

Above diagram explains the Hibernate architecture broadly. Use of Hibernate to persist data at first we will need an Entity class (mapped with database table) that will be instantiated by created by Hibernate, an instance of SessionFactory will be made. Factory design pattern is implemented by SessionFactory and it loads hibernate.cfg.xml file. Session object that is made from SessionFactory object is acted as a single connection with database, this connection is made by an instance of Session interface. Transaction API in Hibernate abstracts the JDBC or JTA transaction to the application.

Configuration : Configuration instance is used to provide the application to assign properties and mapped the documents when the SessionFactory is creating.

SessionFactory : SessionFactory is responsible for creating a Session instances to communicate with the database. **SessionFactory object is a thread safe (immutable) and it is created once as per database.**

**TransactionFactory** : TransactionFactory is responsible for generating a Transaction instances.

**ConnectionProvider** : Using ConnectionProvider Hibernate obtains a JDBC connections.

**Session** : Session interface is a major runtime interface between an Application and Hibernate. Main functionality of Session is to create, read, and delete operations of the mapped entity classes instance.

**Query** : This interface represents a Hibernate query as Object-Oriented. An instance of Query is found by Session.createQuery().

**Criteria** : Criteria is used to retrieve entities by criterion objects.

**JPA/java persistant Api** is a standard for Java object-relational mapping - it specifies a set of annotations and an interface -EntityManager to perform persistence operations with the mapped objects. Hibernate implements the JPA standard.

**plain JDBC** is a technology for accessing databases. It is what Hibernate actually uses to perform the database operations, "under the hood". It uses JDBC to send queries to the database.

**JTA** is a transaction API, and it is optional in Hibernate. It handles (logically) the transaction behaviour.

**JNDI java naming directory interface :** a mechanism to bind an object to a name

**Components of Hibernate Architecture**

There are three main components of Hibernate Architecture :(broadly classified)

**Connection Management** : This service allows database connection management in an effective manner.

**Transaction Management** : This service allows for executing a several database statements at a time.

**Object Relational Mapping** : ORM technique is used for mapping the data representation from an object-oriented model to a relational database.

**Common jar file used in hibernate**

**Hibernate-core** : provides core hibernate functionality like session factory, session , configuration , query , transaction etc

**Hibernate-validator** : provides some validator classes like email validator, credit card validator, pattern validator etc.

**Hibernate-commons-annotations** : provides classes which support annotations**.**

**Hibernate-jpa-2.0-api** : provides classes to map object properties with table columns.

**Hibernate-entitymanager** : support beans/entities

**Hibernate-entitymanager** : bean property validtion. Ex : person.age >18 and <28

**Slf4j** : simple log facade for java : used for logging. Slf4j give generic access to all logging frameworks.

Once you have placed both(slf4j and log4j) jar files in your application classpath, SLF4j will automatically detect it and start using log4j for processing the log statements based on configuration you provided in log4j configuration file.

**Mysql / oracle connector** depending on which database you use.