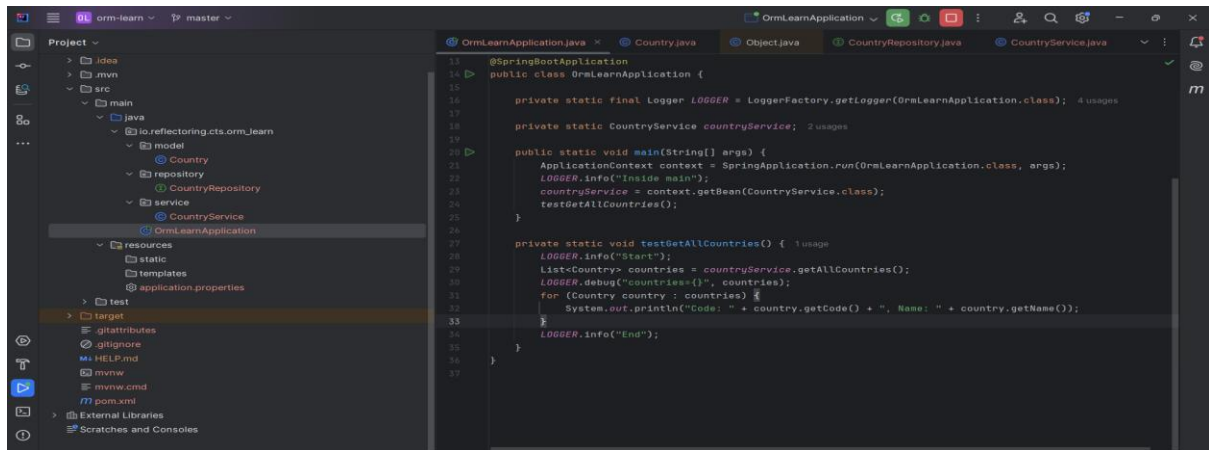


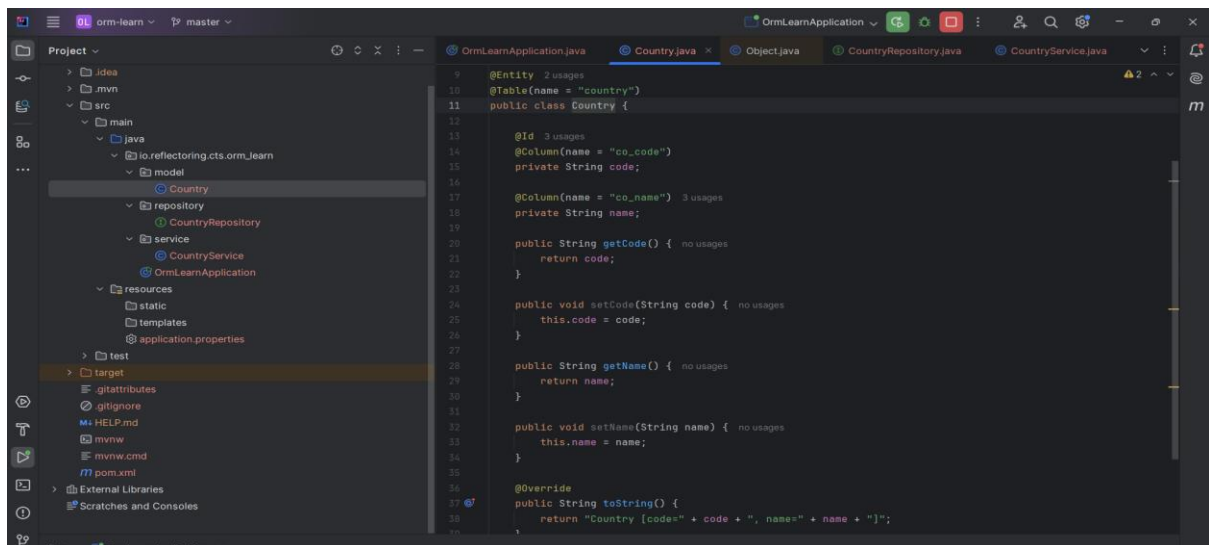
Week 3

Spring Data JPA - Quick Example

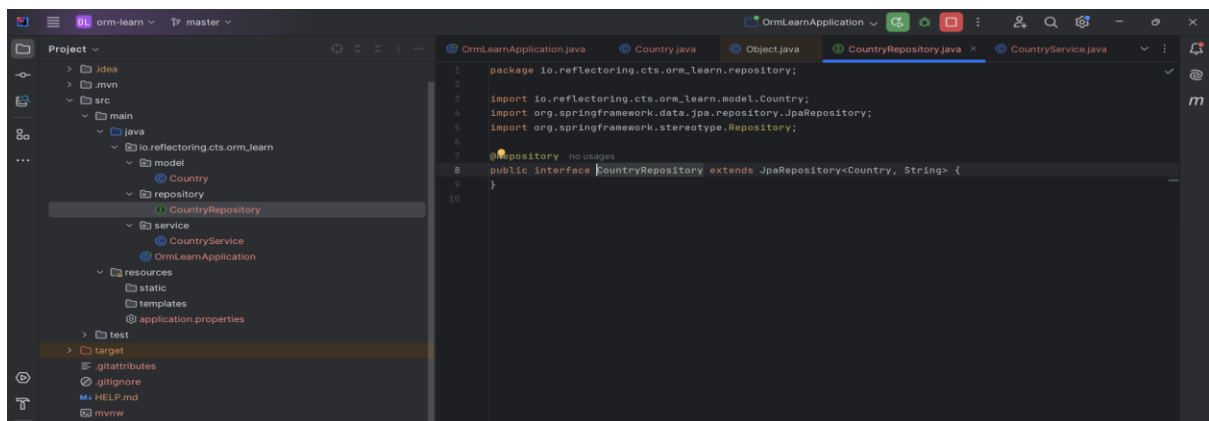
Main.java



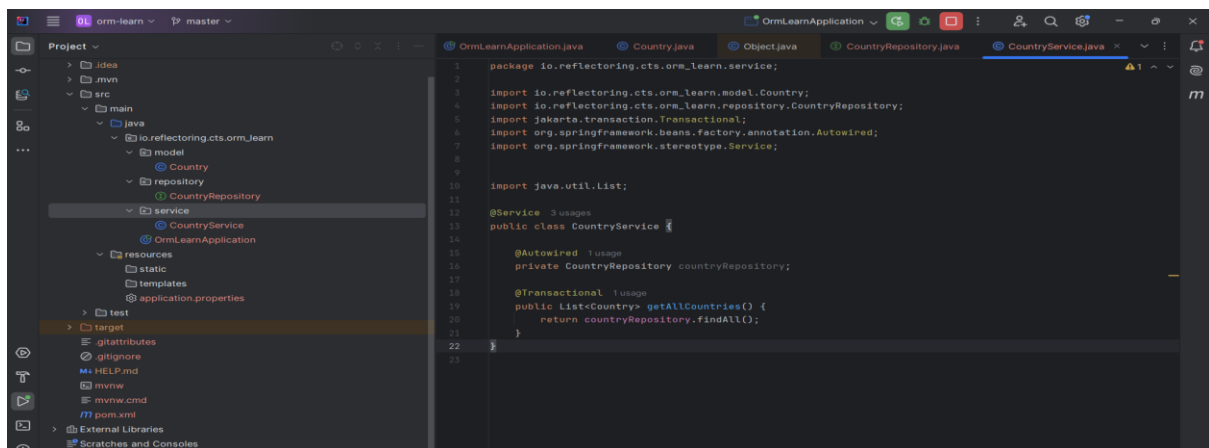
Model : Country.java



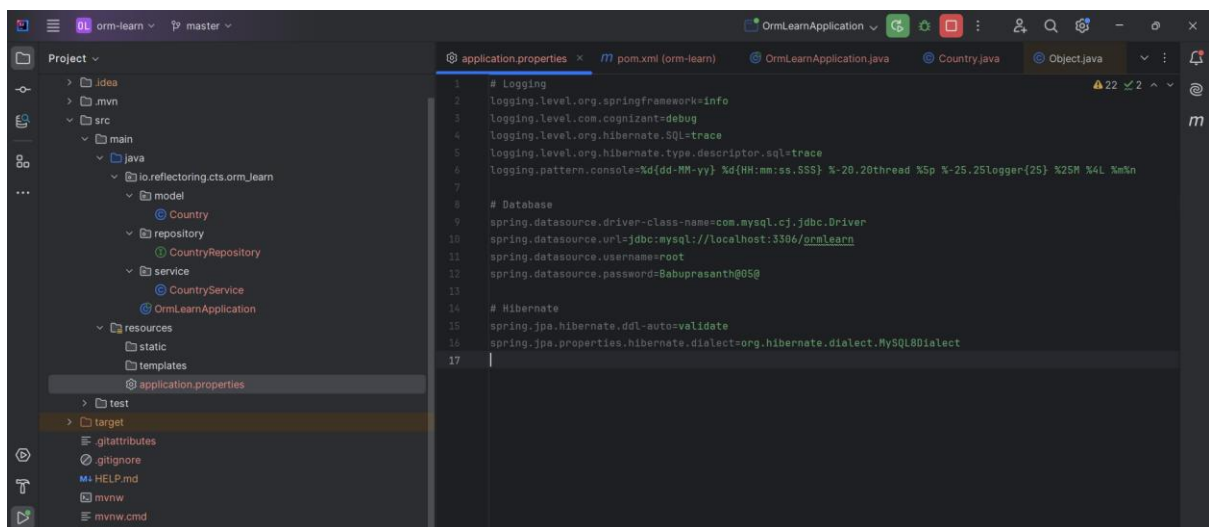
Repository : CountryRepository.java



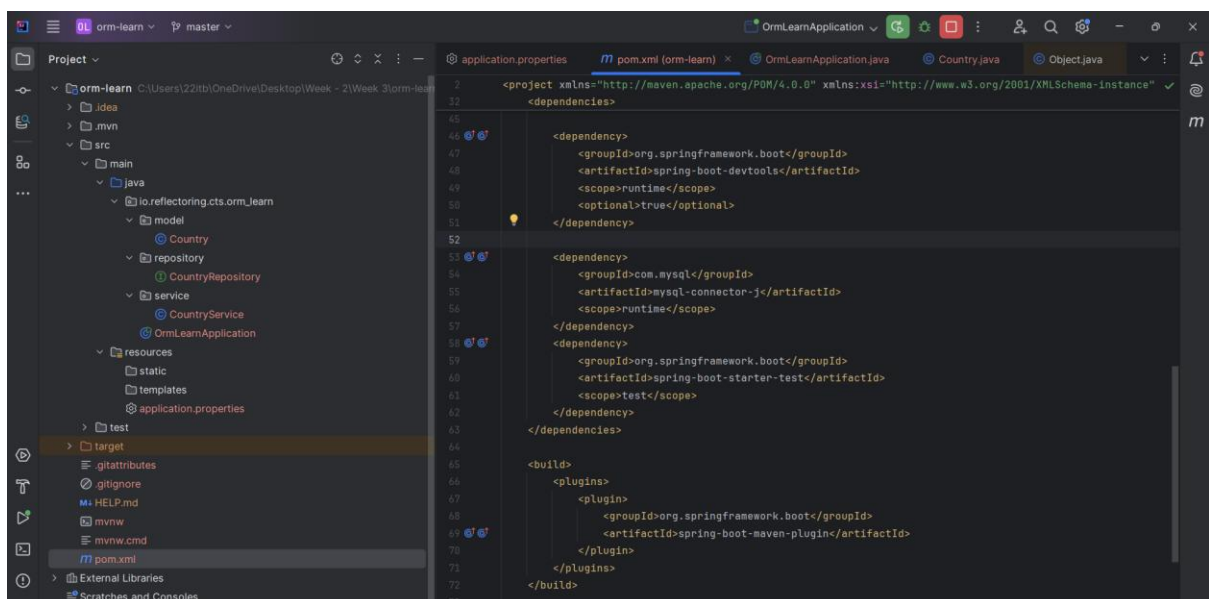
Service : CountryService.java



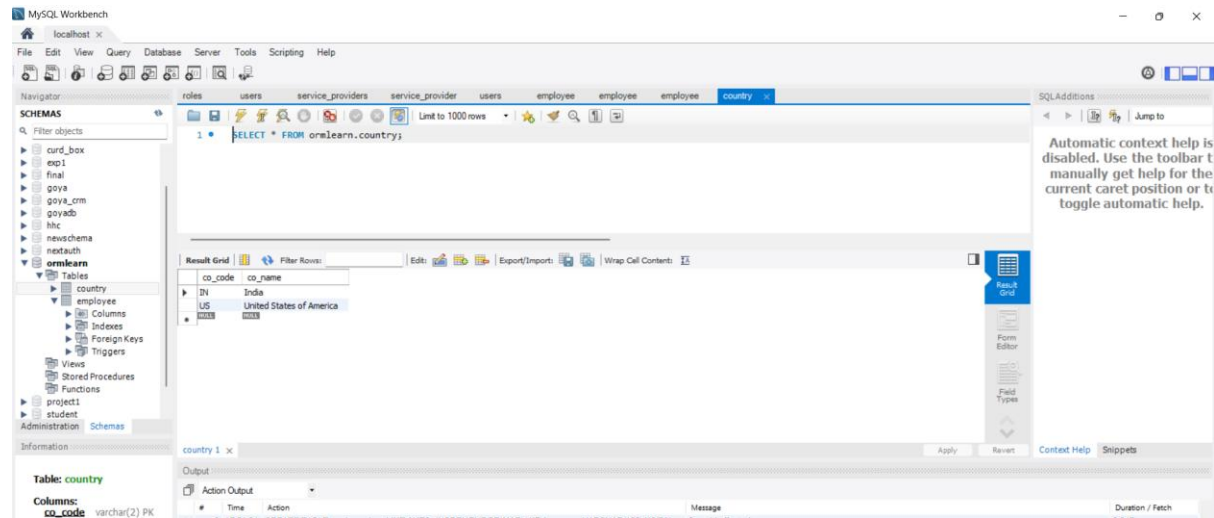
Application.property



Pom.xml



Mysql



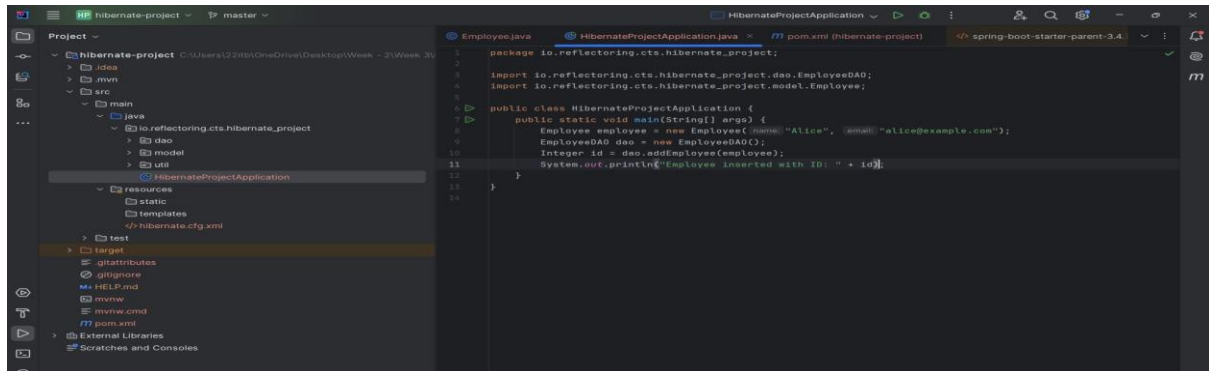
Output



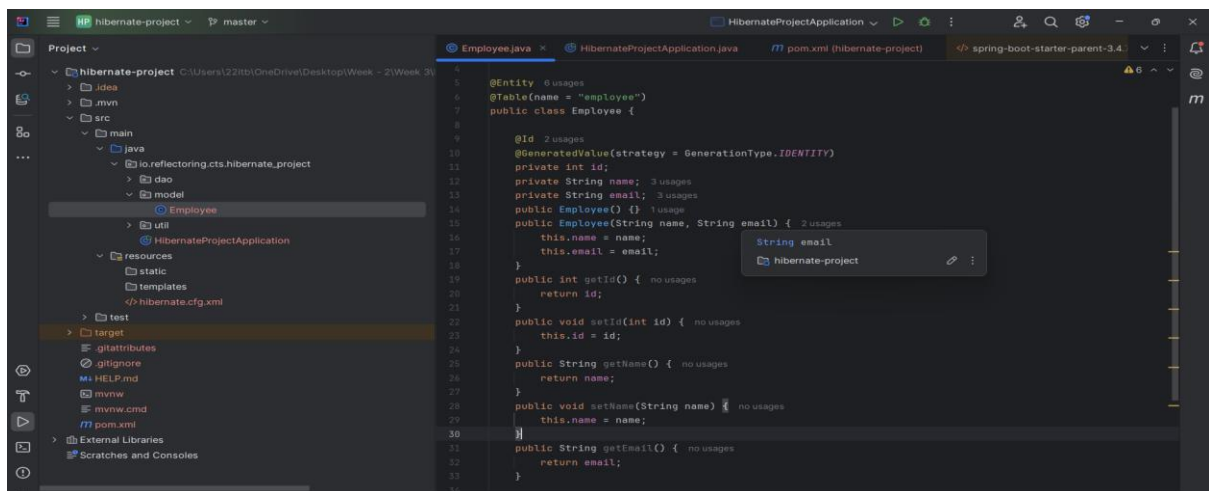
Difference between JPA, Hibernate and Spring Data JPA

Hibernate Project

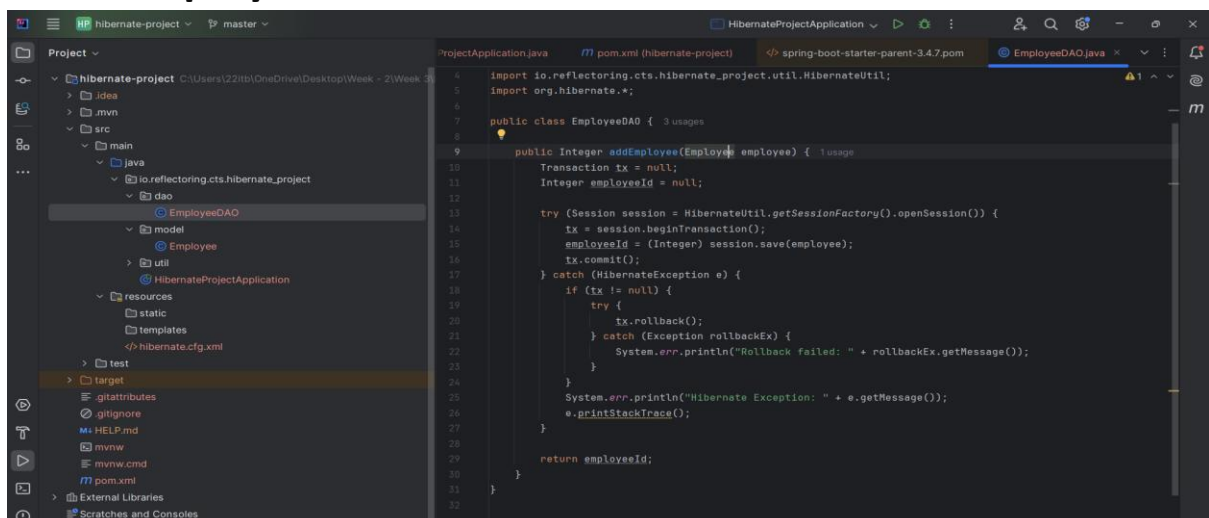
Main.java



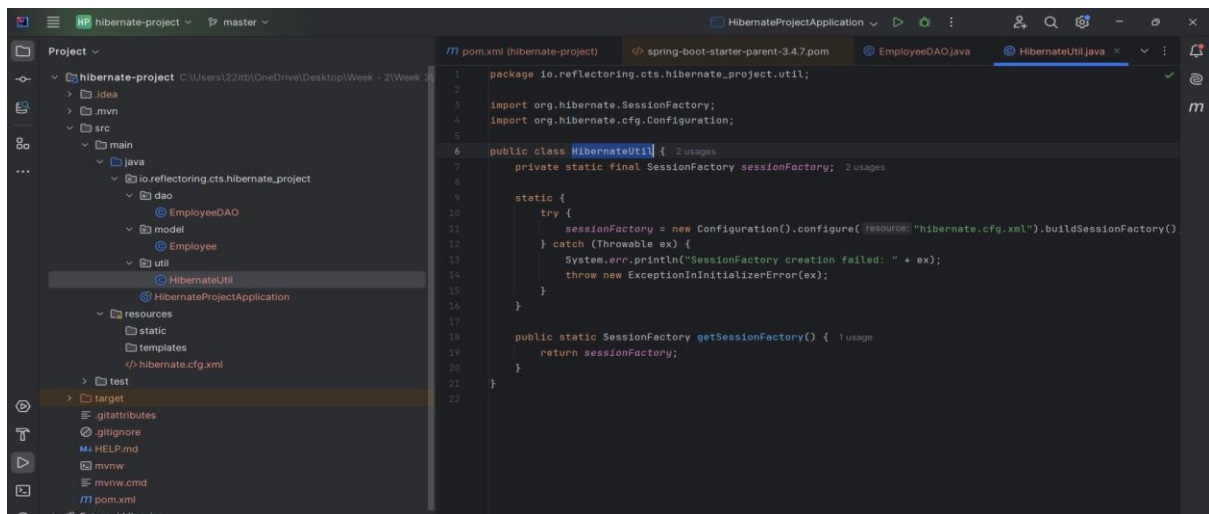
Model : Employee



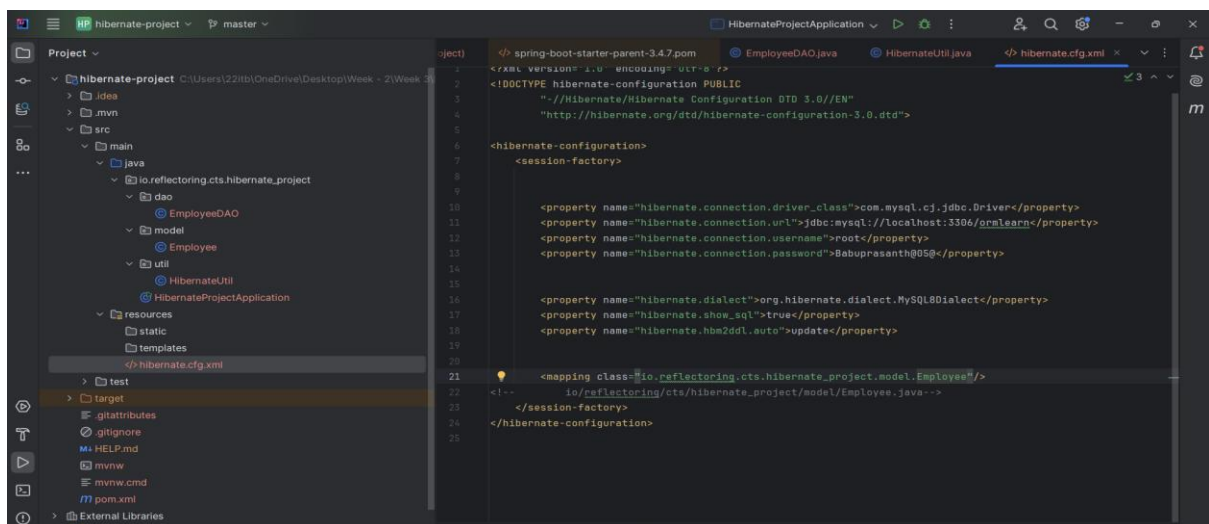
DAO :EmployeeDAO



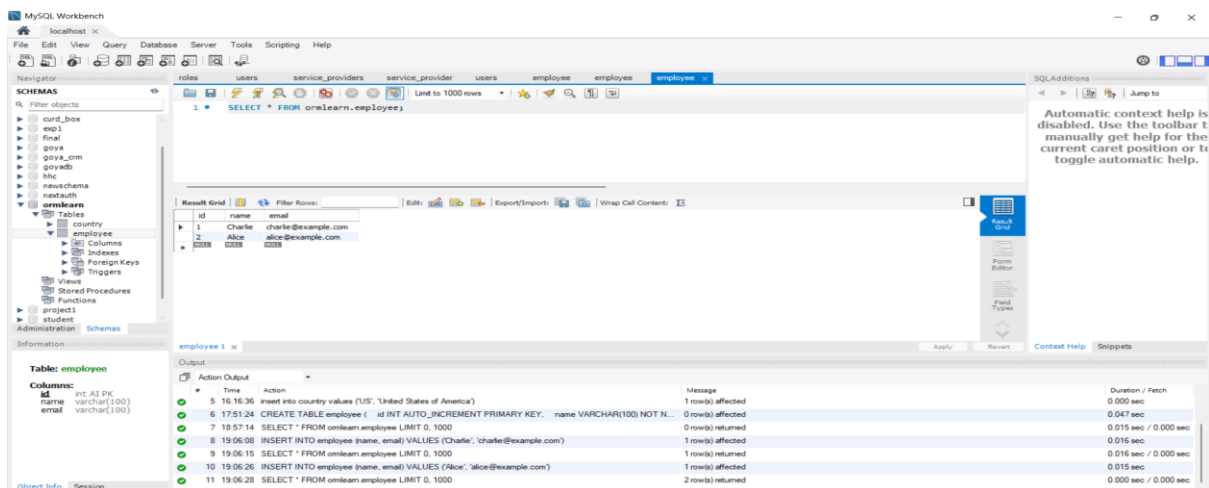
util : HibernateUtil



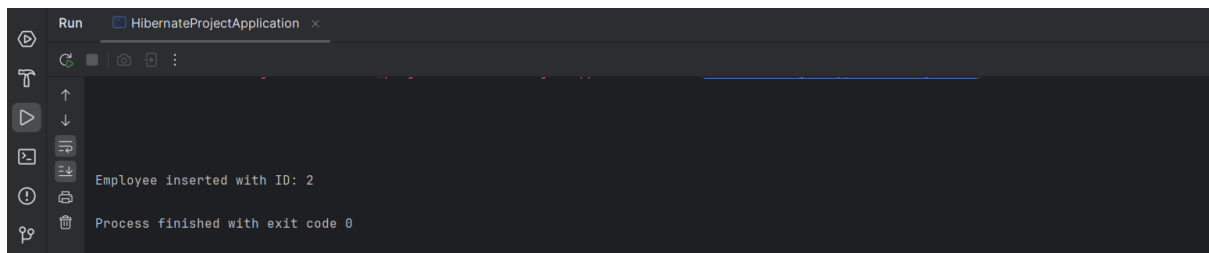
Resources : hibernate.cfg.xml



Mysql



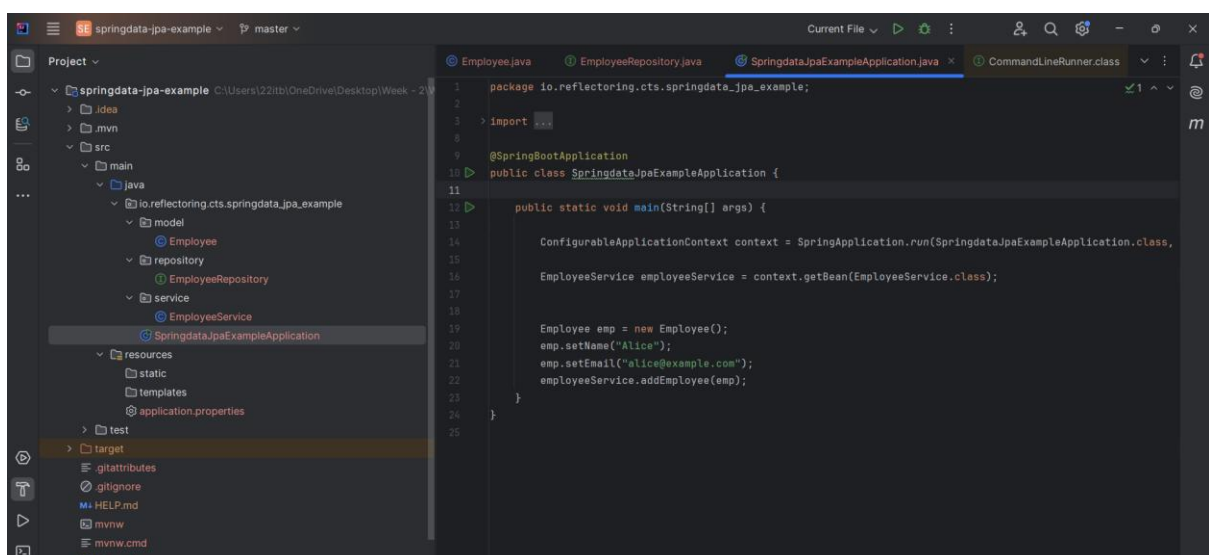
Output



```
Run HibernateProjectApplication x
Employee inserted with ID: 2
Process finished with exit code 0
```

Spring Data JPA

Main.java



```
package io.reflectoring.cts.springdata_jpa_example;

import ...

@SpringBootApplication
public class SpringdataJpaExampleApplication {

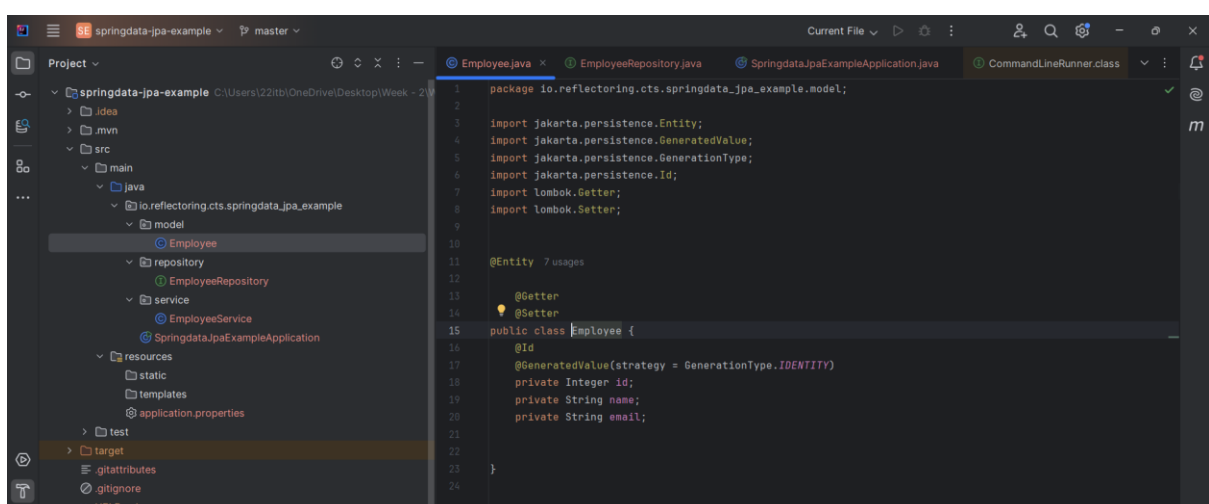
    public static void main(String[] args) {

        ConfigurableApplicationContext context = SpringApplication.run(SpringdataJpaExampleApplication.class,
            args);

        EmployeeService employeeService = context.getBean(EmployeeService.class);

        Employee emp = new Employee();
        emp.setName("Alice");
        emp.setEmail("alice@example.com");
        employeeService.addEmployee(emp);
    }
}
```

Model : Employee



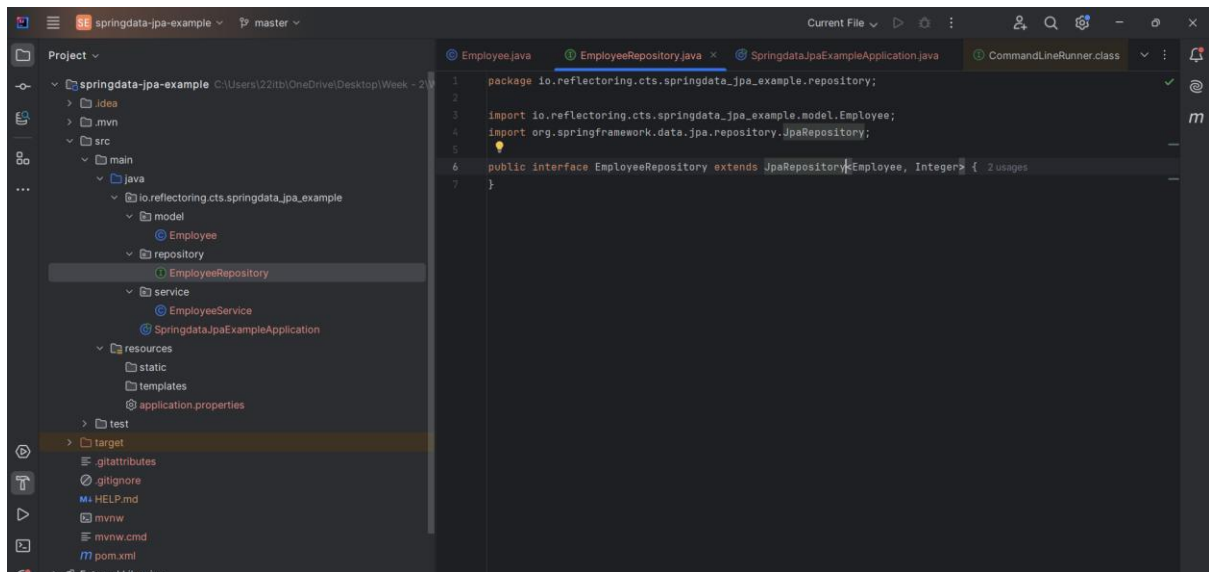
```
package io.reflectoring.cts.springdata_jpa_example.model;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import lombok.Getter;
import lombok.Setter;

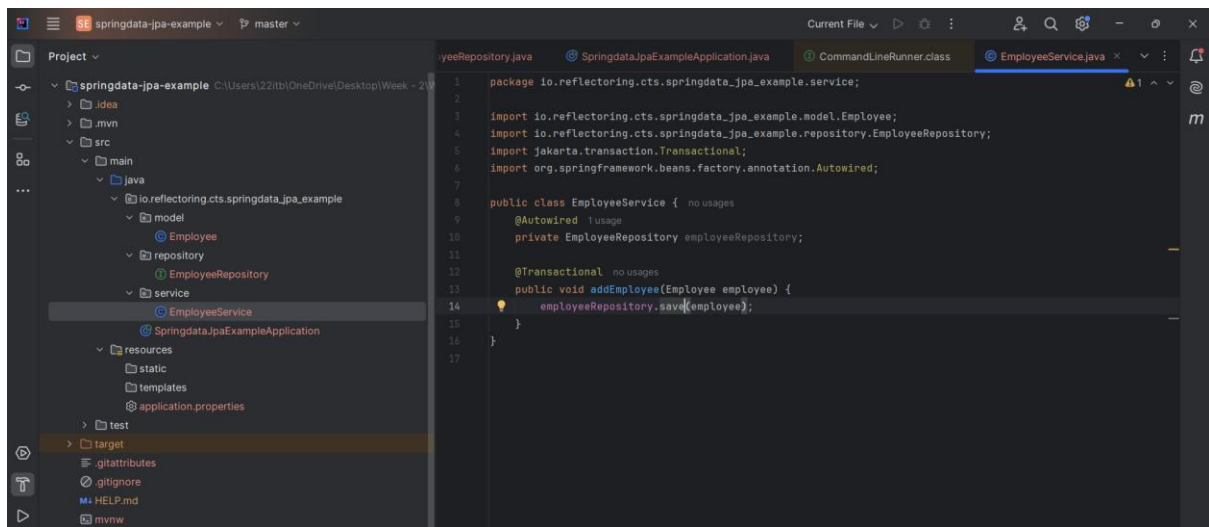
@Entity
public class Employee {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String name;
    private String email;
}
```

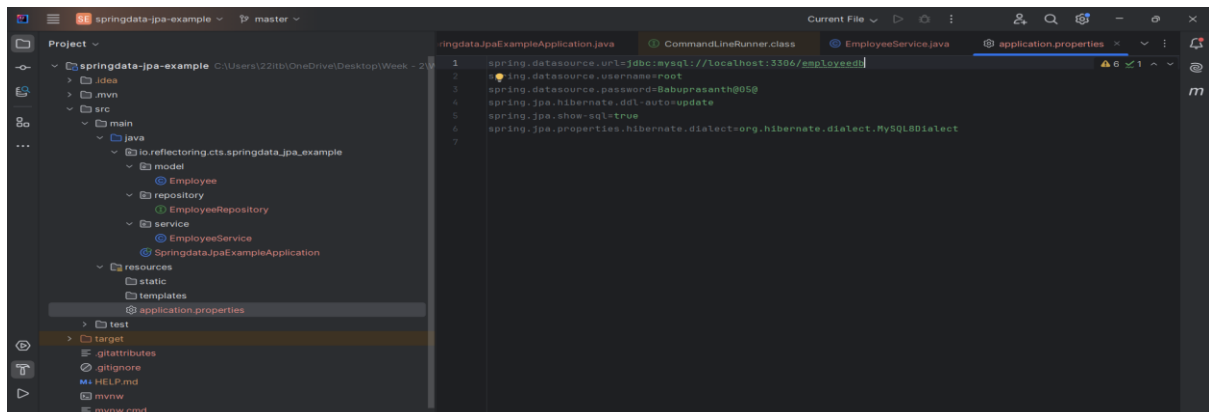

Repository : EmployeeRepository



Service : EmployeeService



application.property



Mysql DB

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar shows the 'SCHEMAS' tree with a search filter. The 'employee' schema is selected, showing tables like 'employee' and 'employeee'. The main query editor contains the SQL statement: `SELECT * FROM employeee.employee;`. The 'Result Grid' shows one row of data: `1 Alice alice@example.com`. The bottom panel shows the 'Output' tab with a table of execution actions and messages.

#	Time	Action	Message	Duration / Fetch
12	19:16:04	SELECT * FROM omlearn.country LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
13	19:27:06	CREATE DATABASE IF NOT EXISTS employeee	1 row(s) affected	0.016 sec
14	19:27:06	USE employeee	0 row(s) affected	0.000 sec
15	19:27:06	CREATE TABLE employee (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(255), email...	0 row(s) affected	0.031 sec
16	19:27:24	SELECT * FROM employeee.employee LIMIT 0, 1000	0 row(s) returned	0.016 sec / 0.000 sec
17	19:44:07	INSERT INTO employee (name, email) VALUES ('Alice', 'alice@example.com')	1 row(s) affected	0.000 sec
18	19:44:09	SELECT * FROM employeee.employee LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

Output

The screenshot shows a terminal window titled 'Run' with the application 'OrmLearnApplication'. The output text is as follows:

```
"C:\Program Files\Java\jdk-22\bin\java.exe" ...  
Employee value inserted Successfully !  
Process finished with exit code 0
```