ANEXO TEMA 3.- HERRAMIENTAS DE MAPEO OBJETO RELACIONAL. HQL.

Hibernate Framework viene con un potente lenguaje de consulta orientado a objetos – Hibernate Query Language (HQL). Hibernate Query Language es igual que SQL (Structured Query Language), pero en lugar de tablas se trata de clases y en lugar de columnas trata de propiedades o atributos de clase.

Las consultas HQL son consultas independientes de la base de datos porque las consultas HQL se convierten internamente en consultas SQL específicas de la base de datos utilizando la clase Dialect mencionada en el archivo hibernate-cfg.xml.

Ventajas de HQL

- Independiente de la base de datos
- Fácil de aprender para el programador de Java

Sintaxis de HQL

La mayoría de la sintaxis y características de HQL son muy similares a SQL. Una consulta HQL puede consistir en los siguientes elementos:

- Cláusulas
- Funciones agregadas
- Subconsultas

a) Cláusulas

Algunas de las cláusulas comúnmente soportadas en HQL son:

- from
- as
- select
- where
- order by
- group by
- update
- delete
- insert

Cláusula from:

Es la consulta de Hibernate más importante. Esta cláusula se utiliza para cargar un objeto persistente completo en la memoria.

Sintaxis:

```
String hql = "FROM Employee";
Query query = session.createQuery(hql);
List results = query.list();
```

Cláusula as:

La cláusula as permitirá asignar alias a las clases.

Sintaxis:

```
String hql = "FROM Employee AS E";
Query query = session.createQuery(hql);
List results = query.list();
```

La palabra clave “as” es opcional; también puede especificar el alias directamente después del nombre de la clase, como se menciona a continuación:

Sintaxis:

```
from Employee e
```

Cláusula select:

La cláusula select se utiliza para obtener algunas propiedades de los objetos en lugar del objeto completo.

Sintaxis:

```
String hql = "SELECT e.firstName FROM  Employee e";
Query query = session.createQuery(hql);
List results = query.list();
```

Cláusula where:

La cláusula where se utiliza para recuperar los objetos específicos de la base de datos.

Sintaxis:

```
String hql = "FROM Employee e WHERE e.id = 100";
Query query = session.createQuery(hql);
List results = query.list();
```

Cláusula order by:

Esta cláusula se utiliza para clasificar los resultados de nuestra consulta HQL. Podemos ordenar el resultado por cualquier propiedad de los objetos del conjunto de resultados, ya sea ascendente (ASC) o descendente (DESC).

Sintaxis:

```
String hql = "FROM Person P WHERE P.id > 10 ORDER BY P.salary DESC";
Query query = session.createQuery(hql);
```

Cláusula group by:

Obtiene información de la base de datos y la agrupa basándose en un valor de un atributo y normalmente utiliza el resultado para incluir un valor agregado. Devuelve valores agregados que se agrupan por cualquier propiedad de una clase o componente devuelto.

Sintaxis:

```
String hql = "SELECT SUM(P.salary), P.firtName FROM Person P” +
"GROUP BY P.firstName";
```

```
Query query = session.createQuery(hql);
List results = query.list();
```

Cláusula de actualización:

Se puede utilizar para actualizar una o más propiedades de uno o más objetos.

Sintaxis:

```
String hql = "UPDATE Employee set salary =: salary "+
"WHERE id =:empId";
Query query = session.createQuery(hql);
query.setParameter("salary", 10000);
query.setParameter("empId", 10);
int result = query.executeUpdate();
System.out.println("Rows Affected: " + result);
```

Cláusula DELETE:

La cláusula DELETE se puede utilizar para eliminar uno o más objetos.

Sintaxis:

```
String hql = "DELETE FROM Employee "+
"WHERE id = :empId";
Query query = session.createQuery(hql);
query.setParameter("empId", 10);
int result = query.executeUpdate();
System.out.println("Rows Affected: " + result);
```

Cláusula Insert:

Insertar en cláusula de soporte de lenguaje de consulta de hibernación donde los registros se pueden insertar de un objeto a otro objeto.

Sintaxis:

```
String hql = "INSERT INTO Person(firstName, lastName, salary)"  +
"SELECT firstName, lastName, salary FROM old_person";
Query query = session.createQuery(hql);
int result = query.executeUpdate();
System.out.println("Rows Affected: " + result);
```

b) Funciones agregadas

avg (...), sum (...), min (...), max (...), count(*), count(...), count(distinct ...), count(all...)

c) Subconsultas (subqueries)

Las subconsultas no son otra cosa que una consulta dentro de otra consulta. Hibernate admite subconsultas si la base de datos subyacente lo admite.

Pasos para utilizar HQL

Paso 1 :

Escriba la consulta HQL según el requisito.

Por ejemplo:

```
from Employee
```

Paso 2 :

Obtener una instancia de org.hibernate.Query pasando HQL query.

Por ejemplo:

```
Query q = session.createQuery(“from Employee”);
```

Paso 3 :

Si la consulta contiene el parámetro establecer esos valores.

Por ejemplo:

```
query.setParameter(index,value);
```

Paso 4: Ejecutar la consulta.

Necesitamos llamar al método list () para ejecutar query select en base de datos, devolverá java.util.List. Para actualizar y eliminar consultas necesitamos llamar a executeUpdate () method.

Por Ejemplo:

```
query.list(); //Ejecuta el query de seleccion
query.executeUpdate();//Ejecuta queries de update/delete
```

Paso 5: Iterar la lista

Necesitamos iterar la colección List.

Por ejemplo:

```
Query q = session.createQuery("from Employee");
List empList= q.list();
for(Employee employee : empList) {
      System.out.println(employee.getEmpno());
      System.out.println(employee.getUserName());
}
```

Query:

Query es una interfaz proporcionada en el paquete org.hibernate. Si queremos ejecutar ejecutar una consulta HQL en una base de datos, necesitamos crear un objeto de consulta.

Para obtener el objeto de consulta, necesitamos llamar al método createQuery () en la interfaz de sesión.

```
Query q = session.createQuery(“from Employee”);
```

La interfaz de consulta proporciona muchos métodos. A continuación se enumeran los más utilizados:

- **Public int executeUpdate ()** : se utiliza para ejecutar las consultas de actualización y eliminación.
- **Public List lista ()** : devuelve el resultado de la consulta de selección como una lista.
- **Public Query setFirstResult (int rowno)** : especifica el numero de fila de donde se recuperará el registro.
- **Public Query setMaxResult (int rowno)** : especifica el número de registros que se recuperarán de la tabla.
- **Public Query setParameter (int position, Object value)** : establece el valor en el parámetro de consulta de estilo JDBC.
- **Public Query setParameter (String name, Object value)** : establece el valor en un parámetro de consulta con nombre.

Obtención de objetos completos.

Si queremos seleccionar un Objeto Completo de la base de datos, necesitamos usar la referencia de clase POJO en lugar de * al escribir la consulta.Exemplo:

En SQL

```
select * from Employee
```

En HQL

```
select e from Employee e
```

o

```
from Employee e
```

Obtención de objetos parciales.

Si queremos cargar el Objeto Parcial desde la base de datos entonces necesitamos reemplazar los nombres de las columnas con nombres de variables de clase POJO. Ejemplo:

En SQL

```
select empid,name from Employee
```

Nota: empid, name son las columnas, Employee es la tabla.

En HQL

```
select e.empId,e.name from Employee e
```

También es posible cargar o seleccionar el objeto de la base de datos pasando valores de tiempo de ejecución a la consulta, en este caso podemos usar "?" o :etiqueta en un comando HQL, el número de índice de"?" se iniciará a partir de cero y no uno.

Ejemplo:

```
select e from Employee e where e.name=?
select e from Employee e where e.name=:param
from Employee e where e.name=?
from Employee e where e.name=:param
```

Enlace de parámetros de Hibernate (Binding)

La vinculación de parámetros (binding) es el proceso de enlazar una variable Java con una instrucción HQL. Es similar a cualquier lenguaje SQL normal.

Consulta simple de selección, sin Binding

```
String name="Mukesh";
Query query = session.createQuery("from Employee where employeeName = '"+name+"' ");
```

Hay dos tipos de bindings en Hibernate, Posicionales y Nombrados. Hibernate recomienda utilizar los parámetros nombrados, ya que es más flexible y potente comparado con el parámetro posicional.

Selección de campos en HQL:

La cláusula select se utiliza para obtener algunas propiedades de los objetos en lugar del objeto completo. La sintaxis "from ClassName" devuelve una lista del tipo de clase con todos los campos de la misma. En su lugar, se puede seleccionar uno o más campos de forma selectiva de la propia tabla. Si está seleccionando una sola columna en el resultado, devolverá una Lista de ese tipo de campo.

Sintaxis:

```
Query query = session.createQuery("Select name from User");
List<String> names = (List<String>) query.list();
for (String n : names) {
System.out.println(n);
}
```

Selección de varios campos en HQL

Si selecciona más de un campo en una instrucción HQL, query.list devolverá una lista de listas, una lista para cada campo.

Sintaxis:

```
Consulta query = session.createQuery ("Seleccionar id, nombre del usuario");
List <Object []> users = (Lista <Object []>) query.list ();
for (objeto [] user : users) {
         Integer id = (Integer) usuario [0];
         System.out.println (id);
         String name = (String) user [1];
         System.out.println (name);
}
```