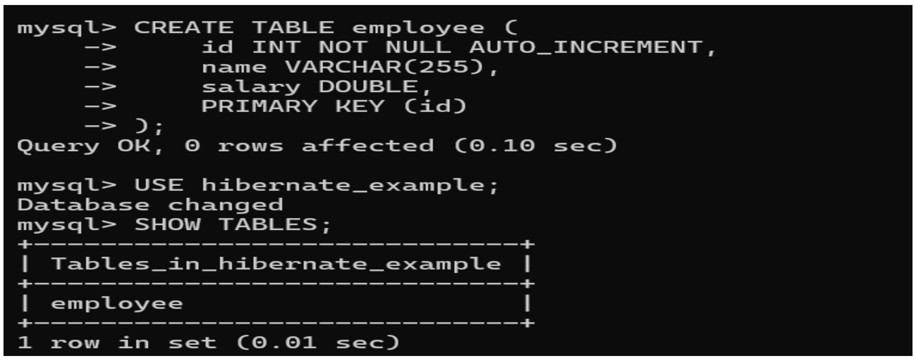
Dhruva Kumar S

ASSIGNMENT6: HibernateExampleProject

Initially, I had installed JDK and MySQL 8.0.37, and Install Apache Maven. I also created a database named `hibernate_example` and a table named `employee`.

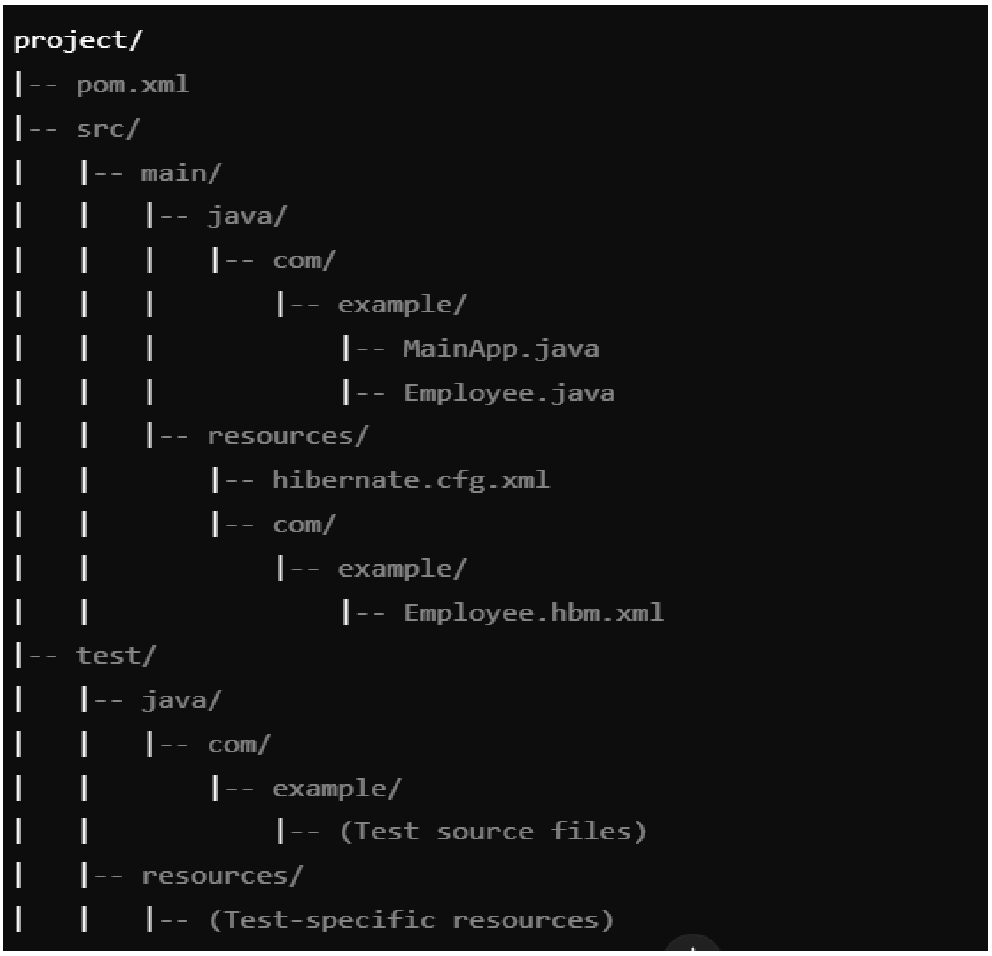


```
mysql> CREATE TABLE employee (
    ->     id INT NOT NULL AUTO_INCREMENT,
    ->     name VARCHAR(255),
    ->     salary DOUBLE,
    ->     PRIMARY KEY (id)
    -> );
Query OK, 0 rows affected (0.10 sec)

mysql> USE hibernate_example;
Database changed
mysql> SHOW TABLES;
+-----------------------------+
| Tables_in_hibernate_example |
+-----------------------------+
| employee                    |
+-----------------------------+
1 row in set (0.01 sec)
```

SQL Database

Maven: Maven is a build automation and dependency management tool primarily used for Java projects. It simplifies and standardizes the project build process by managing project dependencies, compiling source code, packaging compiled code into distributable units such as JAR files, and facilitating project documentation and reporting.



```
project/
|-- pom.xml
|-- src/
|   |-- main/
|   |   |-- java/
|   |   |   |-- com/
|   |   |   |   |-- example/
|   |   |   |       |-- MainApp.java
|   |   |   |       |-- Employee.java
|   |   |-- resources/
|   |       |-- hibernate.cfg.xml
|   |       |-- com/
|   |           |-- example/
|   |               |-- Employee.hbm.xml
|-- test/
|   |-- java/
|   |   |-- com/
|   |       |-- example/
|   |           |-- (Test source files)
|   |-- resources/
|   |   |-- (Test-specific resources)
```

Structure of the project

Dhruva Kumar S

Initially this database having employee table and it is empty

Question 1: Create a Persistent Class

How do I create a persistent class in VS Code?

Answer: Create a file named Employee.java in the src/main/java/com/example directory with the necessary fields, constructors, getters, and setters.

Question 2: Create the Mapping File for Persistent Class

How do I create the mapping file for a persistent class?

Answer: Create a file named Employee.hbm.xml in the src/main/resources directory with the appropriate XML mapping configuration for the Employee class.

Question 3: Create the Configuration File

How do I create the Hibernate configuration file?

Answer: Create a file named hibernate.cfg.xml in the src/main/resources directory with the database connection settings and Hibernate properties.

Question 4: Create the Class Which Retrieves or Persists the Object

How do I create a class that retrieves or persists objects?

Answer: Rename App.java to MainApp.java in the src/main/java/com/example directory and implement the code to open a Hibernate session, create an Employee object, and persist it to the database.

Question 5: Load the JAR File

How do I load the necessary JAR files?

Answer: Ensure that your pom.xml file includes dependencies for Hibernate and MySQL. Maven will automatically download and manage these dependencies.

Question 6: Run the Application

How do I run the application in VS Code?

Answer: Open the terminal, navigate to your project root, compile the project using mvn compile, and run the application using mvn exec:java -Dexec.mainClass="com.example.MainApp".

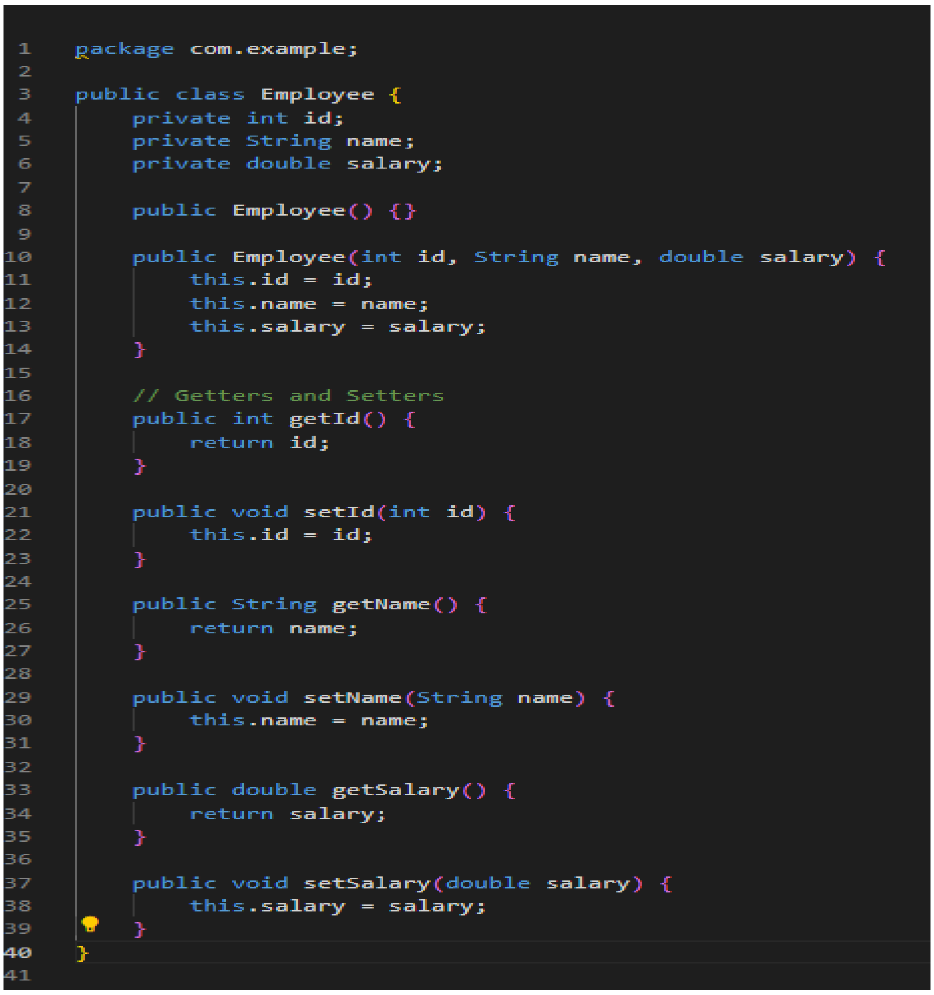
Dhruva Kumar S

1. Create a Persistent Class

A persistent class is a simple Java class (also known as a POJO - Plain Old Java Object) that represents an entity that will be mapped to a database table.

Step:

- In your Maven project directory (hibernate-example/src/main/java/com/example), create a file named Employee.java and add the following code:



```java
package com.example;

public class Employee {
    private int id;
    private String name;
    private double salary;

    public Employee() {}

    public Employee(int id, String name, double salary) {
        this.id = id;
        this.name = name;
        this.salary = salary;
    }

    // Getters and Setters
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public double getSalary() {
        return salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }
}
```

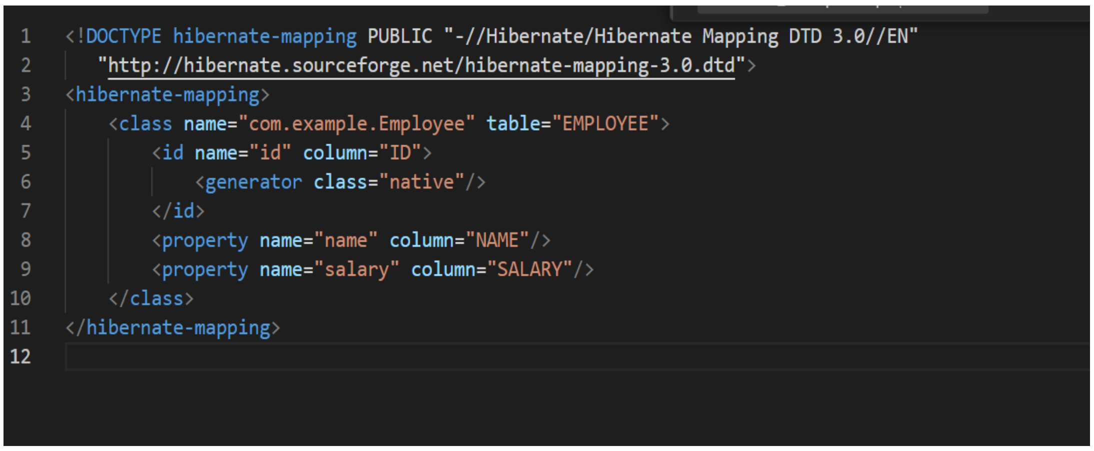
Fig1: Employee.java

2. Create the Mapping File for Persistent Class

A mapping file tells Hibernate how to map the class to a database table.

Step:

- In your Maven project directory (hibernate-example/src/main/resources), create a file named Employee.hbm.xml and add the following configuration:



```xml
<!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.example.Employee" table="EMPLOYEE">
        <id name="id" column="ID">
            <generator class="native"/>
        </id>
        <property name="name" column="NAME"/>
        <property name="salary" column="SALARY"/>
    </class>
</hibernate-mapping>

```

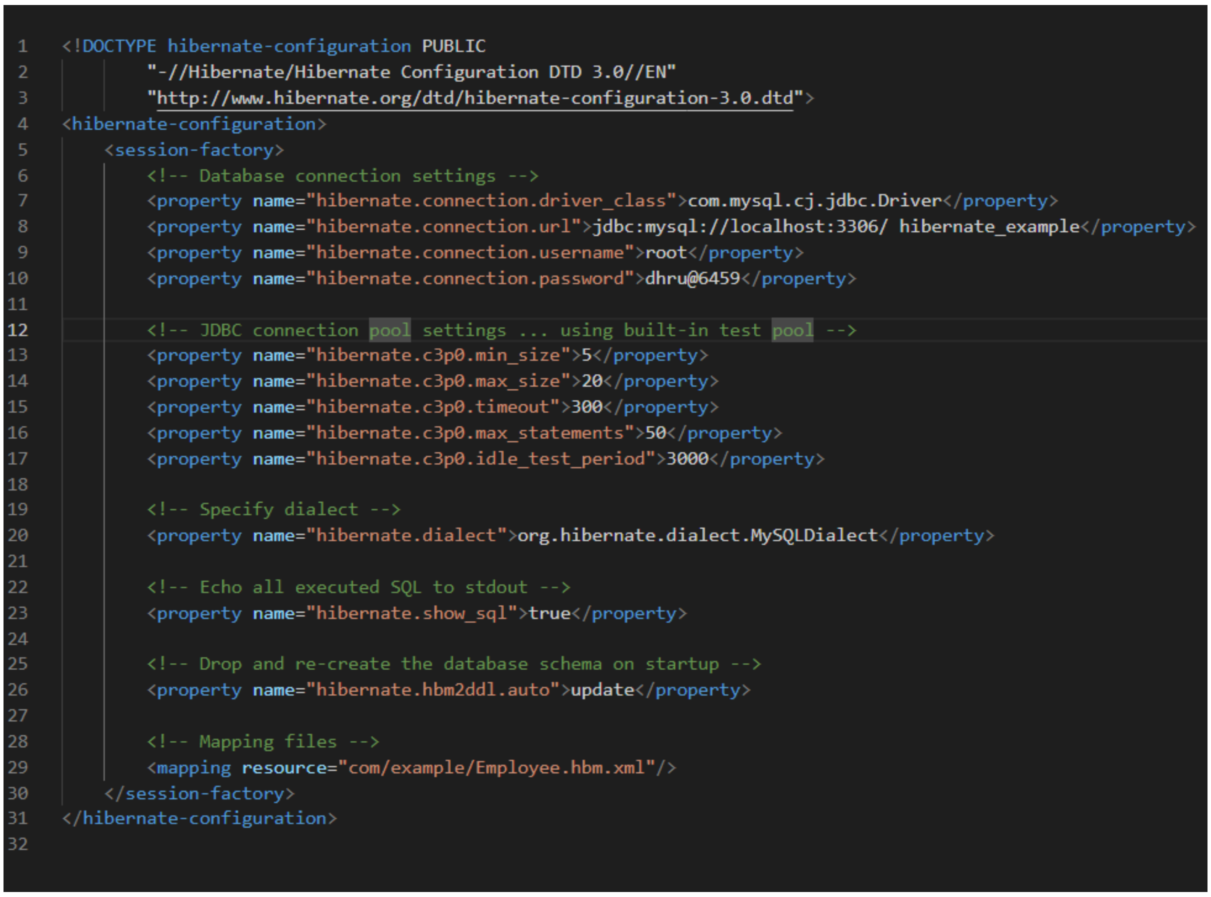
Fig2: **Mapping File for Persistent Class**

3. Create the Configuration File

The Hibernate configuration file (hibernate.cfg.xml) contains database connection settings and other Hibernate properties.

Step:

- In your Maven project directory (hibernate-example/src/main/resources), create a file named hibernate.cfg.xml and add the following configuration:



```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>
        <!-- Database connection settings -->
        <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">dhru@6459</property>

        <!-- JDBC connection pool settings ... using built-in test pool -->
        <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>

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

        <!-- Echo all executed SQL to stdout -->
        <property name="hibernate.show_sql">true</property>

        <!-- Drop and re-create the database schema on startup -->
        <property name="hibernate.hbm2ddl.auto">update</property>

        <!-- Mapping files -->
        <mapping resource="com/example/Employee.hbm.xml"/>
    </session-factory>
</hibernate-configuration>
```

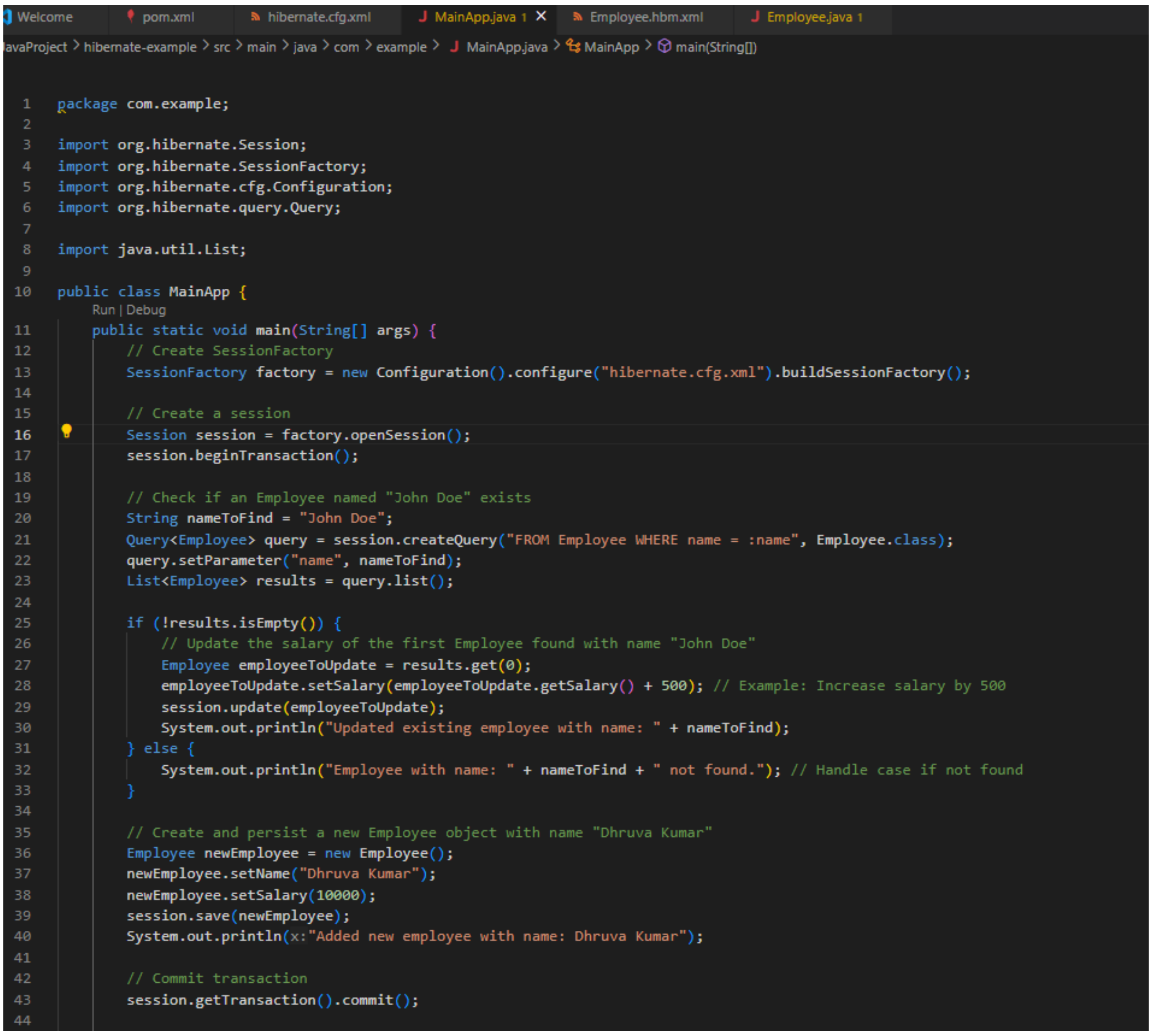
Fig3: **Configuration File**

4. Create the Class Which Retrieves or Persists the Object

This is the main application class that uses Hibernate to persist an object to the database.

Step:

- In your Maven project directory (hibernate-example/src/main/java/com/example), rename App.java to MainApp.java and add the following code:


Welcome
pom.xml
hibernate.cfg.xml
MainApp.java 1
Employee.hbm.xml
Employee.java 1
avaProject > hibernate-example > src > main > java > com > example > MainApp.java > MainApp > main(String[])
Run | Debug

```java
package com.example;

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 nameToFind = "John Doe";
        Query<Employee> query = session.createQuery("FROM Employee WHERE name = :name", Employee.class);
        query.setParameter("name", nameToFind);
        List<Employee> results = query.list();

        if (!results.isEmpty()) {
            // Update the salary of the first Employee found with name "John Doe"
            Employee 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: " + nameToFind);
        } else {
            System.out.println("Employee with name: " + nameToFind + " not found."); // Handle case if not found
        }

        // Create and persist a new Employee object with name "Dhruva Kumar"
        Employee newEmployee = new Employee();
        newEmployee.setName("Dhruva Kumar");
        newEmployee.setSalary(10000);
        session.save(newEmployee);
        System.out.println(x:"Added new employee with name: Dhruva Kumar");

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

```

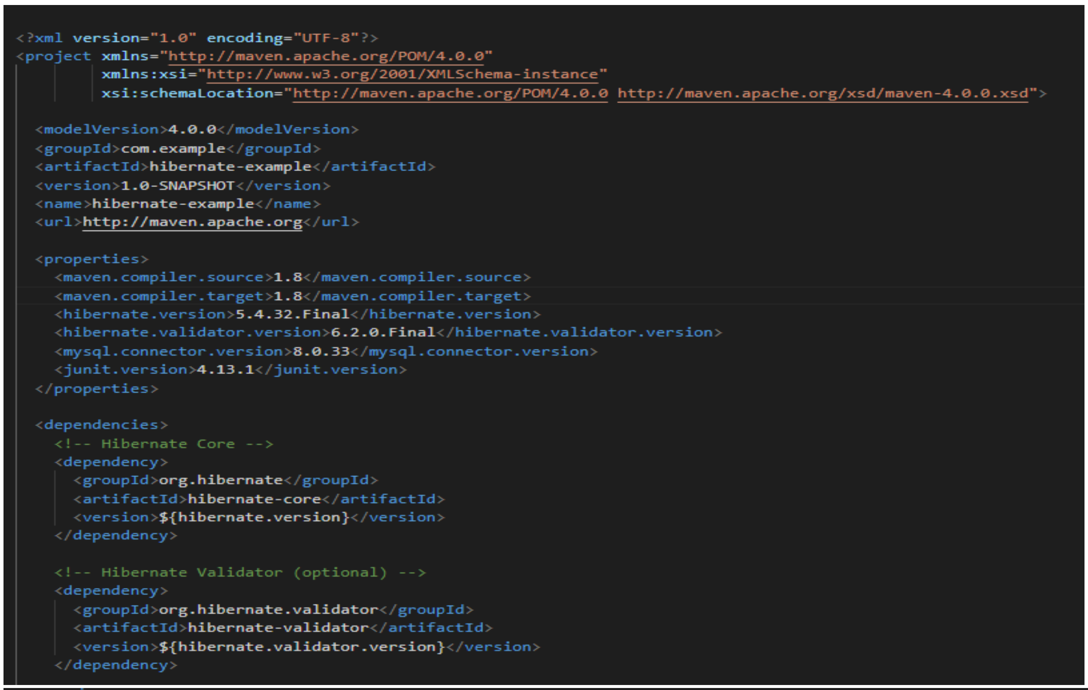
Fig4: MainApp.java

5. Load the JAR File

Ensure that the necessary dependencies are included in your pom.xml file. Maven will automatically download and manage the required JAR files.

Step:

- Our pom.xml should already include dependencies for Hibernate and MySQL:



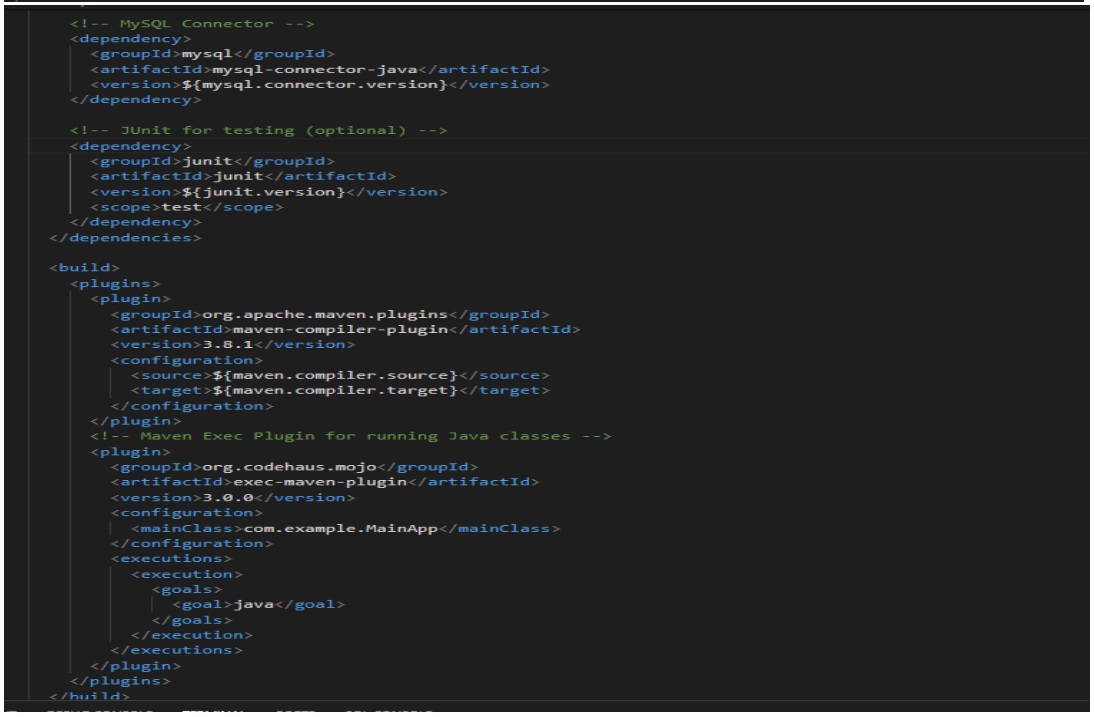
```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId>hibernate-example</artifactId>
  <version>1.0-SNAPSHOT</version>
  <name>hibernate-example</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
    <!-- 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>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.1</version>
        <configuration>
          <source>${maven.compiler.source}</source>
          <target>${maven.compiler.target}</target>
        </configuration>
      </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.example.MainApp</mainClass>
        </configuration>
        <executions>
          <execution>
            <goals>
              <goal>java</goal>
            </goals>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
```

Fig5:Pom.xml

6. Run the Application

Compile and run your Maven project using the following commands:

Step:

- **Open the terminal in VS Code and navigate to your project root:**

```
bash
Copy code
cd path/to/your/project/hibernate-example
```

- **Compile the project:**

```
mvn compile
```

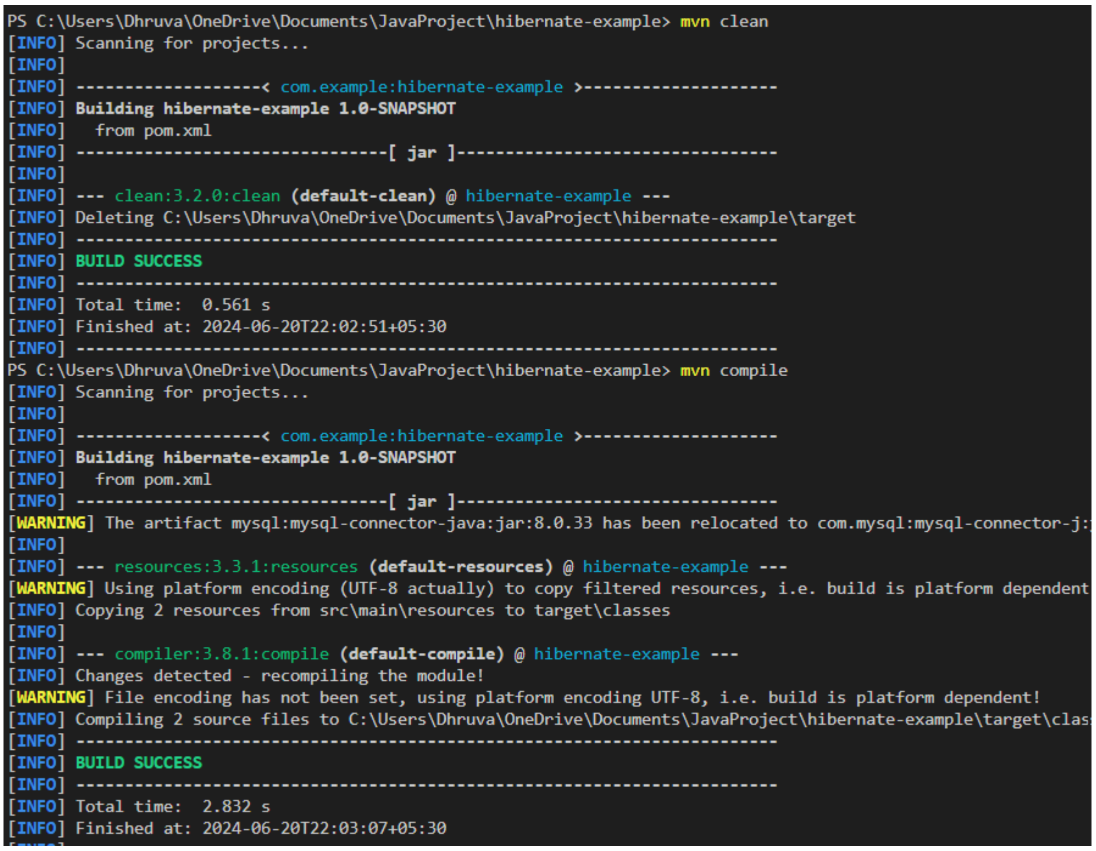
- **Run the application:**

```
mvn exec:java -Dexec.mainClass="com.example.MainApp"
```

Results:

 Maven Build Process:

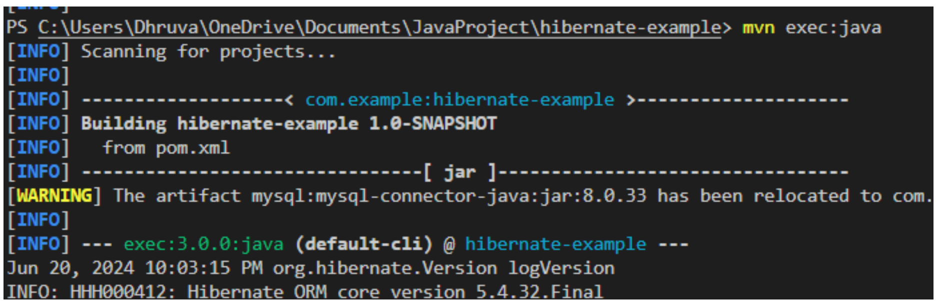
- **Clean Phase:** Maven starts by cleaning the target directory where previous build artifacts are deleted.
- **Compile Phase:** Compiles the Java source code (MainApp.java and Employee.java) into bytecode (*.class files) under target/classes.



```
PS C:\Users\Dhruva\OneDrive\Documents\JavaProject\hibernate-example> mvn clean
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< com.example:hibernate-example >-------------------
[INFO] Building hibernate-example 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]--------------------------------
[INFO]
[INFO] --- clean:3.2.0:clean (default-clean) @ hibernate-example ---
[INFO] Deleting C:\Users\Dhruva\OneDrive\Documents\JavaProject\hibernate-example\target
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  0.561 s
[INFO] Finished at: 2024-06-20T22:02:51+05:30
[INFO] ------------------------------------------------------------------------
PS C:\Users\Dhruva\OneDrive\Documents\JavaProject\hibernate-example> mvn compile
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< com.example:hibernate-example >-------------------
[INFO] Building hibernate-example 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]--------------------------------
[WARNING] The artifact mysql:mysql-connector-java:jar:8.0.33 has been relocated to com.mysql:mysql-connector-j:
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ hibernate-example ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent
[INFO] Copying 2 resources from src\main\resources to target\classes
[INFO]
[INFO] --- compiler:3.8.1:compile (default-compile) @ hibernate-example ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 2 source files to C:\Users\Dhruva\OneDrive\Documents\JavaProject\hibernate-example\target\clas
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.832 s
[INFO] Finished at: 2024-06-20T22:03:07+05:30
```

 Hibernate Initialization:

- **Configuration:** Hibernate initializes using settings from hibernate.cfg.xml.
- **Connection Setup:** Establishes a connection to the MySQL database (jdbc:mysql://localhost:3306/hibernate_example) using JDBC.
- **Connection Pooling:** Configures a built-in connection pool (not for production) with settings specified in hibernate.cfg.xml.



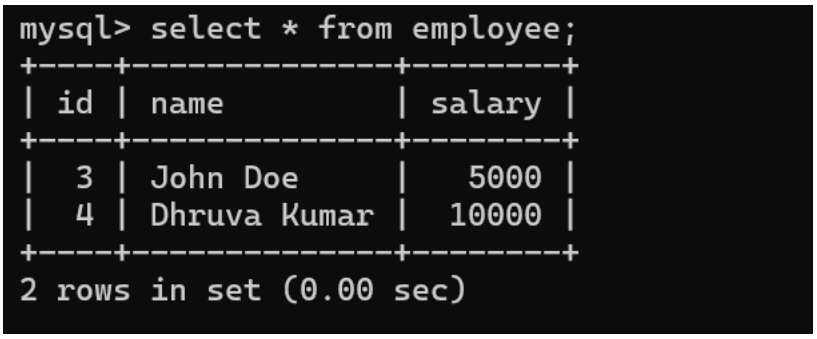
```
PS C:\Users\Dhruva\OneDrive\Documents\JavaProject\hibernate-example> mvn exec:java
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< com.example:hibernate-example >-------------------
[INFO] Building hibernate-example 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]--------------------------------
[WARNING] The artifact mysql:mysql-connector-java:jar:8.0.33 has been relocated to com.
[INFO]
[INFO] --- exec:3.0.0:java (default-cli) @ hibernate-example ---
Jun 20, 2024 10:03:15 PM org.hibernate.Version logVersion
INFO: HHH000412: Hibernate ORM core version 5.4.32.Final
```

Execution Flow:

- **Session and Transaction:** Opens a Hibernate session and starts a transaction.
- **Data Operations:**
  - **Update:** Executes an update query (update EMPLOYEE set NAME=?, SALARY=? where ID=?) for an existing employee named "John Doe" with updated salary.
  - **Insert:** Inserts a new record into the EMPLOYEE table with name "Dhruva Kumar" and salary.
- **Commit Transaction:** Commits the transaction, saving changes to the database.
- **Close Resources:** Closes the Hibernate session and releases database resources.

Hibernate SQL Output:

- **Hibernate SQL Logging:** Outputs SQL statements executed by Hibernate (select, insert, update).
- **Information Messages:** Shows Hibernate version, dialect used (MySQLDialect), and connection pool details.



```
mysql> select * from employee;
+----+--------------+--------+
| id | name         | salary |
+----+--------------+--------+
|  3 | John Doe     |   5000 |
|  4 | Dhruva Kumar |  10000 |
+----+--------------+--------+
2 rows in set (0.00 sec)
```

Fig6: Updated table by using Hibernate