Modulo Java
Guia de Ejercicios JDBC - JPA

JDBC

Ejercicio I

1. Abrir el administrador de PostgreSQL elegido
2. Ejecutar el comando para crear esta tabla de ejemplo

```
CREATE TABLE contacto (
   nombre VARCHAR(49),
   email VARCHAR(30),
   telefono VARCHAR(15),
   nacimiento date
   )
```

3. Insertar uno a mas registros en la tabla usando esta linea

```
   INSERT INTO contacto (nombre, telefono, email)
   VALUES ('Juan José', '999-99-99-99', 'juan@jose.com')
```

4. Crear un proyecto Java en eclipse
5. Convertir el proyecto en proyecto maven
6. Agregar la dependencia con los drivers de JDBC

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <scope>runtime</scope>
</dependency>
```

7. Hacer un update del proyecto y verificar que los drivers se encuentren en la seccion de "Dependencias Maven"

1. Crear una clase con un metodo main que se conecte a la base de datos y ejecute la sentencia
2. Verificar que se haya agregado el registro en la base de datos

Ejercicio II

3. Modificar el metodo main de la clase anterior para que ejecute la consulta

```
SELECT * FROM CONTACTO
```

4. Iterar sobre los resultados e imprimir por consola los nombre y mails de los contactos.

JPA - Hibernate

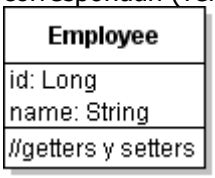
Ejercicio III

1. Agregar las siguientes dependencias al proyecto

```
<dependency>
        <groupId>javax.persistence</groupId>
        <artifactId>javax.persistence-api</artifactId>
        <version>2.2</version>
</dependency>
```

```
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<scope>runtime</scope>
</dependency>
```

Crear la clase Employee, con las annotations que correspondan (ver slides)



2. Dentro de la carpeta src crear la carpeta META-INF (respetar las mayusculas)
3. Crear el archivo persistence.xml dentro de la carpeta META-INF con el siguiente contenido

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/persistence" version="1.0">
<persistence-unit transaction-type="RESOURCE_LOCAL" name="DemoJPA">

<provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
<!-- <provider>org.hibernate.ejb.HibernatePersistence</provider> -->

<properties>

<property name="hibernate.connection.username" value="root"/>
<property name="hibernate.connection.driver_class" value="com.mysql.jdbc.Driver"/>
<property name="hibernate.connection.password" value="1234"/>
<property name="hibernate.connection.url" value="jdbc:mysql://localhost:3306/elca"/>
<property name="hibernate.cache.provider_class"
value="org.hibernate.cache.NoCacheProvider"/>
<property name="hibernate.hbm2ddl.auto" value="update"></property>
<!-- <property name="hibernate.hbm2ddl.auto" value="create-drop"></property> -->
</properties>
</persistence-unit>
```

```
</persistence>
```

4. Modificar los datos de conexión **driver**, **url**, **user** y **password** según su base de datos
5. Dentro de la clase que contiene el metodo main agregar el atributo

```
private static EntityManagerFactory managerFactory = Persistence
      .createEntityManagerFactory("ejsHibernate");
```

Notar que el nombre coincide con el nombre de la persistent unit en el XML.
Hacemos esto así porque queremos tener exactamente un EntityManagerFactory en nuestra aplicación.

6. Dentro del metodo main , crear un EntityManager

```
EntityManager em = managerFactory.createEntityManager();
```

7. Luego crear una instancia de Employee
8. Agregar el codigo para persistir su instancia en la base de datos

```
EntityTransaction tran = em.getTransaction();
tran.begin();
em.persist(tuInstanciaDeEmployee);
tran.commit();
em.close();
```

9. Ejecutar el codigo y verificar que se creo la tabla y se inserto un registro en la base de datos (tambien se va a mostrar lo que se ejecuta por consola)

Ejercicio IV

Crear y persistir las siguientes clases:

Usuario, con los siguientes atributos:

```
private Long id;
private String firstname;
private String lastname;
private String username;
private String password;
private String email;
private int ranking;
private boolean admin;
private Address domicilioParticular;
private Address domicilioTrabajo;
```

Address, con los siguientes atributos:

```
private String calle;
private int numero;
private String codigoPostal;
private String ciudad;
```

Implementar un TestCase de JUnit para

a.- Crear y Persistir un usuario (con dos direcciones)

b.- Cerrar el em, y consultar la BD en otra transacción para que nos devuelva el usuario que persistimos:

Hint: em.createQuery("select u from User u").getSingleResult();
c.- Supongamos que conocemos el id del objeto que buscamos. Qué método de los de em convendría usar?

Ejercicio V

Convertir en persistentes las clases 'BankingTransaction', 'CashTransaction' y 'StockTransaction'. Utilizar sucesivamente las tres estrategias de herencia, para ver cómo quedan las tablas en la BD (Cambiando unas unas annotations)

```
public abstract class BankingTransaction {
	private Long id = null;
	private String txType = null;
	private Date txDate = null;
	private String txDescription = null;
	private Double txFee = null;
	//getters y setters
}

public class CashTransaction extends BankingTransaction {
	private boolean isDeposit = false;
	private Double moneyAmount = 0.0;
	//getters y setters
}

public class StockTransaction extends BankingTransaction {
	private boolean isSale = false;
	private String stockSymbol = null;
	private String companyName = null;
	private Integer numShares = 0;
	private Double pricePerShare = 0.0;
	//getters y setters
}
```