### 1. Creating a Spring Boot Project

#### 1.1 Initialize the Project

You can initialize the Spring Boot project using the Spring Initializr. Here’s how to do it:

**Visit Spring Initializr**: Go to [Spring Initializr](https://start.spring.io/" \t "_new).

**Project Metadata**:

* 1. **Project**: Maven Project (or Gradle if preferred)
  2. **Language**: Java
  3. **Spring Boot Version**: Choose the latest stable version (e.g., 3.x.x)
  4. **Group**: com.example
  5. **Artifact**: EmployeeManagementSystem
  6. **Name**: EmployeeManagementSystem
  7. **Description**: A project for managing employee data
  8. **Package Name**: com.example.employeemanagementsystem
  9. **Packaging**: Jar
  10. **Java Version**: Choose a version compatible with your environment (e.g., 17 or 21)

**Add Dependencies**:

* 1. **Spring Data JPA**
  2. **H2 Database**
  3. **Spring Web**
  4. **Lombok**

**Generate the Project**: Click on the "Generate" button to download a ZIP file containing the project. Extract this ZIP file to your working directory.

#### 1.2 Import the Project into an IDE

1. **Import into IDE**:
   1. Open your IDE (IntelliJ IDEA, Eclipse, VS Code, etc.).
   2. Import the project as a Maven or Gradle project depending on what you selected.

### 2. Configuring Application Properties

#### 2.1 Open application.properties

Navigate to src/main/resources/application.properties in your project.

#### 2.2 Configure H2 Database Connection

Add the following configuration to application.properties to set up the H2 database connection:

# H2 Database Configuration

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=password

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

# Enable H2 Console (optional, useful for debugging)

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console

* spring.datasource.url=jdbc:h2:mem:testdb sets up an in-memory H2 database.
* spring.jpa.database-platform=org.hibernate.dialect.H2Dialect specifies the H2 dialect for Hibernate.
* spring.h2.console.enabled=true enables the H2 database console, which you can access for debugging at http://localhost:8080/h2-console.

### 3. Basic Setup and Structure

#### 3.1 Create Entity Classes

Define your entity classes for employees and departments.

**Employee.java**

package com.example.employeemanagementsystem.model;

import lombok.Data;

import lombok.NoArgsConstructor;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

@Data

@NoArgsConstructor

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String firstName;

private String lastName;

private String email;

// Add additional fields as necessary

}

Department.java:

package com.example.employeemanagementsystem.model;

import lombok.Data;

import lombok.NoArgsConstructor;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

@Data

@NoArgsConstructor

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

// Add additional fields as necessary

}

#### 3.2 Create Repository Interfaces

Create repository interfaces to manage database operations.

**EmployeeRepository.java**

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.model.Employee;

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

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

}

DepartmentRepository.java:

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.model.Department;

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

public interface DepartmentRepository extends JpaRepository<Department, Long> {

}

#### 3.3 Create REST Controller

Define a REST controller to expose endpoints for managing employees.

**EmployeeController.java**

package com.example.employeemanagementsystem.controller;

import com.example.employeemanagementsystem.model.Employee;

import com.example.employeemanagementsystem.repository.EmployeeRepository;

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

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/employees")

public class EmployeeController {

@Autowired

private EmployeeRepository employeeRepository;

@GetMapping

public List<Employee> getAllEmployees() {

return employeeRepository.findAll();

}

@PostMapping

public Employee createEmployee(@RequestBody Employee employee) {

return employeeRepository.save(employee);

}

// Add other endpoints as needed (e.g., GET by ID, DELETE, PUT for updates)

}

### 4. Running the Application

**Run the Application**:

* 1. Open a terminal or command prompt.
  2. Navigate to the root directory of your project.
  3. Execute the following command:

./mvnw spring-boot:run

or, if using Gradle:

./gradlew bootRun

**Access the Application**:

* Your application will be running at http://localhost:8080.
* If you enabled the H2 console, you can access it at http://localhost:8080/h2-console with JDBC URL jdbc:h2:mem:testdb, username sa, and password password.