**Hibernate:**

**What is Hibernate & ORM tool?**

**Hibernate -**

* It is Java Frame work that simplifies development of Java Application to interaction with database.
* It creates SQL (Structured Query Language) queries at runtime according to database.
* It provide automatic table creation feature, relationship mapping like IS-A relationship (Default, Single Table, Joint, table per Class) & HAS-A relationship (one to one, one to many, many to one, many to many).
* It provide primary key auto increment feature.
* It converts all checked exception into uncheck exception.
* It provides cache mechanism.
* It provide HQL (Hibernate Query Language) query and Criteria API.

**ORM-**

* It is a programming technique that maps the object to the data stored in the database.
* It overcomes the mismatch between OOP language & database.
* It reduces developer’s efforts, time and cost.
* What is disadvantage of hibernate?
* Can’t perform multiple insert operations.
* Debugging is difficult as compare to JDBC.
* Contain lots of boiler plate code.
* Can’t be used for small type of applications.
* Slow execution as it perform SQL queries at runtime.

**Difference between get( ) & load( )?**

|  |  |
| --- | --- |
| get ( ) | load ( ) |
| Eager Loading | Lazy Loading |
| If value is absent in database then it returns null. | If value is absent in database then  hibernate exception (ObjectNotFoundException) occurs. |
| It always hit database. | It may or may not be hit to database. |

**What is Session Factory?**

* SessionFactory is an interface.
* It contain all DB related property details (pulled from either hibernate.cfg.xml file or hibernate.properties file)
* SessionFactory is a factory for Session objects. (allow to create any number of session objects developer wants)
* We can create one SessionFactory per database in any application.
* It is usually created during application start up.
* It holds 2nd level cache data.

**What are methods present in Session?**

* It wraps the JDBC connection.

(Session)

* It is factory of Transaction, Query and Criteria.
* The session object provides an interface between the application and data stored in the database.
* It holds a first-level cache (mandatory) of data.
* The org.hibernate.Session interface provides methods to insert, update and delete the object.
* (Session Method)
* beginTransaction()
* save()
* update()
* persist()
* delete()
* saveOrUpdate()
* createQuery()
* Create a new instance of Query for the given HQL query string.
* createSQLQuery()
* Create a new instance of SQLQuery for the given SQL query string.
* merge()

**Difference between save( ) & persist( )?**

|  |  |
| --- | --- |
| Save ( ) | Persist ( ) |
| Its return type is Serializable object. | Its return type is void. |
| It can save object within transaction  boundaries and outside boundaries. | It can only save object within the  transaction boundaries. |
| It is only supported by Hibernate. | It is supported by Hibernate and also by  JPA (Java Persistence API). |
| It will create a new row in the table for  detached object. | It will throw persistence exception for  detached object. |

**What are the states of an object? Explain them?**

In Hibernate following are the states of an object: -

**Transient state-**

* The transient state is the initial state of an object.
* Once we create an instance of POJO class, then the object entered in the transient state.
* Here, an object is not associated with the Session. So, the transient state is not related to any database. Hence, modifications in the data don't affect any changes in the database.
* The transient objects exist in the heap memory. They are independent of Hibernate.

**Persistent state-**

* When object associates with the Session, it entered in the persistent state.
* Hence, we can say that an object is in the persistence state when we use save( ), persist( ), update( ), saveOrUpdate( ), lock( ), merge( ).
* Here, each object represents the row of the database table. So, modifications in the data make changes in the database.

**Detached state-**

* Once we either close the session or clear its cache, then the object entered into the detached state.
* As an object is no more associated with the Session, modifications in the data don't affect any changes in the database.
* However, the detached object still has a representation in the database.
* If we want to persist the changes made to a detached object, it is required to reattach the application to a valid Hibernate session.
* To associate the detached object with the new hibernate session, use any of these methods - load(), merge(), refresh(), update() or save() on a new session with the reference of the detached object.

**What is hibernate transaction management?**

* There are some methods we can use within transaction boundaries like update & delete method.
* Also Commit & Rollback need transaction’s object always.
* Commit = to save object into database after save method we need to commit that.
* Rollback = checked to unchecked exception but hibernate does that internally.

**What is cache? What is 1st level cache?**

* 1st level cache is associated with session, it is always enabled.
* When query is fired to get data from database, then that data is stored in local session for that session object.
* Then next time if we try to get same data for same session object then that time it doesn’t hit database it get data from local session.
* For every session object newly query get fired.

**What is 2nd level cache?**

* It is associated with session factory, it need to be enabled.
* It is available globally for all sessions.
* It stores data from database to 1st level cache then to 2nd level cache also, then if we try to get same record then first it see within 1st level cache if data present or not, if not then it goes to 2nd level cache, if there also data not found then it goes to fire query from database.
* For another session object it has its own local session (1st level cache) but 2nd level cache is same as before because it acts globally.

**How to remove particular object from cache?**

* Session has Evict( ) method used for removing particular cache.

**How to clean cache?**

* Session has Clear( ) method to clear all cache.

**In One-to-Many & Many-to-One, how many tables are created by-default and if mapped by is used?**

**By-default it will create 3 tables.**

* For mapped by it will create 2 tables.

**In Many-to-Many, how many tables are created by-default and if mapped by is used?**

**By-default it will create 4 tables**

* For mapped by it will create 3 tables.

**What is Dirty Checking?**

* If we get record & we set again then it is updated without calling update method, its because of dirty checking.
* This can be avoided by using @Immutable annotation.

**What is process for Automatic ID generation from any random number?**

* @SequenceGenerator(name = "mySeqGen", sequenceName = "mySeq", initialValue = 500, allocationSize=1)
* @GeneratedValue(generator = "mySeqGen")

**How to show SQL queries at run time?**

* While specifying hibernate properties, add Show\_SQL property as a true,
* e.g. <property name="show\_sql">true</property>

**What does hbm2ddl does?**

* It validates number of column.

**How to disable 1st level cache?**

* We can’t disable cache but we can clear all cache using clear( ) method of session, then it works as disabled.

**How to enable 2nd level cache?**

* Add 3 rd party jar files. (ehcache)
* Add Annotation below to Entity class. @Cache(usage=CacheConcurrencyStrategy.READ\_WRITE)
* Include two extra lines in settings in ‘HibernateUtil.class’ settings.put(Environment.USE\_SECOND\_LEVEL\_CACHE, "true"); settings.put(Environment.CACHE\_REGION\_FACTORY, "org.hibernate.cache.ehcache.EhCach-eRegionFactory");

**Named Queries-Definition, syntax, advantages?**

* Instead of writing multiple case, HQL query or Native query we can write in a single place which is entity class.
* Because of that HQL query and Native query will not be scattered.
* It is easy to reuse and maintain.
* There are Four Annotations for Named query :
* @NamedQuery.
* @NamedQueries.
* @NamedNativeQuery.
* @NamedNativeQueries.

**Native Queries-Definition, syntax, advantages?**

* You can use native SQL to express database queries if you want to utilize database- specific features.
* Hibernate 3.x allows you to specify handwritten SQL, including stored procedures, for all create, update, delete, and load operations.
* Your application will create a native SQL query from the session with the createSQLQuery() method.
* You need to pass a string containing the SQL query to the createSQLQuery() method.