How to Create a Spring Boot Web Service in IntelliJ IDEA and MySQL

Raman Deep Singh

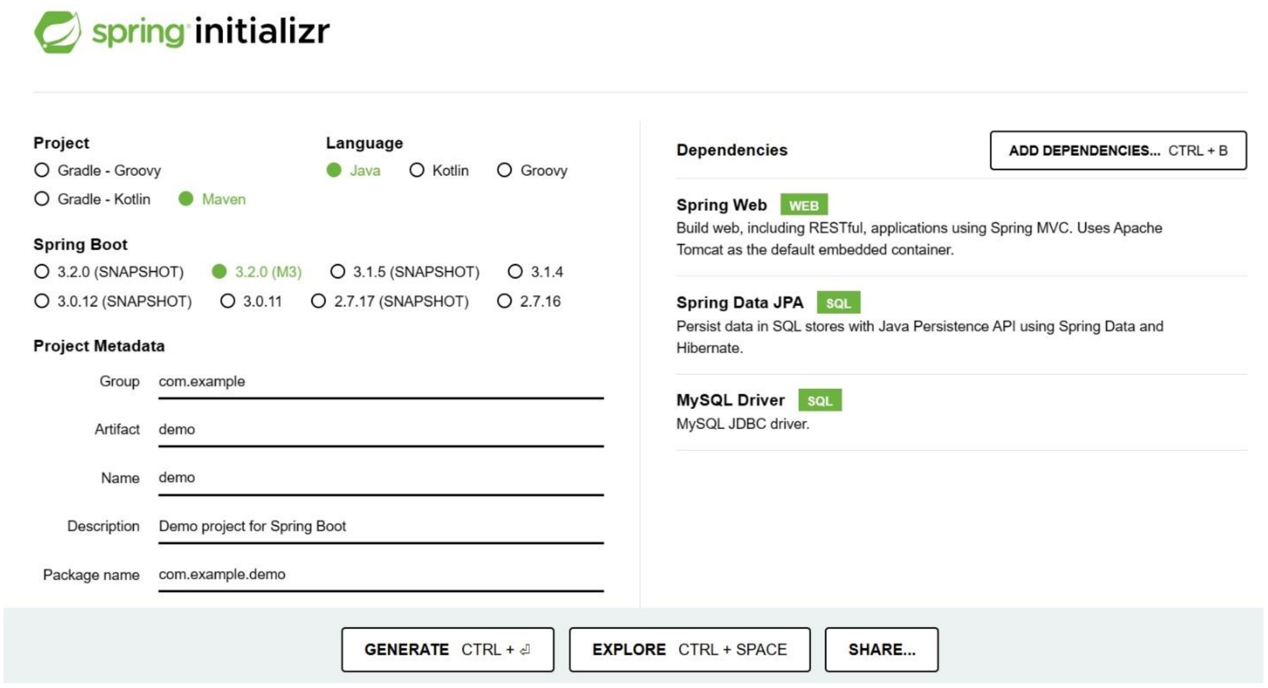
This tutorial will guide you to create Spring Boot Web Service using MySQL.

It is an Employee Management System.

SQL Scripts are

```sql
CREATE DATABASE  IF NOT EXISTS `employee_directory`;
USE `employee_directory`;

--
-- Table structure for table `employee`
--

DROP TABLE IF EXISTS `employee`;

CREATE TABLE `employee` (
  `id` int NOT NULL AUTO_INCREMENT,
  `first_name` varchar(45) DEFAULT NULL,
  `last_name` varchar(45) DEFAULT NULL,
  `email` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=latin1;

--
-- Data for table `employee`
--

INSERT INTO `employee` VALUES
  (1,'Leslie','Andrews','leslie@luv2code.com'),
  (2,'Emma','Baumgarten','emma@luv2code.com'),
  (3,'Avani','Gupta','avani@luv2code.com'),
  (4,'Yuri','Petrov','yuri@luv2code.com'),
  (5,'Juan','Vega','juan@luv2code.com');
```

Open https://start.spring.io and generate the following spring project


spring initializr

Project
- Gradle - Groovy
- Gradle - Kotlin
- [✓ Maven]

Language
- [✓ Java]
- Kotlin
- Groovy

Spring Boot
- 3.2.0 (SNAPSHOT)
- [✓ 3.2.0 (M3)]
- 3.1.5 (SNAPSHOT)
- 3.1.4
- 3.0.12 (SNAPSHOT)
- 3.0.11
- 2.7.17 (SNAPSHOT)
- 2.7.16

Project Metadata

Group	com.example
Artifact	demo
Name	demo
Description	Demo project for Spring Boot
Package name	com.example.demo

Dependencies

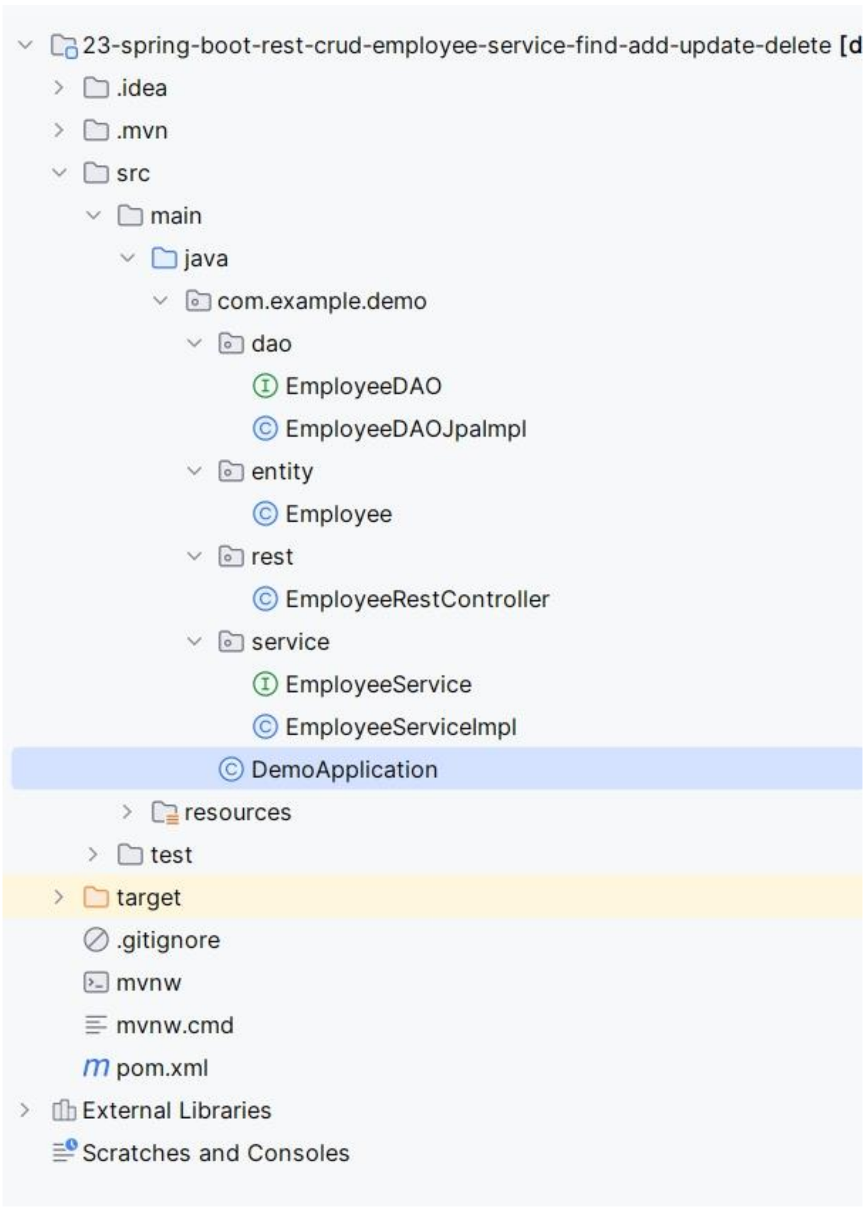
Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

MySQL Driver SQL
MySQL JDBC driver.

Click on Generate and open the project in IntelliJ IDEA Community Edition

Following is the project structure



```
23-spring-boot-rest-crud-employee-service-find-add-update-delete [d
  .idea
  .mvn
  src
    main
      java
        com.example.demo
          dao
            EmployeeDAO
            EmployeeDAOJpaImpl
          entity
            Employee
          rest
            EmployeeRestController
          service
            EmployeeService
            EmployeeServiceImpl
          DemoApplication
      resources
    test
  target
  .gitignore
  mvnw
  mvnw.cmd
  pom.xml
External Libraries
Scratches and Consoles
```

EmployeeDAO is an interface

EmployeeDAOJpaImpl is a class

Employee is a Class

Employee Rest Controller is a Class

Employee Service is a interface

EmployeeServiceImpl is a Class

Demo Application is a Class

Code for application.properties file

```
spring.datasource.url=jdbc:mysql://localhost:3306/employee_directory
spring.datasource.username=springstudent
spring.datasource.password=springstudent
```

Code for EmployeeDAO

```java
package com.example.demo.dao;

import com.example.demo.entity.Employee;

import java.util.List;

public interface EmployeeDAO {

    List<Employee> findAll();

    Employee findById(int theId);

    Employee save(Employee theEmployee);

    void deleteById(int theId);
}
```

____

Code for EmployeeDAOJpaImpl

```java
package com.example.demo.dao;

import com.example.demo.entity.Employee;
import jakarta.persistence.EntityManager;
import jakarta.persistence.TypedQuery;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public class EmployeeDAOJpaImpl implements EmployeeDAO {

    // define field for entitymanager
    private EntityManager entityManager;


    // set up constructor injection
    @Autowired
    public EmployeeDAOJpaImpl(EntityManager theEntityManager) {
        entityManager = theEntityManager;
    }


    @Override
    public List<Employee> findAll() {

        // create a query
        TypedQuery<Employee> theQuery = entityManager.createQuery("from
Employee", Employee.class);

        // execute query and get result list
        List<Employee> employees = theQuery.getResultList();

        // return the results
        return employees;
    }
    @Override
    public Employee findById(int theId) {

        // get employee
        Employee theEmployee = entityManager.find(Employee.class, theId);

        // return employee
        return theEmployee;
    }

    @Override
    public Employee save(Employee theEmployee) {

        // save employee
        Employee dbEmployee = entityManager.merge(theEmployee);

        // return the dbEmployee
        return dbEmployee;
    }

    @Override
    public void deleteById(int theId) {

        // find employee by id
        Employee theEmployee = entityManager.find(Employee.class, theId);

        // remove employee
        entityManager.remove(theEmployee);
    }
}
```

____

Code for Employee Class

```
package com.example.demo.entity;

import jakarta.persistence.*;

@Entity
@Table(name="employee")
public class Employee {

    // define fields
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name="id")
    private int id;

    @Column(name="first_name")
    private String firstName;

    @Column(name="last_name")
    private String lastName;

    @Column(name="email")
    private String email;


    // define constructors
    public Employee() {
```

```java
    }

    public Employee(String firstName, String lastName, String email) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
    }

    // define getter/setter

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    // define toString
    @Override
    public String toString() {
        return "Employee{" +
                "id=" + id +
                ", firstName='" + firstName + '\'' +
                ", lastName='" + lastName + '\'' +
                ", email='" + email + '\'' +
                '}';
    }
}
```

____

Code for EmployeeRestController

```java
package com.example.demo.rest;
```

```java
import com.example.demo.service.EmployeeService;
import com.example.demo.entity.Employee;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api")
public class EmployeeRestController {

    private EmployeeService employeeService;

    // quick and dirty: inject employee dao (use constructor injection)
    @Autowired
    public EmployeeRestController(EmployeeService theEmployeeService) {
        employeeService = theEmployeeService;
    }

    // expose "/employees" and return a list of employees
    @GetMapping("/employees")
    public List<Employee> findAll() {
        return employeeService.findAll();
    }

    @GetMapping("/employees/{eId}")
    public Employee findById(@PathVariable int eId) {

        return employeeService.findById(eId);
    }
    @PostMapping("/employees")
    public Employee addEmployee(@RequestBody Employee theEmployee) {

        // also just in case they pass an id in JSON ... set id to 0
        // this is to force a save of new item ... instead of update

        theEmployee.setId(0);

        Employee dbEmployee = employeeService.save(theEmployee);

        return dbEmployee;
    }

    @DeleteMapping("/delemployee/{eId}")
    public void deleteById(@PathVariable int eId) {

        employeeService.deleteById(eId);
    }

}
```

____

Code for EmployeeService Interface

```java
package com.example.demo.service;

import com.example.demo.entity.Employee;

import java.util.List;
```

```java
public interface EmployeeService {

    List<Employee> findAll();

    Employee findById(int theId);

    Employee save(Employee theEmployee);

    void deleteById(int theId);

}
```

Code for EmployeeServiceImpl

```java
package com.example.demo.service;

import com.example.demo.dao.EmployeeDAO;
import com.example.demo.entity.Employee;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;

@Service
public class EmployeeServiceImpl implements EmployeeService {

    private EmployeeDAO employeeDAO;

    @Autowired
    public EmployeeServiceImpl(EmployeeDAO theEmployeeDAO) {
        employeeDAO = theEmployeeDAO;
    }

    @Override
    public List<Employee> findAll() {
        return employeeDAO.findAll();
    }

    @Override
    public Employee findById(int theId) {
        return employeeDAO.findById(theId);
    }

    @Transactional
    @Override
    public Employee save(Employee theEmployee) {
        return employeeDAO.save(theEmployee);
    }

    @Transactional
    @Override
    public void deleteById(int theId) {
        employeeDAO.deleteById(theId);
    }
}
```

____

Code for class DemoApplication

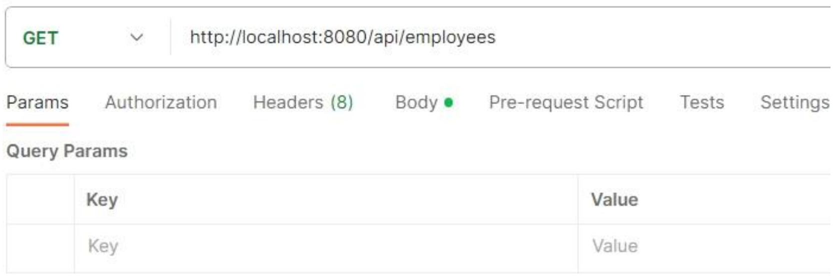
```java
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }

}
```

Now we will get Output of this project using POSTMAN

Send a GET request to

http://localhost:8080/api/employees

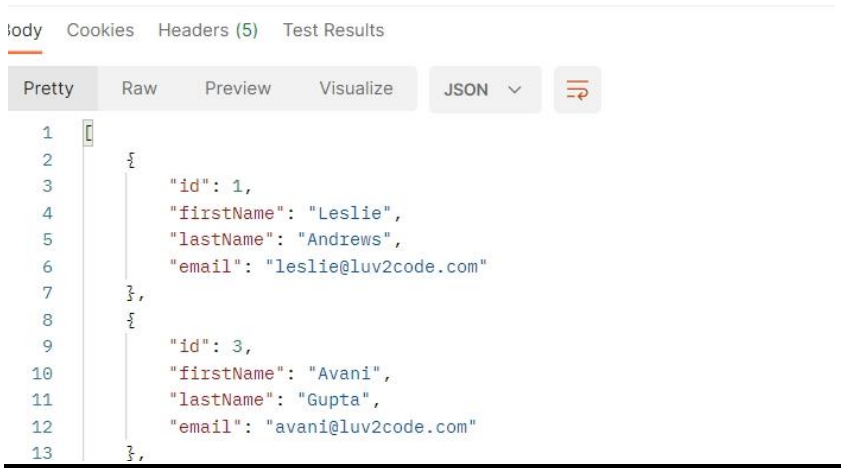
and you will get the output in Response Body which is a list of all employees in the database table.



http://localhost:8080/api/employees

Query Params

	Key	Value
	Key	Value

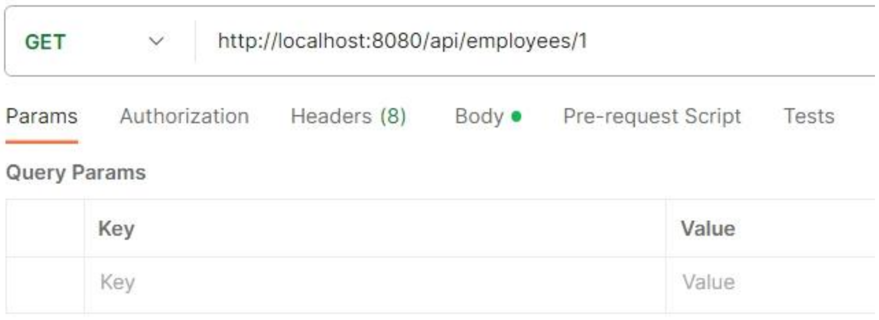


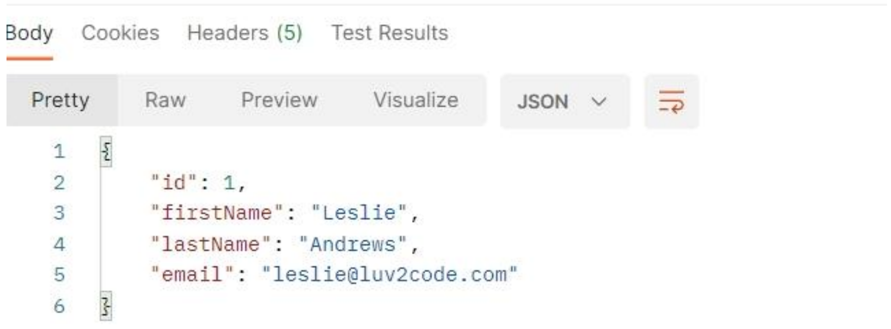
```json
[
    {
        "id": 1,
        "firstName": "Leslie",
        "lastName": "Andrews",
        "email": "leslie@luv2code.com"
    },
    {
        "id": 3,
        "firstName": "Avani",
        "lastName": "Gupta",
        "email": "avani@luv2code.com"
    },
```

Now second output

http://localhost:8080/api/employees/1

Output will be employee record with employee id as 1





```
{
    "id": 1,
    "firstName": "Leslie",
    "lastName": "Andrews",
    "email": "leslie@luv2code.com"
}
```

Now how to store a record in database table employee using POSTMAN

we will send a json code to add record in table

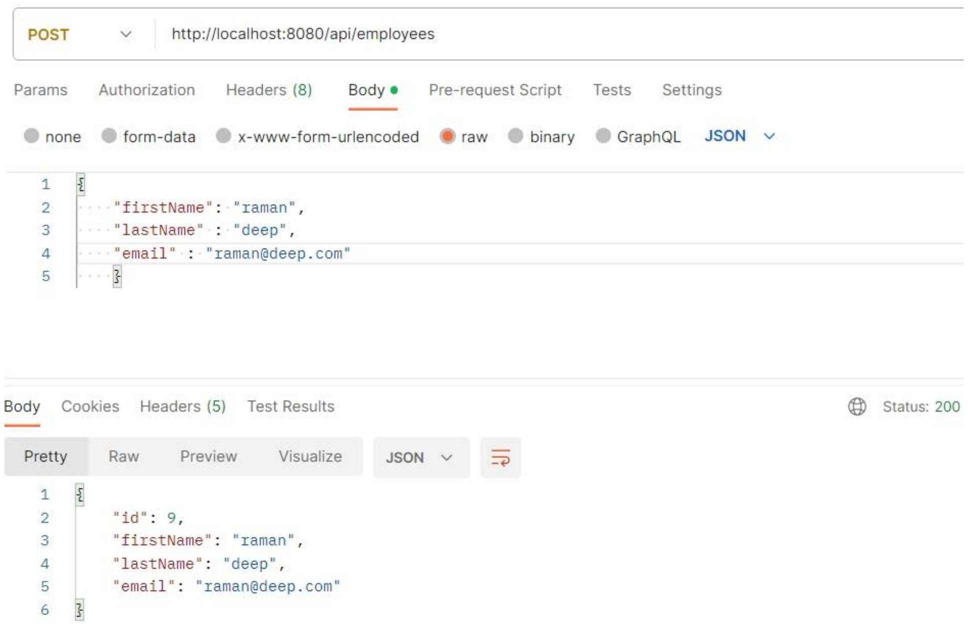
type of request is POST

and Body Type data is raw

code for json is

```
{
    "firstName": "raman",
    "lastName" : "deep",
    "email" : "raman@deep.com"
    }
```

Output



http://localhost:8080/api/employees

```
{
    "firstName": "raman",
    "lastName" : "deep",
    "email" : "raman@deep.com"
    }
```

Status: 200

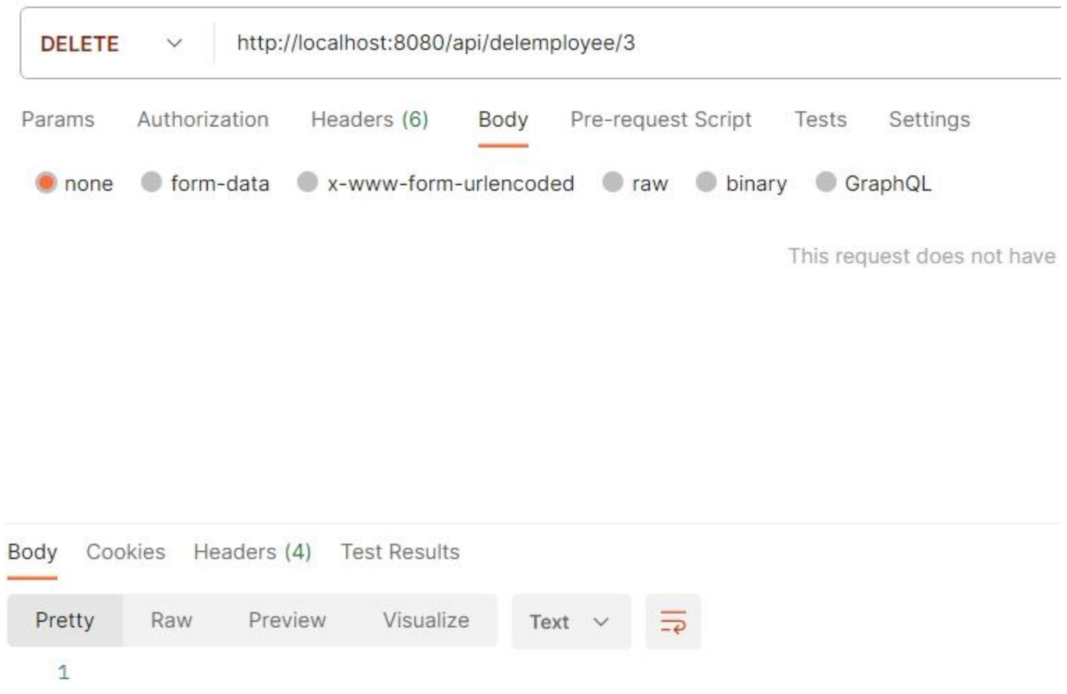
```
{
    "id": 9,
    "firstName": "raman",
    "lastName": "deep",
    "email": "raman@deep.com"
}
```

Now how to delete a record

send a DELETE request as

http://localhost:8080/api/delemployee/3

and you will get output as



http://localhost:8080/api/delemployee/3

This request does not have

After the above DELETE request record with id as 3 will be deleted from the table