Hibernate Architecture: -

Hibernate is broadly divided into four layers

* Hibernate Architecture is divided into four layers.
* Java application layer
* Hibernate framework layer
* Backhand API layer
* Database layer



Following is a detailed view of the Hibernate Application Architecture with its important core classes.



## So, here we can see that Hibernate needs multiple objects like configuration object, SessionFactory object, Session object, Transaction object, Query Object, Criteria Object.

## Configuration Object

## The Configuration object is the first Hibernate object you create in any Hibernate application. It is usually created only once during application initialization. It represents a configuration or properties file required by the Hibernate.

## The Configuration object provides two keys components −

## Database Connection − This is handled through one or more configuration files supported by Hibernate. These files are hibernate.properties and hibernate.cfg.xml.

## Class Mapping Setup − This component creates the connection between the Java classes and database tables.

## Configuration is class present in org.hibernate.cfg package.

## This Object takes hibernate configuration file name and location as input value and also takes hibernate mapping file name and location through hibernate configuration file.

## This class is based on builder design pattern

## Using configuration Object we can create SessionFactory Object.

## In case if we won’t provide hibernate configuration file then it takes hibernate.cfg.xml file from classpath.

## Configuration config = new Configuration();

## SessionFactory Object

## SessionFactory is an interface present in org.hibernate package.

## To check it out this check official documentation of this interface. <https://docs.jboss.org/hibernate/orm/3.5/api/org/hibernate/SessionFactory.html#:~:text=Interface%20SessionFactory&text=The%20main%20contract%20here%20is,of%20a%20SessionFactory%20is%20immutable>.

## We can’t create object of Interface, but we can use it’s implemented classes to create it’s object. Visit above link and see it’s implemented classes.

## It’s design is based on factory design pattern. It provides abstraction toward Session Object creation.

## Factory Design pattern -> It is design pattern under which it will create objects for us without knowing how it’s getting created. Simply understand it with real factory example.

## Here using this object we can use a method of implemented class that is openSession()

## 

## Session Object

## Now, using SessionFactory session object can be created.

## 