SESIONS HIBERNATE

Las interfaces principales de programación que forman parte de Hibernate son los siguientes:

org.hibernate.SessionFactory se utiliza básicamente para obtener una instancia de session, y puede ser visto como una analogía con el mecanismo de agrupación de conexiones. Se trata de hilos de seguridad, como todos los hilos de aplicación puede utilizar un SessionFactory único (siempre que Hibernate utiliza una sola base de datos). Esta interfaz se configura mediante el archivo de configuración, que determina la asignación de archivo a ser cargado.

org.hibernate.Session proporciona un único hilo que determina la conversación entre la aplicación y la base de datos. Esto es análogo a una conexión específica. Una instancia de Session es "poco pesada" y su creación y destrucción es muy "barata". Esto es importante, ya que nuestra aplicación necesitará crear y destruir sesiones todo el tiempo, quizá en cada petición. Puede ser útil pensar en una sesión como en una cache o colección de objetos cargados (a o desde una base de datos) relacionados con una única unidad de trabajo.

org.hibernate.Transaction proporciona un único objeto hilo que se extiende a través de la solicitud y determina una unidad atómica de trabajo.

necesitamos hacernos una pequeña clase, *HibernateUtil*, que nos de las conexiones con la base de datos. (crea una instancia de sesión)

```
package com.chuidiang.ejemplos.hibernate.ejemplo1;

import java.io.File;

import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateUtil {
    private static final SessionFactory sessionFactory;
    static {
        try {
            // Create the SessionFactory from hibernate.cfg.xml
            sessionFactory = new Configuration().configure().buildSessionFactory();
        } catch (Throwable ex) {
            // Make sure you log the exception, as it might be swallowed
            System.err.println("Initial SessionFactory creation failed." + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}
```

Amb aquest exemple guardariem a la base de dades l'objecte.

```
Session sesión=HibernateUtil.getSessionFactory().openSession();
     sesion.beginTransaction();
     sesion.save(alumno1);
     sesion.getTransaction().commit();
     sesion.close();
```

Generator	Description
increment	It generates identifiers of type long, short or int that are unique only when no other process is inserting data into the same table. It should not the used in the clustered environment.
identity	It supports identity columns in DB2, MySQL, MS SQL Server, Sybase and HypersonicSQL. The returned identifier is of type long, short or int.
sequence	The sequence generator uses a sequence in DB2, PostgreSQL, Oracle, SAP DB, McKoi or a generator in Interbase. The returned identifier is of type long, short or int
hilo	The hilo generator uses a hi/lo algorithm to efficiently generate identifiers of type long, short or int, given a table and column (by default hibernate_unique_key and next_hi respectively) as a source of hi values. The hi/lo algorithm generates identifiers that are unique only for a particular database. Do not use this generator with connections enlisted with JTA or with a user-supplied connection.
seqhilo	The seqhilo generator uses a hi/lo algorithm to efficiently generate identifiers of type long, short or int, given a named database sequence.
uuid	The uuid generator uses a 128-bit UUID algorithm to generate identifiers of type string, unique within a network (the IP address is used). The UUID is encoded as a string of hexadecimal digits of length 32.
guid	It uses a database-generated GUID string on MS SQL Server and MySQL.
native	It picks identity, sequence or hilo depending upon the capabilities of the underlying database.
assigned	lets the application to assign an identifier to the object before save() is called. This is the default strategy if no <generator> element is specified.
select	retrieves a primary key assigned by a database trigger by selecting the row by some unique key and retrieving the primary key value.
foreign	uses the identifier of another associated object. Usually used in conjunction with a <one-to-one> primary key association.