Hibernate Framework
By Ram Lovewanshi

Need of Hibernate Framework

- Hibernate is used to overcome the limitations of JDBC like:
- JDBC code is dependent upon the Database software being used i.e. our persistence logic is dependent, because of using JDBC. Here we are inserting a record into Employee table but our query is Database software-dependent i.e. Here we are using MySQL. But if we change our Database then this query won't work.
- If working with JDBC, changing of Database in middle of the project is very costly.
- JDBC code is not portable code across the multiple database software.
- In JDBC, Exception handling is mandatory. Here We can see that we are handling lots of Exception for connection.

Need of Hibernate Framework

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
public class InsertDataIntoEmp
{
    public static void main(String[] args)
    {
        final String URL="jdbc:mysql://localhost:3306?user=root&&password=root";
        String query="insert into EmployeeDB.Employee values(1,'Bikash Dubey','300000','CSE')";
        try
        {
            Class.forName("com.mysql.jdbc.Driver");
            Connection con=DriverManager.getConnection(URL);
            System.out.println("Connection established");
            Statement st=con.createStatement();
            st.execute(query);
            System.out.println("Data inserted");
            st.close();
            con.close();
        }catch(SQLException | ClassNotFoundException e)
        {
            e.printStackTrace();
        }
    }
}
```

Need of Hibernate Framework

```
import java.io.IOException;
import java.sql.*;
public class Demo
{
    public static void main(String[] args) throws IOException, SQLException
    {
        try {
            /*loading & register of the driver class*/
            Class.forName("com.mysql.jdbc.Driver");
            Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306","root", "root");
            System.out.println("Driver class loaded successfully");
        } catch (ClassNotFoundException e)
        {
            System.out.println("Driver Class not loaded");
        }
    }
}
```

Need of Hibernate Framework

```
import java.io.IOException;
import java.sql.*;
public class Demo
{
    public static void main(String[] args) throws IOException, SQLException
    {
        try {
            /*loading & register of the driver class*/
            Class.forName("com.mysql.jdbc.Driver");
            Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306","root", "root");
            System.out.println("Driver class loaded successfully");
        } catch (ClassNotFoundException e)
        {
            System.out.println("Driver Class not loaded");
        }
    }
}
```

Need of Hibernate Framework

- While working with JDBC, There is no support Object-level relationship.
- In JDBC, there occurs a Boilerplate problem i.e. For each and every project we have to write the below code. That increases the code length and reduce the readability.

```
try
{
    Class.forName("com.mysql.jdbc.Driver");
    Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306?user=root&&password=root");
    System.out.println("Connection established");
    con.close();
}catch(SQLException | ClassNotFoundException e)
{
    e.printStackTrace();
}
```

Hibernate Framework

- To overcome the above problems we use ORM tool i.e. nothing but Hibernate framework. By using Hibernate we can avoid all the above problems and we can enjoy some additional set of functionalities.
- Hibernate is a framework which provides some abstraction layer, meaning that the programmer does not have to worry about the implementations, Hibernate does the implementations for you internally like Establishing a connection with the database, writing query to perform CRUD operations etc.

Hibernate Framework

- It is a java framework which is used to develop persistence logic.

  Persistence logic means to store and process the data for long use. More precisely Hibernate is an open-source, non-invasive, light-weight java ORM(Object-relational mapping) framework to develop objects which are independent of the database software and make independent persistence logic in all JAVA, JEE.

- Framework means it is special install-able software that provides an abstraction layer on one or more technologies like JDBC, Servlet, etc to simplify or reduce the complexity for the development process.

Open Source means:

- Hibernate framework is available for everyone without any cost.
- The source code of Hibernate is also available on the Internet and we can also modify the code.

Light-weight means:

- Hibernate is less in size means the installation package is not big is size.
- Hibernate does not require any heavy container for execution.
- It does not require POJO and POJI model programming.
- Hibernate can be used alone or we can use Hibernate with other java technology and framework.

Non-invasive means:

- The classes of Hibernate application development are loosely coupled classes with respect to Hibernate API i.e. Hibernate class need not implement hibernate API interfaces and need not extend from Hibernate API classes.

Functionalities supported by Hibernate framework

- Hibernate framework support Auto DDL operations. In JDBC manually we have to create table and declare the data-type for each and every column. But Hibernate can do DDL operations for you internally like creation of table,drop a table,alter a table etc.
- Hibernate supports Auto Primary key generation. It means in JDBC we have to manually set a primary key for a table. But Hibernate can this task for you.

Functionalities supported by Hibernate framework

- Hibernate framework is independent of Database because it supports HQL (Hibernate Query Language) which is not specific to any database, whereas JDBC is database dependent.
- In Hibernate, Exception Handling is not mandatory, whereas In JDBC exception handling is mandatory.
- Hibernate supports Cache Memory whereas JDBC does not support cache memory.

Functionalities supported by Hibernate framework

- Hibernate is a ORM tool means it support Object relational mapping. Whereas JDBC is not object oriented moreover we are dealing with values means primitive data. In hibernate each record is represented as a Object but in JDBC each record is nothing but a data which is nothing but primitive values.

Hibernate Architecture

- Hibernate: Hibernate is a framework which is used to develop persistence logic which is independent of Database software. In JDBC to develop persistence logic we deal with primitive types. Whereas Hibernate framework we use Objects to develop persistence logic which are independent of database software.

  Hibernate Architecture:

Hibernate Architecture

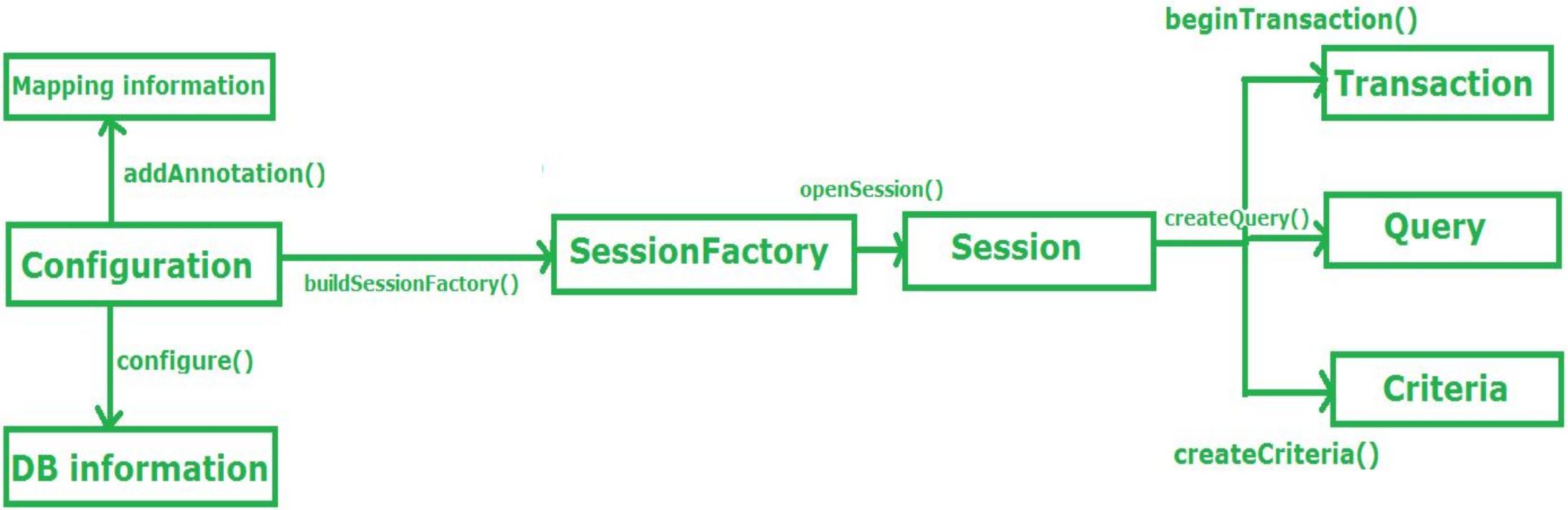

Fig: Hibernate Architecture

Configuration:

- Configuration is a class which is present in org.hibernate.cfg package. It activates Hibernate framework. It reads both configuration file and mapping files.

It activate Hibernate Framework

Configuration cfg=new Configuration();

It read both cfg file and mapping files

cfg.configure();

Configuration:

- It checks whether the config file is syntactically correct or not.
- If the config file is not valid then it will throw an exception. If it is valid then it creates a meta-data in memory and returns the meta-data to object to represent the config file.

SessionFactory:

- SessionFactory is an Interface which is present in org.hibernate package and it is used to create Session Object.
- It is immutable and thread-safe in nature.

buildSessionFactory() method gathers the meta-data which is in the cfg Object.

From cfg object it takes the JDBC information and create a JDBC Connection.

SessionFactory factory=cfg.buildSessionFactory();

Session:

- Session is an interface which is present in org.hibernate package. Session object is created based upon SessionFactory object i.e. factory.
- It opens the Connection/Session with Database software through Hibernate Framework.
- It is a light-weight object and it is not thread-safe.
- Session object is used to perform CRUD operations.

Session session=factory.buildSession();

Transaction:

- Transaction object is used whenever we perform any operation and based upon that operation there is some change in database.
- Transaction object is used to give the instruction to the database to make the changes that happen because of operation as a permanent by using commit() method.

Transaction tx=session.beginTransaction();

tx.commit();

Query:

- Query is an interface that present inside org.hibernate package.
- A Query instance is obtained by calling Session.createQuery().
- This interface exposes some extra functionality beyond that provided by Session.iterate() and Session.find():

1. A particular page of the result set may be selected by calling setMaxResults(), setFirstResult().
2. Named query parameters may be used.

Criteria:

- Criteria is a simplified API for retrieving entities by composing Criterion objects.
- The Session is a factory for Criteria. Criterion instances are usually obtained via the factory methods on Restrictions.

Flow of working during operation in Hibernate Framework

- Suppose We want to insert an Object to the database. Here Object is nothing but persistence logic which we write on java program and create an object of that program. If we want to insert that object in the database or we want to retrieve the object from the database. Now the question is that how hibernate save the Object to the database or retrieve the object from the database. There are several layers through which Hibernate framework go to achieve the above task. Let us understand the layers/flow of Hibernate framework during performing operations:

Stage I:

- In first stage, we will write the persistence logic to perform some specific operations to the database with the help of Hibernate Configuration file and Hibernate mapping file. And after that we create an object of the particular class on which we wrote the persistence logic.

Stage II

- In second stage, our class which contains the persistence logic will interact with the hibernate framework where hibernate framework gives some abstraction do perform some task. Now here the picture of java class is over. Now Hibernate is responsible to perform the persistence logic with the help of layers which is below of Hibernate framework or we can say that the layers which are the internal implementation of Hibernate.

Stage III

- In third stage, our hibernate framework interact which JDBC, JNDI, JTA etc to go to the database to perform that persistence logic

Stage IV & V

- In fourth & fifth stage, hibernate is interact with Database with the help of JDBC driver. Now here hibernate perform that persistence logic which is nothing but CRUD operation. If our persistence logic is to retrieve an record then in the reverse order it will display on the console of our java program in terms of Object.

Some Hibernate Properties

1. Automatic Table Creation Using Hibernate

<property name=”hibernate.hbm2ddl.auto”>create</property>

create: If the value is created, Hibernate creates a new table in the database when the SessionFactory object is created. In case, a table exists in the database with the same name, it deletes the table along with data and creates a new table.

update: If the value is updated then hibernate first validates whether the table is present in the database or not , if present alters that table as per the changes, if not creates a new one.

Some Hibernate Properties

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.formate_sql=true;