Hibernate Framework

Hibernate framework simplifies the development of java application to interact with the database. Hibernate is an open source, lightweight, [ORM (Object Relational Mapping)](http://en.wikipedia.org/wiki/Object-relational_mapping) tool.

An ORM tool simplifies the data creation, data manipulation and data access. It is a programming technique that maps the object to the data stored in the database.



The ORM tool internally uses the JDBC API to interact with the database.

Advantages of Hibernate Framework

There are many advantages of Hibernate Framework. They are as follows:

**1) Opensource and Lightweight:** Hibernate framework is opensource under the LGPL license and lightweight.

**2) Fast performance:** 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 bydefault.

**3) Database Independent query:** 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) Automatic table creation:** 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) Simplifies complex join:** To fetch data from multiple tables is easy in hibernate framework.

**6) Provides query statistics and database status:** Hibernate supports Query cache and provide statistics about query and database status.

Next Top

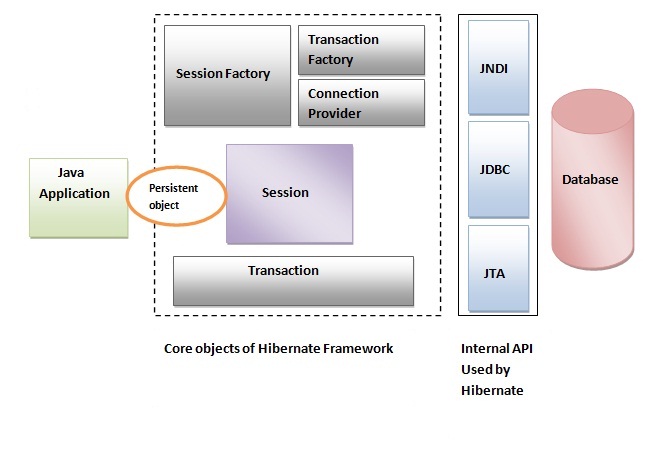
# Hibernate Architecture

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, backhand api layer and database layer.Let's see the diagram of hibernate architecture:



This is the high level architecture of Hibernate with mapping file and configuration file.



Hibernate framework uses many objects session factory, session, transaction etc. alongwith existing Java API such as JDBC (Java Database Connectivity), JTA (Java Transaction API) and JNDI (Java Naming Directory Interface).

## 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.

# Hibernate Example

For creating the first hibernate application, we need to follow following steps:

1. Create the Persistent class
2. Create the mapping file for Persistent class
3. Create the Configuration file
4. Create the class that retrieves or stores the persistent object
5. Load the jar file
6. Run the first hibernate application without IDE

### 1) Create the Persistent class

A simple Persistent class should follow some rules:

* **A no-arg constructor:** It is recommended that you have a default constructor at least package visibility so that hibernate can create the instance of the Persistent class by newInstance() method.
* **Provide an identifier property (optional):** It is mapped to the primary key column of the database.
* **Declare getter and setter methods (optional):** The Hibernate recognizes the method by getter and setter method names by default.
* **Prefer non-final class:** Hibernate uses the concept of proxies, that depends on the persistent class. The application programmer will not be able to use proxies for lazy association fetching.

Let's create the simple Persistent class:

#### Employee.java

**package** com.javatpoint.mypackage;

**public** **class** Employee {

**private** **int** id;

**private** String firstName,lastName;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getLastName() {

**return** lastName;

}

**public** **void** setLastName(String lastName) {

**this**.lastName = lastName;

}

}

### 2) Create the mapping file for Persistent class

The mapping file name conventionally, should be class\_name.hbm.xml. There are many elements of the mapping file.

* **hibernate-mapping** is the root element in the mapping file.
* **class** It is the sub-element of the hibernate-mapping element. It specifies the Persistent class.
* **id**It is the subelement of class. It specifies the primary key attribute in the class.
* **generator** It is the subelement of id. It is used to generate the primary key. There are many generator classes such as assigned (It is used if id is specified by the user), increment, hilo, sequence, native etc. We will learn all the generator classes later.
* **property** It is the subelement of class that specifies the property name of the Persistent class.

Let's see the mapping file for the Employee class:

#### employee.hbm.xml

<?xml version='1.0' encoding='UTF-8'?>

<!DOCTYPE hibernate-mapping PUBLIC

 "-//Hibernate/Hibernate Mapping DTD 3.0//EN"

 "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

 <hibernate-mapping>

  <**class** name="com.javatpoint.mypackage.Employee" table="emp1000">

    <id name="id">

     <generator **class**="assigned"></generator>

    </id>

    <property name="firstName"></property>

    <property name="lastName"></property>

  </**class**>

 </hibernate-mapping>

### 3) Create the Configuration file

The configuration file contains informations about the database and mapping file. Conventionally, its name should be hibernate.cfg.xml .

#### hibernate.cfg.xml

<?xml version='1.0' encoding='UTF-8'?>

<!DOCTYPE hibernate-configuration PUBLIC

          "-//Hibernate/Hibernate Configuration DTD 3.0//EN"

          "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

    <session-factory>

        <property name="hbm2ddl.auto">update</property>

        <property name="dialect">org.hibernate.dialect.Oracle9Dialect</property>

        <property name="connection.url">jdbc:oracle:thin:@localhost:1521:xe</property>

        <property name="connection.username">system</property>

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

        <property name="connection.driver\_class">oracle.jdbc.driver.OracleDriver</property>

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

    </session-factory>

</hibernate-configuration>

### 4) Create the class that retrieves or stores the object (JavaBean)

In this class, we are simply storing the employee object to the database.

**package** com.javatpoint.mypackage;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.hibernate.cfg.Configuration;

**public** **class** StoreData {

**public** **static** **void** main(String[] args) {

    //creating configuration object

    Configuration cfg=**new** Configuration();

    cfg.configure("hibernate.cfg.xml");//populates the data of the configuration file

    //creating seession factory object

    SessionFactory factory=cfg.buildSessionFactory();

    //creating session object

    Session session=factory.openSession();

    //creating transaction object

    Transaction t=session.beginTransaction();

    Employee e1=**new** Employee();

    e1.setId(115);

    e1.setFirstName("sonoo");

    e1.setLastName("jaiswal");

    session.persist(e1);//persisting the object

    t.commit();//transaction is commited

    session.close();

    System.out.println("successfully saved");

}

}

### 5) Load the jar file

For successfully running the hibernate application, you should have the hibernate4.jar file.

[download the latest hibernate jar file](http://www.hibernate.org/downloads). Some other jar files or packages are required such as

* cglib
* log4j
* commons
* SLF4J
* dom4j
* xalan
* xerces

[download the required jar files for hibernate](https://www.javatpoint.com/src/hb/hibernatejar.zip)