3. El lenguaje de consulta HQL

HQL son las siglas de *Hibernate Query Language*. Es un lenguaje inspirado en SQL. Tiene sentencias SELECT, INSERT, UPDATE y DELETE, similares a las de SQL.

El IDE Eclipse tiene un intérprete de HQL que permite ejecutar directamente las sentencias, tal y como vimos durante el proceso de instalación de Hibernate.

3.1. La interfaz Query

La API de Hibernate permite utilizar sentencias HQL mediante la interfaz **Query**. Aunque esto abre muchas posibilidades, hay que tener cuidado con las operaciones de actualización y borrado masivo. En la guía de usuario de Hibernate se dice lo siguiente:

Debe actuarse con precaución cuando se ejecutan operaciones masivas de actualización o borrado porque pueden provocar inconsistencias entre la base de datos y las entidades dentro del contexto de persistencia activo. En general, las operaciones masivas de actualización y borrado solo deberían realizarse dentro de una transacción en un nuevo contexto de persistencia, o antes de recuperar o acceder a entidades cuyo estado pueda verse afectado por dichas operaciones.

Se puede obtener una instancia de *Query* a partir de una instancia de *Session*, mediante el método **createQuery(String sentenciaHQL)**. Hay que tener en cuenta que *createQuery* devuelve una instancia de la interfaz **org.hibernate.query.Query**. No hay que confundirla con la interfaz org.hibernate.Query, que está obsoleta (*deprecated*).

Podemos consultar la documentación de esta interfaz en el siguiente enlace:

https://docs.jboss.org/hibernate/orm/5.6/javadocs/org/hibernate/query/Query.html

El funcionamiento de la interfaz *Query* es equivalente al de *PreparedStatement* de JDBC. Una *Query* puede tener parámetros que se pueden identificar por nombre además de por posición. La interfaz Query realmente se llama, hablando con propiedad, Query<R>, es decir, es una interfaz genérica con un parámetro de tipo R, que es el tipo de los objetos con los que trabaja la Query.

Los principales métodos de esta interfaz son:

- **List<R> getResultList():** devuelve la lista de resultados de una consulta HQL.
- **R getSingleResult():** devuelve un único resultado para una consulta. Este método puede lanzar varias excepciones, entre ellas NoResultException cuando la consulta no devuelve ningún resultado y NonUniqueResultException cuando devuelve más de uno.
- **int executeUpdate():** ejecuta la sentencia de tipo UPDATE o DELETE y devuelve el número de objetos afectados.
- **Query<R> setParameter(…):** asigna un valor a un parámetro de la sentencia.

- **<P> Query <R> setParameterList(…):** este método tiene diversas variantes que permiten asignar a un parámetro una lista de valores. Es útil cuando se especifica en una sentencia de HQL una restricción del tipo valor IN :listaValores.
- **Query<R> setReadOnly(boolean readOnly):** especifica que los resultados recuperados serán solo para lectura, y no para modificación.
- **Query<R> setFirstResult(int posInicial):** especifica el primer resultado a recuperar de entre los obtenidos por la consulta.
- **Query<R> setMaxResults(int maxResult):** especifica el máximo de resultados a recuperar.

3.2. Sentencias SELECT

Podemos obtener una *Query* utilizando la *Session* actual. Para realizar una consulta utilizaremos la función *createQuery("consulta")* de la siguiente forma:

```
Query q = sesion.createQuery("from Depto");
```

Para utilizar consultas preparadas, podemos utilizar el método *setParameter(nombreParametro, valor).* Como en este caso no se van a modificar los objetos persistentes recuperados, también utilizamos *setReadOnly().* Por ejemplo:

```
Query q = sesion.createQuery("from Depto where nomDepto like :nombre").setParameter("nombre", "I+D").setReadOnly(true);
```

Para recuperar los resultados, podemos utilizar *getResultList():*

```
List<Depto> listaDepartamentos = (List<Depto>) q.getResultList();
```

Podemos utilizar el método *getSingleResult()* si solo queremos recuperar un objeto. Por ejemplo, si queremos recuperar el empleado con mayor sueldo. Si hubiera más de uno, o si no existieran datos profesionales para ningún empleado, se lanzaría una excepción y podríamos mostrar un mensaje de error específico.

También se pueden recuperar objetos de una consulta y modificarlos, pero hay que incluirlos dentro de una transacción.

3.3. Sentencias INSERT, UPDATE y DELETE

Para ejecutar cualquiera de estas sentencias podemos utilizar el método **query.executeUpdate().**

3.4. Consultas con SQL

Con Hibernate también se pueden realizar consultas directamente en SQL. Conviene, de todas formas, no abusar de esta posibilidad y recurrir a ella solo en casos muy justificados. Por ejemplo, cuando supone un aumento del rendimiento importante, cuando se requiere una ordenación particular de los resultados, o para consultas muy complejas que no es posible realizar en HQL. También si ya se dispone de una consulta SQL y utilizarla ahorra tiempo.

Podemos introducir consultas SQL utilizando el método **createNativeQuery().**

EJERCICIOS PROPUESTOS

1. Crea una clase *SentenciasHQL* e implementa un método que recupere el número y nombre de los empleados que sean comerciales.
2. Implementa un método que devuelva al empleado con mayor sueldo.
3. Implementa un método que modifique el sueldo de todos los manager a una cantidad determinada.
4. Implementa un método que borre los empleados de un departamento determinado.