**Hibernate configuration file holds database specific data like table name , database name , username , password , type of database, connection pool and other details.**

To define a Hibernate Configuration info a resource file named **hibernate.cfg.xml** is used.

<?xml version='1.0' encoding='utf-8'?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

**<session-factory>**

<property name="hibernate.connection.driver\_class">com.mysql.jdbc.Driver</property>

<property name="hibernate.connection.url">jdbc:mysql://192.168.10.13:3306/data

</property>

<property name="hibernate.connection.username">root</property>

<property name="hibernate.connection.password">root</property>

<property name="hibernate.connection.pool\_size">10</property>

<property name="show\_sql">true</property>

<property name="dialect">org.hibernate.dialect.MySQLDialect</property>

<mapping class="/worldclock/Employee" />

**</session-factory>**

</hibernate-configuration>

Using this configuration file various of information is provided to the Hibernate such as :

* connection.driver\_class : This property element is used to provide the name of the driver class.
* connection.url : This property element is used to provide the url for the database connection.
* connection.username : This property element is used to provide the username by which you are logging to the database.
* connection.password : This property element is used to provide the password by which you are logging to the database.
* connection.pool\_size : This property element configures the number of connections of connection pool of the built-in Hibernate.
* dialect : This property defines the particular SQL variant with which the Hibernate will carry on a conversation. Use of this property is necessary when there are multiple databases are targeted by an application.

**Session Factory**

The core purpose of org.hibernate.SessionFactory is to get org.hibernate.Session instances. The org.hibernate.SessionFactory is a thread-safe global object which is instantiated once and and threads servicing client requests obtain Session instances from this factory.

Previously, we get the Session instance by instantiating SessionFactory, But in the Hibernate 4, the method buildSessionFactory() from the type Configuration is deprecated.

In the below example, you will learn how to get the Session instance by instantiating SessionFactory instance in Hibernate 4 :

Configuration configuration =

new Configuration().configure(HibernateUtil.class.getResource("/hibernate.cfg.xml"));

StandardServiceRegistryBuilder serviceRegistryBuilder =

new StandardServiceRegistryBuilder();

serviceRegistryBuilder.applySettings(configuration.getProperties());

ServiceRegistry serviceRegistry = serviceRegistryBuilder.build();

SessionFactory sessionFactory = configuration.buildSessionFactory(serviceRegistry);

Hibernate Generator tag

<generator> is an optional child element of the element <id>. In <generator> a class name is to be assigned using which unique identifier is generated for the new record at the time of new record is saved.

EX:

<class name="worldclock.Employee" table="employee">  
<id name="empId" type="long" column="Id" >  
<generator class="assigned"/>  
</id>

Generators in hibernate <generator />

Generator classes are used to generate the 'identifier' for a persistent object. i.e., While saving an object into the database, the generator informs to the hibernate that, how the primary key value for the new record is going to generate.

List of Hibernate built-in generator classes

|  |  |
| --- | --- |
| 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. |

**Hql Hibernate query language**

Hibernate Query Language (HQL) is same as SQL (Structured Query Language) but it doesn't depends on the table of the database. Instead of table name, we use class name in HQL. So it is database independent query language.

Query query=session.createQuery("from Emp");//here persistent class name is Emp

Query q=session.createQuery("update User set name=:n where id=:i");

Query query=session.createQuery("delete from Emp where id=100");

session.save(emp)//directly saves to db

**Query Interface**

It is an object oriented representation of Hibernate Query. The object of Query can be obtained by calling the createQuery() method Session interface.

Ex : update

Transaction tx=session.beginTransaction();

Query q=session.createQuery("update User set name=:n where id=:i");

q.setParameter("n","Udit Kumar");

q.setParameter("i",111);

int status=q.executeUpdate();

System.out.println(status);

tx.commit();

**Sql dialects**

For connecting any hibernate application with the database, you must specify the SQL dialects. There are many Dialects classes defined for RDBMS in the org.hibernate.dialect package.

Ex : mysql : org.hibernate.dialect.MySQLDialect

Ex : oracle : org.hibernate.dialect.OracleDialect