HIBERNATE, An ORM Framework for Accessing Database

[1]Jagjit Kaur, [2]Neha sikka, [3]Sunil Kumar Chawla
Assistant Professor
[1]CGC College of Engineering

***Abstract*— The article describes about the Hibernate Framework, which is used to interact with heavy backend databases for the enterprise applications. Earlier the interaction was possible because of "Persistence layer". But the approach of persistence layer costs heavily in terms of time and man work, when the backend schema changes and the whole persistence layer is rewritten which consumes a lot of time and energy. So, here the Hibernate Framework comes into rescue to overcome the problems and the plus point is that it's also an Open source framework. It uses HQL as its query Language and is database independent, it is based on JOM (Java Object Model) and the persistent objects are based on JOM. It is an ORM framework that is used in converting data between Relational data and Object oriented programming languages like java, c#, JavaScript etc.**

***Index Terms*— Hibernate, HQL, ORM, Database, JOM, and POJO.**

I. INTRODUCTION

Hibernate is a Java framework and just like SQL is an Object Oriented Query Language and is a high Performance Query service plus it is open Source. We have tables and columns in SQL whereas HQL works with persistent objects and properties [2]. By the use of SQL we can convert HQL to Hibernate. In this, to get the HQL commands, the POJO class variable name replaces the table column name & POJO class name replaces the table names. By using the HQL, we can run any Database because it is independent of database, more formally, Database Independent [3]. This means, we can modify the schema at one level without interfering with the next level schema, three-schema architecture describes the data independency. As this is object oriented, so it supports features like inheritance, polymorphism and associations etc. The DML operations like insert, update can be performed through it. DML is simply the data manipulations language, as we can manipulate the data by performing several operations as written above. Hibernate is used for the relation between the object world of java with the relational database and that relation is called object relationship mapping (ORM). Using Session Factory in the Session class we can save our object directly into database [4].

For clarification of the data manipulation and data creation an ORM tool is used. ORM is a well-planned tool so if you are making an application and where you want to persist data this is something you can actually use[5].Now, Let us talk about how to setup the Hibernate Framework for use.

II. HIBERNATE SETUP

Use NetBeans IDE for the setup of Hibernate.

1) Firstly go to new and select new project of java web application

2) After selecting, We have to Name the Project : As you click Next, write the project Name for e.g. Hibernate Demo

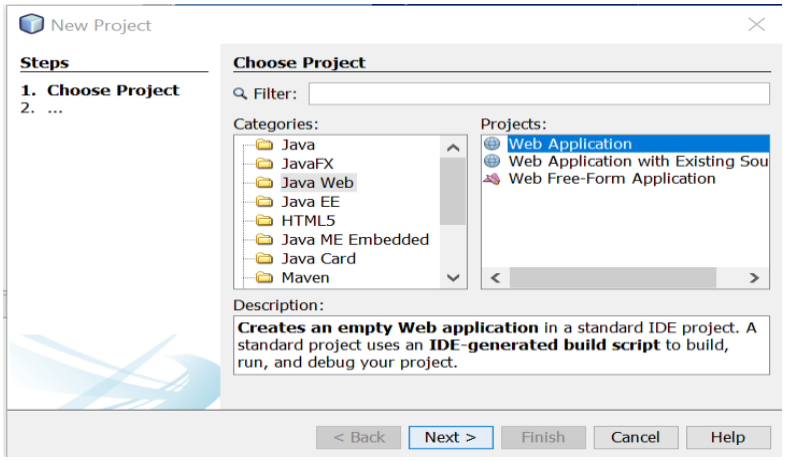

Fig. 1: IDE Environment: Naming

3) Then we need to setup the Server, Select the Server as Apache Tomcat, as you select Tomcat server, you don't need to mess up with JDBC connections [6].

4) Select the collection cache.

5) Select one by one firstly hibernate reverse engineering wizard then hibernate mapping and POJO file then the last one is hibernate.java where you will write the java code for hibernate implementation.

- **The use of revenge.xml:** The revenge, configure.xml contains the proper mapping of database .Db name its password, Drivers and all.

- **Use of POJO class:** This class is the SessionBean class which contains the getter and setter to set and get the value of object variable [4].

- **More about Hibernate:** Hibernate uses its own query Language i.e. HQL (Hibernate query language) which are database independent query language. The queries against database are achieved by HQL. The queries are itself generate by HQL and are executed against fundamental database if HQL is used in the application [9]. HQL makes the SQL object oriented and uses Relational object model. Classes are used in HQL and their properties and objects instead of tables and columns. It supports polymorphism and it is really powerful.

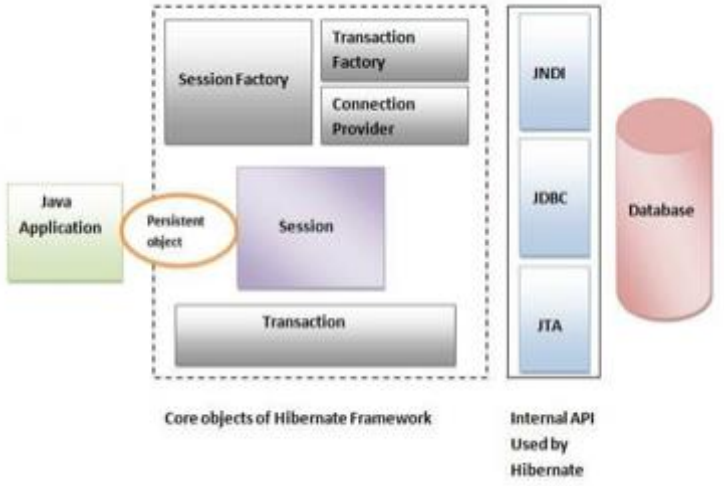

Fig. 2: Hibernate Structure

III. POJO

POJO is the abbreviation of "*Plain Old Java Object*". It is nothing but a pure Data structure having getters and setters [5]. And those getters and setters are publically defined. POJO like its name suggests, is compiled under the JDK. The following code snippet shows the setters and getters of the POJO class:

```
packagemypack;
public class complex1{

  privateint x ;
  privateint y ;
  //Getters
  publicintgetX(){
        return x ;
  }
  publicintgetY(){
        return y ;
  }
  //Setters
  public void setX(int a){
        x = a ;
  }
  public void setY(int b){
        y = b ;
  }
}
```

IV. DATA FLOW

- **First we need to create a persistent class.**
- **Then mapping takes place by the procedure**

We need to map Hib.Cnf.xml from where we get all information regarding Database, its driver, username and password of database, port no., no. of table contains in database.

Mapping of each table:

Then we have revenge.xml where mapping of each table is done inside the database. The file className.xml is the one in which mapping of each column of every table is done [1]. As discussed in introduction part, POJO class is for setting the values and when needed then get the calling its method.

- Create SessionFactory class:

To create the SessionFactory Class from the Hibernate Configfile(Hib.Cnf.xml), we need to create a HibernateProg class. SessionFactory Class is a thread safe class, many threads can have access to it simultaneously and also it has an immutable structure.

Create an object of SessionFactory:

To instantiate SessionFactory, the static singleton pattern is used. The HibernateProg class which we have made is implemented below in the program section [4].

```
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

class HibernateProg{

    private static final SessionFactory sessFactory;
    static {
            try{

                sessFactory = new Configuration().configure().buildSessionFactory();
            }
            catch(Throwable t){
                System.err.print("sessionFactory object creation failed "+t);
                throw new ExceptionInInitializerError(t) ;
            }
    }
    public static SessionFactory getSessionFactory(){
        return sessFactory;
    }
}
```

Fig. 3: Hibernate Program Class Implementation

V. HIBERNATE STRUCTURE

The layered Architecture of Hibernate lets the programmer or the user to operate and use the database without going into depth of the backend APIs. JDBC, JNDI, JTA are some of the APIs used by Hibernate. The integration of Hibernate with J2EE application servers is achieved by JTA and JNDI [6].So, starting with the programming of Hibernate, what are the classes, objects used in it?

An object known as Configuration object is created firstly whenever we create our Hibernate application. It is created only once in an application.

Role of Configuration Object:

The database connection is handled by it and the class mapping setup which is used to create the connection between classes of Java and database tables [8]. **S**essionFactory object is created by configuration object. The former is used to configure the Hibernate for the application, as discussed before; the SessionFactory Class is thread safe. For multiple databases, we need to create multiple objects of it. To get the physical connection with the database, the Session object is created. Whenever we need to connect with the database or to interact with it, the instantiation is needed for the Session object [7]. The transaction manager handles the transaction in Hibernate. It is optional object. Query Object uses HQL string to get the data from database, and execute them.

To execute object oriented criteria query, criteria object are used.

Some properties of Hibernate

1) hibernate.connection.username
2) hibernate.connection.password
3) hibernate.connection.url
4) hibernate.connection.pool_size
5) hibernate.connection.autocommit
6) hibernate.connection.driver_class

VI. HIBERNATE QUERY LANGUAGE

Just like SQL, Hibernate also has the query language HQL, if you have a good hand on SQL and RDBMS earlier, HQL is just a cakewalk. Similar keywords are used like SELECT, UPDATE, WHERE, FROM etc. and they are not case sensitive, write "select" or "SELECT", they mean the same. Let us look at some code snippets to use them:

1) FROM

```
String h = "FROM Student";  //Student is the Table
Query q = session.createQuery(h);
List list = q.list() ; //List collection holds the persistent objects
```

2) AS (To make aliases)

```
String h = "FROM Student as S"; //Now, College table can be referred to as E.
Query q = session.createQuery(h);
List list = q.list() ;
```

we can also write :

FROM Student S" This means the same.

3) SELECT (Precise over FROM,only certain properties can be obtained)

```
String h = "SELECT S.firstName from S";
Query q = session.createQuery(h);
List list = q.list() ;
```

Only firstName Property will be loaded in this case [4].

4) ORDER BY (Sorting the result of the HQL queries)

```
String h = "FROM Student S WHERE S.rollNo> 10 ORDER BY S.marks DESC";
Query q = session.createQuery(h);
List list = q.list();
```

//This will give sort the Students' marks in decreasing order whose rollno is greater than 10.

5) UPDATE (Use to update one or more properties)

```
String h = "UPDATE Student set marks =: marks" + "WHERE rollNo =: student_rollno";
Query q = session.createQuery(h);
q.setParameter("marks", 96); //Setting the parameters(updating)
q.setParameter("student_rollno", 10);
int result = q.executeUpdate(); //Executing the update
System.out.println("Rows affected: " + result); //Print
```

//This will update the marks of student to 96, where rollNo is 10.

6) DELETE (To delete one or more than one object)

```
String h = "DELETE FROM Student "  + "WHERE rollNo =  :student_rollno";
Query q = session.createQuery(h);
q.setParameter("student_rollno", 10);
int result = q.executeUpdate();
System.out.println("Rows affected: " + result);
```

//This will delete the data of the student where roll no. is 10.

7) INSERT (To insert the data into the table)

```
String h = "INSERT INTO Student (firstName, lastName, marks)"  +
 "SELECT firstName, lastName, marks FROM old_student";
Query q = session.createQuery(h);
int result = q.executeUpdate();
System.out.println("Rows affected: " + result);
```

Advantages

Hibernate framework has the following advantages:

i. **Database Independent:** It does not require any database specific language.
ii. **Performance Efficient:** This framework uses cache internally. We don't need to rely upon DB for similar queries, we can cache them and just use them.
iii. **It is Open Source:** It is an open Source API for java which can be used to access the database with application.
iv. **Table Creation is automatic:** There is no need to create the table manually, it already has a feature to create the tables automatically.
v. **Joining is Simple:** Operation on data from multiple tables is simplified than JDBC. i,e Joins[6].
vi. **Statistics for Database status:** As discussed on point 2, it uses query cache and also it provides stats and status of the database.

vii. **Sessions are used to create Connection:** The database connections from an application are made using sessions which are helpful for saving the persistent object.

Disadvantage of Hibernate:-

i. Slower than JDBC:- JDBC is faster than Hibernate as hibernate require multiple SQL statements during runtime.
ii. Debugging and Error Control:-Sometimes debugging & performance tuning becomes difficult[2].
iii. Not handy for a project with few tables:-There is no need for this ORM framework if the tables are less, then JDBC can perform well.
iv. Boilerplate Code issue :-This means lots of code is written for some minimal functionality and Hibernate suffers from this.
v. Not easy for beginners :-As it has a lot of APIs to learn and remember, it's not suitable for beginners to learn it.
vi. Debugging is not easy:- Hibernate uses the concept of reordering of SQL statements, how they are executed and uses an identity map that makes debugging tough.
vii. Static in nature :-So during runtime, the named queries cannot be customised and not even the sorting.

Why Hibernate over JDBC?

Whenever we connect a relational database with an application of OOP style, The Hibernate solves the problem of Object-relational impedance mismatch. It usually occurs due to data type, transactional and manipulative differences.

Mapping of Objects:-You have to write code for mapping of data representation of object models with a relational database and the corresponding schema in JDBC. Whereas hibernate using XML itself maps Java class with the database tables or with the help of annotations [9]. Following program helps us to understand the above said statements.

From the below example we can see that by using JDBC while fetching the data we have to set every property of object, But in hibernate we just have to map the table with java class.

```
//JDBC
List<User>users=newArrayList<User>();
while(rs.next()) {
Useruser=newUser();
user.setUserId(rs.getString("UserId"));
user.setName(rs.getString("FirstName"));
user.setEmail(rs.getString("Email"));
users.add(user);
 }
//Hibernate
@Entity
@Table(name="user")
publicclassUserModel {
@Id
@GeneratedValue(strategy=GenerationType.IDENTITY)
privateBigIntegerid;
@NotEmpty
@Column(name="email", unique=true)
privateStringemail;
@NotEmpty
@Column(name="name")
privateStringname;
publicBigIntegergetId() {
returnthis.id;
   }
publicvoidsetId(BigIntegerid) {
this.id=id;
```

```
publicStringgetEmail() {
returnemail;
   }
publicvoidsetEmail(Stringemail) {
this.email=email;
   }
publicStringgetName() {
returnthis.name;
   }
publicvoidsetName(Stringname) {
  this.name=name;
     }
  }
```

Fig. 4: Mapping of table with Java Class

Advantages of Hibernate over JDBC:-

1. **HQL**:-It is the object oriented category of SQL.Database independent queries are generated by it. There is no need to write those queries which are database specific. Earlier, if we changed the database for our project, we had to change those SQL queries also; these things show the maintenance problems we had to come across [6].

2. **Independent of Database**:-There is no need to change the HQL queries when you switch from one DB to another, like oracle to MySQL etc. Hence switching is easy and friendly.

3. **Reduction in Code Repetition**:-There is less repeating amount of code in Hibernate unlike JDBC. Below is an example of the same.

```
Class.forName("com.mysql.jdbc.Driver");
Connectioncon=DriverManager.getConnection( "jdbc:mysql://localhost:3306/sonoo",
"root","root");
PreparedStatementstmt=con.prepareStatement("insert into Emp values(?,?)");
stmt.setInt(1,101);
stmt.setString(2,"Ratan");
stmt.executeUpdate();
con.close();
```

Fig. 5: Code Reduction

As we can see each statement occur only once, this is a short program, but in the bigger programs also, the repetition is very less.

For inserting a record in JDBC a prepared statement and will be created and also it will be set to every column. Larger number of columns mean larger the number of statements. We just have to save that object with persist to Hibernate.

4. **Avoiding Try-Catch Blocks:-**Hibernate handles the "try-catch" statements. How? All the JDBC exceptions are converted into unchecked exceptions. In JDBC, one will have to write "try-catch" blocks in the code. So, one doesn't have to waste her time writing and implementing the try-catch blocks by using Hibernate [5].

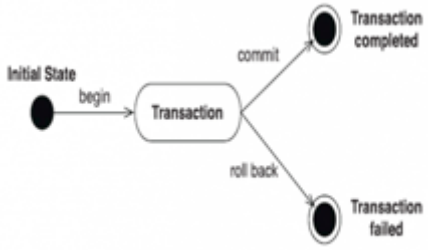

Fig. 6: Transaction Management

Transaction means, a group of many operations that are operated in just one task. The transaction is completed when all the operations in a group succeed. A transaction fails when an operation in a group fails. You need to commit in JDBC if a transaction is successfully completed, otherwise perform rollback operation.

- con.commit();
- con.rollback();

These are implicitly provided by Hibernate, you don't need to write the above statements there.

5. **Associations**:-Using Hibernate, it is very easy to use associations between the tables. Like one-one, one-many, many-one, many-many are easy to achieve by using annotations.

VII. CONCLUSION

This paper has illustrated various aspects of hibernate technology over the old JDBC. We had discussed concept of POJO class having getters and setters that is useful in the Hibernate. It reduces the amount of time and efforts putted in writing code, test, and deploy applications. Hibernate is a most powerful technology with high performance, ORM solution for Java along with mapping objects to a database. The integrated environment of Hibernate framework will provide a better database access mechanism and understand ability than the other frameworks. The integrated framework will provide a better database access with better frontend design.

VIII. ACKNOWLEDGMENT

The authors are thankful to the Management of CGC College of Engineering and anonymous reviewers of the paper.

REFERENCES

[1] S. Barahmand. Benchmarking Interactive Social Networking Actions, Ph.D. thesis, Computer Science Department, USC, 2014.
[2] S. Barahmand and S. Ghandeharizadeh. BG: A Benchmark to Evaluate Interactive Social Networking Actions. CIDR, January 2013.
[3] S. Barahmand, S. Ghandeharizadeh, and J. Yap.A Comparison of Two Physical Data Designs for Interactive Social Networking Actions.CIKM, 2013.
[4] E. Cecchet, J. Marguerite, and W. Zwaenepoel.Performance and scalability of EJB applications. In OOPSLA, pages 246–261, 2002.
[5] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking Cloud Serving Systems with YCSB.In Cloud Computing, 2010.
[6] Transaction Processing Performance Council. TPC Benchmarks, http://www.tpc.org/information/benchmarks.asp.
[7] C. Bauer G. King, M. R. Andersen, E. Bernard, S. Ebersole, and H. Ferentschik. HIBERNATE - Relational Persistence for Idiomatic Java. Red Hat Middleware, Inc., June 2009.
[8] ObjectWeb. RUBBoS: Bulletin Board Benchmark, http://jmob.ow2.org/rubbos.html.
[9] S. Patil, M. Polte, K. Ren, W. Tantisiriroj, L. Xiao, J. Lopez, ´ G. Gibson, A. Fuchs, and B. Rinaldi. YCSB++: Benchmarking and Performance Debugging Advanced Features in Scalable Table Stores. In Cloud Computing, New York, NY, USA, 2011.ACM.