Adv.JAVA
Practical Lab_01

Crud Operations using hibernate

Submitted By :-

Name:- Praveer Kishore

Enrollment No.:- A45349521001

Cousre:- BCA+MCA (Dual)

Semester:- 6th 'B'

About Our Project:-

hibernate CRUD Operations for Student Management System:

The JDBC CRUD Operations for Student Management System project is a Java application designed to perform basic Create, Read, Update, and Delete (CRUD) operations on a student table within a MySQL database. The project utilizes Java Database Connectivity (JDBC) to interact with the database. It provides a simple yet effective way to manage student data within a relational database management system.

Key Features:

1. **Create Operation:**

   Allows users to create a new student record by specifying the student's ID, name, marks, age, and gender. The application establishes a connection to the database, executes an SQL query to create a new table if it does not exist, and inserts the provided student data into the table.

2. **Read Operation:**

   Enables users to retrieve and display student records from the database. Users can view all students' details or search for specific students based on their ID, name, or other attributes. The application establishes a connection to the database, executes an SQL query to fetch data from the table, and displays it to the user.

3. **Update Operation:**

   Allows users to modify existing student records by updating their information such as name, marks, age, or gender. Users can specify the student ID of the record they want to update along with the new data. The application

establishes a connection to the database, executes an SQL query to update the corresponding record in the table.

4. Delete Operation:

Provides users with the capability to delete unwanted student records from the database. Users can specify the student ID of the record they wish to delete, and the application will remove it from the table. The application establishes a connection to the database, executes an SQL query to delete the specified record from the table.

5. HQL :

The provided HQL code demonstrates CRUD operations using Hibernate Query Language (HQL). Firstly, it inserts a new student record into the database. Then, it executes a select query to retrieve all student records from the database and prints them. Next, it updates the name of a student with a specific ID using an update query, and it prints the number of rows affected. Finally, it deletes a student record with a specific ID using a delete query and prints the number of rows affected. The code ensures data manipulation in the SQL table through HQL queries, providing a high-level, object-oriented approach to interact with the database.

Tech Stack Used in this CRUD operations:-

1. Integrated Development Environment (IDE):

Eclipse: The project is developed using the Eclipse IDE, providing a robust and user-friendly development environment for Java applications.

2. Database Management System:

MySQL Workbench: MySQL Workbench is used as the database management system to design, model, create, and manage the MySQL database that stores customer information.

3. **Programming Language:**

Java: The entire application is written in Java, a versatile and platform-independent programming language. Java is used for its object-oriented principles, extensive libraries, and wide industry adoption.

4. **Java Database Connectivity (JDBC) Architecture:**

JDBC is employed for database connectivity, allowing Java applications to interact with relational databases like MySQL. The JDBC architecture is utilized to establish connections, execute SQL queries, and manage data retrieval and updates.

Project in Eclipse:-

hibernatecreate

App.java :-

```
package student.hibernatecreate;


import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;


public class App{
   public static void main(String[] args) {
      Configuration config = new Configuration();
      config.configure("hibernate.cfg.xml");


      SessionFactory sf = config.buildSessionFactory();
      Session session = sf.openSession();


      // Insert
      student s1 = new student();
      s1.setName("Praveer");
      s1.setAge(21);
      s1.setMarks(85.25);
      s1.setGender("male");


      Transaction tx = session.beginTransaction();
```

```
        session.save(s1);
        tx.commit();

        session.close();
        sf.close();
    }
}
```

hibernateread

App.java :-

```
package student.hibernateread;

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

public class App
{
    public static void main(String[] args) {
        Configuration config = new Configuration();
        config.configure("hibernate.cfg.xml");

        SessionFactory sf = config.buildSessionFactory();
        Session session = sf.openSession();

        // Fetch
        student fetchedStudent = session.get(student.class, 1);
```

```
            System.out.println(fetchedStudent);


            session.close();
            sf.close();
        }
}
```

hibernateupdate

App.java :-

```
package student.hibernateupdate;


import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;


public class App
{
    public static void main(String[] args) {
      Configuration config = new Configuration();
      config.configure("hibernate.cfg.xml");


      SessionFactory sf = config.buildSessionFactory();
      Session session = sf.openSession();


      // Edit or modify the data (transaction commit)
```

```
        Transaction tx = session.beginTransaction();
        student sToUpdate = session.get(student.class, 1);
        sToUpdate.setName("Golu");
        sToUpdate.setAge(22);
        session.saveOrUpdate(sToUpdate);
        tx.commit();

        session.close();
        sf.close();
    }
}
```

hibernatedelete

App.java :-

```
package student.hibernatedelete;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

public class App
{
        public static void main(String[] args) {
    Configuration config = new Configuration();
    config.configure("hibernate.cfg.xml");
```

```
        SessionFactory sf = config.buildSessionFactory();
        Session session = sf.openSession();

        // Delete (transaction commit)
        Transaction tx = session.beginTransaction();
        student sToDelete = session.get(student.class, 1);
        session.delete(sToDelete);
        tx.commit();

        session.close();
        sf.close();
    }
}
```

hibernateHQL

App.java:-

```
        package student.hibernatehql;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;
import java.util.List;

public class App
{
        public static void main(String[] args) {
```

```
Configuration config = new Configuration();
config.configure("hibernate.cfg.xml");

SessionFactory sf = config.buildSessionFactory();
Session session = sf.openSession();

Transaction tx = session.beginTransaction();

// Insert
student s1 = new student();
s1.setName("Praveer");
s1.setAge(21);
s1.setMarks(85.25);
s1.setGender("male");
session.save(s1);

// HQL Select
@SuppressWarnings("unchecked")
        List<student> students = session.createQuery("from student").list();
for (student student : students) {
   System.out.println(student);
}

// HQL Update
int updatedRows = session.createQuery("update student set name = :name where id = :id")
                .setParameter("name", "Golu")
                .setParameter("id", 1)
                .executeUpdate();
```

```
        System.out.println("Updated " + updatedRows + " rows.");


        // HQL Delete
        int deletedRows = session.createQuery("delete from student where id = :id")
                        .setParameter("id", 1)
                        .executeUpdate();
        System.out.println("Deleted " + deletedRows + " rows.");


        tx.commit();
        session.close();
        sf.close();
    }
}
```
