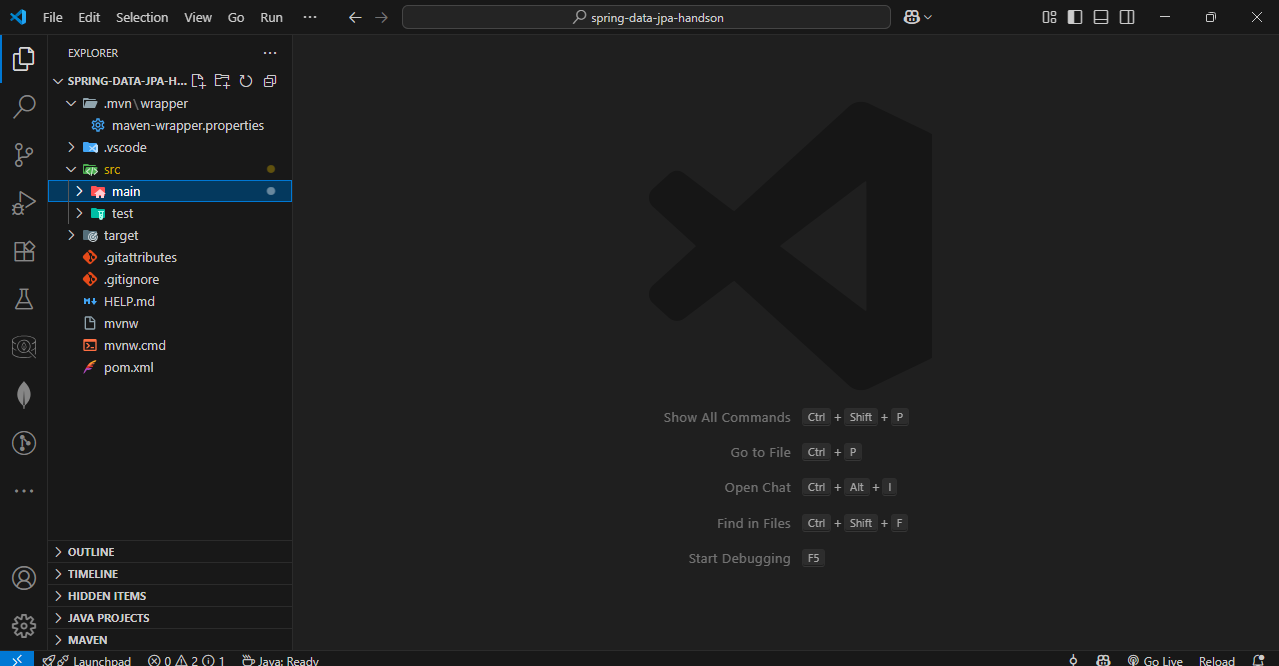
Spring data JPA OUTPUTS

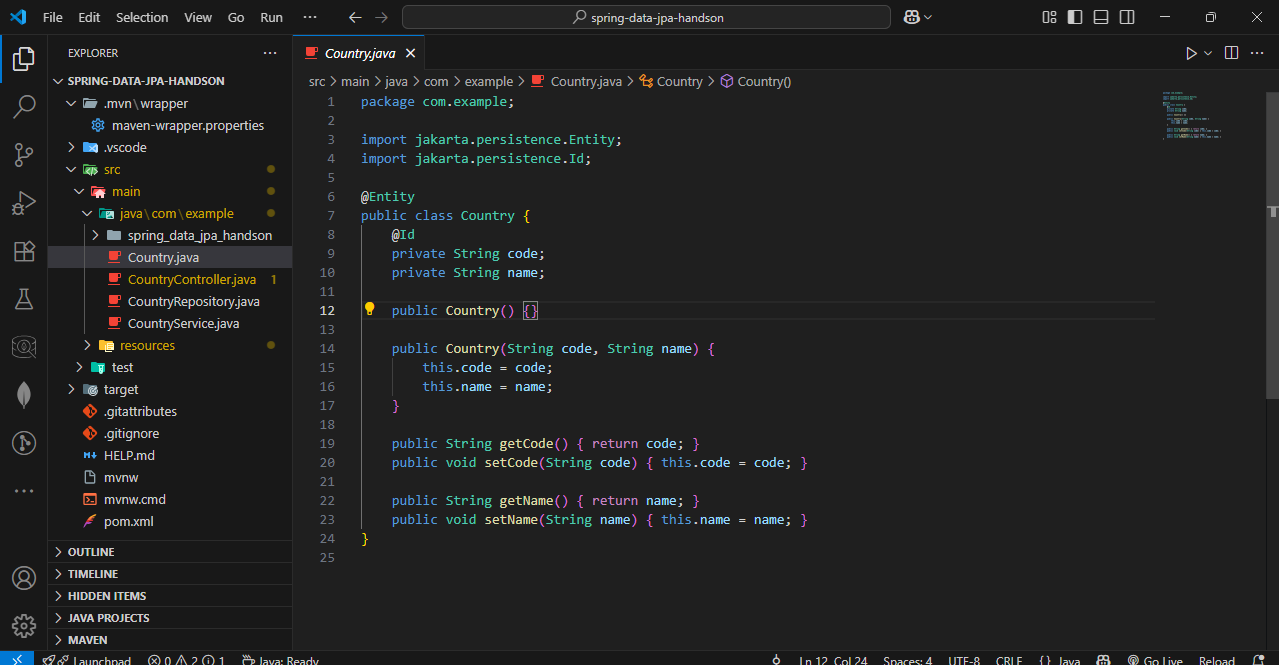
Name : N. Khanishk

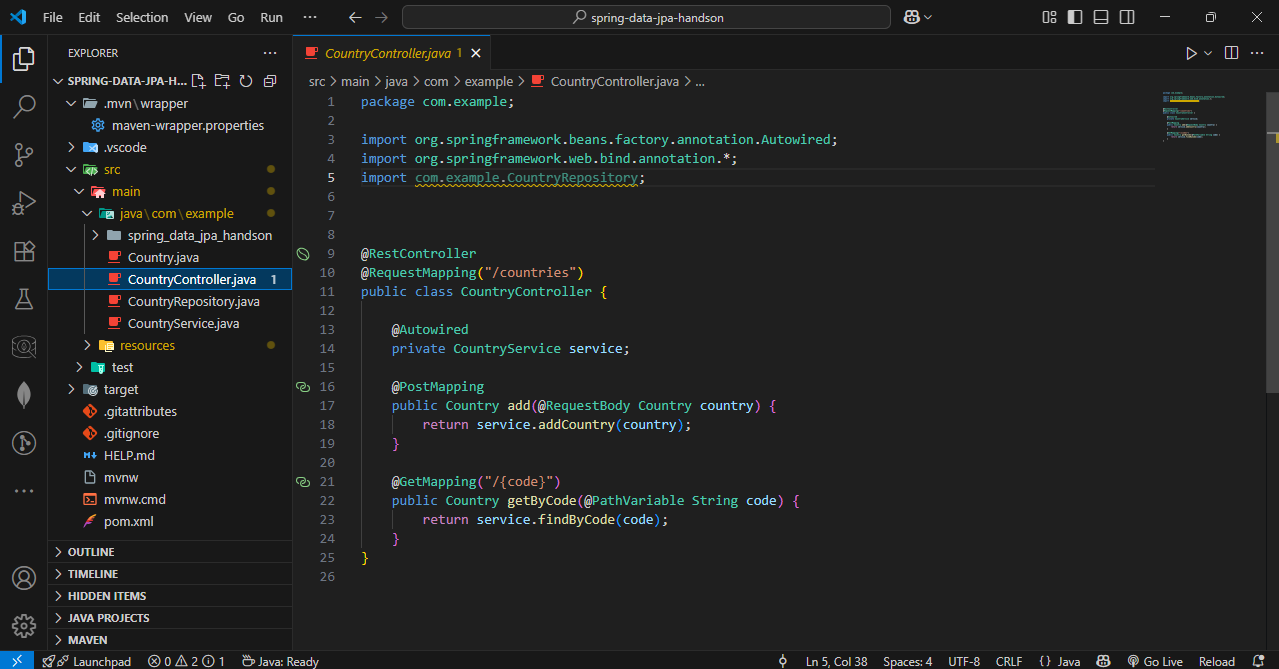
Superset Id : 6124965

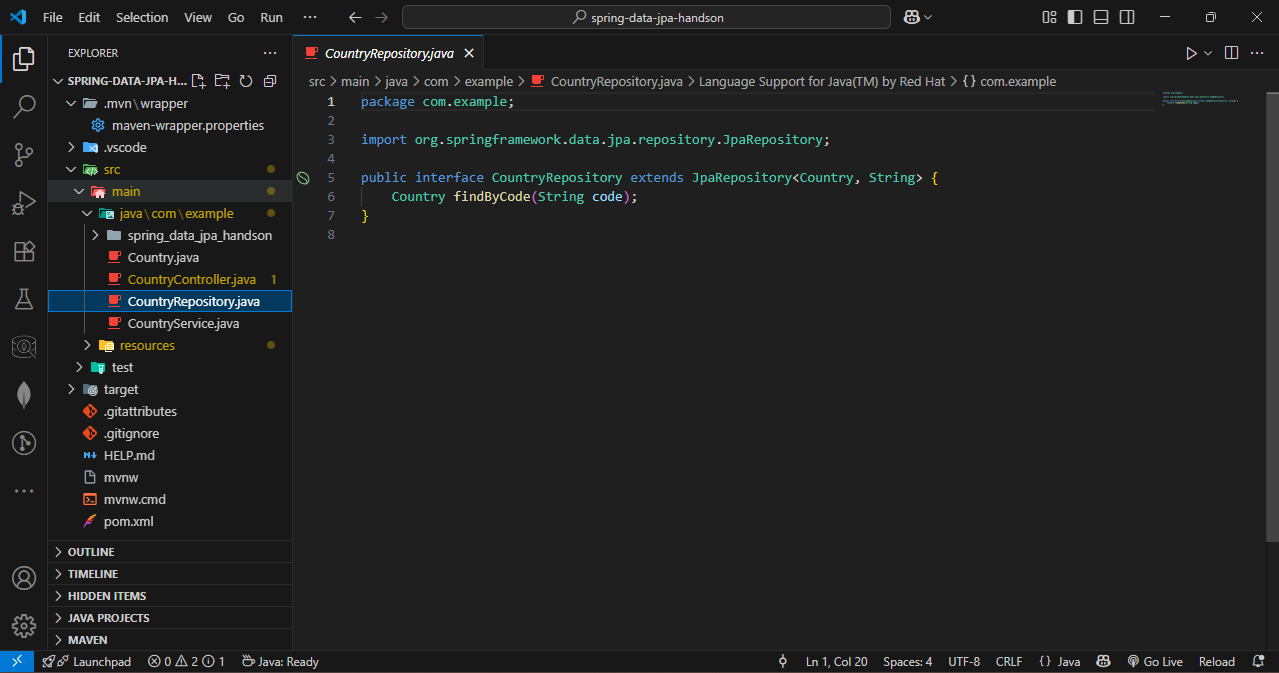
1. Spring Data JPA - Quick Example

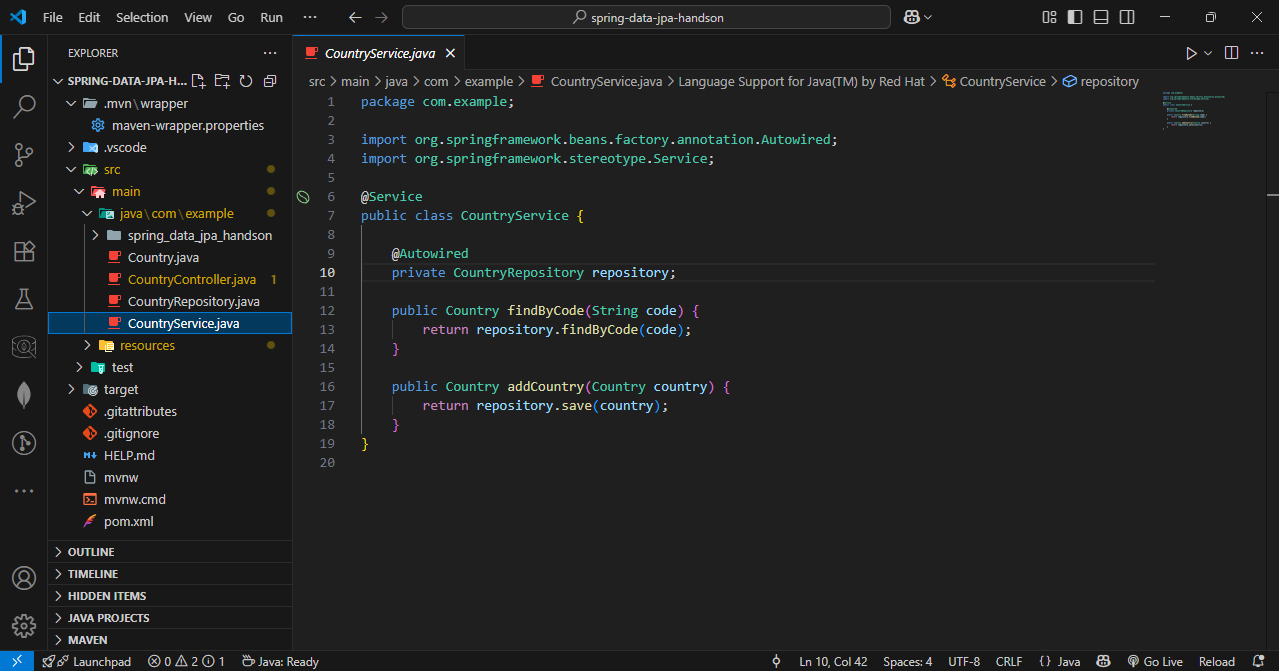
**Step – 1 :** Create a Spring initializr : create maven project in vs code, Enter the group id and artifact id.  
Select the dependencies and Spring Web, Spring Data JPA, H2 Database.



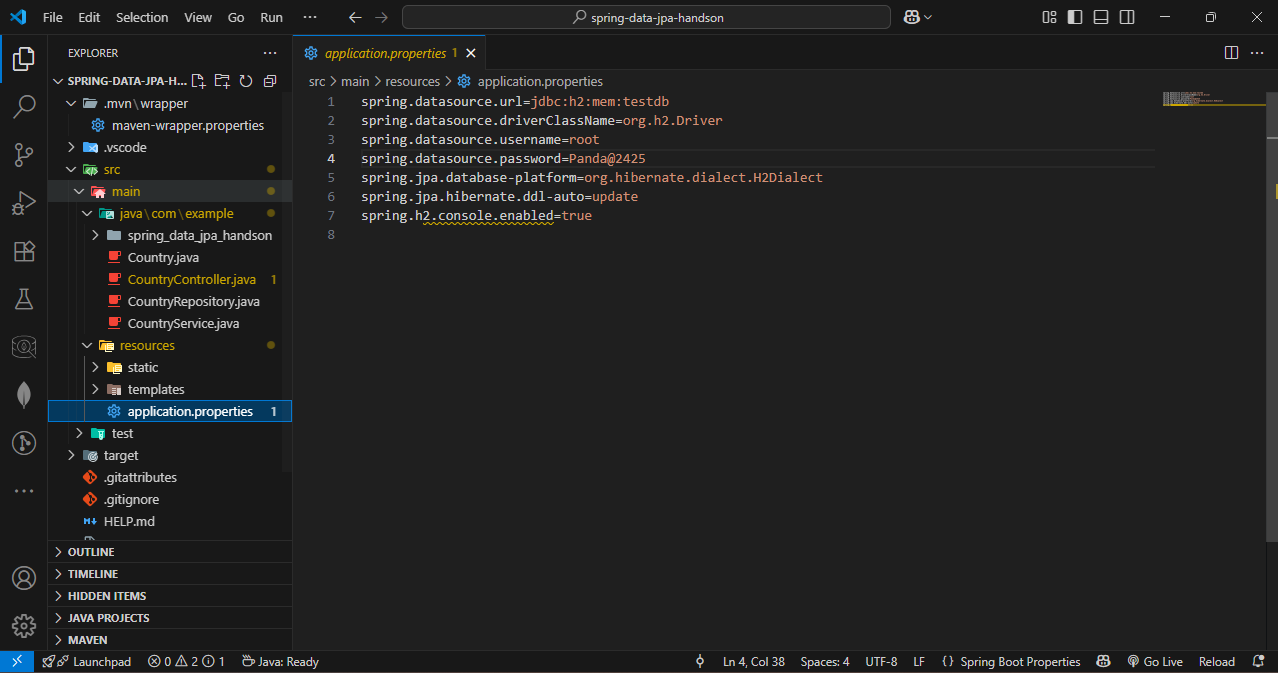
**Step – 2 :** Create classed Country.java, CountryService.java, CountryController.java and Country Repository.java and write codes.  




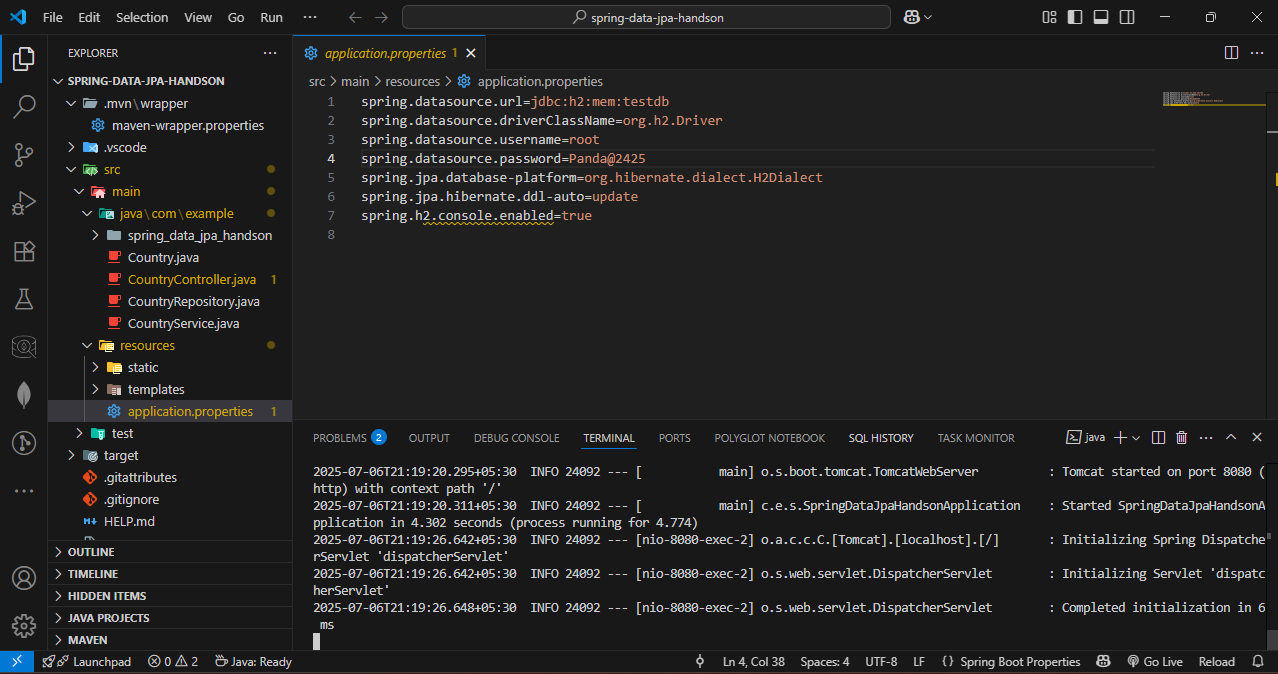




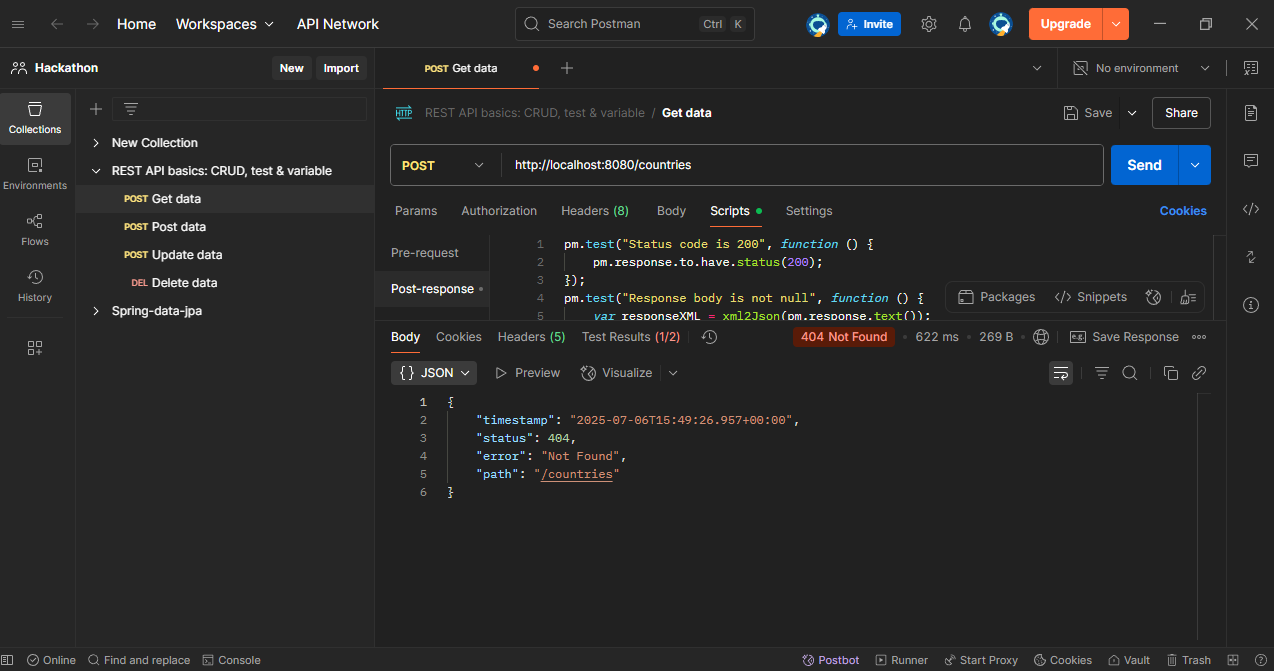
Step – 3 : Add database information in src/main/java/resourcesapplications.properties.



Step – 4 : Run the project.

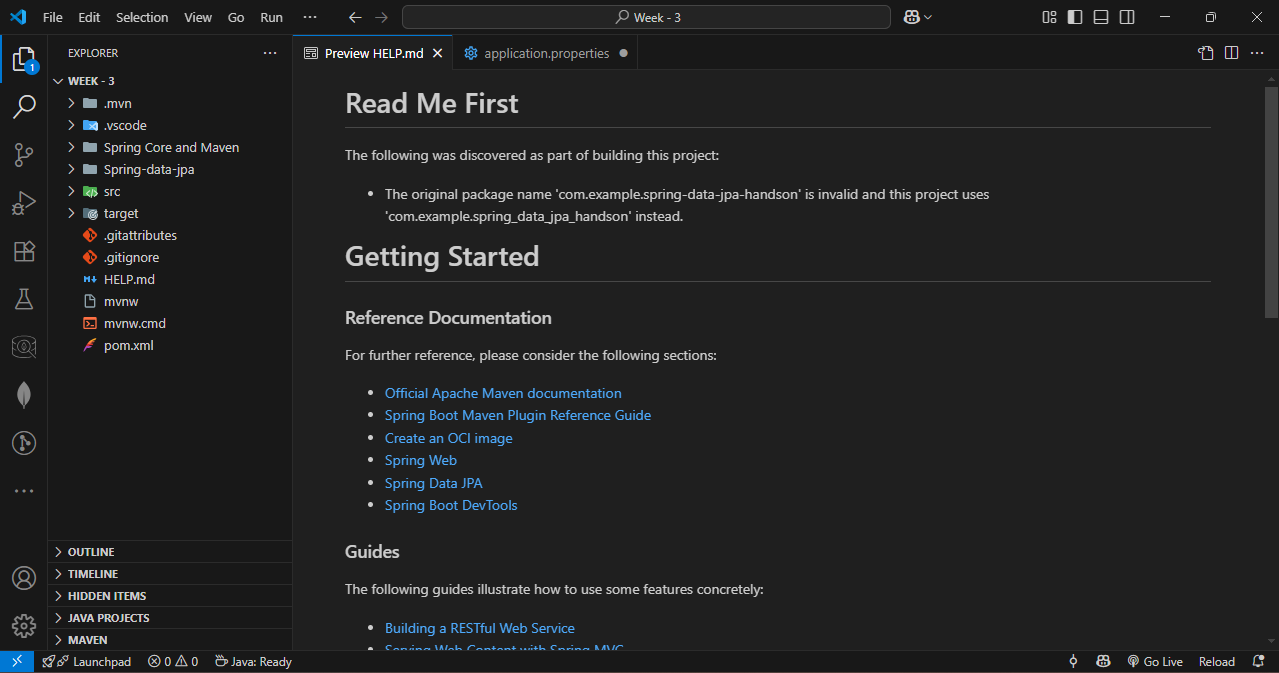


Step – 5 : Test API in postman

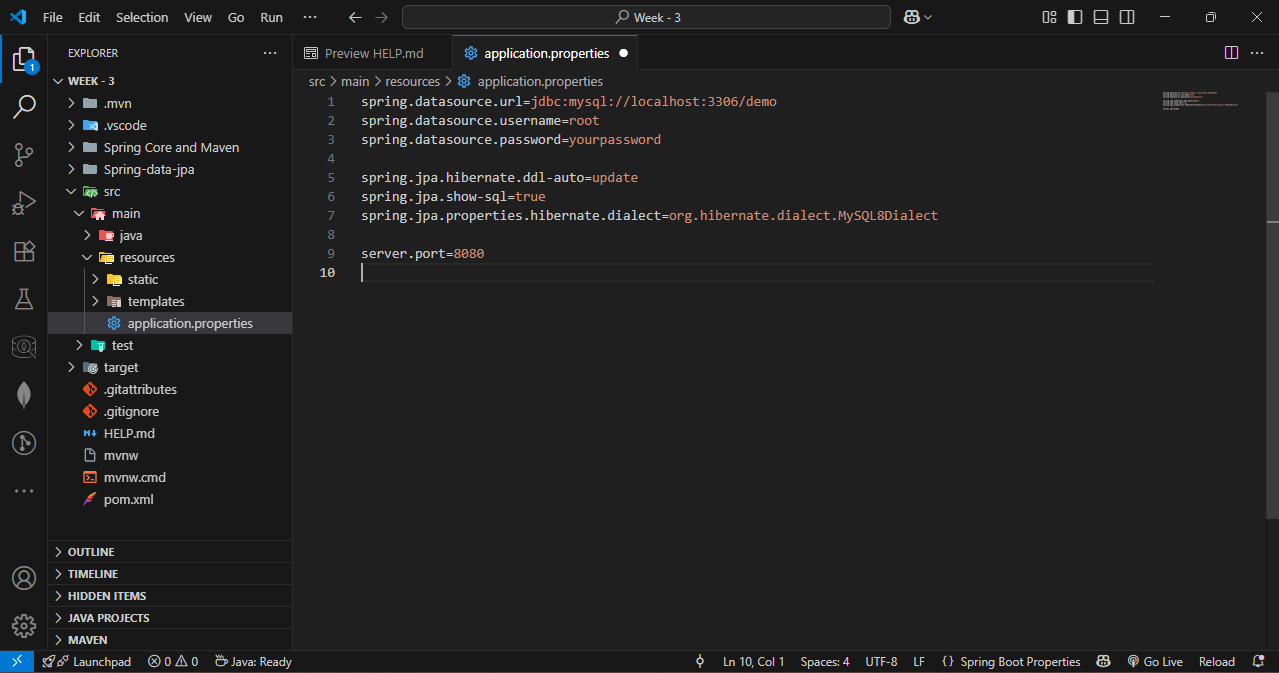


1. Difference between JPA, Hibernate and Spring Data JPA

**Step – 1 :** Create a Spring initializr : create maven project in vs code, Enter the group id and artifact id.  
Select the dependencies and Spring Web, Spring Data JPA, MySQL Driver, Spring Boot DevTools.



Step – 2 : Write code in src/java/resources/application.properties



Step – 3 : Create 2 packages models and repository

In repository add Employee .java and Department.java

Code :

package com.example.spring\_data\_jpa\_handson.model;

import jakarta.persistence.\*;

@Entity

public class Employee {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    private String name;

    private double salary;

    @ManyToOne

    @JoinColumn(name = "department\_id") // FK column in employee table

    private Department department;

    public Employee() {

    }

    public Employee(String name, double salary) {

        this.name = name;

        this.salary = salary;

    }

    // Getters and Setters

    public Long getId() {

        return id;

    }

    public void setId(Long 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;

    }

    public Department getDepartment() {

        return department;

    }

    public void setDepartment(Department department) {

        this.department = department;

    }

    @Override

    public String toString() {

        return "Employee{" +

               "id=" + id +

               ", name='" + name + '\'' +

               ", salary=" + salary +

               ", department=" + (department != null ? department.getName() : null) +

               '}';

    }

}

Department.java

Code :

package com.example.spring\_data\_jpa\_handson.model;

import jakarta.persistence.\*;

import java.util.ArrayList;

import java.util.List;

@Entity

public class Department {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    private String name;

    @OneToMany(mappedBy = "department", cascade = CascadeType.ALL)

    private List<Employee> employees = new ArrayList<>();

    public Department() {}

    public Department(String name) {

        this.name = name;

    }

    // Getters and Setters

    public Long getId() {

        return id;

    }

    public void setId(Long id) {

        this.id = id;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public List<Employee> getEmployees() {

        return employees;

    }

    public void setEmployees(List<Employee> employees) {

        this.employees = employees;

    }

    @Override

    public String toString() {

        return "Department{" +

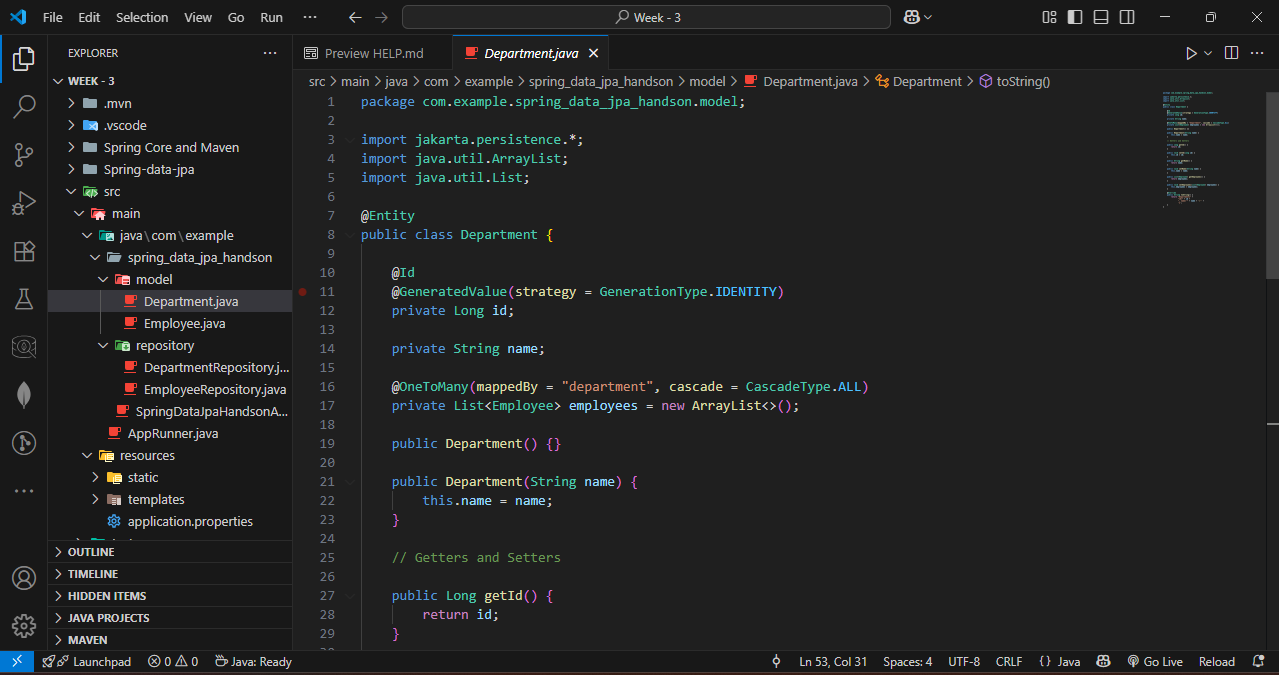
               "id=" + id +

               ", name='" + name + '\'' +

               '}';

    }

}



In repository add classes EmployeeRepository.java and DepartmentRepository.java

EmployeeRepository.java

Code :

package com.example.spring\_data\_jpa\_handson.repository;

import com.example.spring\_data\_jpa\_handson.model.Employee;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import org.springframework.data.repository.query.Param;

import org.springframework.stereotype.Repository;

import java.util.List;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

    @Query("SELECT e FROM Employee e WHERE e.department.name = :dept")

    List<Employee> findByDepartmentName(@Param("dept") String dept);

    @Query("SELECT e FROM Employee e WHERE e.department.name = :dept")

    List<Employee> fetchByHQL(@Param("dept") String dept);

    @Query(value = "SELECT \* FROM employee WHERE salary > ?1", nativeQuery = true)

    List<Employee> fetchByNative(double salary);

}

DepartmentRepository.java

Code :

package com.example.spring\_data\_jpa\_handson.repository;

import com.example.spring\_data\_jpa\_handson.model.Department;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

@Repository

public interface DepartmentRepository extends JpaRepository<Department, Long> {

}

**Step – 4 :** Create a class AppRunner.java as main

Code :

package com.example;

import com.example.spring\_data\_jpa\_handson.model.Department;

import com.example.spring\_data\_jpa\_handson.model.Employee;

import com.example.spring\_data\_jpa\_handson.repository.DepartmentRepository;

import com.example.spring\_data\_jpa\_handson.repository.EmployeeRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

import java.util.List;

@Component

public class AppRunner implements CommandLineRunner {

    @Autowired

    private EmployeeRepository employeeRepository;

    @Autowired

    private DepartmentRepository departmentRepository;

    @Override

    public void run(String... args) throws Exception {

        Department dept = new Department("Finance");

        Employee emp1 = new Employee("Alice", 50000);

        emp1.setDepartment(dept);

        Employee emp2 = new Employee("Bob", 60000);

        emp2.setDepartment(dept);

        dept.setEmployees(List.of(emp1, emp2));

        departmentRepository.save(dept);

        System.out.println("All Employees in Finance:");

        employeeRepository.findByDepartmentName("Finance").forEach(System.out::println);

        System.out.println("HQL Query:");

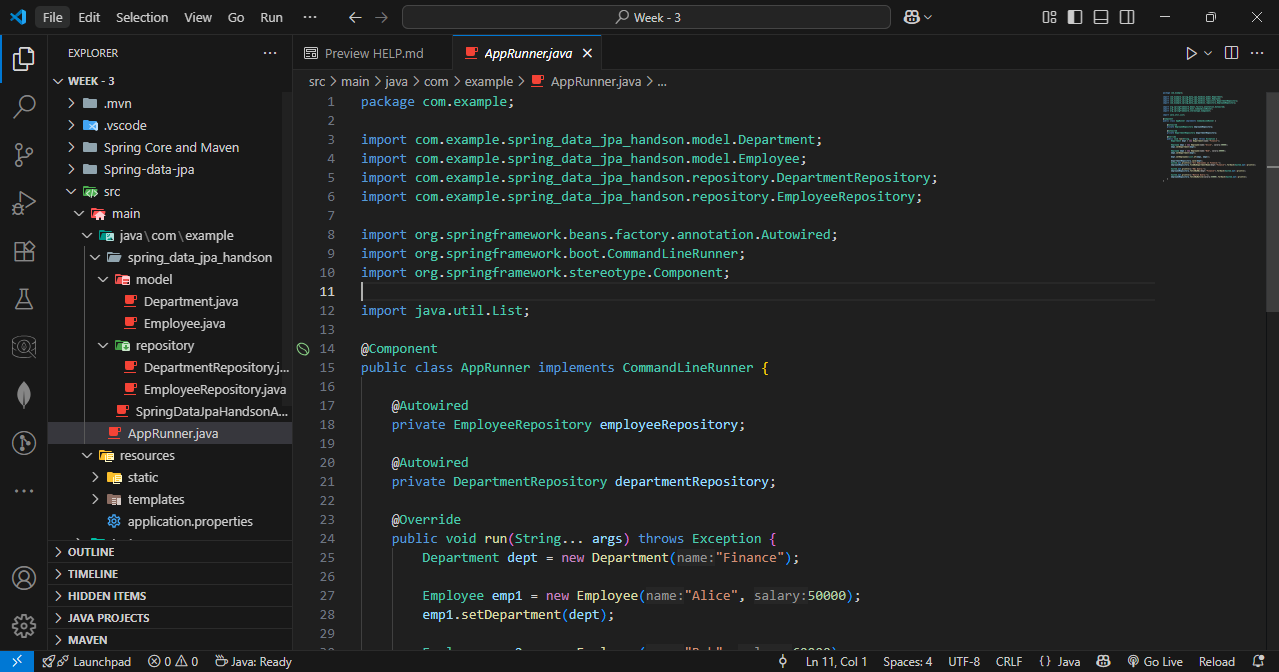
        employeeRepository.fetchByHQL("Finance").forEach(System.out::println);

        System.out.println("Native Query:");

        employeeRepository.fetchByNative(55000).forEach(System.out::println);

    }

}



OUTPUT :

Run the code :

