Proyecto de Desarrollo

Actividad N° 4

Framework: Hibernate

Integrantes:

Maglione, Jorge

Santinelli, Paola

Tejera, Guillermo

Framework: Hibernate

Hibernate es una herramienta de mapeo objeto-relacional (ORM), que se utiliza en la plataforma Java y facilita el mapeo de atributos entre una base de datos relacional y el modelo de objetos de una aplicación mediante archivos declarativos (XML) o anotaciones (@) en los beans de las entidades que permiten estableces estas relaciones.

¿Para qué se usa?

Se utiliza para solucionar el problema que se presenta entre la diferencia entre los dos modelos de datos coexistentes en una aplicación. Estos modelos son el usado en la memoria de la computadora (orientación a objetos) y el usado en las bases de datos (modelo relacional).

¿Cómo se usa?

Permite a los desarrolladores detallar cómo es su modelo de datos, qué relaciones existen y qué forma tienen.

Con esta información hibernate le permite a la aplicación manipular los datos en la base de datos operando sobre objetos, con todas las características de la Poo, éste convertirá los datos entre los tipos utilizados por java y los definidos por SQL.

Hibernate genera las sentencias sql y libera a los desarrolladores del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todos los motores de bases de datos con un ligero incremento en el tiempo de ejecución.

Uso de objetos

Archivo de configuración hibernate (.xml): Representa un archivo de configuración o propiedades requeridas y proporciona dos componentes claves:

- conexión de base de datos
- configuración de mapeo clase

Objetivo sessionfactory: objeto de configuración se utiliza para crear un objeto sessionfactory para la aplicación que utiliza el archivo de configuración suministrada, y permite un objeto session ser ejecutado. Es una sola instancia del objeto.

Objeto session: se utiliza para obtener una conexión física con una base de datos. Los objetos de sesión no deben mantenerse abiertas durante mucho tiempo, ya que no suelen ser seguros para subprocesos y deben ser creados y destruidos, según sea necesario.

Objeto transaction: este es un objeto opcional y las aplicaciones de hibernate puede optar por no utilizar esta interfaz, en lugar gestionar las transacciones en su propio código de la aplicación.

Objeto query: objetos de consulta utilizan sql o hibernate query language (hql) cadena para recuperar datos de la base de datos y crear objetos. Una instancia de consulta se utiliza para enlazar los parámetros de consulta, limitar el número de resultados devueltos por la consulta, y finalmente, para ejecutar la consulta.

Objeto criteria: los objetos criteria se utilizan para crear y ejecutar consultas con objetos y recuperar objetos.

Cualidades

Está diseñado para ser flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente. También tiene la funcionalidad de crear la base de datos a partir de la información disponible.

Lenguaje

Hibernate ofrece un lenguaje de consulta de datos llamado hql (hibernate query language), al mismo tiempo que una api para construir las consultas de lo que quiere hacer.
Para java puede ser utilizado en aplicaciones java independientes o en aplicaciones JAVA EE, mediante el componente hibernate annotatios que implementa el estándar jpa, que es parte de esta plataforma.

Sesiones

Para poder utilizar los mecanismos de Hibernate se debe inicializar el entorno y obtener un objeto Session utilizando la clase SessionFactory de Hibernate, de la siguiente manera:

> Se inicializa el entorno

```
Configuration cfg = new Configuration().configure();
```

> Se crea la "session factory"

```
SessionFactory SessionFactory = configuration.buildSessionFactory(serviceRegistry);
```

> Se abre la sesión con la base de datos.

```
Session session = sessionfactory.openSession();
```

> Cuando se finaliza debemos cerrar las sesiones abiertas.

- session.close ();
- sessionFactory.close();

Ejemplo: Código perteneciente al Hibernate.cfg.xml

Información mínima que debe tener el archivo .xml para poder usar hibernet

```
<?xml version='1.0' encoding='utf-8’?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 2.0//EN“
        "http://hibernate.sourceforge.net/hibernate-configuration-2.0.dtd">
<hibernate-configuration>
        <session-factory>
            <property name="hibernate.dialect">org.hibernate.dialect.PostgreSQLDialect</property>
            <property name="hibernate.connection.driver_class">org.postgresql.Driver</property>
            <property name="hibernate.connection.url">jdbc:postgresql://localhost:5432/nom_BD</property>
            <property name="hibernate.connection.username">postgres</property>
            <property name="hibernate.connection.password">password</property>
                <mapping class="proyecto.act2modelo.Usuario"/>
                <mapping class="proyecto.act2modelo.Documento"/>
                <mapping class="proyecto.act2modelo.Etiqueta"/>
        </session-factory>
</hibernate-configuration>
```

Instalación

Trabajando en un proyecto Maven lo único que tendríamos que hacer es agregar las siguientes líneas de código a nuestro archivo **pom.xml**:

```
<dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-core</artifactId>
        <version> aquí va la versión que se desea usar </version>
</dependency>
```

Una vez hecho esto, se tiene que ejecutar el proyecto para que se instale automáticamente.

Ejemplo de los artículos que pertenecen a ciertos rubros.

```
import javax.persistence.*;

@Entity
@Table(name = "articulos")
public class Articulo {
   @Id
   @GeneratedValue(strategy =
GenerationType.IDENTITY)
   @Column(name = "id_articulo")
   private int id;

   @ManyToOne()
   @JoinColumn(name = "id_rubro")
   private Rubro rubro;

   @Column(name = "nombre")
   private String nombre;

   @Column(name = "descripcion")
   private String descripcion;

   @Column(name = "precio_unit")
   private float precioUnitario;

   @Column(name = "stock_actual")
   private int stockActual;

   @Column(name = "stock_minimo")
   private int stockMinimo;


   public Articulo() {
   }
```

```
import javax.persistence.*;

@Entity
@Table(name = "rubros")
public class Rubro {

   @Id
   @GeneratedValue(strategy =
GenerationType.IDENTITY)
   @Column(name = "id_rubro")
   private int id;

   @Column(name = "nombre")
   private String nombre;

   @Column(name = "descripcion")
   private String descripcion;

   @OneToMany(mappedBy = "rubro", cascade =
CascadeType.PERSIST)
   private List<Articulo> articulos;

   public Rubro() {
   }
```