Hibernate

1. Steps to create hibernate application…?

1**1. Create the persistence POJO class.**

**2. Create Hibernate mapping file.(Student.hbm.xml)**

**3. Create configuration files.(hibernate.cfg.xml)**

**4. Create class that retrieve and store the persistence object.**

**5. Load the jar.**

**6. Run the Hibernate Application.**

Composite key in Hibernate :

1. **Create a class and define two or more field in it that you want to make it unique and extend that class with serializable and override equals method and hash code method, and @classId(EmployeeId.class) and add two Id annotation for both fields. (First Approach).**
2. **Create a class and define two or more field in it and extend that class with serializable and override equals method and hash code method, and add @Embeddable anotation in this new class and define this class in main Employee class.(Second Approach)**

Foreign Key In Hibernate :

1. **Consider a class forming one to many mapping, primary class is customer and contains product list(Product Class).**

* Class Customer{
* @Id
* @Generated Value
* Filed…………
* @OneToMany(targetEntity = Product.class, cascade= CascadeType.ALL)
* @joinColumn(name=”product\_foreginKey”, referenceColumnName =”id”)// This foreign key column
* }

4. Difference between openSession and getCurrentSession…?

1**1. getCurrentSession method returns the session bound to context. Since this object belongs to context of the hibernate, it is ok if we don’t close it. Once the sessionFactory closed, this session object gets closed. It is created new session object if not exists else use same session object which is current hibernate context.**

**2. openSession() method helps in opening new session. You should close this session object once you are done with all database operation. Ans also, you should open a new session object for each request in a multi-threaded environment. It is always create new session.**

5. Difference between get() and load() method…?

|  |  |
| --- | --- |
| get() | load() |
| **1. It is early loading, performance is slow.** | **1. It is lazy loading, and hence performance is faster.** |
| **2. if object not found for the given identifier then it will return null object.** | **2. if object not found then it will throw objectNotFoundException.** |
| **3. After creating object it will direct interact with database.** | **3. It will not interact with database after creating object, it will create proxy object and after calling getName() and other fields then it will interact with database** |
| **4. It should be used if you are not sure that the data exists in the database or not.** | **4. It should be used if you are sure that the data exists in the database.** |

6. What is Hibernate Cache and It’s Type…????

**1. First level cache.**

**i. Hibernate caches query data to make our application faster and improve performance.**

**ii. The idea behind cache is to reduce the number of database queries.**

**iii. Hibernate first level cache is associated with the session object.**

**iv. Hibernate first level catch is enabled by default and there is not way to disable it.**

**v. Still Hibernate provides methods through which we can delete selected object from the cache or clear the cache completely.**

**vi. Any object cached in a section will not be visible to other sessions and when the session is closed, all the cached objects will be lost.**

**2. Second level cache :**

**i. Hibernate second level cache is disable by default, but we can enable it through configuration.**

**ii. Currently EHCache and Infinispan provides implementation for Hibernate second cache and we can use them.**

**iii. Add hibernate-ehcache dependency in your maven project, if it’s not maven then add corresponding jars.**

**<dependency>**

**<groupId>org.hibernate</groupId>**

**<artifactId>hibernate-ehcache</artifactI**

**</dependency>**

**iv. Add below property in hibernate configuration file :**

**<property name**="hibernate.cache.use\_second\_level\_cache"**>**true**</property>**

**<property name**="hibernate.cache.region.factory\_class"**>**org.hibernate.cache.ehcache.EhCacheRegionFactory**</property>**

**<property key**="hibernate.cache.use\_query\_cache"**>**true**</property>**

**Query level cache in not work with first level cache it is always associate with second level cache only.**

**<property name**="net.sf.ehcache.configurationResourceName"**>**/ehcache.xml**</property>**

**v. Create EHCache configuration file, a sample file configuration.xml.**

**vi. Use your cache, Annotate entity beans with @Cache annotation and caching strategy to use**

**@Entity**

**@Table( name = “Address”)**

**@Cache( usage = cacheConcurrencyStrategy.READ\_ONLY, region=”employee”)**

**Public class Address{}**

### 6. What are the states of the object in hibernate…????

There are mainly four states of the Hibernate Lifecycle :

1. **Transient State**
2. **Persistent State**
3. **Detached State**
4. **Removed State**

**1. Transient State :**

**The transient state is the first state of an entity object. When we instantiate an object of pojo classes using the new operator then the object is in the transient state. This object is not connected with any hibernate session. As it is not connected to any Hibernate Session, so this state is not connected to any database table. So, if we make any changes in the data of the POJO Class then the database table is not altered. Transient objects are independent of Hibernate, and they exist in the heap memory.**

**There are two layouts in which transient state will occur as follows:**

1. **When objects are generated by an application but are not connected to any session.**
2. **The objects are generated by a closed session.**

**2. Persistent State :**

**Once the object is connected with the Hibernate Session then the object moves into the Persistent State. So, there are two ways to convert the Transient State to the Persistent State :**

1. **Using the hibernated session, save the entity object into the database table.**
2. **Using the hibernated session, load the entity object into the database table.**

**In this state. each object represents one row in the database table. Therefore, if we make any changes in the data then hibernate will detect these changes and make changes in the database table.**

**Following are the methods given for the persistent state:**

* **session.persist(e);**
* **session.save(e);**
* **session.saveOrUpdate(e);**
* **session.update(e);**
* **session.merge(e);**
* **session.lock(e);**

**3. Detached State :**

**For converting an object from Persistent State to Detached State, we either have to close the session or we have to clear its cache. As the session is closed here or the cache is cleared, then any changes made to the data will not affect the database table. Whenever needed, the detached object can be reconnected to a new hibernate session. To reconnect the detached object to a new hibernate session, we will use the following methods as follows:**

**Following are the methods used for the detached state :**

* **session.detach(e);**
* **session.evict(e);**
* **session.clear();**
* **session.close();**

**4. Removed State :**

**In the hibernate lifecycle it is the last state. In the removed state, when the entity object is deleted from the database then the entity object is known to be in the removed state. It is done by calling the *delete() operation*. As the entity object is in the removed state, if any change will be done in the data will not affect the database table.**

***Note: To make a removed entity object we will call session.delete().***

##### **What is Cascade…??**

Cascade is the feature provided by hibernate to **automatically manage the state of mapped entity** whenever the state of its relationship owner entity is affected.

In other words,  
When **relationship owner is saved**,then **associated mapped entity**should also be **saved** and when relationship owner is **deleted**, associated **mapped entity** should also be **deleted** etc.