Hands-On: Implementing ORM with Hibernate using XML and Annotation Configuration

Prerequisites

- Java JDK 8 or higher installed on your computer
- A Java IDE, such as Eclipse or IntelliJ IDEA
- Hibernate ORM framework installed in your project
- MySQL database server installed on your computer

Set up the project

1. Create a new Maven project in your IDE
2. Add the following dependencies to your **`pom.xml`** file:

```
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-core</artifactId>
  <version>5.5.7.Final</version>
</dependency>
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>8.0.27</version>
</dependency>
```

XML Configuration

Create a database

1. Open MySQL Workbench and create a new database called **`hibernate_example`**.
2. Create a new table called **`employee`** with the following fields:

```
CREATE TABLE employee (
  id INT(11) NOT NULL AUTO_INCREMENT,
  first_name VARCHAR(50) NOT NULL,
  last_name VARCHAR(50) NOT NULL,
  email VARCHAR(50) NOT NULL,
  PRIMARY KEY (id)
);
```

Create a Hibernate configuration file

1. Create a new file called **`hibernate.cfg.xml`** in your project's root directory.
2. Add the following code to the file:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!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.MySQL8Dialect</property>
    <property name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
    <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/hibernate_example?useSSL=false&am
p;serverTimezone=UTC</property>
    <property name="hibernate.connection.username">root</property>
    <property name="hibernate.connection.password">password</property>
    <property name="hibernate.hbm2ddl.auto">update</property>
    <mapping class="com.example.Employee"/>
  </session-factory>
</hibernate-configuration>
```

Create an Entity class

1. Create a new class called `**Employee**` in your project's `**src/main/java**` directory.
2. Add the following code to the class:

```
package com.example;

import javax.persistence.*;

@Entity
@Table(name="employee")
public class Employee {
  @Id
  @GeneratedValue(strategy=GenerationType.IDENTITY)
  @Column(name="id")
  private int id;

  @Column(name="first_name")
  private String firstName;

  @Column(name="last_name")
  private String lastName;

  @Column(name="email")
  private String email;

  public Employee() {}

  public Employee(String firstName, String lastName, String email) {
    this.firstName = firstName;
    this.lastName = lastName;
    this.email = email;
  }
```

```
  // getters and setters
  ...
}
```

Create a Hibernate Util class

1. Create a new class called `**HibernateUtil**` in your project's `**src/main/java**` directory.
2. Add the following code to the class:

```
package com.example;

import org.hibernate.SessionFactory;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;

public class HibernateUtil {

  private static final SessionFactory sessionFactory = buildSessionFactory();

  private static SessionFactory buildSessionFactory() {
    try {
      Configuration configuration = new Configuration().configure();
      return configuration.buildSessionFactory(
          new StandardServiceRegistryBuilder()
              .applySettings(configuration.getProperties())
              .build());
    } catch (Throwable ex) {
      System.err.println("Initial SessionFactory creation failed." + ex);
      throw new ExceptionInInitializerError(ex);
    }
  }

  public static SessionFactory getSessionFactory() {
    return sessionFactory;
  }

  public static void shutdown() {
    getSessionFactory().close();
  }

}
```

Create a main method

1. Create a new class called `**App**` in your project's `**src/main/java**` directory.
2. Add the following code to the class:

```
package com.example;

import org.hibernate.Session;
import org.hibernate.Transaction;
```

```
public class App {

  public static void main(String[] args) {
    Session session = HibernateUtil.getSessionFactory().openSession();

    Transaction transaction = null;
    try {
      transaction = session.beginTransaction();

      // create a new employee object
      Employee employee = new Employee("John", "Doe", "jdoe@example.com");

      // save the employee object
      session.save(employee);

      // commit the transaction
      transaction.commit();
    } catch (Exception e) {
      if (transaction != null) {
        transaction.rollback();
      }
      e.printStackTrace();
    } finally {
      session.close();
      HibernateUtil.shutdown();
    }
  }
}
```

Run the application

1. Right-click on the `**App**` class and select `**Run As > Java Application**`.
2. Check the console output for any errors.
3. Open MySQL Workbench and execute the following query:

```
SELECT * FROM employee;
```

You should see one record with the values you entered in the `**App**` class.

Annotation Configuration

Create a database

1. Open MySQL Workbench and create a new database called `**hibernate_annotation**`.
2. Create a new table called `**customer**` with the following fields:

```
CREATE TABLE customer (
  id INT(11) NOT NULL AUTO_INCREMENT,
```

```
  first_name VARCHAR(50) NOT NULL,
  last_name VARCHAR(50) NOT NULL,
  email VARCHAR(50) NOT NULL,
  PRIMARY KEY (id)
);
```

Update the Hibernate configuration file

1. Open the `**hibernate.cfg.xml**` file and remove the `**mapping**` element.
2. Add the following line to the `**session-factory**` element:

```
<property name="hibernate.hbm2ddl.auto">create</property>
```

Update the Entity class

1. Open the `**Employee**` class and change the class name to `**Customer**`.
2. Replace all occurrences of `**Employee**` with `**Customer**`.
3. Replace all occurrences of `**employee**` with `**customer**`.
4. Add the following code to the `**Customer**` class:

```
@Id
@GeneratedValue(strategy=GenerationType.IDENTITY)
@Column(name="id")
private int id;

@Column(name="first_name")
private String firstName;

@Column(name="last_name")
private String lastName;

@Column(name="email")
private String email;

public Customer() {}

public Customer(String firstName, String lastName, String email) {
  this.firstName = firstName;
  this.lastName = lastName;
  this.email = email;
}

// getters and setters
...
```

Update the Hibernate Util class

1. Open the `**HibernateUtil**` class and replace all occurrences of `**Employee**` with `**Customer**`.

2. Change the `**buildSessionFactory**` method to use annotation-based configuration:

```
private static SessionFactory buildSessionFactory() {
  try {
    Configuration configuration = new Configuration().configure();
    ServiceRegistry serviceRegistry = new StandardServiceRegistryBuilder()
        .applySettings(configuration.getProperties()).build();
    Metadata metadata = new MetadataSources(serviceRegistry)
        .addAnnotatedClass(Customer.class)
        .getMetadataBuilder()
        .build();
    return metadata.getSessionFactoryBuilder().build();
  } catch (Throwable ex) {
    System.err.println("Initial SessionFactory creation failed." + ex);
    throw new ExceptionInInitializerError(ex);
  }
}
```

Create a main method

1. Create a new class called `**App**` in your project's `**src/main/java**` directory.
2. Add the following code to the class:

```
package com.example;

import org.hibernate.Session;
import org.hibernate.Transaction;

public class App {

  public static void main(String[] args) {
    Session session = HibernateUtil.getSessionFactory().openSession();

    Transaction transaction = null;
    try {
      transaction = session.beginTransaction();

      // create a new customer object
      Customer customer = new Customer("John", "Doe", "jdoe@example.com");

      // save the customer object
      session.save(customer);

      // commit the transaction
      transaction.commit();
    } catch (Exception e) {
      if (transaction != null) {
        transaction.rollback();
      }
      e.printStackTrace();
    } finally {
      session.close();
```

```
        HibernateUtil.shutdown();
    }
  }
}
```

Run the application

1. Right-click on the `**App**` class and select `**Run As > Java Application**`.
2. Check the console output for any errors.
3. Open MySQL Workbench and execute the following query:

```
SELECT * FROM customer;
```

You should see one record with the values you entered in the `**App**` class.