Hands-On: Creating a Simple Employee Management Application with Hibernate and JPA

Let's assume that you have a requirement to create a simple application that manages the employees' information. For this exercise, you need to create a database table called "Employee" with the following fields:

- Id (int) - primary key
- Name (varchar)
- Age (int)
- Salary (double)

You will use Hibernate and JPA to create and manage this database table. Follow the steps below to complete the exercise:

Step 1: Create a Java project in your IDE (Eclipse or IntelliJ IDEA) and add the following dependencies to your pom.xml file:

```
<dependencies>
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-core</artifactId>
        <version>5.4.32.Final</version>
    </dependency>
    <dependency>
        <groupId>javax.persistence</groupId>
        <artifactId>persistence-api</artifactId>
        <version>2.2</version>
    </dependency>
    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <version>1.4.200</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Step 2: Create a new Java class called "Employee" with the following fields:

```
@Entity
@Table(name = "Employee")
public class Employee {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @Column(name = "Name")
    private String name;
```

```
    @Column(name = "Age")
    private int age;

    @Column(name = "Salary")
    private double salary;

    // Constructors, getters, and setters
}
```

Step 3: Create a new Java class called "App" and add the following code to create a new employee:

```
public class App {

    public static void main(String[] args) {
        EntityManagerFactory entityManagerFactory = Persistence.createEntityManagerFactory("employee-
unit");
        EntityManager entityManager = entityManagerFactory.createEntityManager();

        Employee employee = new Employee();
        employee.setName("John");
        employee.setAge(35);
        employee.setSalary(50000.0);

        entityManager.getTransaction().begin();
        entityManager.persist(employee);
        entityManager.getTransaction().commit();

        entityManager.close();
        entityManagerFactory.close();
    }
}
```

Step 4: Run the "App" class to create a new employee in the database.

Step 5: Add the following code to the "App" class to retrieve the employee from the database:

```
Employee employee = entityManager.find(Employee.class, 1);
System.out.println("Employee ID: " + employee.getId());
System.out.println("Employee Name: " + employee.getName());
System.out.println("Employee Age: " + employee.getAge());
System.out.println("Employee Salary: " + employee.getSalary());
```

Step 6: Run the "App" class to retrieve the employee from the database.

Step 7: Add the following code to the "App" class to update the employee's salary:

```
Employee employee = entityManager.find(Employee.class, 1);
employee.setSalary(60000.0);

entityManager.getTransaction().begin();
```

```
entityManager.merge(employee);
entityManager.getTransaction().commit();
```

Step 8: Run the "App" class to update the employee's salary.

Step 9: Add the following code to the "App" class to delete the employee:

```
Employee employee = entityManager.find(Employee.class, 1);

entityManager.getTransaction().begin();
entityManager.remove(employee);
entityManager.getTransaction().commit();
```

Step 10: Run the "App" class to delete the employee.

Congratulations! You have successfully completed the hands-on exercise based on Hibernate and JPA. In this exercise, you have created a Java project, added Hibernate and JPA dependencies, created a database table, created a Java class to represent the table, and used JPA to create, retrieve, update, and delete an employee from the database. This exercise provides a basic understanding of how Hibernate and JPA work together to manage database operations.