Below is a simple HR Application implemented using the \*\*Spring Framework\*\*. This application will include basic functionality for managing employees (CRUD operations). It uses Spring Boot to simplify configuration.

---

### \*\*Prerequisites\*\*

1. \*\*Tools and Dependencies\*\*:

- Install JDK 8 or later.

- Set up your IDE (e.g., IntelliJ IDEA or Eclipse).

- Install Maven or Gradle.

- Use Spring Boot Starter.

2. \*\*Dependencies in `pom.xml`\*\*:

Add the following dependencies:

```xml

<dependencies>

<!-- Spring Boot Starter Web -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!-- Spring Boot Starter Data JPA -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<!-- H2 Database -->

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<!-- Spring Boot Starter Test -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

```

---

### \*\*Project Structure\*\*

```

hr-application/

├── src/main/java/com/example/hr

│ ├── HrApplication.java # Main Application

│ ├── controller

│ │ └── EmployeeController.java # REST Controller

│ ├── model

│ │ └── Employee.java # Employee Entity

│ ├── repository

│ │ └── EmployeeRepository.java # JPA Repository

│ └── service

│ └── EmployeeService.java # Service Layer

└── src/main/resources

├── application.properties # Spring Boot Configuration

└── data.sql # Sample Data

```

---

### \*\*Code Implementation\*\*

#### 1. `HrApplication.java` (Main Class)

```java

package com.example.hr;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class HrApplication {

public static void main(String[] args) {

SpringApplication.run(HrApplication.class, args);

}

}

```

#### 2. `Employee.java` (Model/Entity)

```java

package com.example.hr.model;

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 Long id;

private String name;

private String department;

private double 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 String getDepartment() {

return department;

}

public void setDepartment(String department) {

this.department = department;

}

public double getSalary() {

return salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

}

```

#### 3. `EmployeeRepository.java` (Repository)

```java

package com.example.hr.repository;

import com.example.hr.model.Employee;

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

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

}

```

#### 4. `EmployeeService.java` (Service Layer)

```java

package com.example.hr.service;

import com.example.hr.model.Employee;

import com.example.hr.repository.EmployeeRepository;

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

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository repository;

public List<Employee> getAllEmployees() {

return repository.findAll();

}

public Employee getEmployeeById(Long id) {

return repository.findById(id).orElse(null);

}

public Employee saveEmployee(Employee employee) {

return repository.save(employee);

}

public void deleteEmployee(Long id) {

repository.deleteById(id);

}

}

```

#### 5. `EmployeeController.java` (Controller)

```java

package com.example.hr.controller;

import com.example.hr.model.Employee;

import com.example.hr.service.EmployeeService;

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

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

import java.util.List;

@RestController

@RequestMapping("/api/employees")

public class EmployeeController {

@Autowired

private EmployeeService service;

@GetMapping

public List<Employee> getAllEmployees() {

return service.getAllEmployees();

}

@GetMapping("/{id}")

public Employee getEmployeeById(@PathVariable Long id) {

return service.getEmployeeById(id);

}

@PostMapping

public Employee addEmployee(@RequestBody Employee employee) {

return service.saveEmployee(employee);

}

@PutMapping("/{id}")

public Employee updateEmployee(@PathVariable Long id, @RequestBody Employee updatedEmployee) {

Employee existingEmployee = service.getEmployeeById(id);

if (existingEmployee != null) {

existingEmployee.setName(updatedEmployee.getName());

existingEmployee.setDepartment(updatedEmployee.getDepartment());

existingEmployee.setSalary(updatedEmployee.getSalary());

return service.saveEmployee(existingEmployee);

}

return null;

}

@DeleteMapping("/{id}")

public void deleteEmployee(@PathVariable Long id) {

service.deleteEmployee(id);

}

}

```

#### 6. `application.properties`

```properties

# 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

# H2 Console

spring.h2.console.enabled=true

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

```

#### 7. `data.sql` (Sample Data)

```sql

INSERT INTO EMPLOYEE (NAME, DEPARTMENT, SALARY) VALUES ('Alice', 'HR', 60000);

INSERT INTO EMPLOYEE (NAME, DEPARTMENT, SALARY) VALUES ('Bob', 'IT', 70000);

INSERT INTO EMPLOYEE (NAME, DEPARTMENT, SALARY) VALUES ('Charlie', 'Finance', 80000);

```

---

### \*\*Run and Test\*\*

1. \*\*Start the Application\*\*: Run the `HrApplication` class.

2. \*\*Access the H2 Console\*\*:

- Go to `http://localhost:8080/h2-console`.

- Use JDBC URL: `jdbc:h2:mem:testdb`.

3. \*\*API Endpoints\*\*:

- `GET /api/employees`: Get all employees.

- `GET /api/employees/{id}`: Get an employee by ID.

- `POST /api/employees`: Add a new employee.

- `PUT /api/employees/{id}`: Update an employee.

- `DELETE /api/employees/{id}`: Delete an employee.

---

### \*\*Sample API Requests\*\*

1. \*\*Add Employee\*\*:

```json

POST /api/employees

{

"name": "Diana",

"department": "Admin",

"salary": 50000

}

```

2. \*\*Update Employee\*\*:

```json

PUT /api/employees/1

{

"name": "Alice Smith",

"department": "HR",

"salary": 65000

}

```

This program demonstrates a simple Spring Boot-based HR application for managing employee data.