**Why ORM?**

When we work with an object-oriented system, there is a mismatch between the object model and the relational database.

RDBMSs represent data in a tabular format whereas object-oriented languages, such as Java or C# represent it as an interconnected graph of objects.

**ORM (O**bject **R**elational **M**apping**)**

It is a programming technique for converting data between relational databases and object oriented programming languages such as Java, C#, etc.

**Advantages of ORM**

Code access objects rather than DB tables.

It hides details of SQL queries from OO logic.

It is based on JDBC 'under the hood.'

No need to deal with the database implementation.

Transaction management and automatic key generation.

Fast development of application.

**Hibernate Framework**

Hibernate is an Object-Relational Mapping (ORM) solution for JAVA. It is an open source persistent framework created by Gavin King in 2001.

It is a powerful, high performance Object-Relational Persistence and Query service for any Java Application.

Hibernate maps Java classes to database tables and from Java data types to SQL data types and relieves the developer from 95% of common data persistence related programming tasks.

Hibernate sits between traditional Java objects and database server to handle all the works in persisting those objects based on the appropriate O/R mechanisms and patterns.

**Hibernate Advantages**

1) Hibernate takes care of mapping Java classes to database tables using XML files and without writing any line of code.

2) Provides simple APIs for storing and retrieving Java objects directly to and from the database.

3) If there is change in the database or in any table, then you need to change the XML file properties only.

4) Hibernate does not require an application server to operate.

5) Manipulates Complex associations of objects of your database.

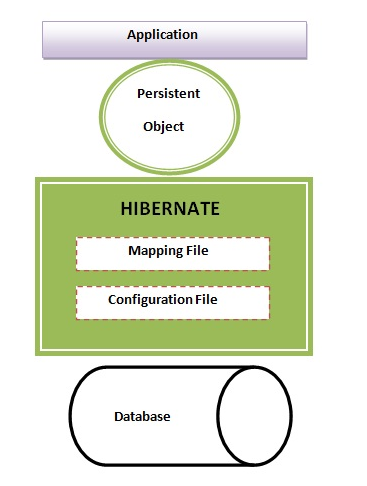
6) Minimizes database access with smart fetching strategies.

7) Provides simple querying of data.

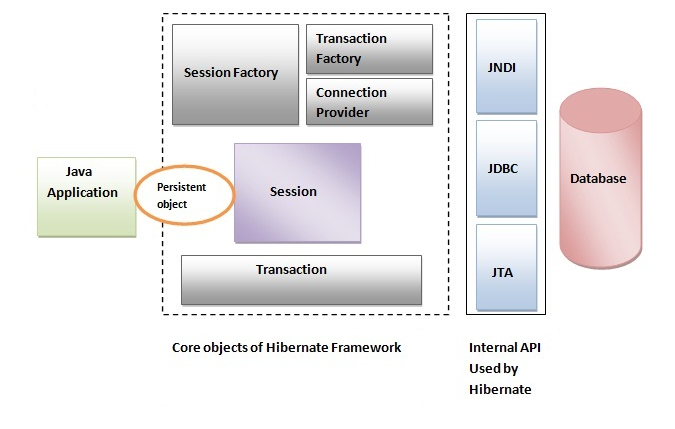
Hibernate Architecture

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

There are 4 layers in hibernate architecture java application layer, hibernate framework layer, backend API layer and database layer.



**High level Architecture of Hibernate**



#### Configuration object

The **org.hibernate.cfg.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.

Configuration object is used to create a SessionFactory object which in turn configures Hibernate for the application using the supplied configuration file.

#### SessionFactory object

The SessionFactory is a factory of session and client of ConnectionProvider. The **org.hibernate.SessionFactory** interface provides factory method to get the object of Session.

#### Session

#### A Session is used to get a physical connection with a database 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.

#### 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.

#### Query object

Query objects use SQL or Hibernate Query Language (HQL) string to retrieve data from

the database and create objects. A Query instance is used to bind query parameters, limit

the number of results returned by the query, and finally to execute the query.

**Hibernate Configuration file**

Hibernate requires a set of configuration settings related to database and other related parameters. Hibernate configuration can be done through standard java properties file called as hibernate.properties or as an XMl file named as hibernate.cfg.xml

**Important hibernate properties:-**

1. hibernate.dialect

This property makes Hibernate generate the appropriate SQL for the chosen database.

eg:-

org.hibernate.dialect.OracleDialect

org.hibernate.dialect.MySQLDialect

org.hibernate.dialect.DB2Dialect

org.hibernate.dialect.PostgreSQLDialect

1. hibernate.connection.driver\_class 🡪 To specify the JDBC driver class.
2. hibernate.connection.url 🡪 The JDBC URL to the database.
3. hibernate.connection.username 🡪 The database username.
4. hibernate.connection.password 🡪 The database password.

E.g.:-

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE hibernate-configuration SYSTEM

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

<property name="hibernate.connection.driver\_class">com.mysql.jdbc.Driver</property>

<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/sasi</property>

<property name="hibernate.connection.username">root</property>

<property name="hibernate.connection.password"></property>

<mapping resource="employee.hbm.xml"/>

</session-factory>

</hibernate-configuration>

**Hibernate Mapping file**

An Object/relational mappings are usually defined in an XML document. This mapping file instructs Hibernate — how to map the defined class to the database table.