

Ejercicio JPA

Introducción

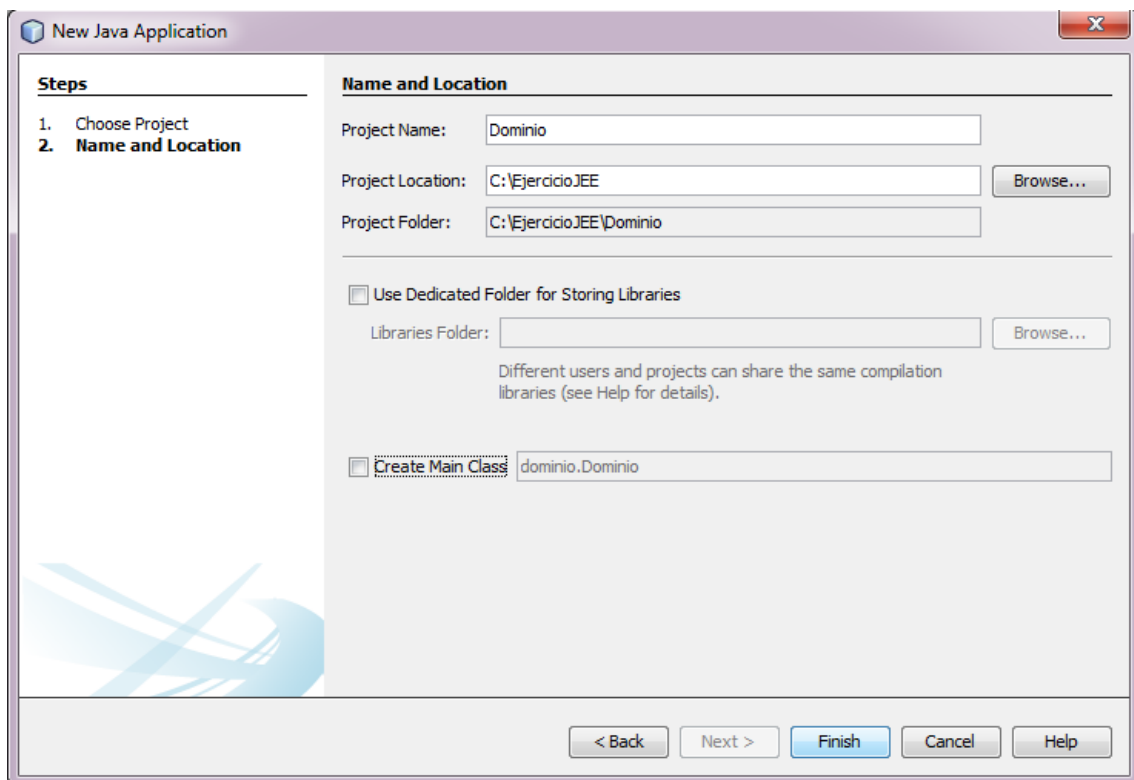
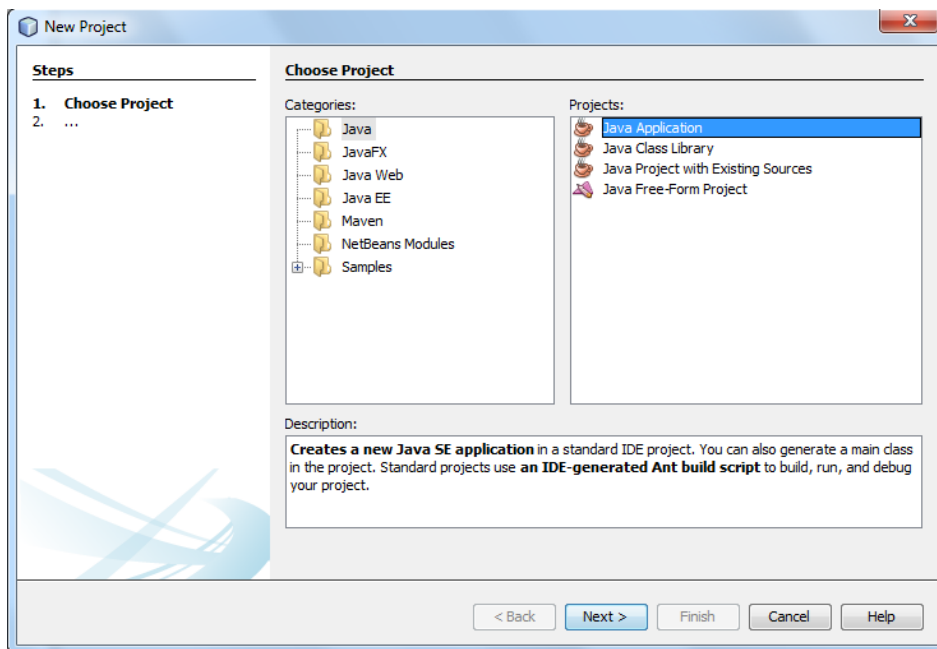
En esta guía se proporciona una introducción a la API de Persistencia provista por la plataforma JEE, para lo cual se manejarán ejemplos de mapeo objeto-relacional, operaciones de persistencia y conexión a la base de datos.

Configuración de la Base de Datos

Se asume para la realización de este ejercicio que el servidor MySQL se está ejecutando en el puerto 3306 (valor por defecto de MySQL) de localhost, y se utilizará un usuario **root** (con clave **root**) con permisos sobre un esquema **mysql**.

A-Crear Dominio

1-Crear un proyecto del tipo Java Application llamado Dominio:



Project Name: Dominio

Project Location : C:\EjercicioJEE

Desmarcar el Create Main Class

2-Crear paquete *uy.edu.ort.dominio*

3-Crear la clase *Persona* en el paquete *uy.edu.ort.dominio*

4-Codigo de la clase *Persona*

```
public class Persona {  
    private long id;  
    private String nombre;  
    private String apellido;  
    private String direccion;  
  
    public String getNombre() {  
        return nombre;  
    }  
    public void setNombre(String nombre) {  
        this.nombre = nombre;  
    }  
    public String getApellido() {  
        return apellido;  
    }  
    public void setApellido(String apellido) {  
        this.apellido = apellido;  
    }  
    public String getDireccion() {  
        return direccion;  
    }  
    public void setDireccion(String direccion) {  
        this.direccion = direccion;  
    }  
}
```

B-Creación de un módulo EJB

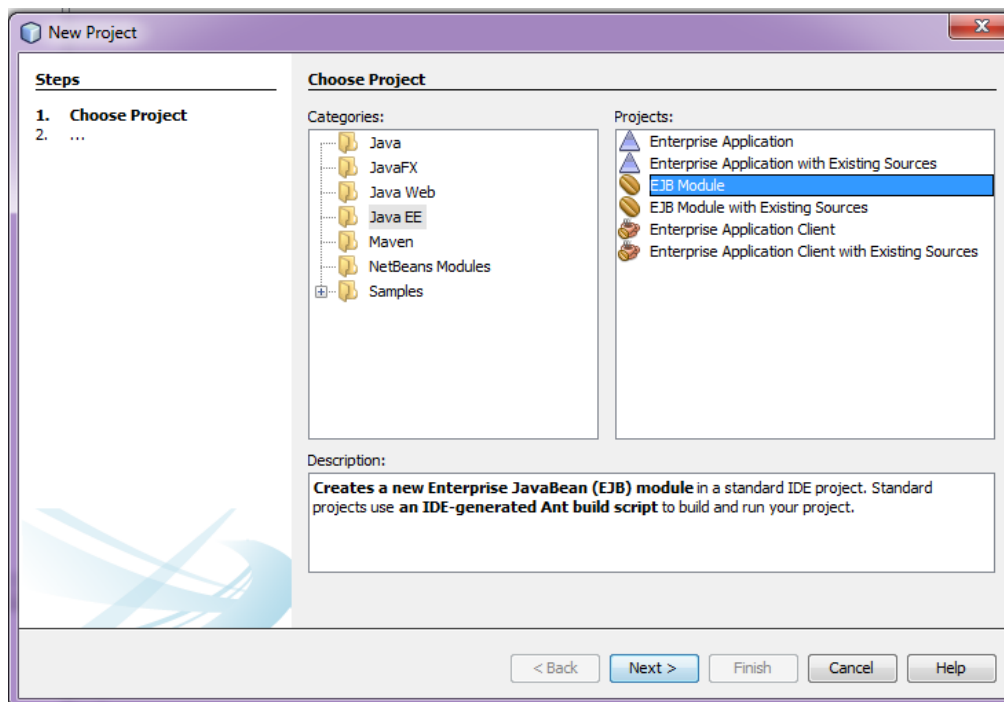
En este módulo EJB se encapsula la lógica para la persistencia de las entidades de nuestra aplicación empresarial.

Este módulo contiene los Session Bean en donde se desarrolla la lógica de persistencia.

Se creará un Session Bean sin Estado con una interface Local y a partir del mismo vamos a crear un Web Services.

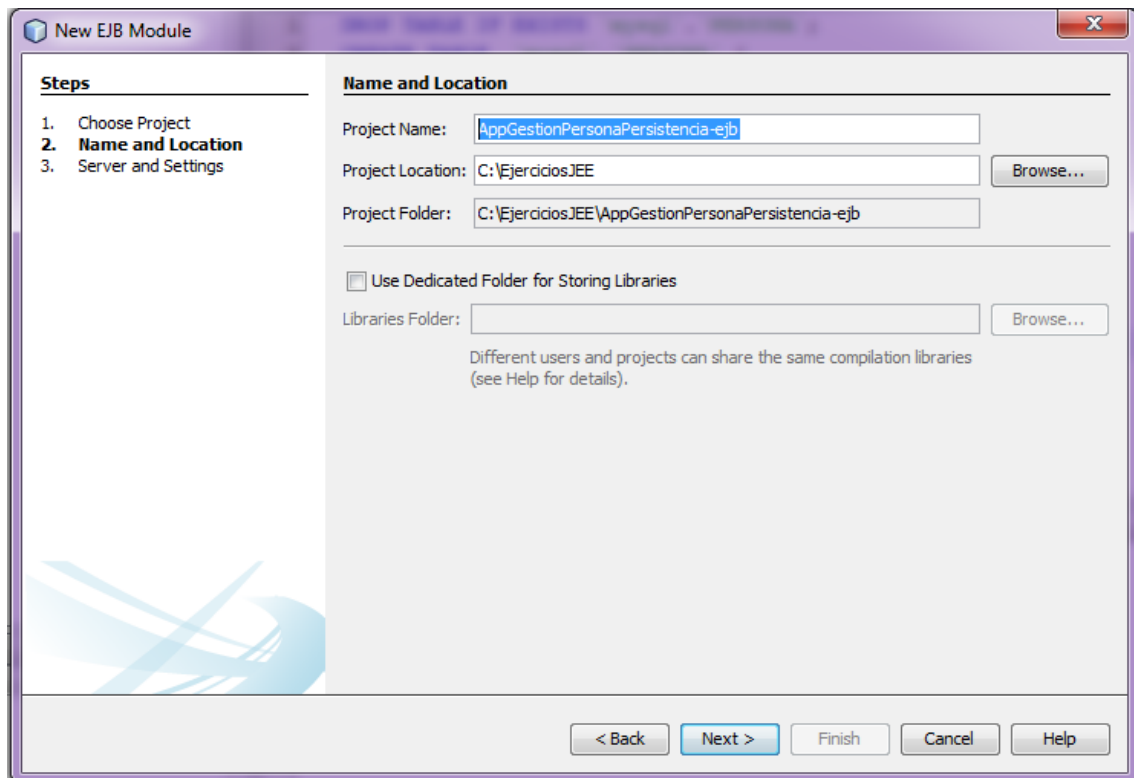
1-Crear un Módulo EJB

1.1-En la IDE de NetBeans, crear un proyecto seleccionando de la categoría JavaEE la opción EJB Module



1.2-Ingresa el nombre del proyecto AppGestionPersonaPersistencia-ejb

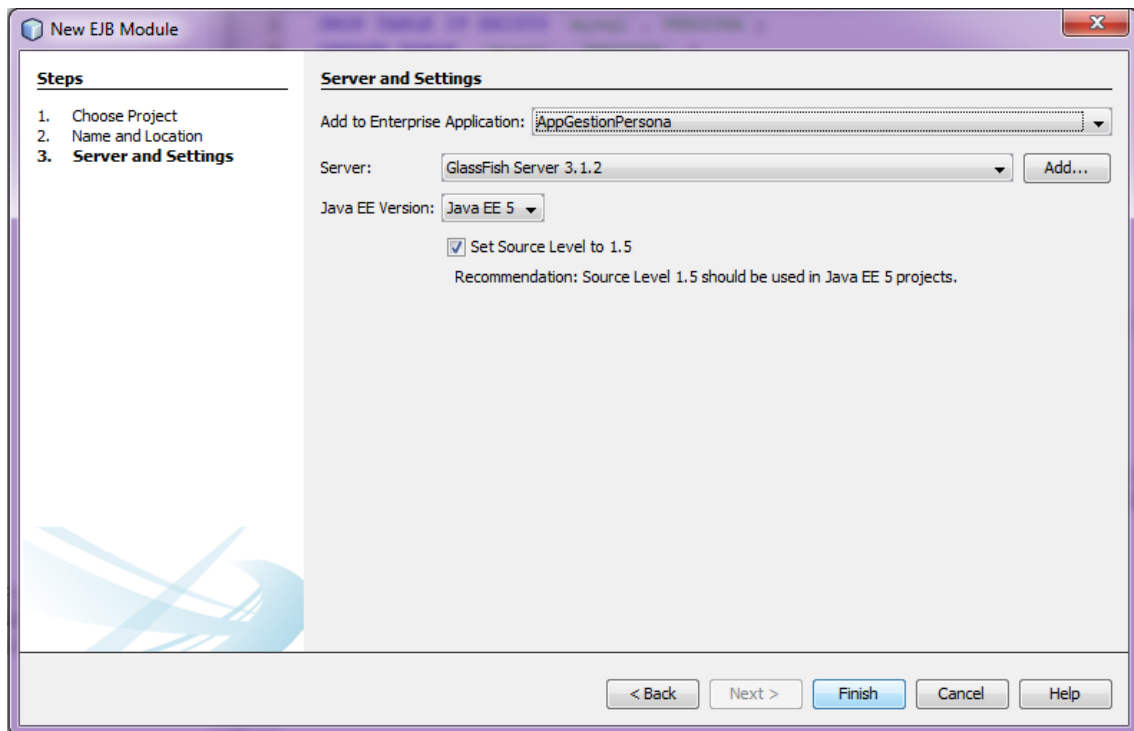
1.3-Ingresa la ubicación del proyecto en la carpeta C:\EjerciciosJEE\



1.4-Add to Enterprise Application: AppGestionPersona

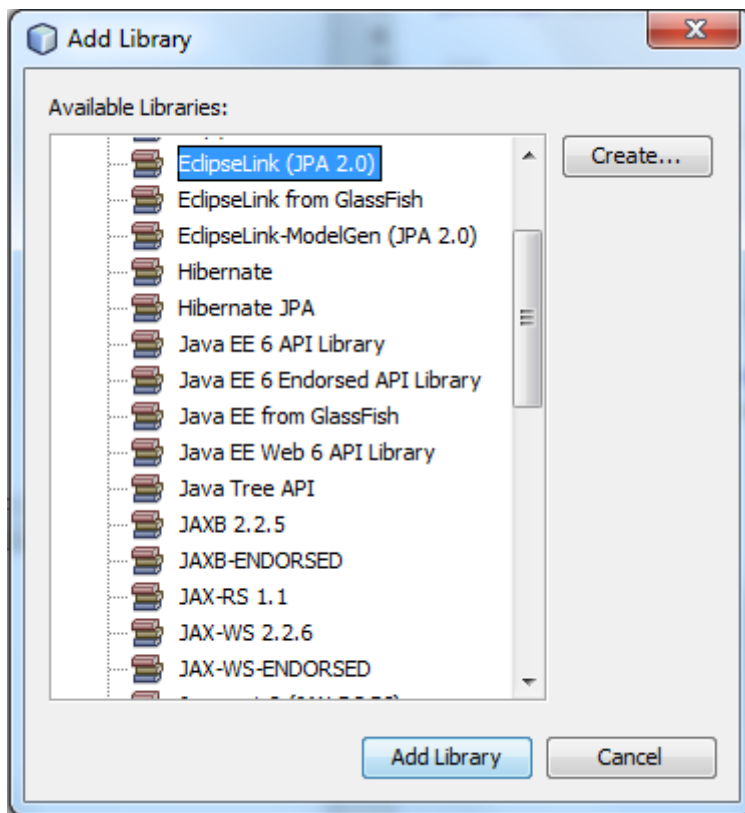
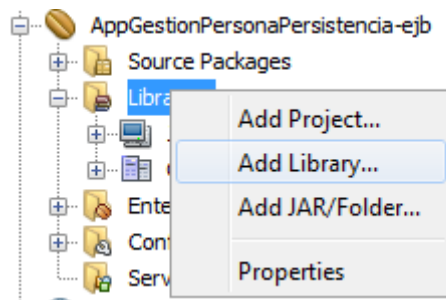
1.5-Seleccionamos el servidor Glassfish

1.6-Seleccionamos la versión de JEE 6



2- Agregar la Libreria

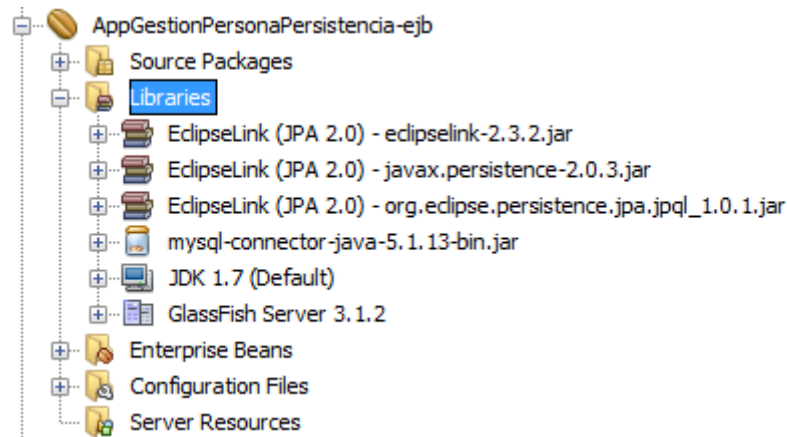
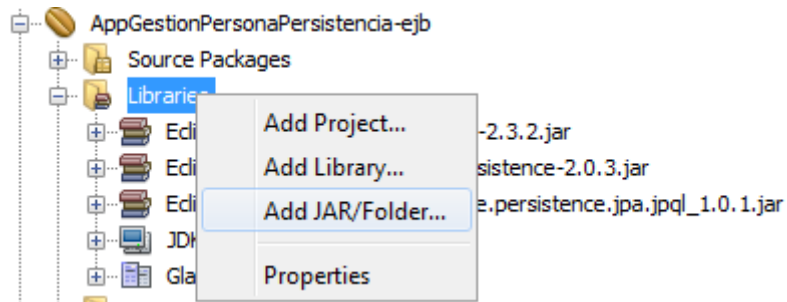
2.1-Agregar la Libreria EclipseLink



Seleccionamos ***EclipseLink(JPA 2.0)***

2.2-Agregamos el jar asociado al driver de la Base de Datos en este caso

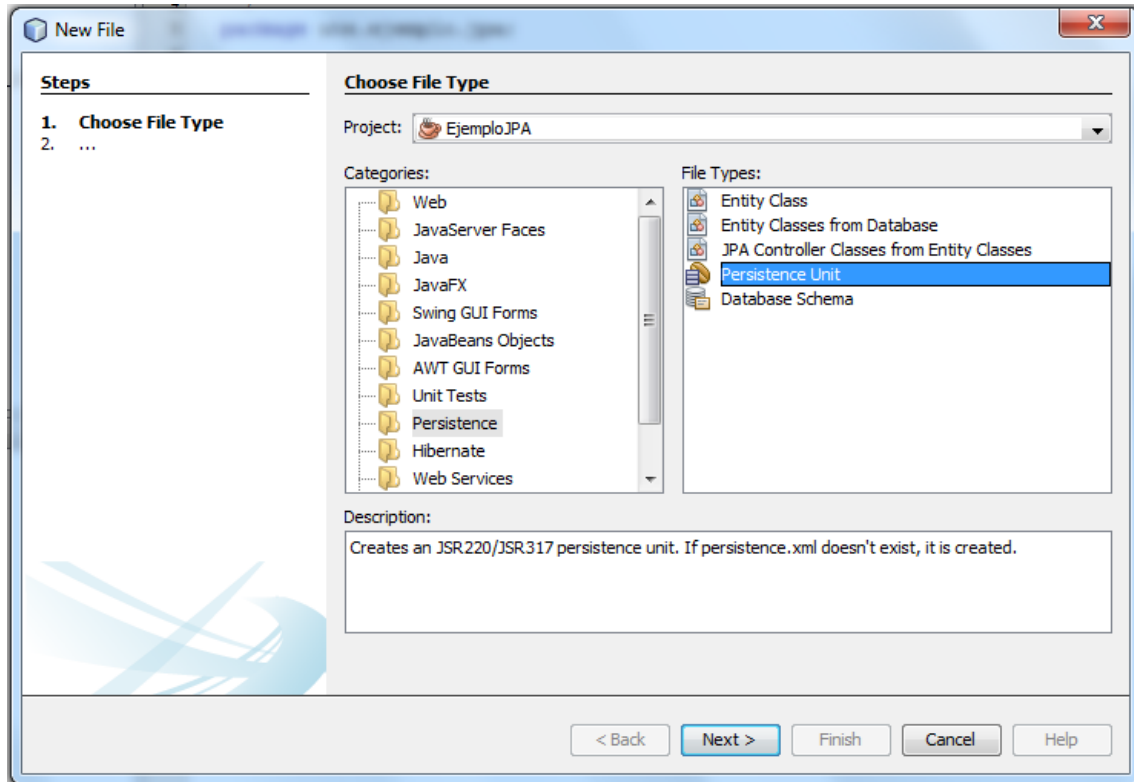
mysql-connector-java-5.1.13-bin.jar (Esta subido a aulas)



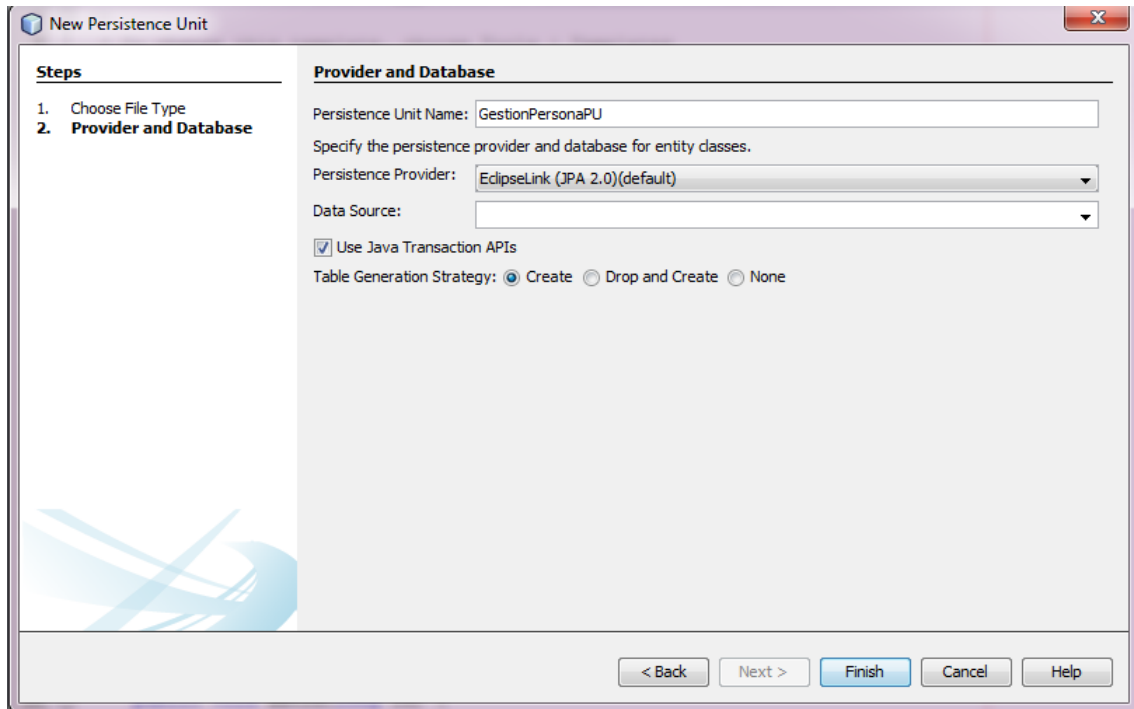
2.3-Agregamos el jar o proyecto asociado Dominio

3-Crear Persistence Unit

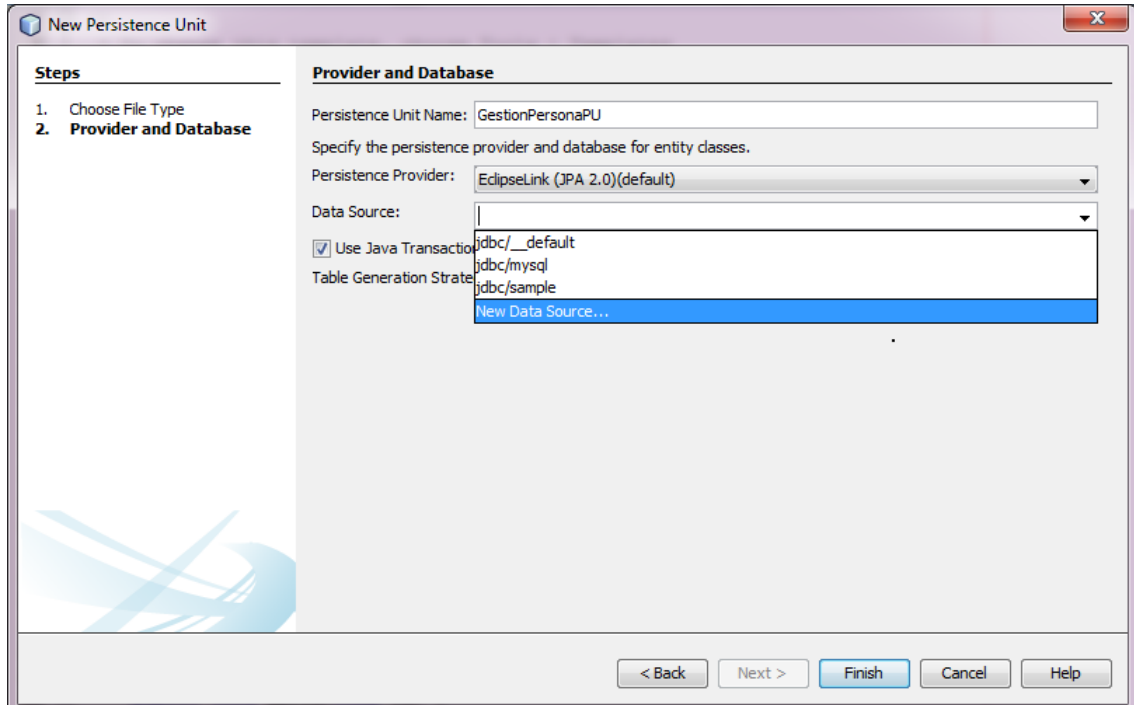
3.1-Hacer click derecho sobre el proyecto AppGestionPersonaPersistencia-ejb y seleccionar New ->Persistence->Persistence Unit



3.2-Dejar el nombre por defecto de la Persistence Unit (GestionPersonaPU), seleccionar EclipseLink (JPA 2.0) en el campo Persistence Library y la opción Create en el campo Table Generation Strategy:

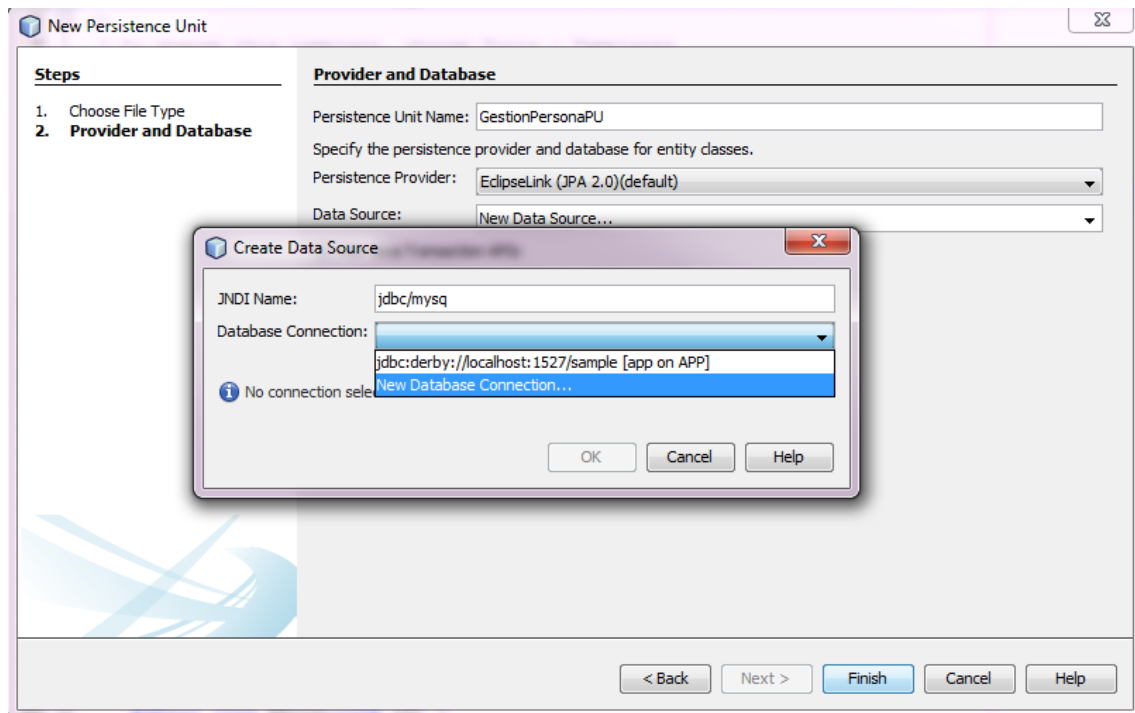


3.3. En el campo Database Source, crear una conexión con los siguientes parámetros:

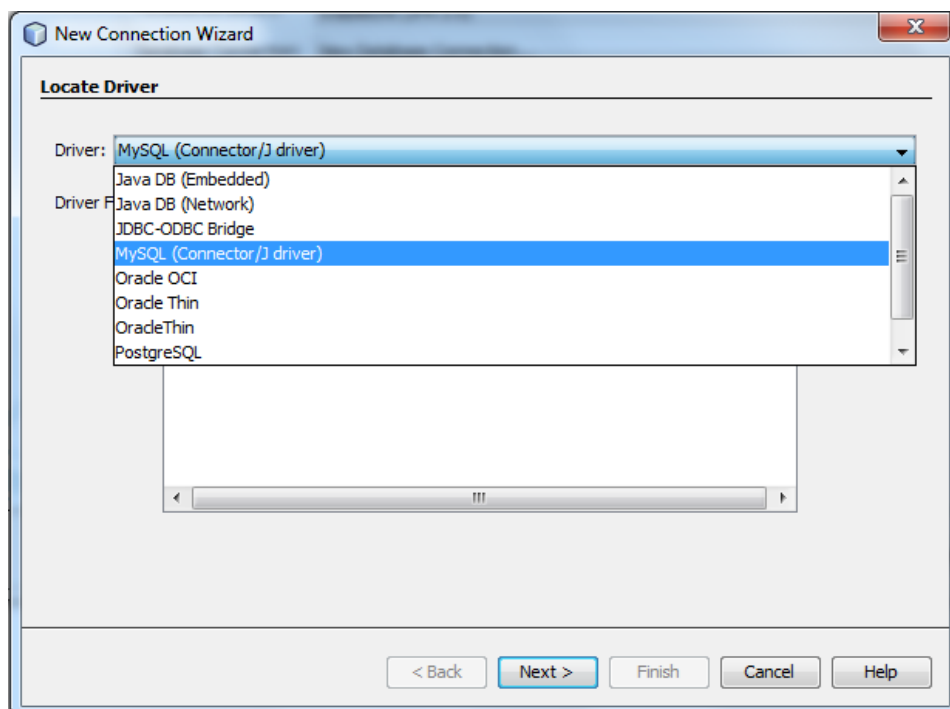


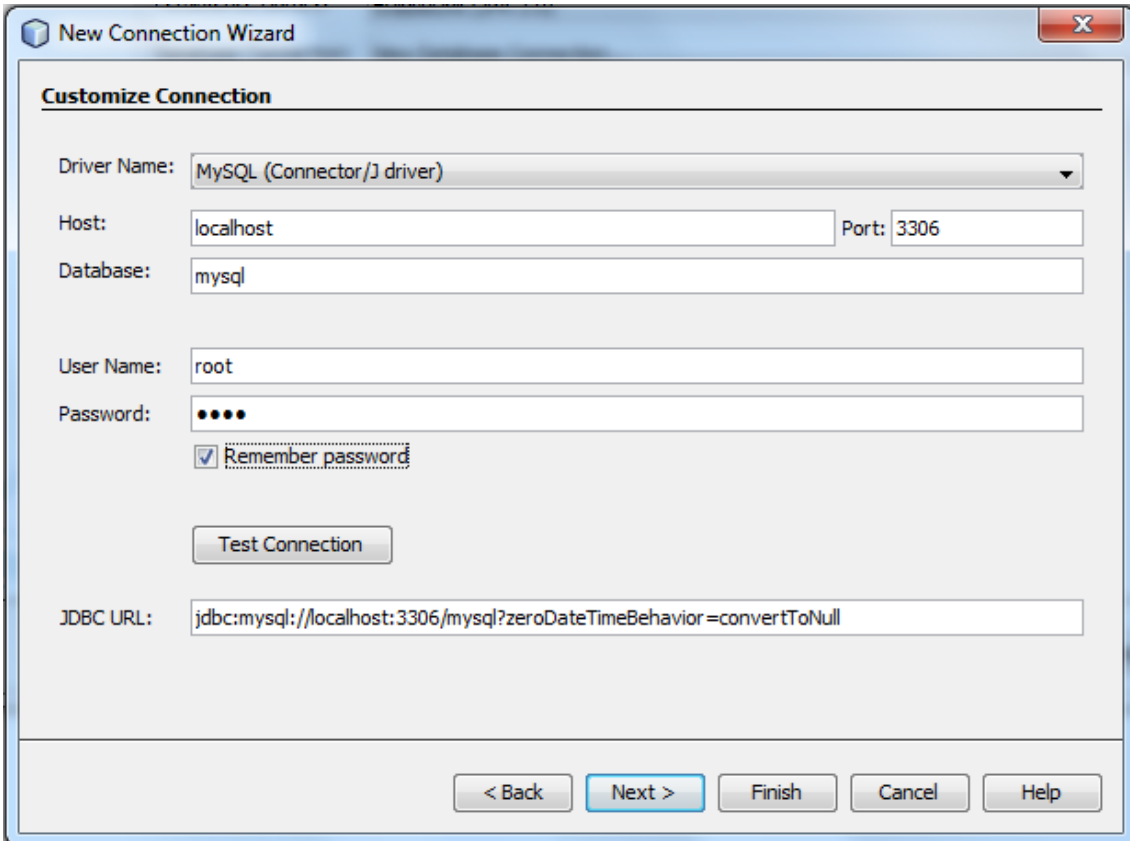
3.3.1-Ingresamos JNDI Name: *jdbc/mysql*

3.3.2-Creamos un database Connection



3.3.2.1 Seleccionamos el driver para la base de datos





The image shows a 'New Connection Wizard' dialog box with the 'Customize Connection' tab selected. The dialog contains several input fields: 'Driver Name' is a dropdown menu showing 'MySQL (Connector/J driver)'; 'Host' is a text field with 'localhost'; 'Port' is a text field with '3306'; 'Database' is a text field with 'mysql'; 'User Name' is a text field with 'root'; 'Password' is a text field with four dots, and a checked checkbox labeled 'Remember password' is below it; a 'Test Connection' button is below the password field; and 'JDBC URL' is a text field with the value 'jdbc:mysql://localhost:3306/mysql?zeroDateTimeBehavior=convertToNull'. At the bottom are five buttons: '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

New Connection Wizard

Customize Connection

Driver Name: MySQL (Connector/J driver)

Host: localhost Port: 3306

Database: mysql

User Name: root

Password: ●●●●

☒ Remember password

Test Connection

JDBC URL: jdbc:mysql://localhost:3306/mysql?zeroDateTimeBehavior=convertToNull

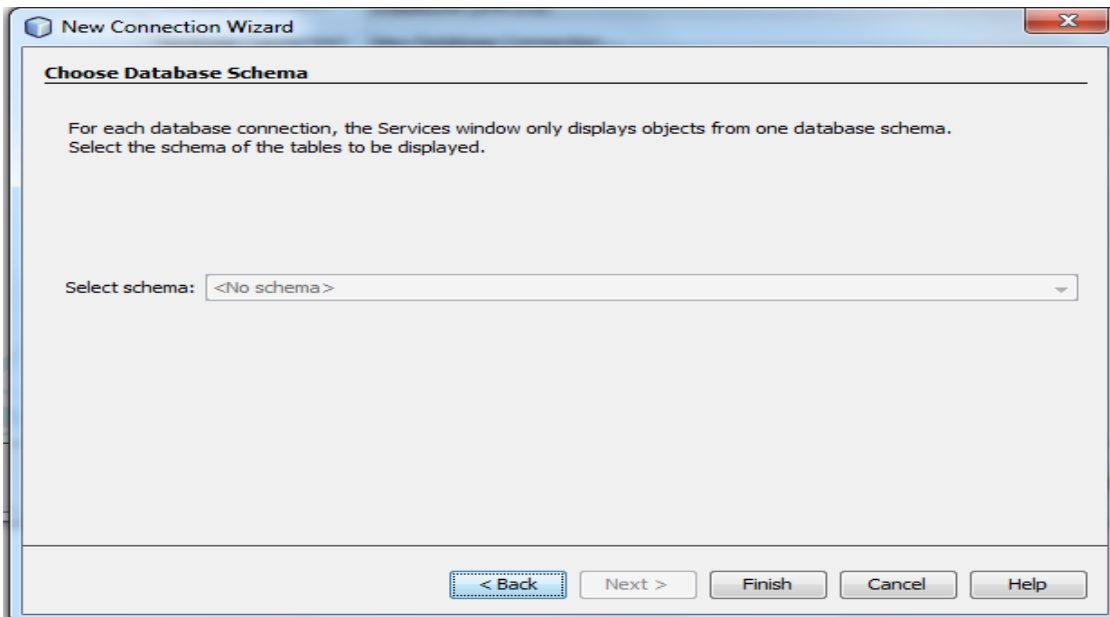
< Back Next > Finish Cancel Help

Host: localhost

Port: 3306

User Name: root

Password: root



The image shows a 'New Connection Wizard' dialog box with the 'Choose Database Schema' tab selected. The dialog contains a text area with the instruction: 'For each database connection, the Services window only displays objects from one database schema. Select the schema of the tables to be displayed.' Below this is a 'Select schema:' label followed by a dropdown menu showing '<No schema>'. At the bottom are five buttons: '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

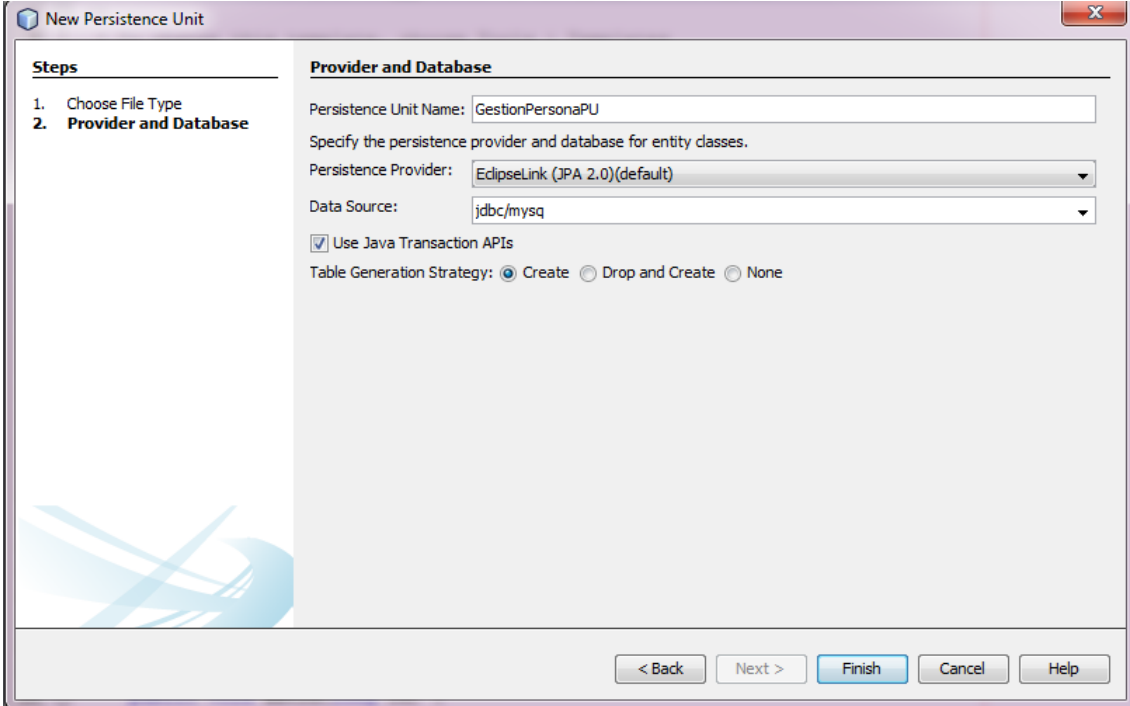
New Connection Wizard

Choose Database Schema

For each database connection, the Services window only displays objects from one database schema. Select the schema of the tables to be displayed.

Select schema: <No schema>

< Back Next > Finish Cancel Help



New Persistence Unit

Steps

1. Choose File Type
2. **Provider and Database**

Provider and Database

Persistence Unit Name:

Specify the persistence provider and database for entity classes.

Persistence Provider:

Data Source:

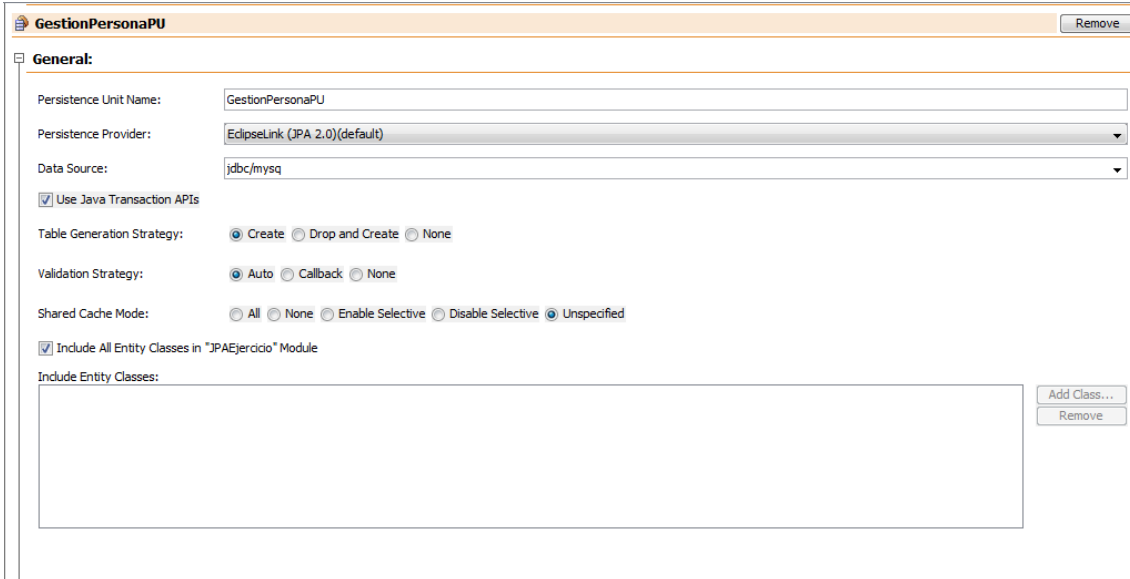
☒ Use Java Transaction APIs

Table Generation Strategy: ☒ Create ☐ Drop and Create ☐ None

< Back Next > **Finish** Cancel Help

Click en el botón Finish

Se genera el archivo persistence.xml con la información del Persistence Unit



GestionPersonaPU Remove

General:

Persistence Unit Name:

Persistence Provider:

Data Source:

☒ Use Java Transaction APIs

Table Generation Strategy: ☒ Create ☐ Drop and Create ☐ None

Validation Strategy: ☒ Auto ☐ Callback ☐ None

Shared Cache Mode: ☐ All ☐ None ☐ Enable Selective ☐ Disable Selective ☒ Unspecified

☒ Include All Entity Classes in "JPAEjercicio" Module

Include Entity Classes:

Add Class...
Remove

4-Servicios-Databases

En la sección Databases de Servicios podemos ver las conexiones de a las bases de datos creadas. Además se pueden realizar consultas SQL sobre la base de datos.

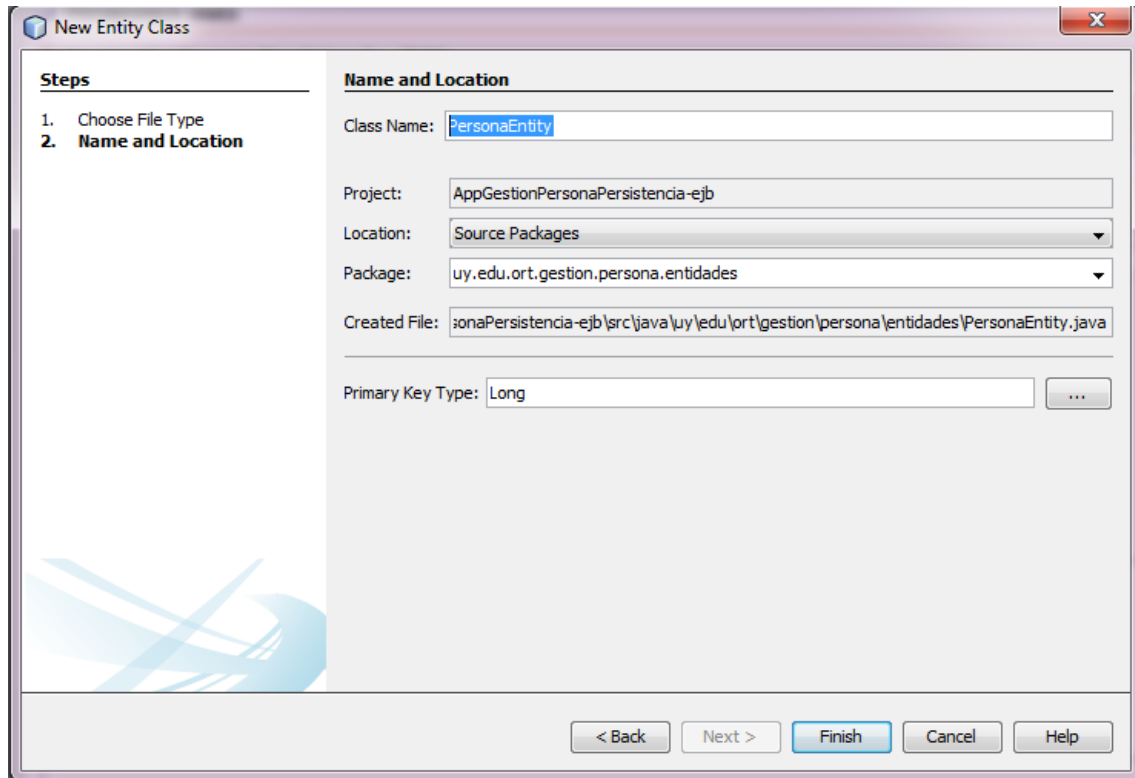


5-Crear Entity Bean

5.1-Creamos el paquete *uy.edu.ort.gestion.persona.entidades*

5.2-Creamos una clase que representara la entidad *PersonaEntity*.

En el paquete *uy.edu.ort.gestion.persona.entidades* New -> Entity Class para crear una entidad Persona en el paquete



5.3-Codigo en la Entity Persona

```
@Entity
@Table(name = "PERSONAS")
public class PersonaEntity implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "ID")
    private Long id;
    @Column(name = "NOMBRE", nullable = false)
    private String nombre;
    @Column(name = "APELLIDO", nullable = false)
    private String apellido;
    @Column(name = "DIRECCION", nullable = false)
    private String direccion;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getApellido() {
        return apellido;
    }

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

    public String getDireccion() {
        return direccion;
    }

    public void setDireccion(String direccion) {
        this.direccion = direccion;
    }
}
```

@Override

```
public int hashCode() {  
    int hash = 0;  
    hash += (id != null ? id.hashCode() : 0);  
    return hash;  
}
```

@Override

```
public boolean equals(Object object) {  
    // TODO: Warning - this method won't work in the case the id fields are not set  
    if (!(object instanceof PersonaEntity)) {  
        return false;  
    }  
    PersonaEntity other = (PersonaEntity) object;  
    if ((this.id == null && other.id != null) || (this.id != null && !this.id.equals(other.id))) {  
        return false;  
    }  
    return true;  
}
```

@Override

```
public String toString() {  
    return "uy.ort.edu.entidades.ClienteEntity[ id=" + id + " ]";  
}  
}
```


6-Crear Session Beans

6.1-Creamos el paquete *uy.edu.ort.gestion.persona.negocio*

6.2-Creamos una Session Beans *PersonaSB* de tipo *Stateless* e interface *Local* en el paquete *uy.edu.ort.gestion.persona.negocio*.

6.3Codigo de la interface Local

```
@Local
public interface PersonaSBLocal {
    public void alta(Persona persona);
    public void eliminar (Persona persona);
    public void modificar(Persona persona);
}
```

6.3 Codigo de Session Beans

```
@Stateless
public class PersonaSB implements PersonaSBLocal {
    @PersistenceContext
    EntityManager em;

    @Override
    public void alta(Persona persona) {
        PersonaEntity personaEntity= new PersonaEntity();
        personaEntity.setDireccion(persona.getDireccion());
        personaEntity.setNombre(persona.getNombre());
        personaEntity.setApellido(persona.getApellido());
        em.persist(personaEntity);
    }

    @Override
    public void eliminar(Persona persona) {
        //Completar por el alumno
    }

    @Override
    public void modificar(Persona persona) {
        //Completar por el alumno
    }

    // Add business logic below. (Right-click in editor and choose
    // "Insert Code > Add Business Method")

}
```

7- Web Services

7.1- Creamos el paquete *uy.edu.ort.gestion.persona.ws*

7.2-Crear un Web Services a partir del Session Beans *PersonaSB* en el paquete *uy.edu.ort.gestion.persona.ws* con el nombre *PersonaWS*

7.3-Codigo del Web Services

```
@WebService(serviceName = "PersonaWS")
@Stateless()
public class PersonaWS {
    @EJB
    private PersonaSBLocal ejbRef; // Add business logic below. (Right-click in editor and choose
    // "Insert Code > Add Web Service Operation")

    @WebMethod(operationName = "alta")
    @Oneway
    public void alta(@WebParam(name = "persona") Persona persona) {
        ejbRef.alta(persona);
    }

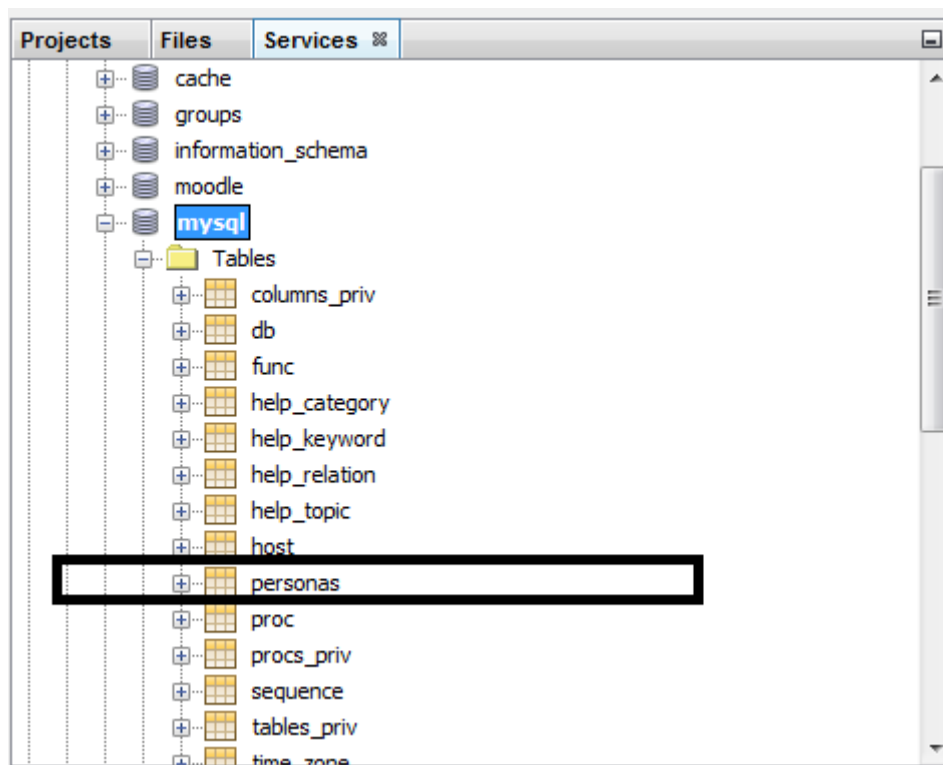
    @WebMethod(operationName = "eliminar")
    @Oneway
    public void eliminar(@WebParam(name = "persona") Persona persona) {
        ejbRef.eliminar(persona);
    }

    @WebMethod(operationName = "modificar")
    @Oneway
    public void modificar(@WebParam(name = "persona") Persona persona) {
        ejbRef.modificar(persona);
    }
}
```

8- Build y Deploy

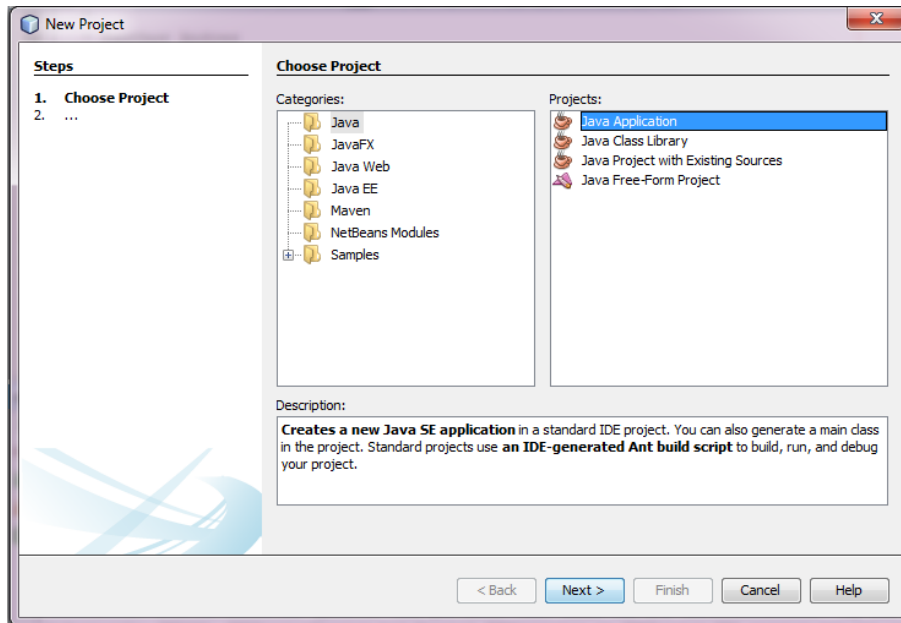
8.1-Hacer build en el proyecto *AppGestionPersona* generando el ear

8.2-Hacer el deploy en el proyecto *AppGestionPersona* , verificar que se generó el Web Services y que se creó la tabla asociada a la entidad

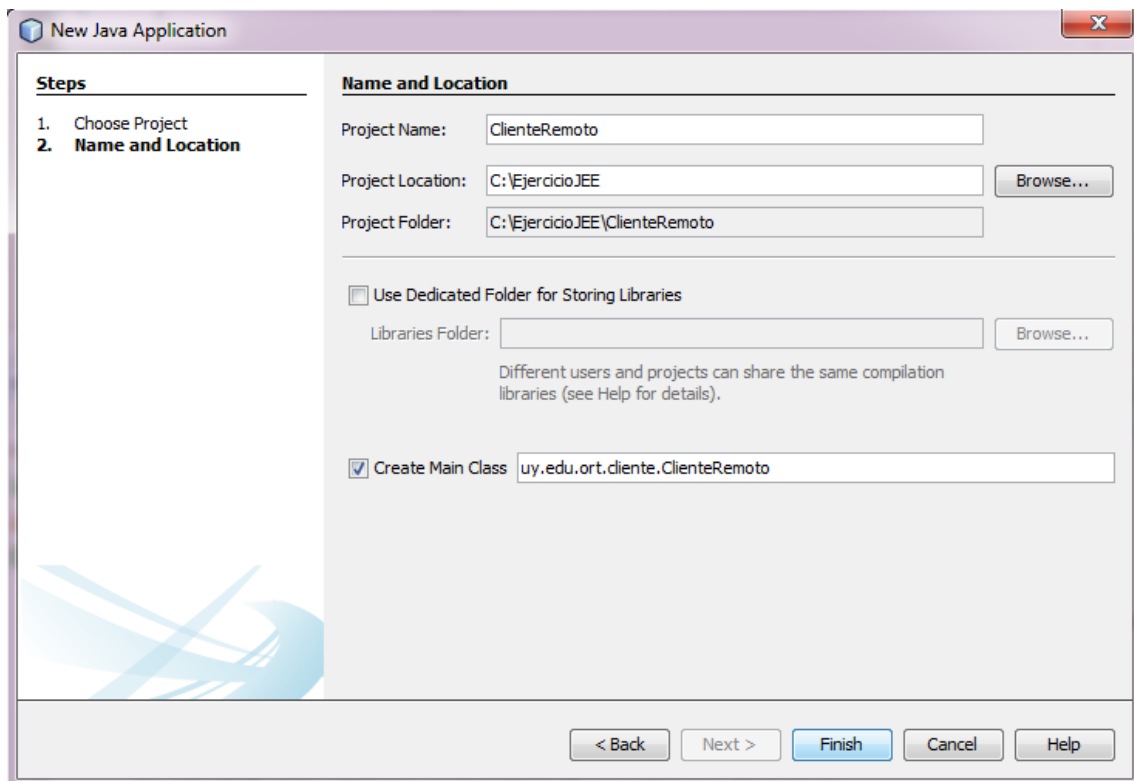


C-Cliente

1. Seleccionar del menú de NetBeans la opción para crear un nuevo proyecto. En la ventana que se muestra, seleccionar de la lista de categorías (a la izquierda) la opción *Java* y de la lista de proyectos la opción *Java Application*.



2. Ingresar ClienteRemoto como nombre de Proyecto, especificar la ubicación del directorio de trabajo, asignar el nombre `uy.edu.ort.cliente.ClienteRemoto` a la clase main del proyecto.



3-Crear un Cliente Web Services con la WSDL generada anteriormente

http://localhost:8080/PersonaWS/PersonaWS?wsdl en el paquete *uy.edu.ort.cliente.ws*.

4-Codigo ClienteRemoto

```
public class ClienteRemoto {  
  
    public static void main(String[] args) {  
  
        PersonaWS port = new PersonaWS_Service().getPersonaWSPort();  
  
        Persona persona = new Persona();  
  
        persona.setDireccion("Lejos");  
  
        persona.setNombre("Juan");  
  
        persona.setApellido("Perez");  
  
        port.alta(persona);    }  
}
```

NOTA: Si tienen problemas al insertar la persona copien el archivo

mysql-connector-java-5.1.13-bin.jar en el directorio

"C:\Program Files\glassfish3.1.2.2\glassfish\domains\domain1\lib\ext"