22/06/24

Chaitra

Hibernate Assignment

I installed all the necessary tools – Eclipse IDE, MySQL 8.0.37, and Apache maven.

I created database- hibernate_example and table “EMPLOYEE”.

```
mysql> CREATE DATABASE hibernate_example;
Query OK, 1 row affected (0.89 sec)

mysql> USE hibernate_example
```

```
mysql> CREATE TABLE EMPLOYEE
    -> (
    ->    id INT AUTO_INCREMENT PRIMARY KEY,
    -> NAME VARCHAR(255) NOT NULL,
    -> SALARY INT NOT NULL
    -> );
Query OK, 0 rows affected (2.72 sec)

mysql> SHOW TABLES;
+------------------------------+
| Tables_in_hibernate_example |
+------------------------------+
| employee                     |
+------------------------------+
1 row in set (0.39 sec)
```

Project Structure

```
Hibernate/
├── src/
│   ├── main/
│   │   ├── java/
│   │   │   └── com/
│   │   │       └── example/
│   │   │           ├── Employee1.java
│   │   │           ├── MainApp.java
│   │   ├── resources/
│   │       ├── hibernate.cfg.xml
│   │       └── employee.hbm.xml
├── pom.xml
```

Maven-

Maven is a build automation and project management tool used primarily for Java projects. It simplifies the build process, dependency management, and project configuration through the use of a project object model (POM) file.

Key Purposes of Maven:

Dependency Management: Automatically downloads Java libraries and Maven plugins from one or more repositories such as the Maven Central Repository.

Manages dependencies and ensures that the correct versions are used.

Build Automation: Automates the process of compiling source code, running tests, packaging the application, and deploying it to production or other environments.

Project Management: Provides a uniform build system, making it easier to understand the project structure and build lifecycle.

Facilitates project information and documentation through reporting features.

Implementation-

Open Eclipse and create a new Java Project.

if you're using Maven:

After creating java project, Right-click on the project -> Configure -> Convert to Maven Project.

Add the dependencies in your pom.xml.

1.Create Persistent class-

A persistent class is a simple Java class that is mapped to a table in your database. Each instance of this class represents a row in the table.

HibernateMf/pom.xml | MainApp.java | Employee1.java | hibernate.cfg.xml | employee.hbm.xml

```java
package com.exmaple.model2;

public class Employee1 {
     private int id;
        private String firstName;
        private String lastName;
        private int salary;
        private String name; // Add this line if 'name' is required

        public Employee1() {}

        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;
        }

        public int getSalary() {
```

2.Create the mapping file Persistent class

The mapping file is an XML file that describes how the persistent class maps to the database table. This file is named as employee.hbm.xml.

HibernateMf/pom.xml | MainApp.java | Employee1.java | hibernate.cfg.xml | employee.hbm.xml

```
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="com.exmaple.model2.Employee1" table="EMPLOYEE">
        <id name="id" type="int">
            <generator class="native"/>
        </id>
        <property name="firstName" type="string"/>
        <property name="lastName" type="string"/>
        <property name="salary" type="int"/>
         <property name="name" type="string"/>
    </class>
</hibernate-mapping>
```

3.Create the Configuration file –

The configuration file (hibernate.cfg.xml) provides Hibernate with information about the database and the mapping files.

HibernateMf/pom.xml | MainApp.java | Employee1.java | hibernate.cfg.xml | employee.hbm.xml

```
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
    "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.cj.jdbc.Driver</property>
        <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/hibernate_example</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password">chaitra@2024</property>

        <!-- JDBC connection pool settings -->
        <property name="hibernate.c3p0.min_size">5</property>
        <property name="hibernate.c3p0.max_size">20</property>
        <property name="hibernate.c3p0.timeout">300</property>
        <property name="hibernate.c3p0.max_statements">50</property>
        <property name="hibernate.c3p0.idle_test_period">3000</property>

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

        <!-- Enable Hibernate's automatic session context management -->
        <property name="hibernate.current_session_context_class">thread</property>
        <property name="hibernate.hbm2ddl.auto">update</property>
        <property name="hibernate.show_sql">true</property>

        <mapping resource="employee.hbm.xml"/>
    </session-factory>
</hibernate-configuration>
```

4.Create the class which retrieves or persistent the object –

This class uses Hibernate to save or retrieve objects from the database.

```
package com.exmaple.model2;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;
import org.hibernate.query.Query;
import java.util.List;

public class MainApp {

    public static void main(String[] args) {

                // Create SessionFactory
                SessionFactory factory = new Configuration().configure("hibernate.cfg.xml").buildSessionFactory();

                // Create a session
                Session session = factory.openSession();
                session.beginTransaction();

                // Check if an Employee named "John Doe" exists
                String firstNameToFind = "John";
                String lastNameToFind = "Doe";
                Query<Employee1> query = session.createQuery("FROM Employee1 WHERE firstName = :firstName AND lastName = :lastName", Employee1.class)
                query.setParameter("firstName", firstNameToFind);
                query.setParameter("lastName", lastNameToFind);
                List<Employee1> results = query.list();

                if (!results.isEmpty()) {
                    // Update the salary of the first Employee found with name "John Doe"
                    Employee1 employeeToUpdate = results.get(0);
                    employeeToUpdate.setSalary(employeeToUpdate.getSalary() + 500); // Example: Increase salary by 500
                    session.update(employeeToUpdate);
                    System.out.println("Updated existing employee with name: " + firstNameToFind + " " + lastNameToFind);
                } else {
                    System.out.println("Employee with name: " + firstNameToFind + " " + lastNameToFind + " not found."); // Handle case if not found
                }
```

```
                // Create and persist a new Employee object with name "Dhruva Kumar"
                Employee1 newEmployee = new Employee1();
                newEmployee.setFirstName("Akshay");
                newEmployee.setLastName("Gonal");
                newEmployee.setSalary(10000);
                session.save(newEmployee);
                System.out.println("Added new employee with name: Akshay Gonal");

                newEmployee.setSalary(newEmployee.getSalary() + 500);
                session.update(newEmployee);

                // Commit transaction
                session.getTransaction().commit();

                // Close the session
                session.close();
                factory.close();

    }
```

5.Load the JAR file using maven

In your pom.xml file, add the required dependencies to include Hibernate and your JDBC driver.

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId>hibernate-example</artifactId>
  <version>1.0-SNAPSHOT</version>
  <name>HibernateMf</name>
  <url>http://maven.apache.org</url>

  <properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
    <hibernate.version>5.4.32.Final</hibernate.version>
    <hibernate.validator.version>6.2.0.Final</hibernate.validator.version>
    <mysql.connector.version>8.0.33</mysql.connector.version>
    <junit.version>4.13.1</junit.version>
  </properties>

  <dependencies>
    <!-- Hibernate Core -->
    <dependency>
      <groupId>org.hibernate</groupId>
      <artifactId>hibernate-core</artifactId>
      <version>${hibernate.version}</version>
    </dependency>

    <!-- Hibernate Validator (optional) -->
    <dependency>
      <groupId>org.hibernate.validator</groupId>
      <artifactId>hibernate-validator</artifactId>
      <version>${hibernate.validator.version}</version>
    </dependency>
```

```xml
      </plugin>
      <!-- Maven Exec Plugin for running Java classes -->
      <plugin>
        <groupId>org.codehaus.mojo</groupId>
        <artifactId>exec-maven-plugin</artifactId>
        <version>3.0.0</version>
        <configuration>
          <mainClass>com.exmaple.model2.MainApp</mainClass>
        </configuration>
        <executions>
          <execution>
            <goals>
              <goal>java</goal>
            </goals>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
</project>
```

```xml
    <!-- MySQL Connector -->
    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>${mysql.connector.version}</version>
    </dependency>

    <!-- JUnit for testing (optional) -->
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>${junit.version}</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
    <resources>
      <resource>
        <directory>src/resources</directory>
      </resource>
      <resource>
        <directory>src/main/resources</directory>
        <includes>
          <include>**/*.xml</include>
        </includes>
      </resource>
    </resources>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.1</version>
        <configuration>
```

6.Run the application

Finally, run your application to test if everything works correctly. You can run your `MianApp.java` class by right-clicking the class and selecting `Run As` -> `Java Application`.

```
Console ×
<terminated> MainApp [Java Application] C:\Program Files\Java\jdk-20\bin\javaw.exe (Jun 24, 2024, 12:07:11 AM – 12:07:17 AM) [pid: 11924]
Jun 24, 2024 12:07:12 AM org.hibernate.Version logVersion
INFO: HHH000412: Hibernate ORM core version 5.4.32.Final
Jun 24, 2024 12:07:13 AM org.hibernate.annotations.common.reflection.java.JavaReflectionManager <clinit>
INFO: HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
Jun 24, 2024 12:07:14 AM org.hibernate.engine.jdbc.connections.internal.ConnectionProviderInitiator instantiateC3p0Provider
WARN: HHH000022: c3p0 properties were encountered, but the c3p0 provider class was not found on the classpath; these proper
Jun 24, 2024 12:07:14 AM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure
WARN: HHH10001002: Using Hibernate built-in connection pool (not for production use!)
Jun 24, 2024 12:07:14 AM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001005: using driver [com.mysql.cj.jdbc.Driver] at URL [jdbc:mysql://localhost:3306/hibernate_example]
Jun 24, 2024 12:07:14 AM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001001: Connection properties: {password=****, user=root}
Jun 24, 2024 12:07:14 AM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001003: Autocommit mode: false
Jun 24, 2024 12:07:14 AM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PooledConnectio
INFO: HHH000115: Hibernate connection pool size: 20 (min=1)
Jun 24, 2024 12:07:15 AM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect
Jun 24, 2024 12:07:16 AM org.hibernate.validator.internal.util.Version <clinit>
INFO: HV000001: Hibernate Validator 6.2.0.Final
Jun 24, 2024 12:07:16 AM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsola
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentIni
Hibernate: select employee1x0_.id as id1_0_, employee1x0_.firstName as firstnam2_0_, employee1x0_.lastName as lastname3_0_,
Employee with name: John Doe not found.
Hibernate: insert into EMPLOYEE (firstName, lastName, salary, name) values (?, ?, ?, ?)
Added new employee with name: Akshay Gonal
Hibernate: update EMPLOYEE set firstName=?, lastName=?, salary=?, name=? where id=?
Jun 24, 2024 12:07:17 AM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PoolState stop
INFO: HHH10001008: Cleaning up connection pool [jdbc:mysql://localhost:3306/hibernate_example]
```

```
mysql> SELECT * FROM EMPLOYEE;
+----+------+--------+-----------+----------+
| id | NAME | SALARY | firstName | lastName |
+----+------+--------+-----------+----------+
|  2 | NULL |  10000 | Chaitra   | Gonal    |
|  3 | NULL |  10000 | Chaitra   | Gonal    |
|  4 | NULL |  10500 | Chaitra   | Gonal    |
|  5 | NULL |  10500 | Akshay    | Gonal    |
+----+------+--------+-----------+----------+
4 rows in set (0.01 sec)

mysql> Select From EMPLOYEE WHERE SALARY >= 10500;
ERROR 1064 (42000): You have an error in your SQL syntax; che
t syntax to use near 'From EMPLOYEE WHERE SALARY >= 10500' at
mysql> SELECT * FROM EMPLOYEE WHERE SALARY >= 10500;
+----+------+--------+-----------+----------+
| id | NAME | SALARY | firstName | lastName |
+----+------+--------+-----------+----------+
|  4 | NULL |  10500 | Chaitra   | Gonal    |
|  5 | NULL |  10500 | Akshay    | Gonal    |
+----+------+--------+-----------+----------+
2 rows in set (0.10 sec)
```

Updated table using hibernate