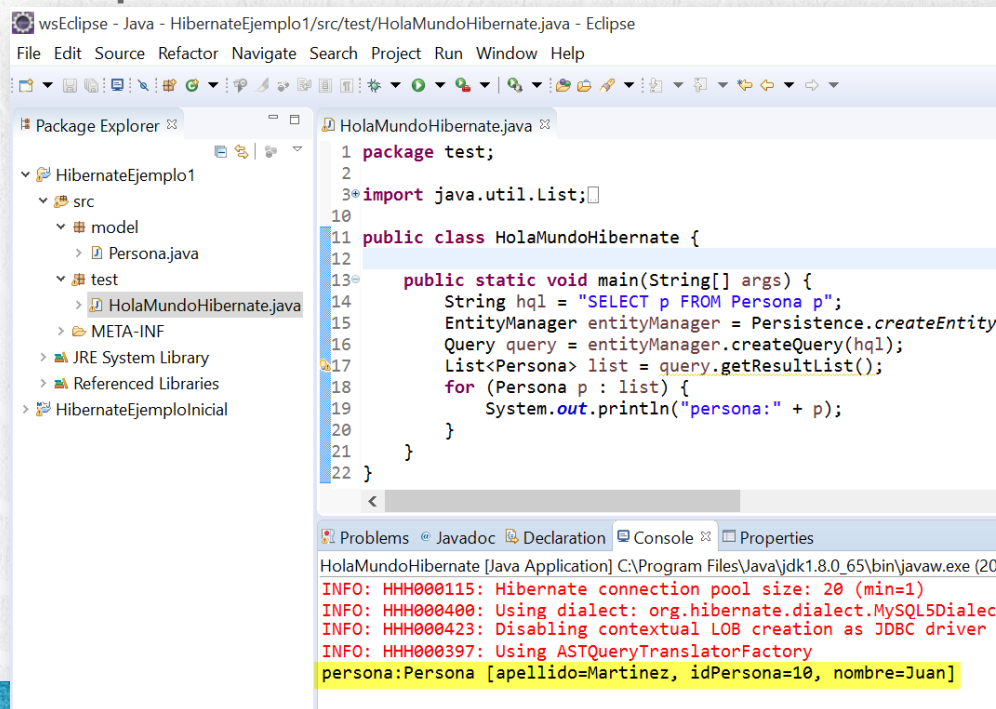


CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx

OBJETIVO DEL EJERCICIO

Crear una aplicación para hacer un HolaMundo con Hibernate/JPA.



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left shows the project structure: HibernateEjemplo1 > src > model > Persona.java and test > HolaMundoHibernate.java. The main editor displays the code for HolaMundoHibernate.java, which is a test class that uses JPA to query a database. The console at the bottom shows the output of the application, including log messages from Hibernate and the final output line: persona:Persona [apellido=Martinez, idPersona=10, nombre=Juan].

```
wsEclipse - Java - HibernateEjemplo1/src/test/HolaMundoHibernate.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer
  HibernateEjemplo1
    src
      model
        Persona.java
      test
        HolaMundoHibernate.java
        META-INF
      JRE System Library
      Referenced Libraries
      HibernateEjemplo1Initial

HolaMundoHibernate.java
1 package test;
2
3 import java.util.List;
10
11 public class HolaMundoHibernate {
12
13     public static void main(String[] args) {
14         String hql = "SELECT p FROM Persona p";
15         EntityManager entityManager = Persistence.createEntityManagerFactory("hibernate").createEntityManager();
16         Query query = entityManager.createQuery(hql);
17         List<Persona> list = query.getResultList();
18         for (Persona p : list) {
19             System.out.println("persona: " + p);
20         }
21     }
22 }

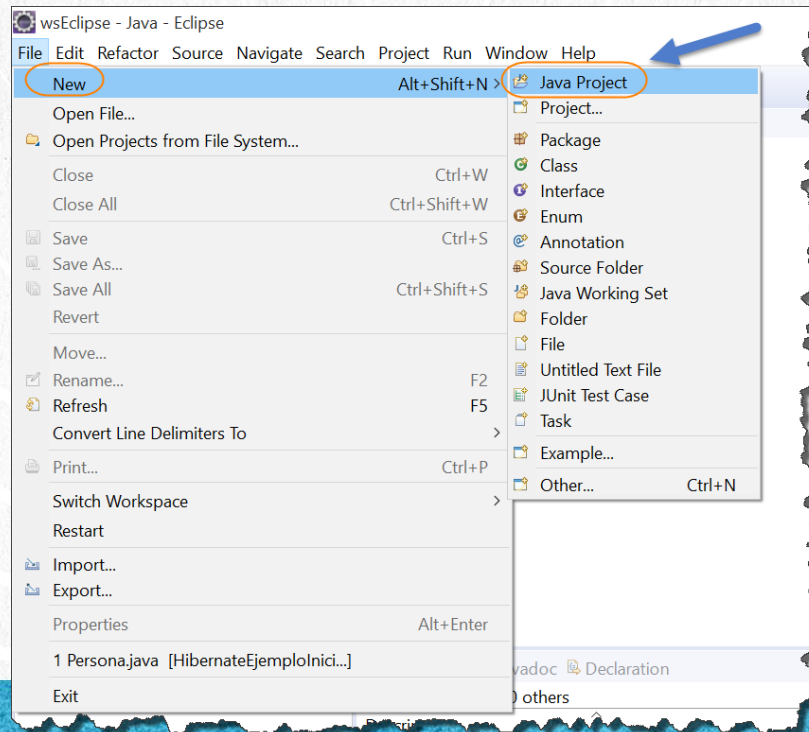
Problems Javadoc Declaration Console Properties
HolaMundoHibernate [Java Application] C:\Program Files\Java\jdk1.8.0_65\bin\javaw.exe (20)
INFO: HHH000115: Hibernate connection pool size: 20 (min=1)
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQL5Dialect
INFO: HHH000423: Disabling contextual LOB creation as JDBC driver
INFO: HHH000397: Using ASTQueryTranslatorFactory
persona:Persona [apellido=Martinez, idPersona=10, nombre=Juan]
```

CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx

1. CREAR PROYECTO

- Creamos un nuevo proyecto Java:

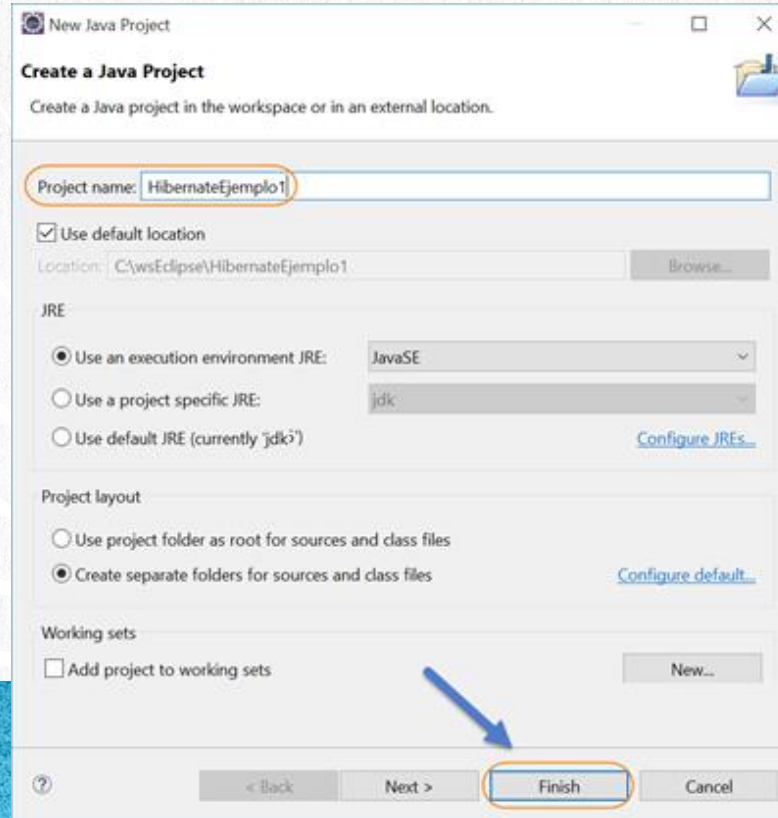


CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx

1. CREAR PROYECTO

- Creamos un nuevo proyecto HibernateEjemplo1:



2. DESCARGA LIBRERÍAS HIBERNATE

Descargamos las librerías necesarias para trabajar con Hibernate/JPA:

<http://icursos.net/cursos/Hibernate/libs/hibernate-libs-1.zip>

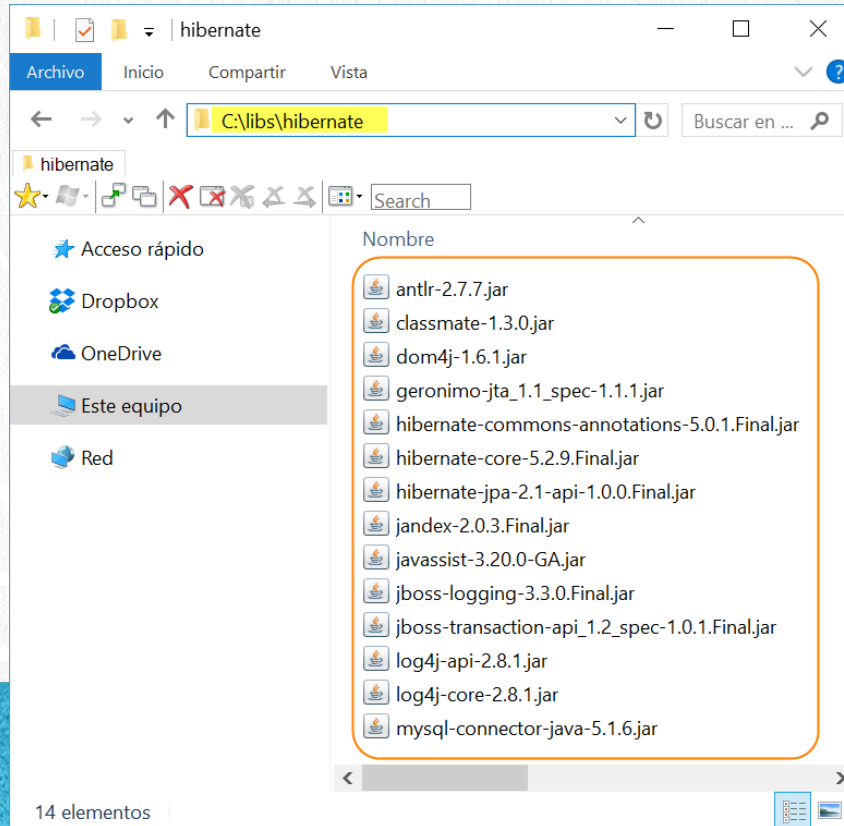
Descomprimir el archivo y guardar todos los archivos .jars en alguna ruta de nuestro disco duro, por ejemplo:

C:\libs\hibernate

NOTA: La versión de las librerías puede ser más reciente, así que se puede usar la última versión proporcionada, se estará actualizando este archivo de librerías, para que sigan siendo compatibles entre ellas y usen la versión más reciente.

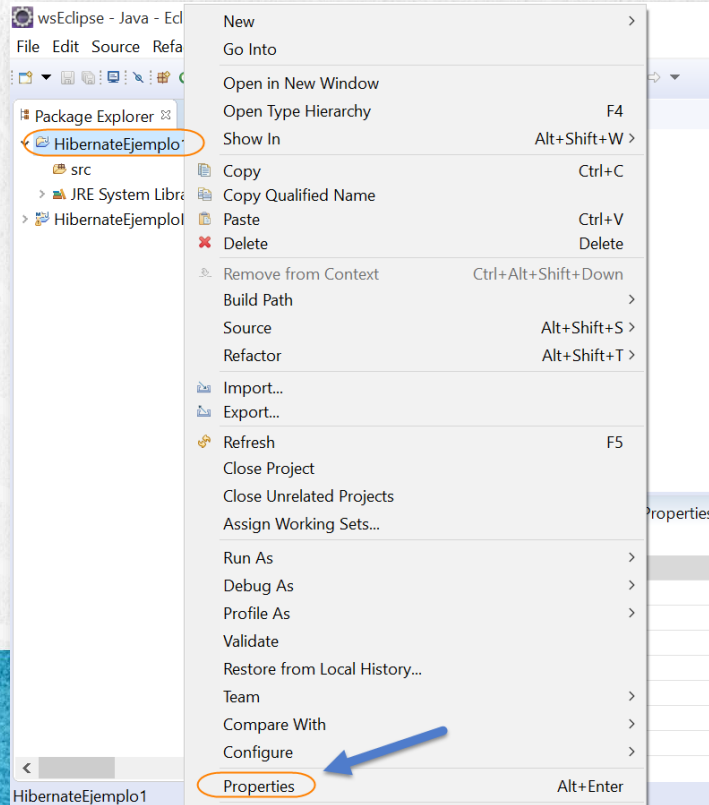
2. DESCARGA LIBRERÍAS HIBERNATE

Así quedan las librerías descomprimidas:



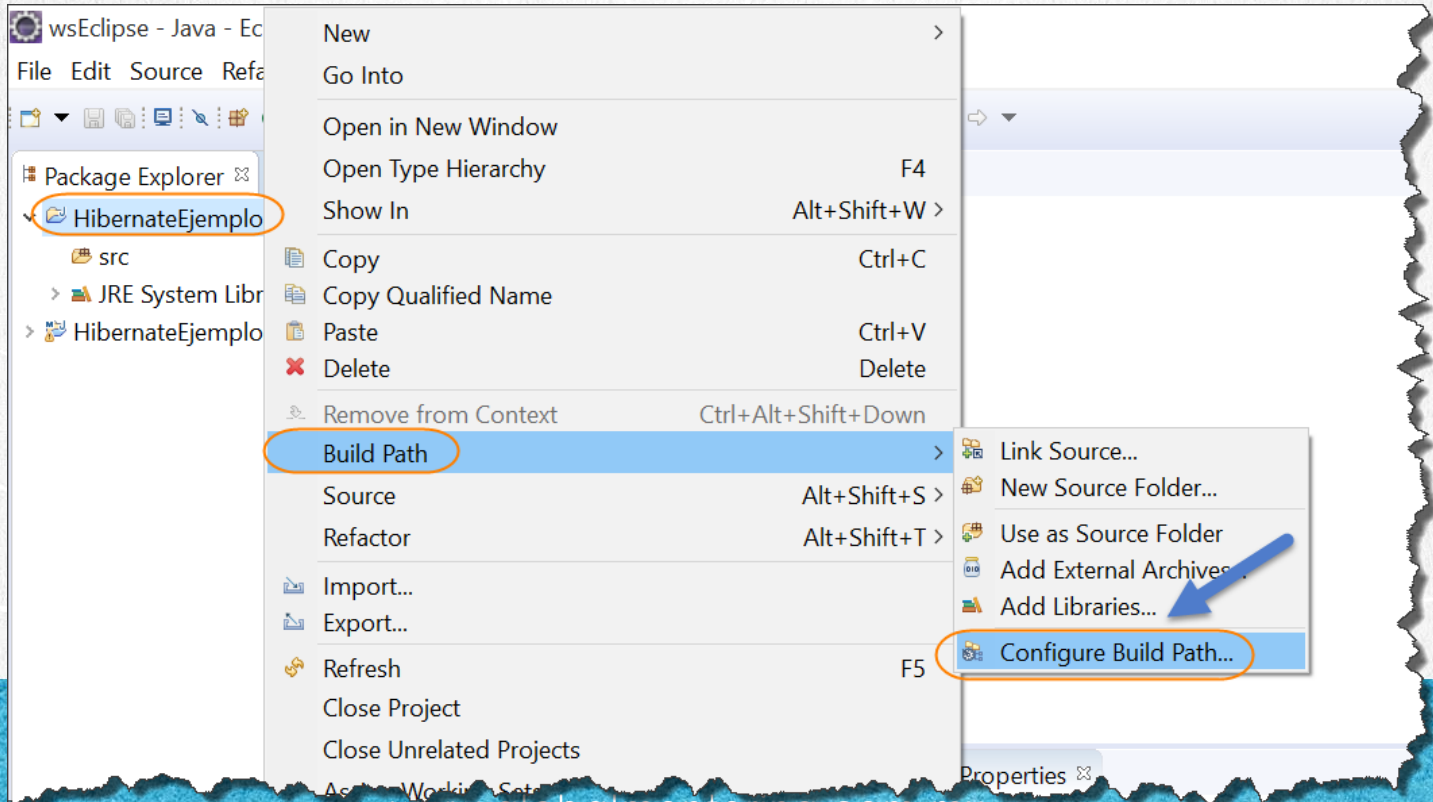
3. AGREGAR LAS LIBRERÍAS

- Agregamos las librerías necesarias para trabajar con Hibernate y MySQL:



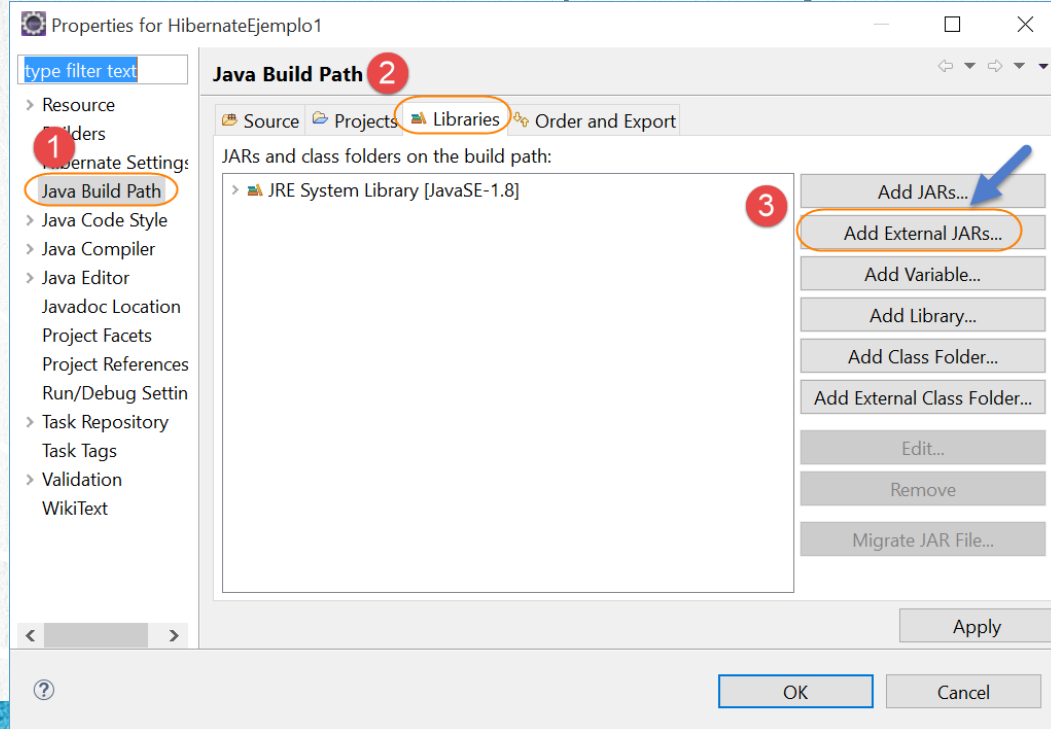
3. AGREGAR LAS LIBRERÍAS

- Agregamos las librerías necesarias para trabajar con Hibernate:



3. AGREGAR LAS LIBRERÍAS

- Agregamos las librerías necesarias para trabajar con Hibernate:

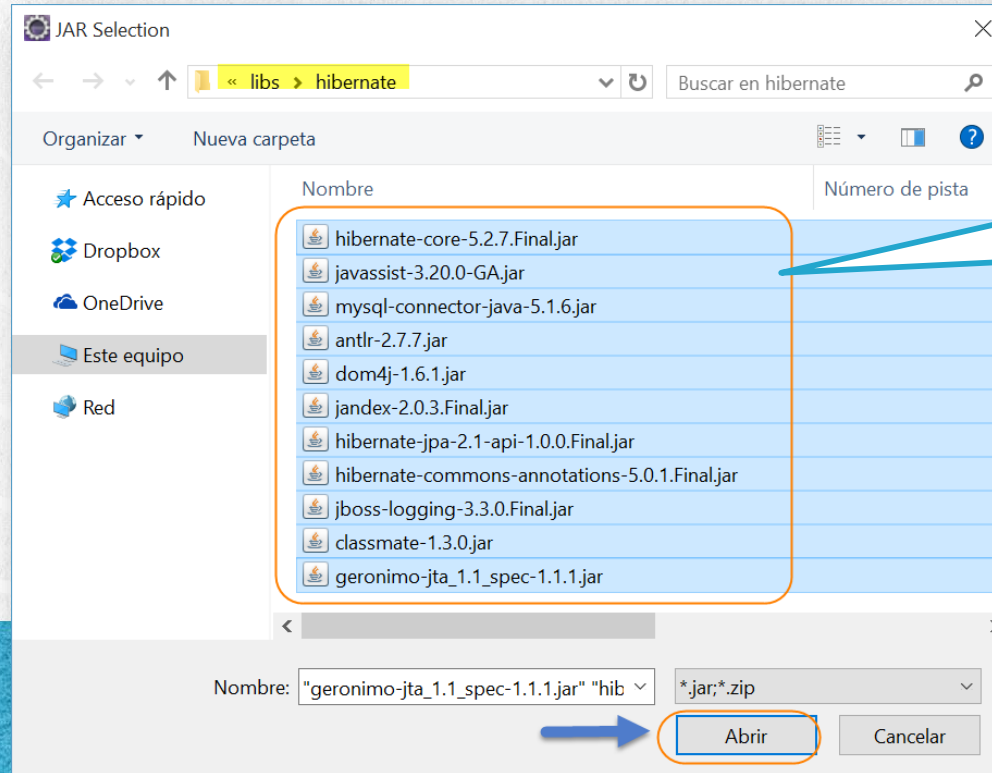


CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx

3. AGREGAR LAS LIBRERÍAS

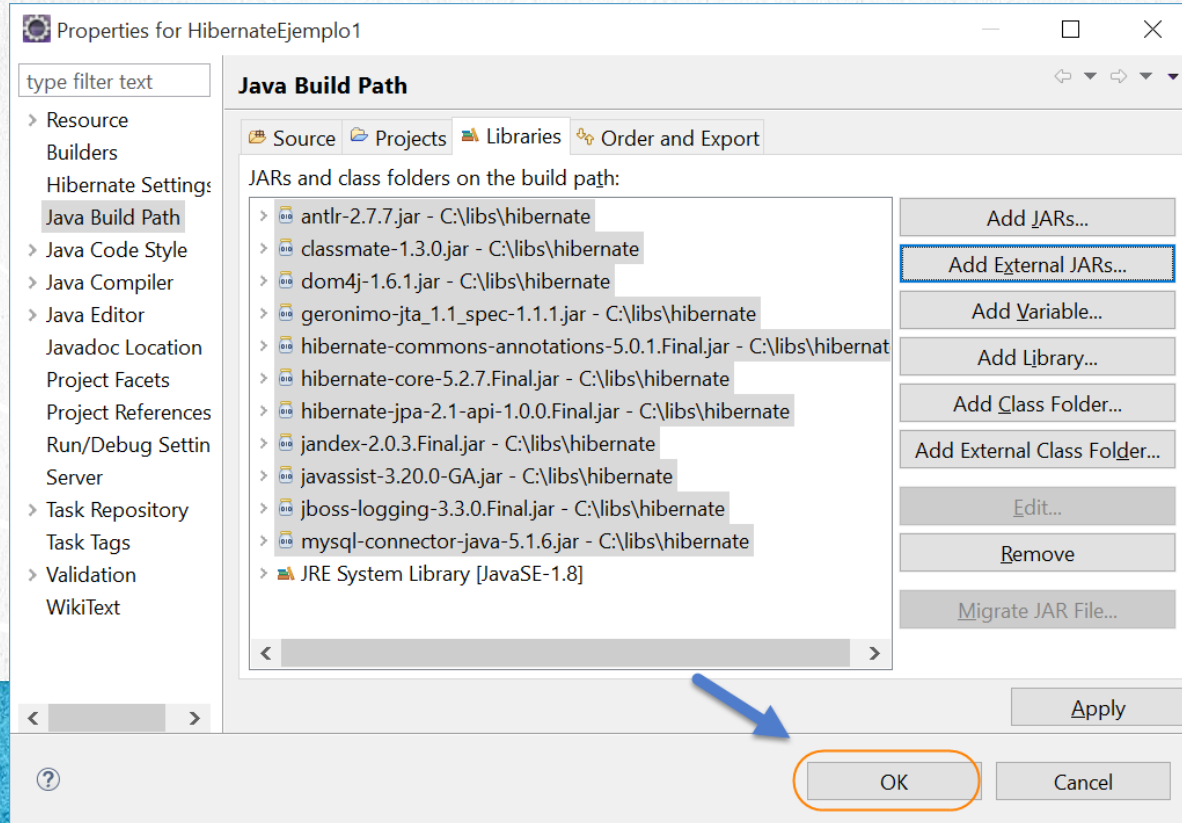
- Agregamos las librerías necesarias para trabajar con Hibernate. Las versiones pueden variar pero pueden agregar las más recientes:



Seleccionar
todas las
librerías

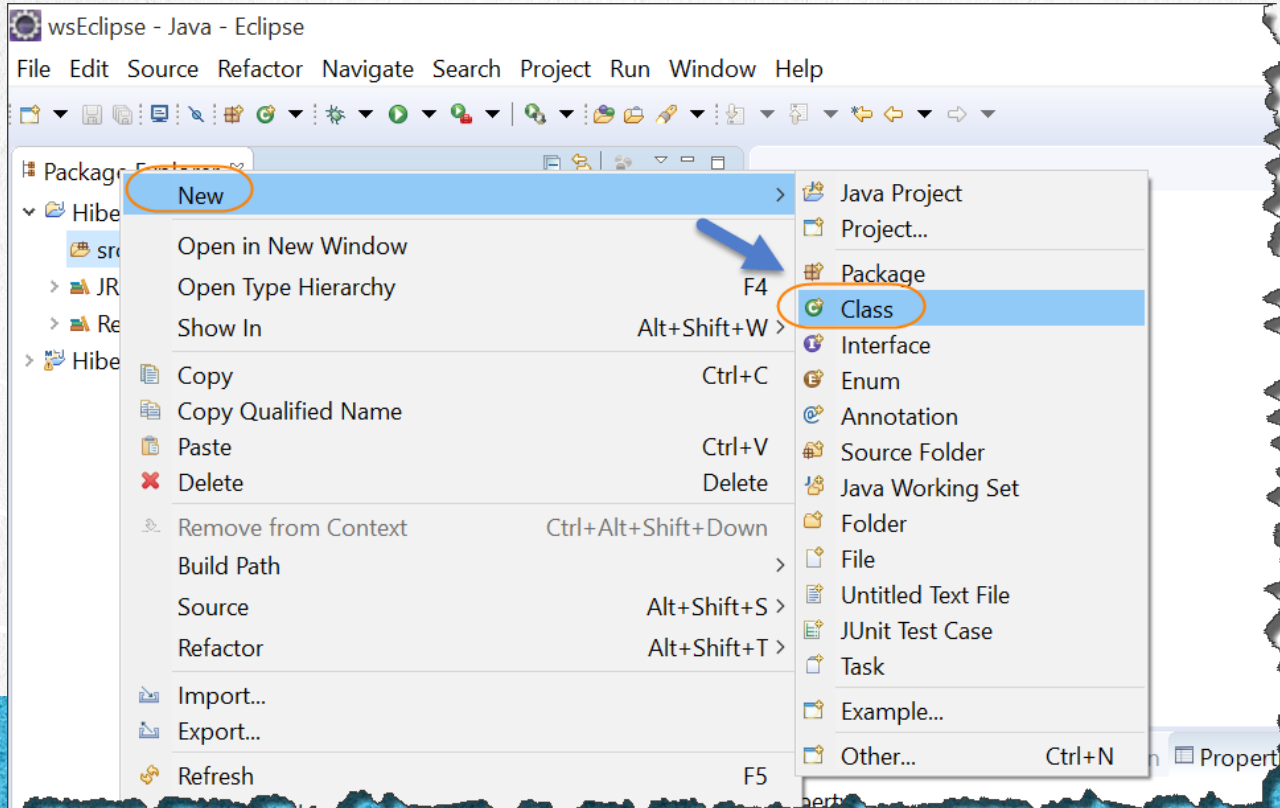
3. AGREGAR LAS LIBRERÍAS

- Agregamos las librerías necesarias para trabajar con Hibernate:



4. CREAR LA CLASE PERSONA

- Creamos la clase Persona.java:



4. CREAR LA CLASE PERSONA

- Creamos la clase Persona.java:

New Java Class

Java Class

Create a new Java class.

Source folder: HibernateEjemplo1/src Browse...

Package: model Browse...

☐ Enclosing type: Browse...

Name: Persona

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object Browse...

Interfaces: Add...
Remove

Which method stubs would you like to create?

☐ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

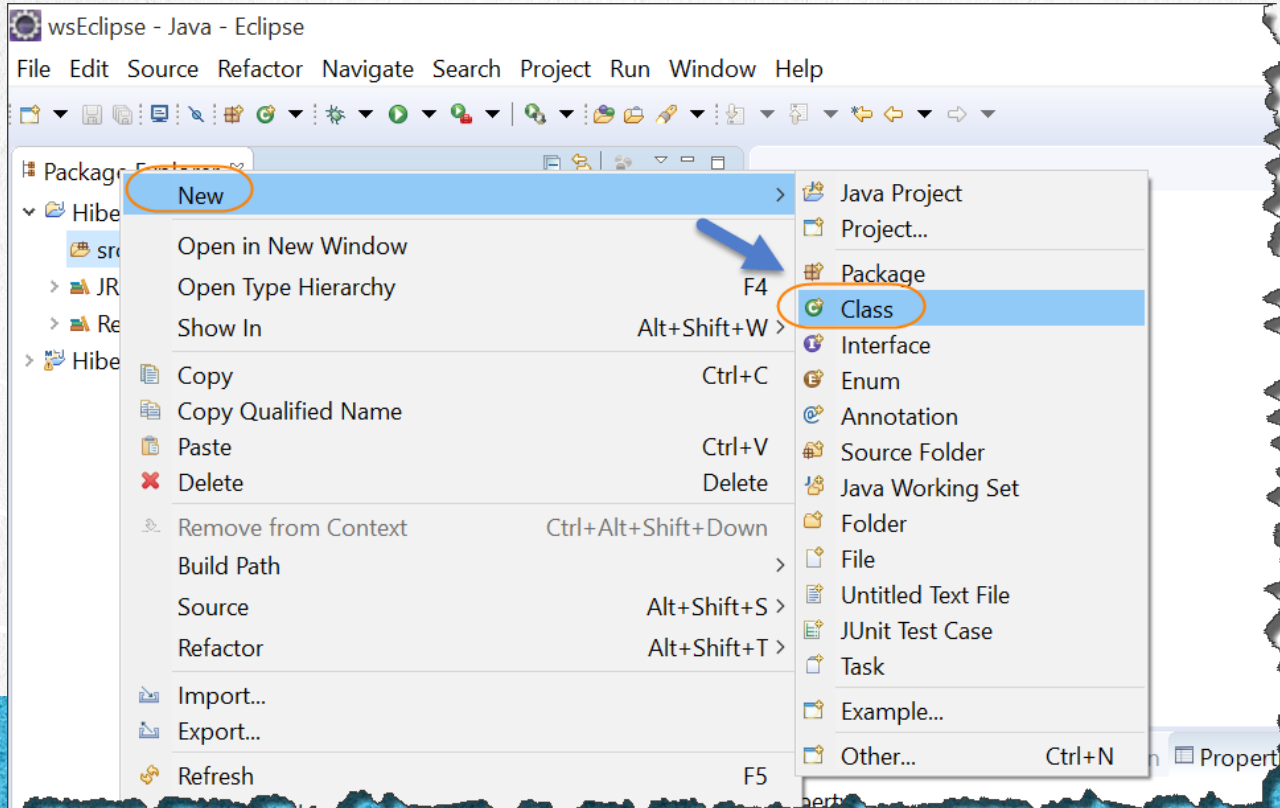
Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

? Finish Cancel

5. CREAR UNA CLASE JAVA

- Creamos la clase HolaMundoHibernate.java:



5. CREAR UNA CLASE JAVA

- Creamos la clase HolaMundoHibernate.java:

New Java Class

Java Class

Create a new Java class.

Source folder: HibernateEjemplo1/src Browse...

Package: test Browse...

☐ Enclosing type: Browse...

Name: HolaMundoHibernate

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object Browse...

Interfaces: Add...
Remove

Which method stubs would you like to create?

☐ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

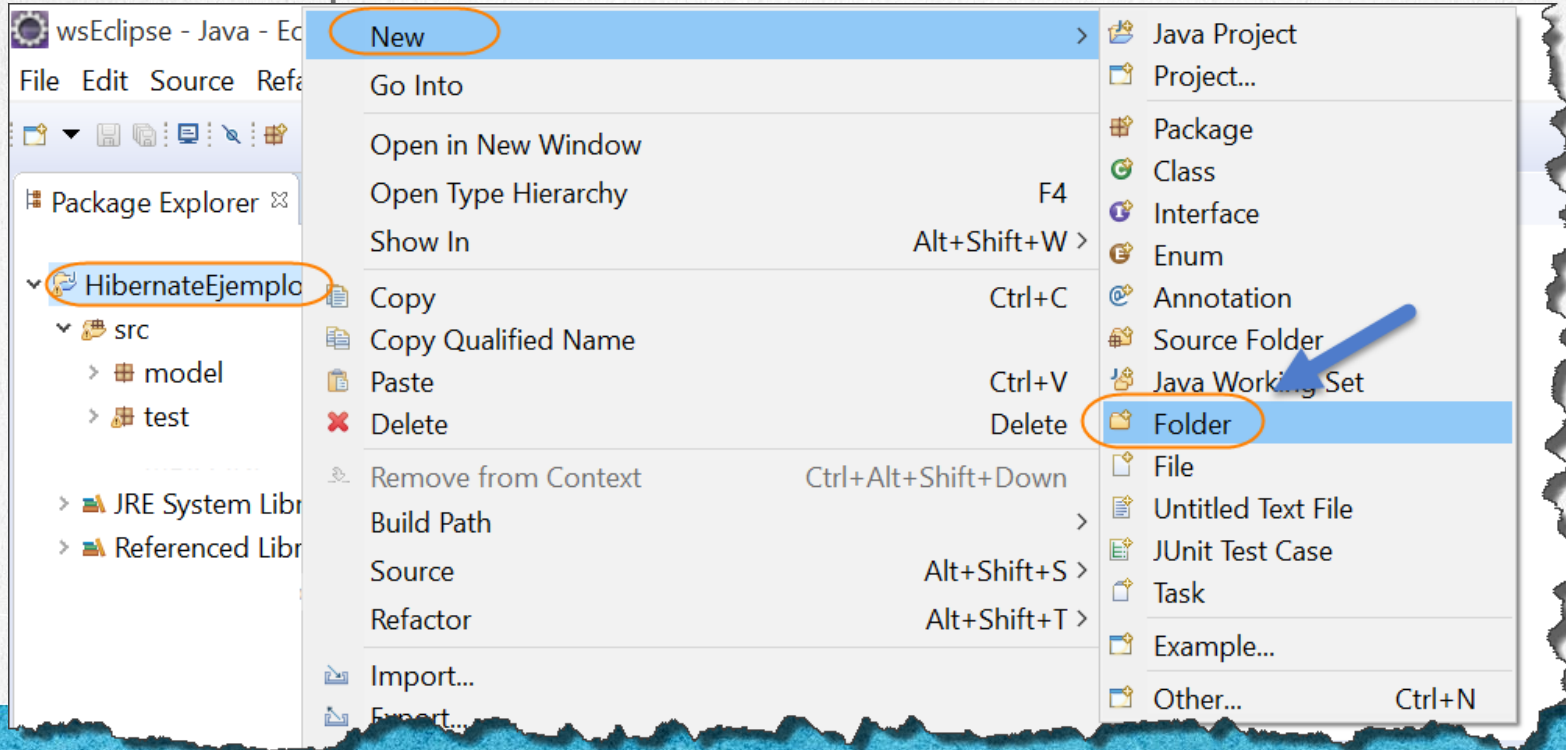
Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

? Finish Cancel

6. CREAR UNA CARPETA

- Creamos la carpeta META-INF:

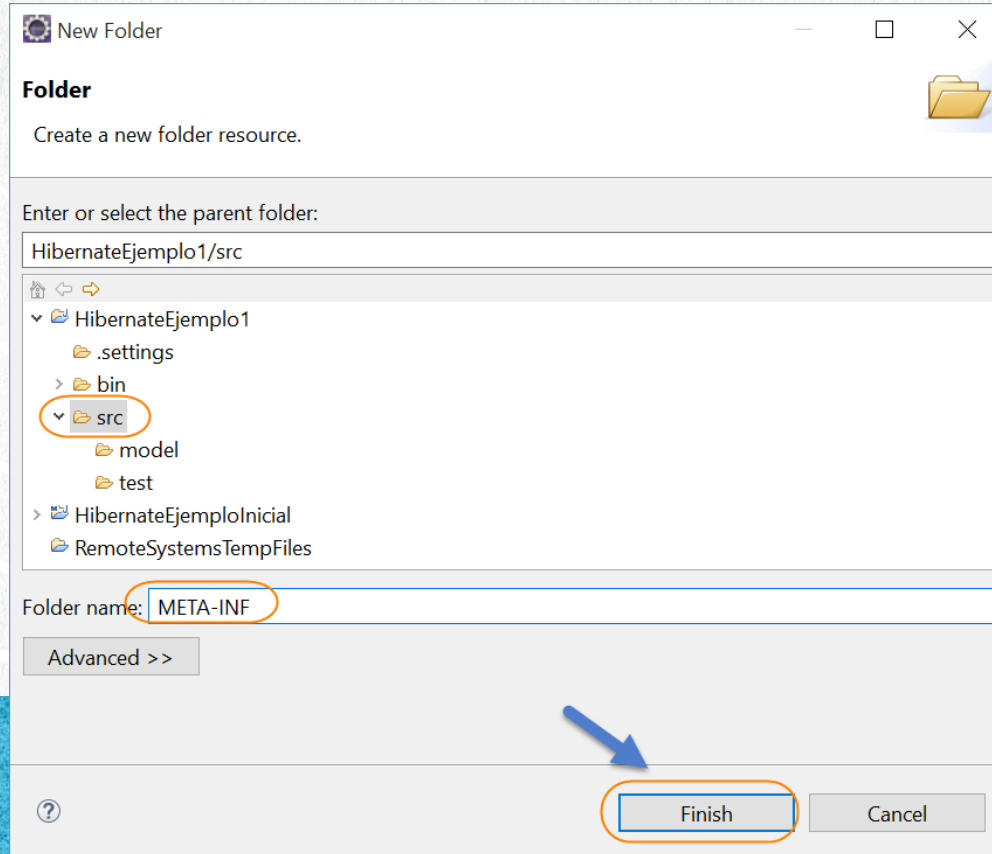


CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx

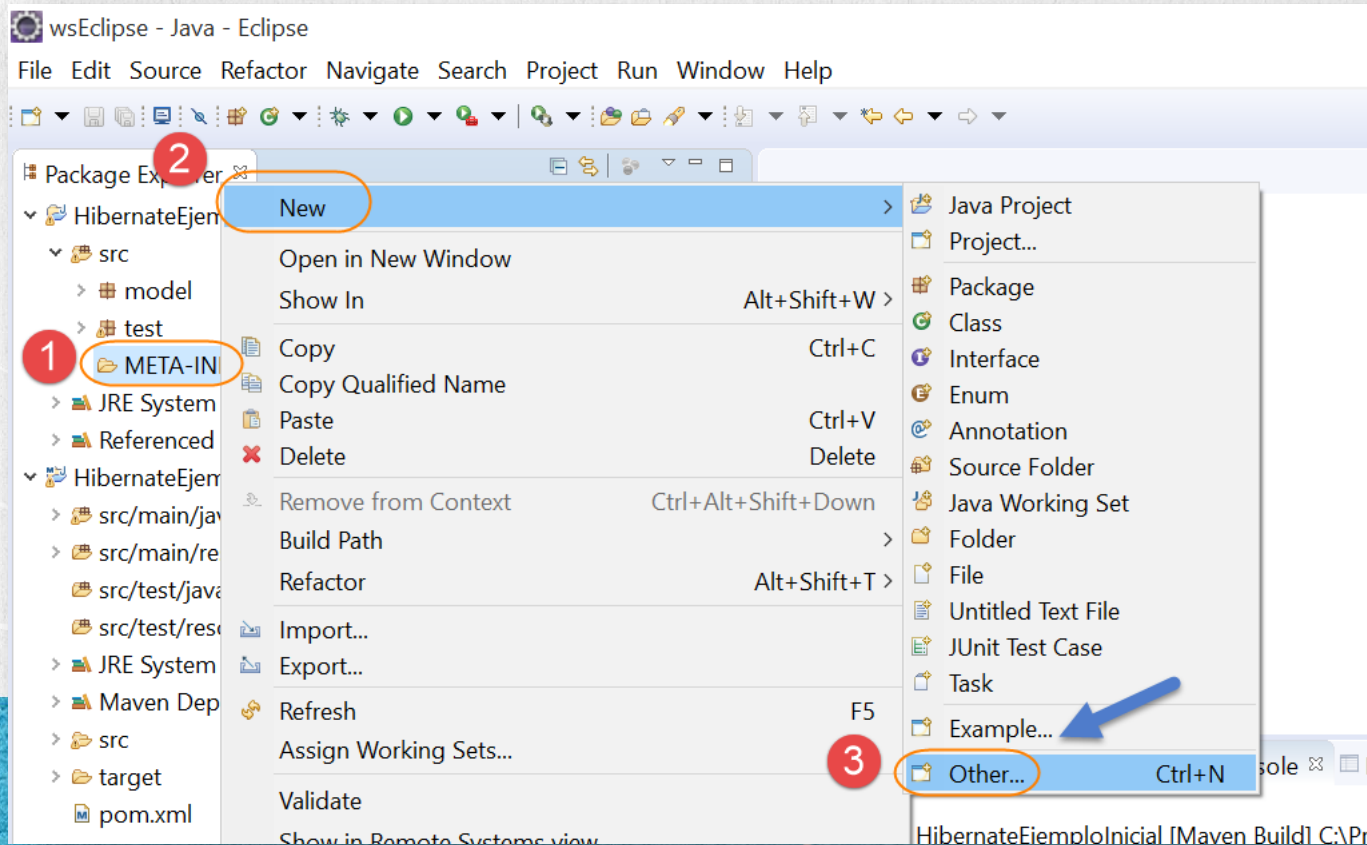
6. CREAR UNA CARPETA

- Creamos la carpeta META-INF:



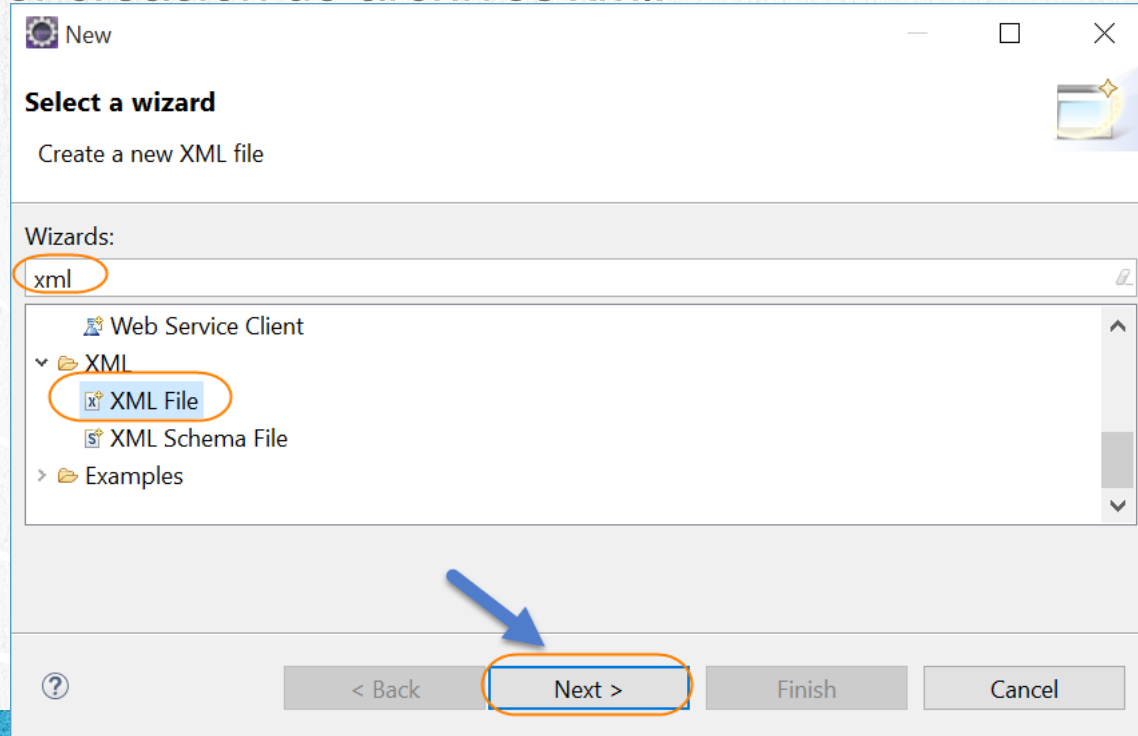
7. CREAR UN ARCHIVO XML

- Creamos el archivo persistence.xml dentro de la carpeta META-INF:



7. CREAR UN ARCHIVO XML

- Filtramos por creación de archivos xml:

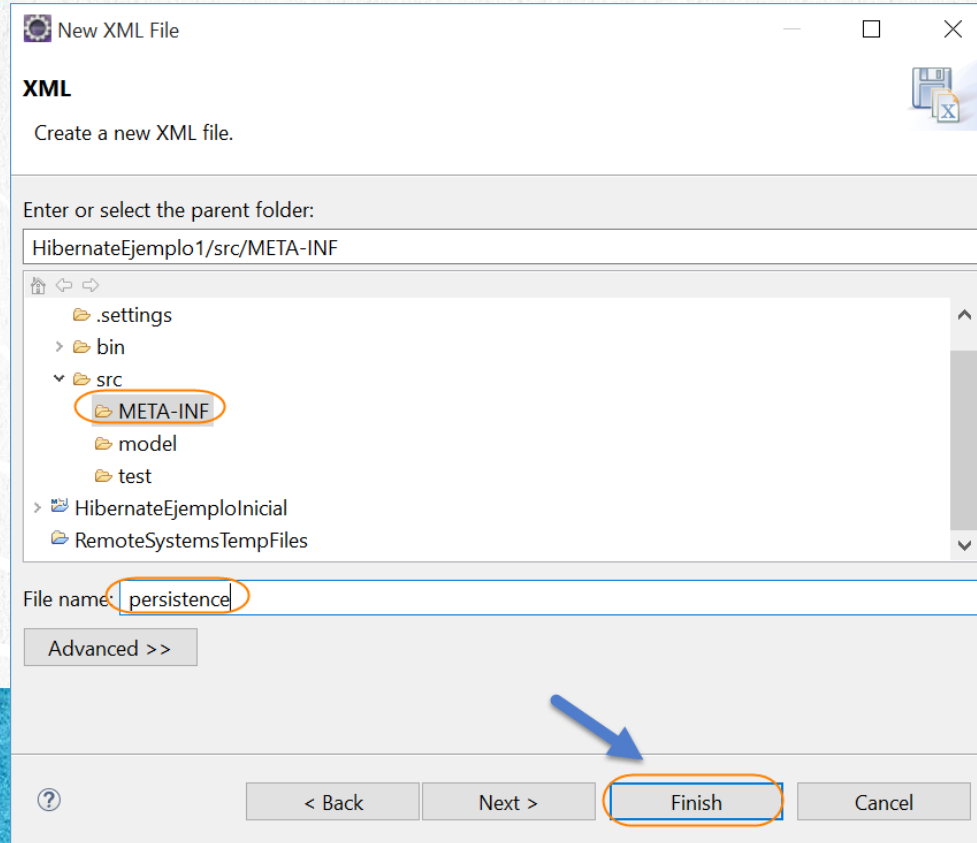


CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx

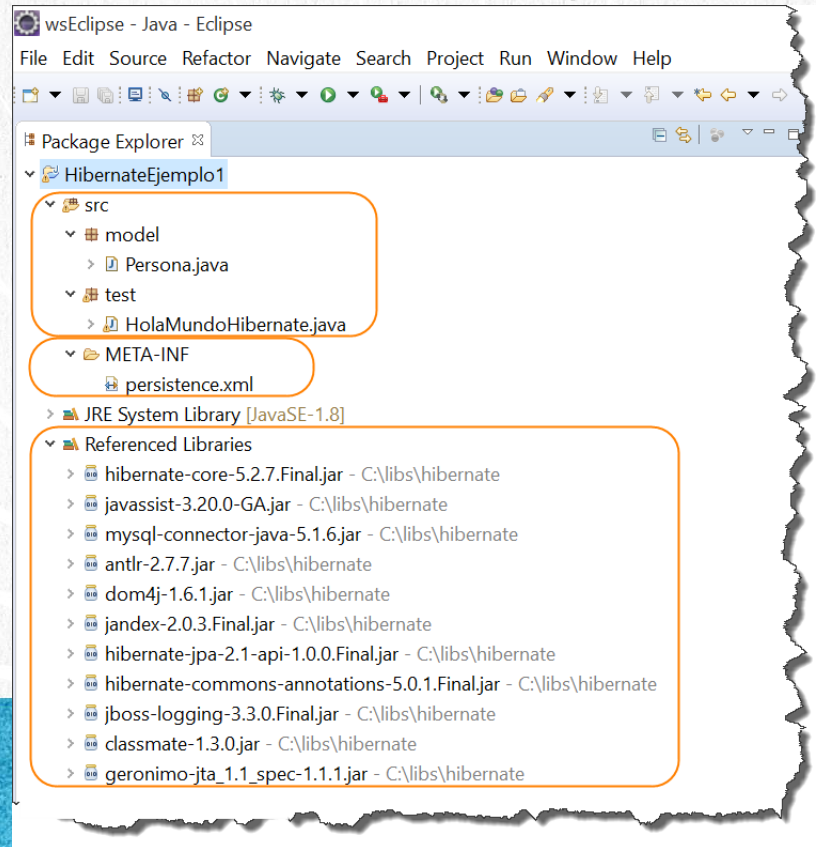
7. CREAR UN ARCHIVO XML

- Creamos el archivo persistence.xml:



8. ESTRUCTURA DEL PROYECTO

- La estructura del proyecto queda como sigue:



PASO 9. MODIFICAMOS EL CÓDIGO

Archivo Persona.java:

[Click para descargar archivo](#)

```
package model;

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

@Entity
public class Persona implements Serializable {

    private static final long serialVersionUID = 1L;

    @Column(name = "id_persona")
    @Id
    private int idPersona;

    private String apellido;

    private String nombre;

    public Persona() {
    }

    public String getApellido() {
        return this.apellido;
    }

    public void setApellido(String apellido) {
        this.apellido = apellido;
    }
}
```


PASO 9. MODIFICAMOS EL CÓDIGO

Archivo Persona.java:

Clic para ver el archivo

```
public int getIdPersona() {  
    return this.idPersona;  
}  
  
public void setIdPersona(int idPersona) {  
    this.idPersona = idPersona;  
}  
  
public String getNombre() {  
    return this.nombre;  
}  
  
public void setNombre(String nombre) {  
    this.nombre = nombre;  
}  
  
@Override  
public String toString() {  
    return "Persona [apellido=" + apellido + ", idPersona=" + idPersona + ", nombre=" + nombre + "];"  
}  
}
```

PASO 10. MODIFICAMOS EL CÓDIGO

Archivo HolaMundoHibernate.java:

Clic para ver el
archivo

```
package test;

import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.Query;

import model.Persona;

public class HolaMundoHibernate {

    public static void main(String[] args) {
        String hql = "SELECT p FROM Persona p";
        EntityManagerFactory fabrica = Persistence.createEntityManagerFactory("HibernateEjemplo1");
        EntityManager entityManager = fabrica.createEntityManager();
        Query query = entityManager.createQuery(hql);
        List<Persona> list = query.getResultList();
        for (Persona p : list) {
            System.out.println("persona:" + p);
        }
    }
}
```


PASO 11. MODIFICAMOS EL CÓDIGO

Archivo persistence.xml:

Clic para ver el
archivo

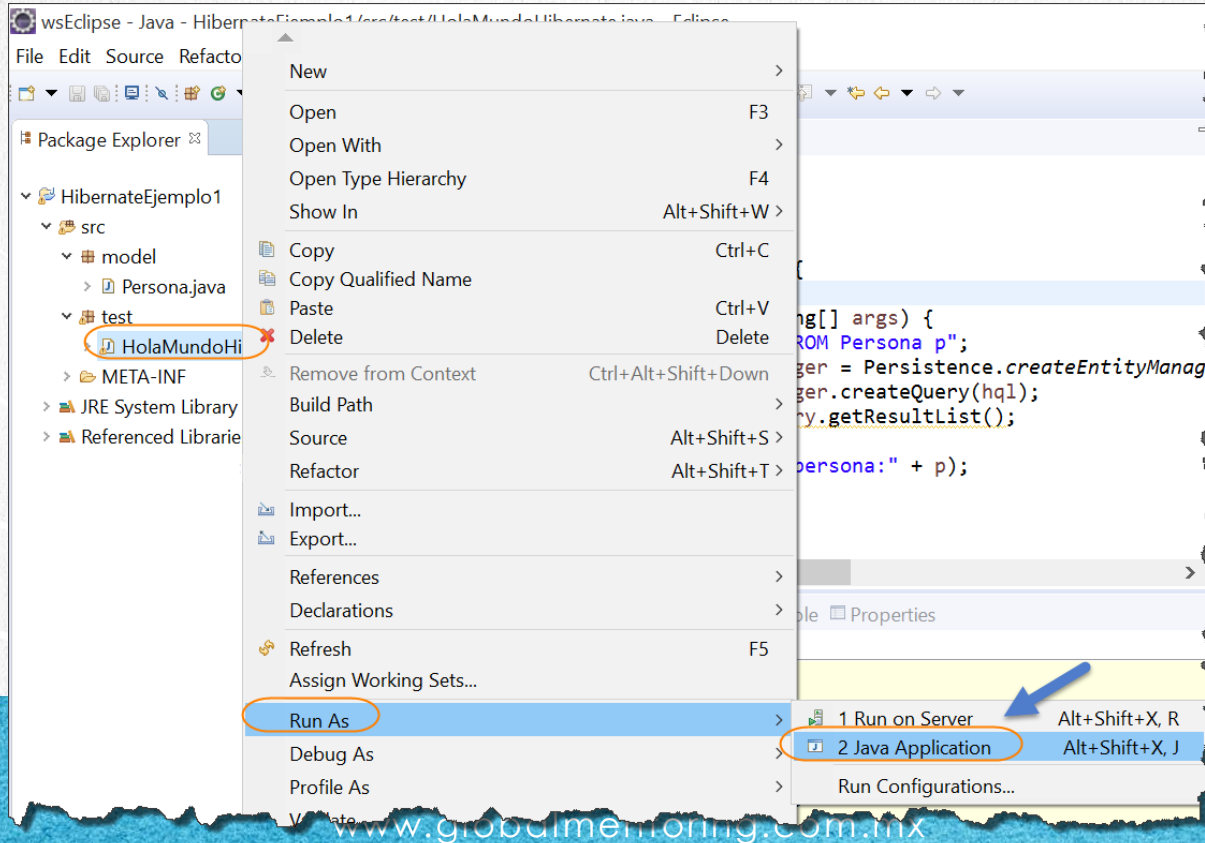
```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
  <persistence-unit name="HibernateEjemplo1" transaction-type="RESOURCE_LOCAL">
    <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
    <class>model.Persona</class>
    <properties>
      <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/sga"/>
      <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"/>
    </properties>
  </persistence-unit>
</persistence>
```

CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx

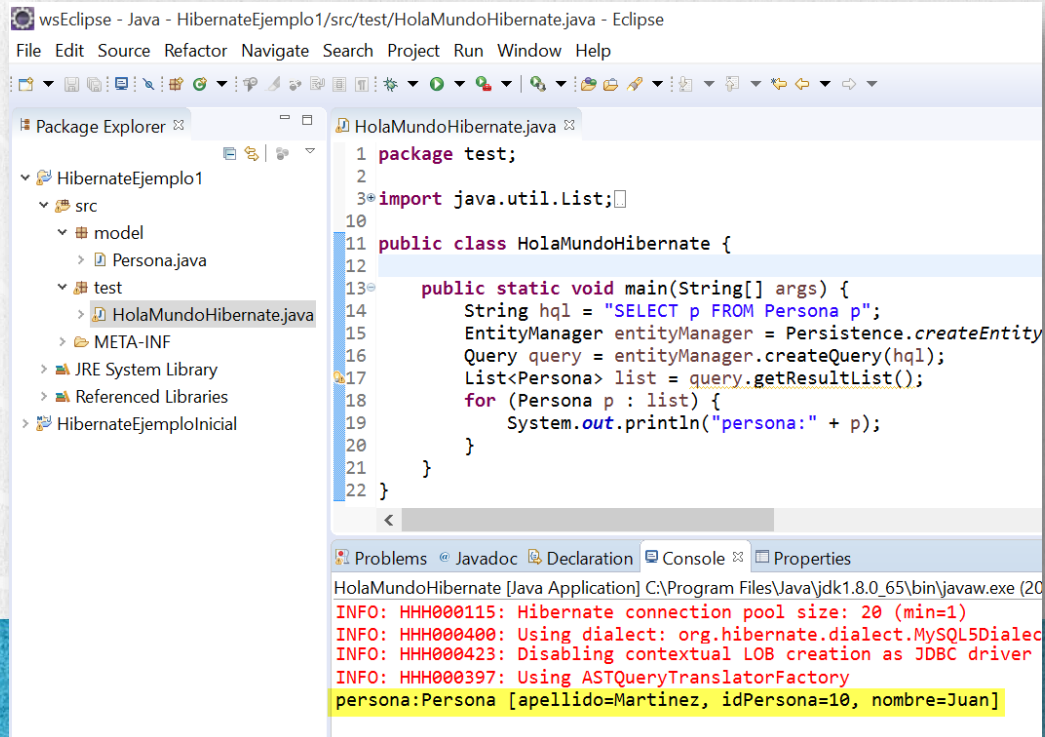
12. EJECUTAMOS LA APLICACIÓN

- Ejecutamos la aplicación desde la clase HolaMundoHibernate.java:



12. EJECUTAMOS LA APLICACIÓN

- Vemos que se listan las personas almacenadas en la tabla de Personas a través del query de Hibernate/JPA que se hemos ejecutado:



```
wsEclipse - Java - HibernateEjemplo1/src/test/HolaMundoHibernate.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer
  HibernateEjemplo1
    src
      model
        Persona.java
      test
        HolaMundoHibernate.java
      META-INF
      JRE System Library
      Referenced Libraries
      HibernateEjemploInicial

HolaMundoHibernate.java
1 package test;
2
3 import java.util.List;
10
11 public class HolaMundoHibernate {
12
13     public static void main(String[] args) {
14         String hql = "SELECT p FROM Persona p";
15         EntityManager entityManager = Persistence.createEntityManagerFactory("hibernate").createEntityManager();
16         Query query = entityManager.createQuery(hql);
17         List<Persona> list = query.getResultList();
18         for (Persona p : list) {
19             System.out.println("persona: " + p);
20         }
21     }
22 }

Problems Javadoc Declaration Console Properties
HolaMundoHibernate [Java Application] C:\Program Files\Java\jdk1.8.0_65\bin\javaw.exe (20
INFO: HHH000115: Hibernate connection pool size: 20 (min=1)
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQL5Dialect
INFO: HHH000423: Disabling contextual LOB creation as JDBC driver
INFO: HHH000397: Using ASTQueryTranslatorFactory
persona:Persona [apellido=Martinez, idPersona=10, nombre=Juan]
```


CONCLUSIÓN DEL EJERCICIO

Con este ejercicio hemos configurado nuestro proyecto para soportar Hibernate apoyándonos del API de JPA que es como generalmente se realizan los proyectos actualmente al utilizar Hibernate.

Configuramos la clase de Entidad llamada Persona. También creamos una clase de prueba donde se utilizó el objeto EntityManager, y con ayuda de queries conocidos como HQL (Hibernate Query Language) o de tipo JPQL (Java Persistence Query Language) es que pudimos obtener el listado de la tabla de Personas.

Además configuramos el archivo persistence.xml el cual nos permite unir tanto los archivos de entidad, así como los datos de conexión a nuestra base de datos, entre varias características más.

No es necesario entender todos estos conceptos, ya que precisamente todo esto es lo que iremos desarrollando a lo largo del curso. Este ejercicio solo lo hemos creado para tener una idea general de lo que vamos a trabajar en las próximas lecciones.

CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx

CURSO ONLINE

HIBERNATE & JPA

Por: Ing. Ubaldo Acosta



Experiencia y Conocimiento para tu vida



CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx