WEEK-12

Roll Number: 238W1A1296

SPRING JPA WITH RELATIONAL DATABASE

Dependencies: Spring web, Spring data jpa and mysql drver

1. application.properties

```
spring.application.name=jpamysql
server.port=88

# JDBC - change user/password/db name as needed
spring.datasource.url=jdbc:mysql://localhost:3306/employee_db?useSSL=false&allowPublicKey
Retrieval=true&serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=
spring.datasource.driver-class-name=com.mysql.jdbc.Driver

# JPA / Hibernate
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
```

2. Create database + sample data (run in MySQL)

```
CREATE DATABASE IF NOT EXISTS testdb;
USE employee_db;

CREATE TABLE IF NOT EXISTS employee (
  id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(100),
  department VARCHAR(100)
);

INSERT INTO employee (name, department) VALUES
('Manogna','IT'),
('Keerthi','CSE'),
('Sai','ECE');
```

3) Entity — Employee.java

src/main/java/com/example/demo/Employee.java

```java
package com.example.demo;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class Employee {

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

    private String name;
    private String department;

    public Employee() {}

    public Employee(String name, String department) {
        this.name = name;
        this.department = department;
    }

    // getters and setters — required for Jackson / JPA
    public Integer getId() { return id; }
    public void setId(Integer id) { this.id = id; }

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

    public String getDepartment() { return department; }
    public void setDepartment(String department) { this.department = department; }
}
```

4) Repository — EmployeeRepository.java

src/main/java/com/example/demo/EmployeeRepository.java

```java
package com.example.demo;

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

public interface EmployeeRepository extends JpaRepository<Employee, Integer> { }
```

5) Controller — returns JSON and HTML view

src/main/java/com/example/demo/EmployeeController.java

```
package com.example.demo;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

// REST endpoints
@RestController
@RequestMapping("/api")
class EmployeeRestController {
    @Autowired
    private EmployeeRepository repo;

    @GetMapping("/employees")
    public List<Employee> getAll() {
        return repo.findAll();
    }
}

// HTML (Thymeleaf) controller
@Controller
class EmployeeViewController {
    @Autowired
    private EmployeeRepository repo;

    @GetMapping("/employees")
    public String employeesPage(Model model) {
        model.addAttribute("employees", repo.findAll());
        return "employees";  // maps to src/main/resources/templates/employees.html
    }
}
```

Note: we used two controllers — one @RestController for JSON (/api/employees) and one @Controller for HTML (/employees). They can coexist.

6) Thymeleaf template — employees.html

Create file: src/main/resources/templates/employees.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
   <meta charset="UTF-8"/>
   <title>Employees</title>
   <style>
     table { border-collapse: collapse; width: 70%; margin-top: 20px; }
     th, td { border: 1px solid #333; padding: 8px; text-align: left; }
     th { background: #efefef; }
   </style>
</head>
<body>
  <h2>Employee List</h2>
  <table>
   <thead>
     <tr><th>ID</th><th>Name</th><th>Department</th></tr>
   </thead>
   <tbody>
     <tr th:each="e : ${employees}">
       <td th:text="${e.id}">1</td>
       <td th:text="${e.name}">Name</td>
       <td th:text="${e.department}">Dept</td>
     </tr>
   </tbody>
  </table>
</body>
</html>
```

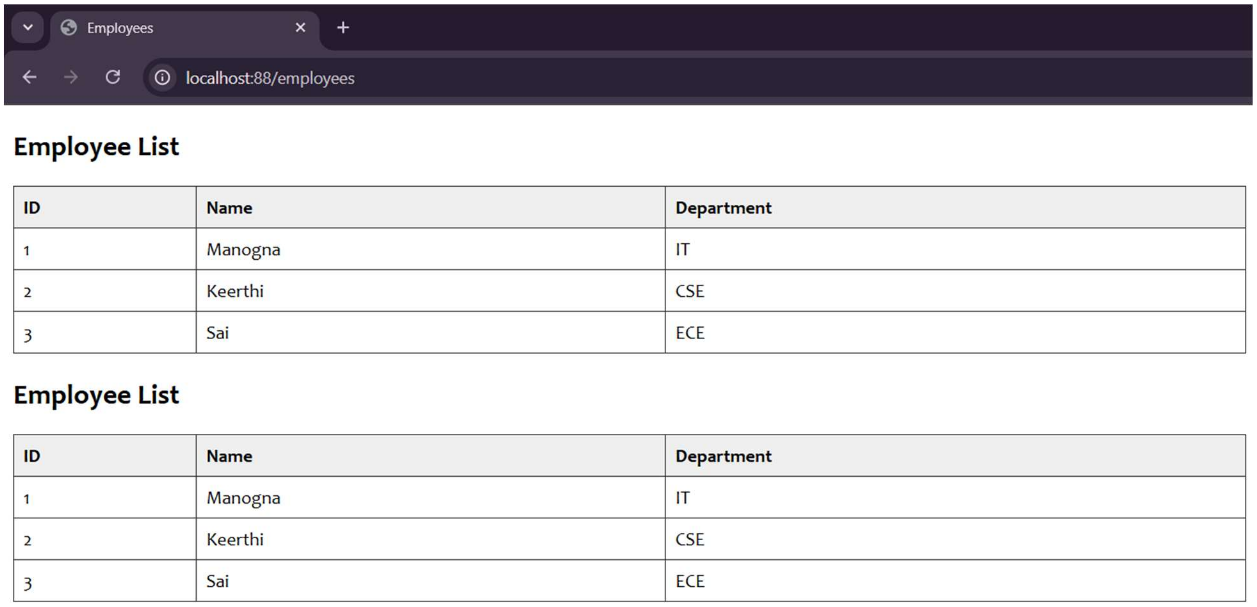
7. pom.xml add these code

```
<dependency>
   <groupId>mysql</groupId>
   <artifactId>mysql-connector-java</artifactId>
   <version>5.1.49</version>
</dependency>

<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```

8) Run in STS

1. Import project into STS (File → Import → Existing Maven Projects).
2. Right-click project → **Run As** → **Spring Boot App**.
3. Visit:
    - HTML table: http://localhost:88/employees
    - JSON API: http://localhost:88/api/employees

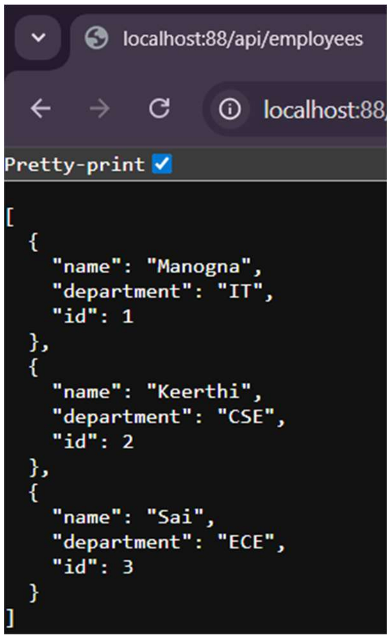


Employee List

ID	Name	Department
1	Manogna	IT
2	Keerthi	CSE
3	Sai	ECE

Employee List

ID	Name	Department
1	Manogna	IT
2	Keerthi	CSE
3	Sai	ECE



```
[
  {
    "name": "Manogna",
    "department": "IT",
    "id": 1
  },
  {
    "name": "Keerthi",
    "department": "CSE",
    "id": 2
  },
  {
    "name": "Sai",
    "department": "ECE",
    "id": 3
  }
]
```

2. Spring jpa with h2

Project name: h2db-jpa

Student.java:

```
package com.example.mysqldemo;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
@Entity
public class Student {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String city;
    public Student() {}
    public Student(String name, String city) {
        this.name = name;
        this.city = city;}
    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 String getCity() {        return city;    }
    public void setCity(String city) {        this.city = city;    }}
```

Student Controller.java:

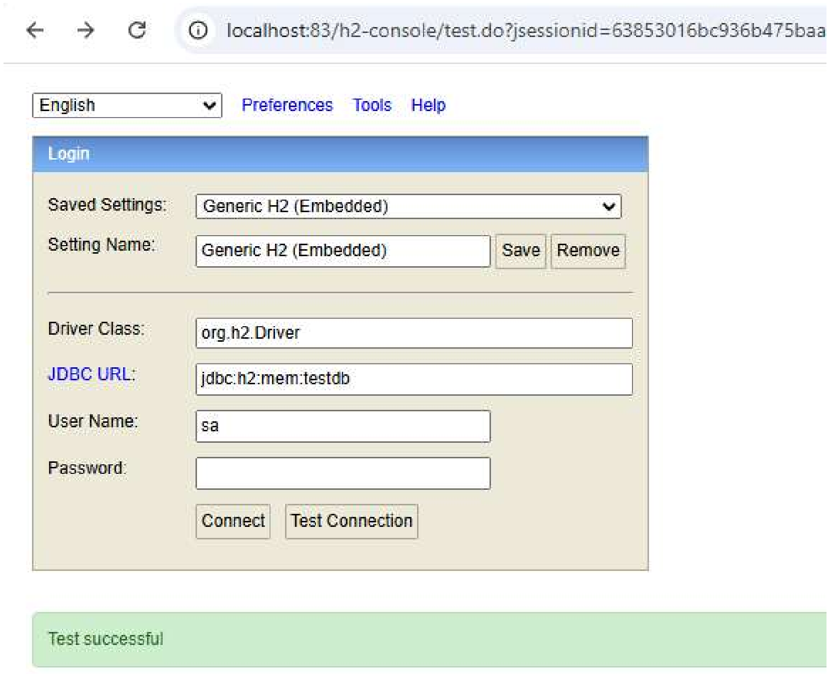
```
package com.example.mysqldemo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import java.util.List;
@RestController
public class StudentController {
   @Autowired
   private StudentRepository repo;
   @GetMapping("/")
   public String home() {
      return "Spring Boot + H2 + JPA Example Running!";    }
   @GetMapping("/add")
   public String addStudent(@RequestParam String name,
                           @RequestParam String city) {
      repo.save(new Student(name, city));
      return "Student Added Successfully!";
   }
   @GetMapping("/students")
   public List<Student> getStudents() {
      return repo.findAll();}}
```

Student Repository.java:

```
package com.example.mysqldemo;
import org.springframework.data.jpa.repository.JpaRepository;
public interface StudentRepository extends JpaRepository<Student, Long> {}
```

Application.properties:

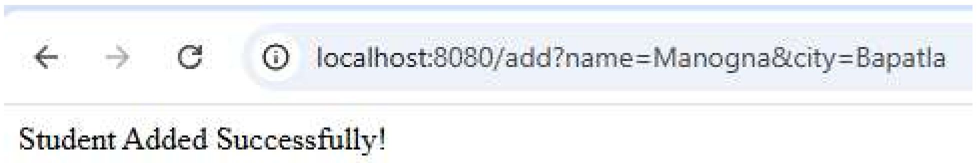
```
spring.application.name=h2db-jpa
spring.h2.console.enabled=true
spring.h2.console.path=/h2-console
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=update
```



localhost:83/h2-console/test.do?jsessionid=63853016bc936b475baa

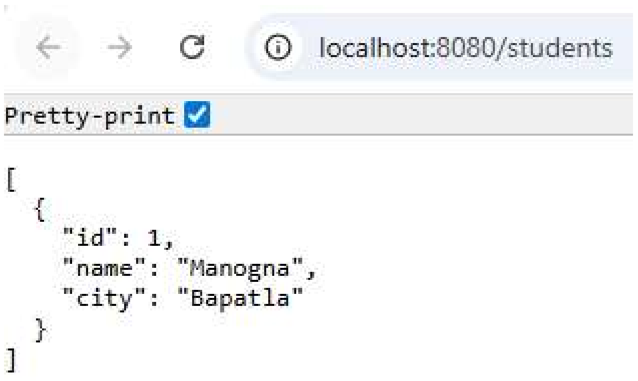
Saved Settings:	Generic H2 (Embedded)
Setting Name:	Generic H2 (Embedded)
Driver Class:	org.h2.Driver
JDBC URL:	jdbc:h2:mem:testdb
User Name:	sa
Password:	

Test successful



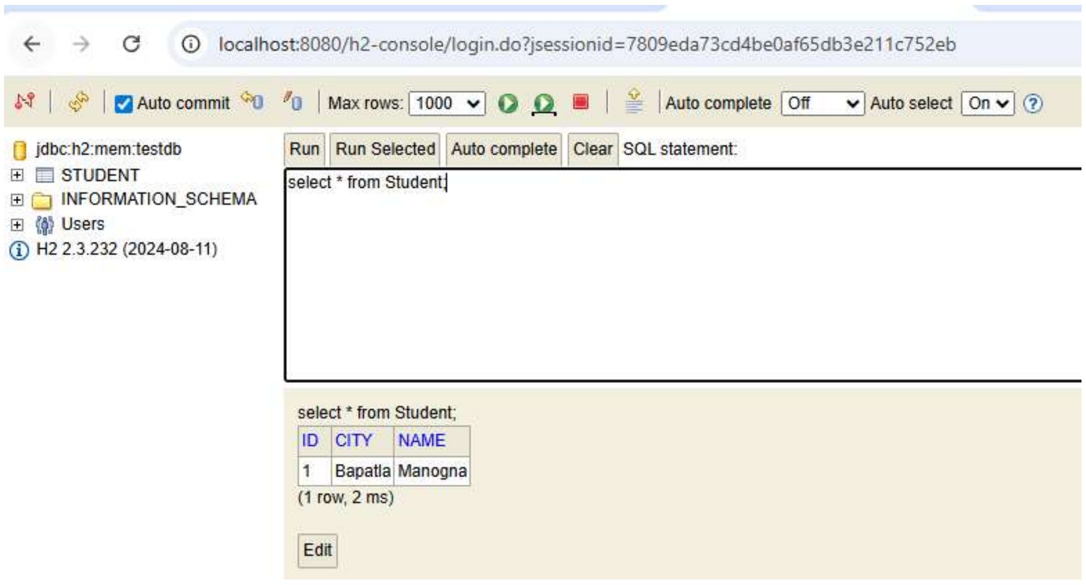
localhost:8080/add?name=Manogna&city=Bapatla

Student Added Successfully!



localhost:8080/students

```
[
  {
    "id": 1,
    "name": "Manogna",
    "city": "Bapatla"
  }
]
```


localhost:8080/h2-console/login.do?jsessionid=7809eda73cd4be0af65db3e211c752eb

Auto commit ✓ Max rows: 1000 Auto complete: Off Auto select: On

jdbc:h2:mem:testdb
STUDENT
INFORMATION_SCHEMA
Users
H2 2.3.232 (2024-08-11)

```
select * from Student;
```

select * from Student;

ID	CITY	NAME
1	Bapatla	Manogna

(1 row, 2 ms)