HIBERNATE - SESSIONS

A Session is used to get a physical connection with a database. The Session object is lightweight and designed to be instantiated each time an interaction is needed with the database. Persistent objects are saved and retrieved through a Session object.

The session objects should not be kept open for a long time because they are not usually thread safe and they should be created and destroyed them as needed. The main function of the Session is to offer create, read and delete operations for instances of mapped entity classes. Instances may exist in one of the following three states at a given point in time:

- **transient:** A new instance of a a persistent class which is not associated with a Session and has no representation in the database and no identifier value is considered transient by Hibernate.
- **persistent:** You can make a transient instance persistent by associating it with a Session. A persistent instance has a representation in the database, an identifier value and is associated with a Session.
- **detached:** Once we close the Hibernate Session, the persistent instance will become a detached instance.

A Session instance is serializable if its persistent classes are serializable. A typical transaction should use the following idiom:

```
Session session = factory.openSession();
Transaction tx = null;
try {
   tx = session.beginTransaction();
   // do some work
   ...
   tx.commit();
}
catch (Exception e) {
   if (tx!=null) tx.rollback();
   e.printStackTrace();
}finally {
   session.close();
}
```

If the Session throws an exception, the transaction must be rolled back and the session must be discarded.

Session Interface Methods:

There are number of methods provided by the **Session** interface but I'm going to list down few important methods only, which we will use in this tutorial. You can check Hibernate documentation for a complete list of methods associated with **Session** and **SessionFactory**.

S.N.	Session Methods and Description
1	**Transaction beginTransaction** Begin a unit of work and return the associated Transaction object.
2	**void cancelQuery** Cancel the execution of the current query.
3	**void clear**

Completely clear the session.

4 **Connection close**

End the session by releasing the JDBC connection and cleaning up.

5 **Criteria createCriteria***ClasspersistentClass*

Create a new Criteria instance, for the given entity class, or a superclass of an entity class.

6 **Criteria createCriteria***StringentityName*

Create a new Criteria instance, for the given entity name.

7 **Serializable getIdentifier***Objectobject*

Return the identifier value of the given entity as associated with this session.

8 **Query createFilter***Objectcollection, StringqueryString*

Create a new instance of Query for the given collection and filter string.

9 **Query createQuery***StringqueryString*

Create a new instance of Query for the given HQL query string.

10 **SQLQuery createSQLQuery***StringqueryString*

Create a new instance of SQLQuery for the given SQL query string.

11 **void delete***Objectobject*

Remove a persistent instance from the datastore.

12 **void delete***StringentityName, Objectobject*

Remove a persistent instance from the datastore.

13 **Session get***StringentityName, Serializableid*

Return the persistent instance of the given named entity with the given identifier, or null if there is no such persistent instance.

14 **SessionFactory getSessionFactory**

Get the session factory which created this session.

15 **void refresh***Objectobject*

Re-read the state of the given instance from the underlying database.

16 **Transaction getTransaction**

Get the Transaction instance associated with this session.

17 **boolean isConnected**

Check if the session is currently connected.

18 **boolean isDirty**

Does this session contain any changes which must be synchronized with the database?

19 **boolean isOpen**

Check if the session is still open.

20 **Serializable save*Objectobject***

Persist the given transient instance, first assigning a generated identifier.

21 **void saveOrUpdate*Objectobject***

Either save*Object* or update*Object* the given instance.

22 **void update*Objectobject***

Update the persistent instance with the identifier of the given detached instance.

23 **void update*StringentityName, Objectobject***

Update the persistent instance with the identifier of the given detached instance.