**What is Hibernate Framework?**

* Object-relational mapping or ORM is the programming technique to map application domain model objects to the relational database tables. Hibernate is java based ORM tool that provides framework for mapping application domain objects to the relational database tables and vice versa.
* Hibernate provides reference implementation of Java Persistence API, that makes it a great choice as ORM tool with benefits of loose coupling. We can use Hibernate persistence API for CRUD operations. Hibernate framework provide option to map plain old java objects to traditional database tables with the use of JPA annotations as well as XML based configuration.

**What is Java Persistence API (JPA)?**

* Java Persistence API (JPA) provides specification for managing the relational data in applications.

**What is Hibernate SessionFactory and how to configure it?**

* SessionFactory is the factory class used to get the Session objects. SessionFactory is responsible to read the hibernate configuration parameters and connect to the database and provide Session objects. Usually an application has a single SessionFactory instance and threads servicing client requests obtain Session instances from this factory.  
  The internal state of a SessionFactory is immutable. Once it is created this internal state is set. This internal state includes all of the metadata about Object/Relational Mapping.
* SessionFactory also provide methods to get the Class metadata and Statistics instance to get the stats of query executions, second level cache details etc.
* Internal state of SessionFactory is immutable, so it’s thread safe. Multiple threads can access it simultaneously to get Session instances.

**What is Hibernate Session and how to get it?**

* Hibernate Session is the interface between java application layer and hibernate. This is the core interface used to perform database operations. Lifecycle of a session is bound by the beginning and end of a transaction.
* Session provide methods to perform create, read, update and delete operations for a persistent object. We can execute HQL queries, SQL native queries and create criteria using Session object.

**What is difference between openSession and getCurrentSession?**

* Hibernate SessionFactory getCurrentSession() method returns the session bound to the context. But for this to work, we need to configure it in hibernate configuration file. Since this session object belongs to the hibernate context, we don’t need to close it. Once the session factory is closed, this session object gets closed.
* <property name="hibernate.current\_session\_context\_class">thread</property>
* Hibernate SessionFactory openSession() method always opens a new session. We should close this session object once we are done with all the database operations. We should open a new session for each request in multi-threaded environment.

**What is difference between Hibernate Session get() and load() method?**

* Hibernate session comes with different methods to load data from database. get and load are most used methods, at first look they seems similar but there are some differences between them.
* get() loads the data as soon as it’s called whereas load() returns a proxy object and loads data only when it’s actually required, so load() is better because it support lazy loading.
* Since load() throws exception when data is not found, we should use it only when we know data exists.
* We should use get() when we want to make sure data exists in the database.

**What is hibernate caching? Explain Hibernate first level cache?**

* As the name suggests, hibernate caches query data to make our application faster. Hibernate Cache can be very useful in gaining fast application performance if used correctly. The idea behind cache is to reduce the number of database queries, hence reducing the throughput time of the application.
* Hibernate first level cache is associated with the Session object. Hibernate first level cache is enabled by default and there is no way to disable it. However hibernate provides methods through which we can delete selected objects from the cache or clear the cache completely.
* Any object cached in a session will not be visible to other sessions and when the session is closed, all the cached objects will also be lost.

**What are different states of an entity bean?**

* An entity bean instance can exist is one of the three states.
* **1. Transient:** When an object is never persisted or associated with any session, it’s in transient state. Transient instances may be made persistent by calling save(), persist() or saveOrUpdate(). Persistent instances may be made transient by calling delete().
* **2. Persistent:** When an object is associated with a unique session, it’s in persistent state. Any instance returned by a get() or load() method is persistent.
* **3. Detached:** When an object is previously persistent but not associated with any session, it’s in detached state. Detached instances may be made persistent by calling update(), saveOrUpdate(), lock() or replicate(). The state of a transient or detached instance may also be made persistent as a new persistent instance by calling merge()

**How to implement Joins in Hibernate?**

* There are various ways to implement joins in hibernate.
* Using associations such as one-to-one, one-to-many etc.
* Using JOIN in the HQL query. There is another form “join fetch” to load associated data simultaneously, no lazy loading.
* We can fire native sql query and use join keyword.

**What is HQL and what are it’s benefits?**

* Hibernate Framework comes with a powerful object-oriented query language – Hibernate Query Language (HQL). It’s very similar to SQL except that we use Objects instead of table names, that makes it more close to object oriented programming.
* Hibernate query language is case-insensitive except for java class and variable names. So select is the same as select is the same as SELECT, but com.amarjit.model.Employee is not same as com.amatjit.model.EMPLOYEE.
* The HQL queries are cached but we should avoid it as much as possible, otherwise we will have to take care of associations. However it’s a better choice than native sql query because of Object-Oriented approach.

**What is Query Cache in Hibernate?**

* Hibernate implements a cache region for queries resultset that integrates closely with the hibernate second-level cache.
* This is an optional feature and requires additional steps in code. This is only useful for queries that are run frequently with the same parameters. First of all we need to configure below property in hibernate configuration file.
* <property name="hibernate.cache.use\_query\_cache">true</property>
* And in code, we need to use setCacheable(true) method of Query, quick example looks like below.
* Query query = session.createQuery("from Employee");
* query.setCacheable(true);
* query.setCacheRegion("ALL\_EMP");

**Can we execute native sql query in hibernate?**

* Hibernate provide option to execute native SQL queries through the use of SQLQuery object.
* For normal scenarios, it is however not the recommended approach because we loose benefits related to hibernate association and hibernate first level caching.

**What is the benefit of native sql query support in hibernate?**

* Native SQL Query comes handy when we want to execute database specific queries that are not supported by Hibernate API such as query hints or the CONNECT keyword in Oracle Database.

**What is Named SQL Query?**

* Hibernate provides Named Query that we can define at a central location and use them anywhere in the code. We can created named queries for both HQL and Native SQL.
* Hibernate Named Queries can be defined in Hibernate mapping files or through the use of JPA annotations @NamedQuery and @NamedNativeQuery

**What are the benefits of Named SQL Query?**

* Hibernate Named Query helps us in grouping queries at a central location rather than letting them scattered all over the code.
* Hibernate Named Query syntax is checked when the hibernate session factory is created, thus making the application fail fast in case of any error in the named queries.
* Hibernate Named Query is global, means once defined it can be used throughout the application.
* However one of the major disadvantage of Named query is that it’s hard to debug, because we need to find out the location where it’s defined.

**What is the benefit of Hibernate Criteria API?**

* Hibernate provides Criteria API that is more object oriented for querying the database and getting results. We can’t use Criteria to run update or delete queries or any DDL statements. It’s only used to fetch the results from the database using more object oriented approach.
* Some of the common usage of Criteria API are:
* Criteria API provides Projection that we can use for aggregate functions such as sum(), min(), max() etc.  
  Criteria API can be used with ProjectionList to fetch selected columns only.
* Criteria API can be used for join queries by joining multiple tables, useful methods are createAlias(), setFetchMode() and setProjection()
* Criteria API can be used for fetching results with conditions, useful methods are add() where we can add Restrictions.
* Criteria API provides addOrder() method that we can use for ordering the results.

**What is Hibernate Proxy and how it helps in lazy loading?**

* Hibernate uses proxy object to support lazy loading. Basically when you load data from tables, hibernate doesn’t load all the mapped objects. As soon as you reference a child or lookup object via getter methods, if the linked entity is not in the session cache, then the proxy code will go to the database and load the linked object. It uses javassist to effectively and dynamically generate sub-classed implementations of your entity objects.

**How to implement relationships in hibernate?**

* We can easily implement one-to-one, one-to-many and many-to-many relationships in hibernate. It can be done using JPA annotations as well as XML based configurations. For better understanding, you should go through following tutorials.

**How transaction management works in Hibernate?**

* Transaction management is very easy in hibernate because most of the operations are not permitted outside of a transaction. So after getting the session from SessionFactory, we can call session beginTransaction() to start the transaction. This method returns the Transaction reference that we can use later on to either commit or rollback the transaction.
* Overall hibernate transaction management is better than JDBC transaction management because we don’t need to rely on exceptions for rollback. Any exception thrown by session methods automatically rollback the transaction.

**What is cascading and what are different types of cascading?**

* When we have relationship between entities, then we need to define how the different operations will affect the other entity. This is done by cascading and there are different types of it.
* Here is a simple example of applying cascading between primary and secondary entities.
* import org.hibernate.annotations.Cascade;
* @Entity
* @Table(name = "EMPLOYEE")
* public class Employee {
* @OneToOne(mapped By = "employee")
* @Cascade(value = org.hibernate.annotations.CascadeType.ALL)
* private Address address;
* }
* Note that Hibernate CascadeType enum constants are little bit different from JPA javax.persistence.CascadeType, so we need to use the Hibernate CascadeType and Cascade annotations for mappings, as shown in above example.  
  Commonly used cascading types as defined in CascadeType enum are:
* **None:** No Cascading, it’s not a type but when we don’t define any cascading then no operations in parent affects the child.
* **ALL:** Cascades save, delete, update, evict, lock, replicate, merge, persist. Basically everything
* **SAVE\_UPDATE:** Cascades save and update, available only in hibernate.
* **DELETE:** Corresponds to the Hibernate native DELETE action, only in hibernate.
* DETATCH, MERGE, PERSIST, REFRESH and REMOVE – for similar operations
* **LOCK:** Corresponds to the Hibernate native LOCK action.
* **REPLICATE:** Corresponds to the Hibernate native REPLICATE action.