Name- Ayan Mondal

Emp ID- 815954

**Hibernate Notes**

**Hibernate Framework**

Hibernate is a Java framework that simplifies the development of Java application to interact with the database. It is an open source, lightweight, ORM (Object Relational Mapping) tool. Hibernate implements the specifications of JPA (Java Persistence API) for data persistence.

**Hibernate Architecture**

The Hibernate architecture includes many objects such as persistent object, session factory, transaction factory, connection factory, session, transaction etc.

The Hibernate architecture is categorized in four layers.

* Java application layer
* Hibernate framework layer
* Backhand api layer
* Database layer

**Elements of Hibernate Architecture**

|  |
| --- |
| For creating the first hibernate application, we must know the elements of Hibernate architecture. They are as follows: |

**SessionFactory**

The SessionFactory is a factory of session and client of ConnectionProvider. It holds second level cache (optional) of data. The org.hibernate.SessionFactory interface provides factory method to get the object of Session.

**Session**

The session object provides an interface between the application and data stored in the database. It is a short-lived object and wraps the JDBC connection. It is factory of Transaction, Query and Criteria. It holds a first-level cache (mandatory) of data. The org.hibernate.Session interface provides methods to insert, update and delete the object. It also provides factory methods for Transaction, Query and Criteria.

**Transaction**

The transaction object specifies the atomic unit of work. It is optional. The org.hibernate.Transaction interface provides methods for transaction management.

ConnectionProvider

It is a factory of JDBC connections. It abstracts the application from DriverManager or DataSource. It is optional.

**TransactionFactory**

It is a factory of Transaction. It is optional

**Advantages of Hibernate**

1) Hibernate framework is open source under the LGPL license and lightweight.

2) The performance of hibernate framework is fast because cache is internally used in hibernate framework. There are two types of cache in hibernate framework first level cache and second level cache. First level cache is enabled by default.

3) HQL (Hibernate Query Language) is the object-oriented version of SQL. It generates the database independent queries. So you don't need to write database specific queries. Before Hibernate, if database is changed for the project, we need to change the SQL query as well that leads to the maintenance problem.

4) Hibernate framework provides the facility to create the tables of the database automatically. So there is no need to create tables in the database manually.

5) Fetching data from multiple tables is easy in hibernate framework.

**What is JPA?**

Java Persistence API (JPA) is a Java specification that provides certain functionality and standard to ORM tools. The javax.persistence package contains the JPA classes and interfaces.

**Supported Databases**

Hibernate supports almost all the major RDBMS. Following is a list of few of the database engines supported by Hibernate −

* HSQL Database Engine
* DB2/NT
* MySQL
* PostgreSQL
* FrontBase
* Oracle
* Microsoft SQL Server Database
* Sybase SQL Server
* Informix Dynamic Server

**Technologies**

Hibernate supports a variety of other technologies, including −

* XDoclet Spring
* J2EE
* Eclipse plug-ins
* Maven

**What is ORM?**

ORM stands for Object-Relational Mapping (ORM) is a programming technique for converting data between relational databases and object oriented programming languages such as Java, C#, etc.

An ORM system has the following advantages over plain JDBC −

|  |  |
| --- | --- |
| Sr.No. | Advantages |
| 1 | Let’s business code access objects rather than DB tables. |
| 2 | Hides details of SQL queries from OO logic. |
| 3 | Based on JDBC 'under the hood.' |
| 4 | No need to deal with the database implementation. |
| 5 | Entities based on business concepts rather than database structure. |
| 6 | Transaction management and automatic key generation. |
| 7 | Fast development of application. |

**Hibernate Mappings**

**Collections Mappings**

If an entity or class has collection of values for a particular variable, then we can map those values using any one of the collection interfaces available in java. Hibernate can persist instances of java.util.Map, java.util.Set, java.util.SortedMap, java.util.SortedSet, java.util.List, and any array of persistent entities or values.

|  |  |
| --- | --- |
| Sr.No. | Collection type & Mapping Description |
| 1 | [java.util.Set](https://www.tutorialspoint.com/hibernate/hibernate_set_mapping.htm)  This is mapped with a <set> element and initialized with java.util.HashSet |
| 2 | [java.util.SortedSet](https://www.tutorialspoint.com/hibernate/hibernate_sortedset_mapping.htm)  This is mapped with a <set> element and initialized with java.util.TreeSet. The sort attribute can be set to either a comparator or natural ordering. |
| 3 | [java.util.List](https://www.tutorialspoint.com/hibernate/hibernate_list_mapping.htm)  This is mapped with a <list> element and initialized with java.util.ArrayList |
| 4 | [java.util.Collection](https://www.tutorialspoint.com/hibernate/hibernate_bag_mapping.htm)  This is mapped with a <bag> or <ibag> element and initialized with java.util.ArrayList |
| 5 | [java.util.Map](https://www.tutorialspoint.com/hibernate/hibernate_map_mapping.htm)  This is mapped with a <map> element and initialized with java.util.HashMap |
| 6 | [java.util.SortedMap](https://www.tutorialspoint.com/hibernate/hibernate_sortedmap_mapping.htm)  This is mapped with a <map> element and initialized with java.util.TreeMap. The sort attribute can be set to either a comparator or natural ordering. |

**Association Mappings**

The mapping of associations between entity classes and the relationships between tables is the soul of ORM. Following are the four ways in which the cardinality of the relationship between the objects can be expressed. An association mapping can be unidirectional as well as bidirectional.

|  |  |
| --- | --- |
| **Sr.No.** | **Mapping type & Description** |
| 1 | [Many-to-One](https://www.tutorialspoint.com/hibernate/hibernate_many_to_one_mapping.htm)  Mapping many-to-one relationship using Hibernate |
| 2 | [One-to-One](https://www.tutorialspoint.com/hibernate/hibernate_one_to_one_mapping.htm)  Mapping one-to-one relationship using Hibernate |
| 3 | [One-to-Many](https://www.tutorialspoint.com/hibernate/hibernate_one_to_many_mapping.htm)  Mapping one-to-many relationship using Hibernate |
| 4 | [Many-to-Many](https://www.tutorialspoint.com/hibernate/hibernate_many_to_many_mapping.htm)  Mapping many-to-many relationship using Hibernate |

**Component Mappings**

It is very much possible that an Entity class can have a reference to another class as a member variable. If the referred class does not have its own life cycle and completely depends on the life cycle of the owning entity class, then the referred class hence therefore is called as the Component class.

The mapping of Collection of Components is also possible in a similar way just as the mapping of regular Collections with minor configuration differences. We will see these two mappings in detail with examples.

|  |  |
| --- | --- |
| **Sr.No.** | **Mapping type & Description** |
| 1 | [Component Mappings](https://www.tutorialspoint.com/hibernate/hibernate_component_mappings.htm)  Mapping for a class having a reference to another class as a member variable. |

**Hibernate - Query Language**

Hibernate Query Language (HQL) is an object-oriented query language, similar to SQL, but instead of operating on tables and columns, HQL works with persistent objects and their properties. HQL queries are translated by Hibernate into conventional SQL queries, which in turns perform action on database.

Although we can use SQL statements directly with Hibernate using Native SQL, but I would recommend to use HQL whenever possible to avoid database portability hassles, and to take advantage of Hibernate's SQL generation and caching strategies.

Keywords like SELECT, FROM, and WHERE, etc., are not case sensitive, but properties like table and column names are case sensitive in HQL.

**Hibernate Transaction Management**

A transaction simply represents a unit of work. In such case, if one step fails, the whole transaction fails (which is termed as atomicity). A transaction can be described by ACID properties (Atomicity, Consistency, Isolation and Durability).

**Transaction Interface in Hibernate**

In hibernate framework, we have Transaction interface that defines the unit of work. It maintains abstraction from the transaction implementation (JTA,JDBC).

A transaction is associated with Session and instantiated by calling session.beginTransaction().

The methods of Transaction interface are as follows:

* **void begin()** starts a new transaction.
* **void commit()** ends the unit of work unless we are in FlushMode.NEVER.
* **void rollback()** forces this transaction to rollback.
* **void setTimeout(int seconds)** it sets a transaction timeout for any transaction started by a subsequent call to begin on this instance.
* **boolean isAlive()** checks if the transaction is still alive.
* **void registerSynchronization(Synchronization s)** registers a user synchronization callback for this transaction.
* **boolean wasCommited()** checks if the transaction is commited successfully.
* **boolean wasRolledBack()** checks if the transaction is rolledback successfully.