# EMPLOYEE DETAILS CURD OPERATIONS DOCUMENTATION

---

Submitted by,

Name: Tumpuri Sai Charan
Regno: 223004093
Mobile: 9390144066
Sastra University ,thanjavur

Its a Spring Boot Rest Apis using Spring Data JPA with H2 Database for a Employees application in that:

- Each Employee has Id, Name, Address, mobile number, Department.
- Apis help to create, retrieve, update, delete Employees.
- Apis also support custom finder methods such as find by Employee.

These are APIs that we need to provide:

| Method | URLs | Actions |
|---|---|---|
| GET | /emp | Get all employees |
| POST | /employees | Create new Employees |
| UPDATE | /employees | Update Employees |
| DELETE | /emp/{empid} | Delete employees with the employee id |

# TECHNOLOGIES USED:

- JAVA
- Spring Boot
- H2 Data Base
- Maven

# Pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
     <modelVersion>4.0.0</modelVersion>
     <parent>
          <groupId>org.springframework.boot</groupId>
          <artifactId>spring-boot-starter-parent</artifactId>
          <version>2.7.4</version>
          <relativePath/> <!-- lookup parent from repository -->
     </parent>
     <groupId>com.example</groupId>
     <artifactId>SpringBootCrud</artifactId>
     <version>0.0.1-SNAPSHOT</version>
     <name>SpringBootCrud</name>
     <description>Demo project for Spring Boot giving details of
tutorial using h2 database</description>
     <properties>
          <java.version>1.8</java.version>
     </properties>
     <dependencies>
          <dependency>
               <groupId>org.springframework.boot</groupId>
               <artifactId>spring-boot-starter-data-jpa</
artifactId>
          </dependency>
          <dependency>
               <groupId>org.springframework.boot</groupId>
               <artifactId>spring-boot-starter-web</artifactId>
          </dependency>

          <dependency>
               <groupId>com.h2database</groupId>
               <artifactId>h2</artifactId>
               <scope>runtime</scope>
          </dependency>
          <dependency>
               <groupId>mysql</groupId>
               <artifactId>mysql-connector-java</artifactId>
               <scope>runtime</scope>
          </dependency>
          <dependency>
               <groupId>org.springframework.boot</groupId>
               <artifactId>spring-boot-starter-test</artifactId>
               <scope>test</scope>
          </dependency>
```

```
     </dependencies>

     <build>
          <plugins>
               <plugin>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-maven-plugin</
artifactId>
               </plugin>
          </plugins>
     </build>

</project>
```

# Application.properties

```
server.port=8087
import com.bezkoder.spring.jpa.h2.employeeservice.Employeeservice
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=charan
spring.datasource.password=password

spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.H2Di
alect
spring.jpa.hibernate.ddl-auto= update

spring.h2.console.enabled=true
# default path: h2-console
spring.h2.console.path=/h2-ui
```

# Models

# Employee.java

```
package com.charan.spring.jpa.h2.model;


import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;
//mark class as an Entity
@Entity
//defining class name as Table name
@Table
public class Employee
{
//Defining book id as primary key
@Id
@Column
private int empid;
@Column
private String empname;
@Column
private String empadress;
@Column
private long phone;
public long getPhone() {
	return phone;
}
public void setPhone(long phone) {
	this.phone = phone;
}
@Column
private String empdept;
public String getEmpdept() {
	return empdept;
}
public void setEmpdept(String empdept) {
	this.empdept = empdept;
```

```
}
public int getEmpid()
{
return empid;
}
public void setEmpid(int empid)
{
this.empid = empid;
}
public String getEmpname()
{
return empname;
}
public void setEmpname(String empname)
{
this.empname = empname;
}
public String getEmpadress()
{
return empadress;
}
public void setEmpadress(String empadress)
{
this.empadress = empadress;
}
}
```

# Controller

# EmployeeController.java

```
package com.charan.spring.jpa.h2.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RestController;

import com.charan.spring.jpa.h2.employeeservice1.EmployeeService1;
import com.charan.spring.jpa.h2.model.Employee;

import java.util.List;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
//mark class as Controller
@RestController
public class EmployeeController
{
@Autowired
EmployeeService1 employeeService;
//creating a get mapping that retrieves all the books detail from the database
@GetMapping("/emp")
private List<Employee> getAllEmployees()
{
return employeeService.getAllBooks() ;
}
//creating a get mapping that retrieves the detail of a specific book
@GetMapping("/emp/{empid}")
private Employee getBooks(@PathVariable("empid") int empid)
{
return employeeService.getBooksById(empid);
}
//creating a delete mapping that deletes a specified book
@DeleteMapping("/emp/{empid}")
private void deleteBook(@PathVariable("empid") int empid)
```

```
{
	employeeService.delete(empid);
}
//creating post mapping that post the book detail in the database
@PostMapping("/employees")
private int saveBook(@RequestBody Employee employees)
{
	employeeService.saveOrUpdate(employees);
	return employees.getEmpid();
}
//creating put mapping that updates the book detail
@PutMapping("/employees")
private Employee update(@RequestBody Employee employees)
{
	employeeService.saveOrUpdate(employees);
	return employees;
}
}
```

# Service layer

# EmployeeService.java

```
package com.charan.spring.jpa.h2.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RestController;

import com.charan.spring.jpa.h2.employeeservice1.EmployeeService1;
import com.charan.spring.jpa.h2.model.Employee;

import java.util.List;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
//mark class as Controller
@RestController
public class EmployeeController
{
@Autowired
EmployeeService1 employeeService;
//creating a get mapping that retrieves all the books detail from the database
@GetMapping("/emp")
private List<Employee> getAllEmployees()
{
return employeeService.getAllBooks() ;
}
//creating a get mapping that retrieves the detail of a specific book
@GetMapping("/emp/{empid}")
private Employee getBooks(@PathVariable("empid") int empid)
{
return employeeService.getBooksById(empid);
}
//creating a delete mapping that deletes a specified book
@DeleteMapping("/emp/{empid}")
private void deleteBook(@PathVariable("empid") int empid)
{
```

```
        employeeService.delete(empid);
}
//creating post mapping that post the book detail in the database
@PostMapping("/employees")
private int saveBook(@RequestBody Employee employees)
{
        employeeService.saveOrUpdate(employees);
        return employees.getEmpid();
}
//creating put mapping that updates the book detail
@PutMapping("/employees")
private Employee update(@RequestBody Employee employees)
{
        employeeService.saveOrUpdate(employees);
        return employees;
}
}
```

# Repository

# EmployeeRepository.java

```
package com.charan.spring.jpa.h2.repository1;

import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import com.charan.spring.jpa.h2.model.Employee;



@Repository
//repository that extends CrudRepository
public interface EmployeeRepository1 extends CrudRepository<Employee,
Integer>
{
}
```

# Application

# Employeeapplication.java

```
package com.charan.spring.jpa.h2.repository1;

import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import com.charan.spring.jpa.h2.model.Employee;



@Repository
//repository that extends CrudRepository
public interface EmployeeRepository1 extends CrudRepository<Employee,
Integer>
{
}
```

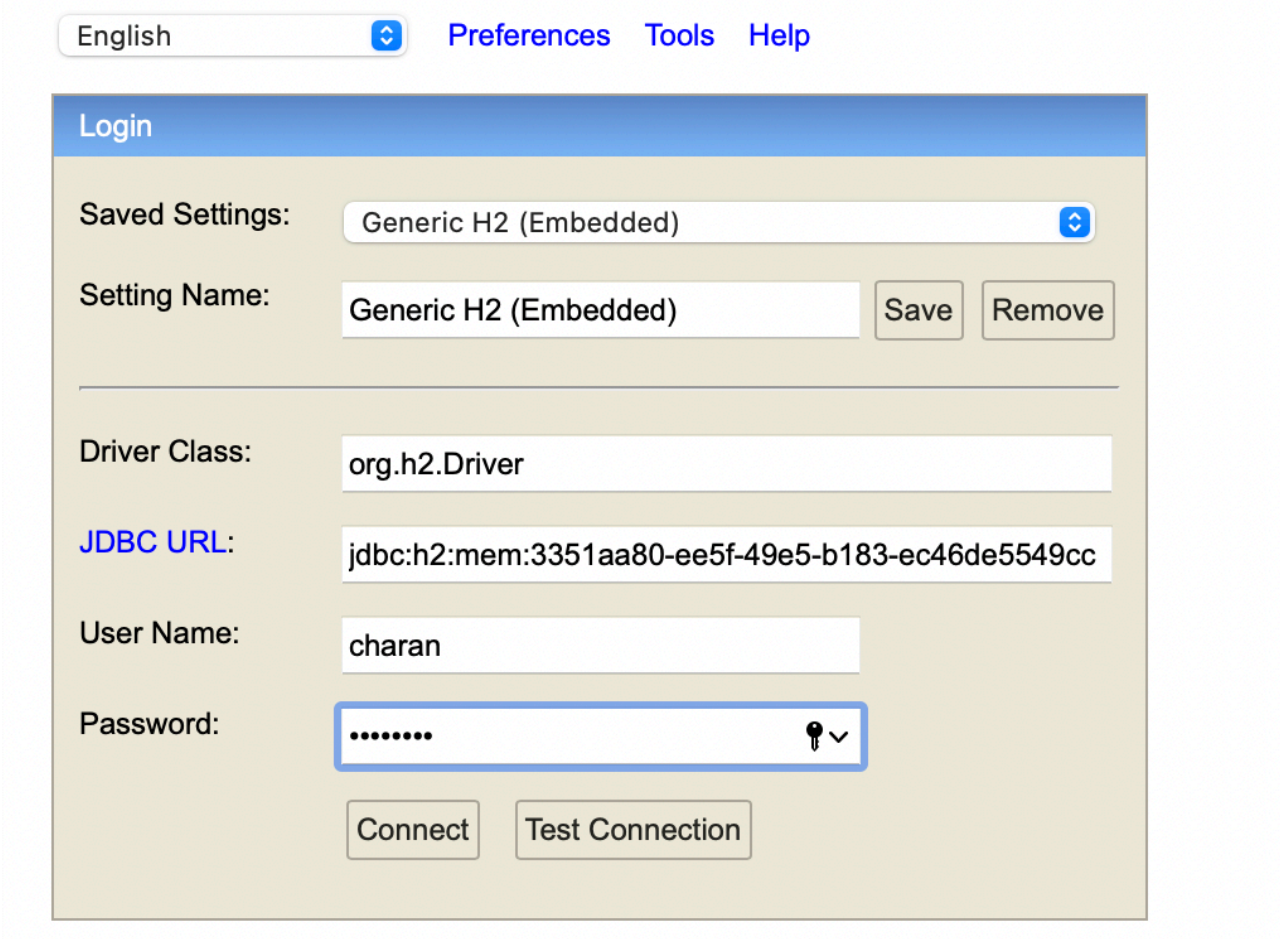
# Snapshots of the output

# Database login:

Run Spring Boot application with command: `mvn spring-boot:run`.

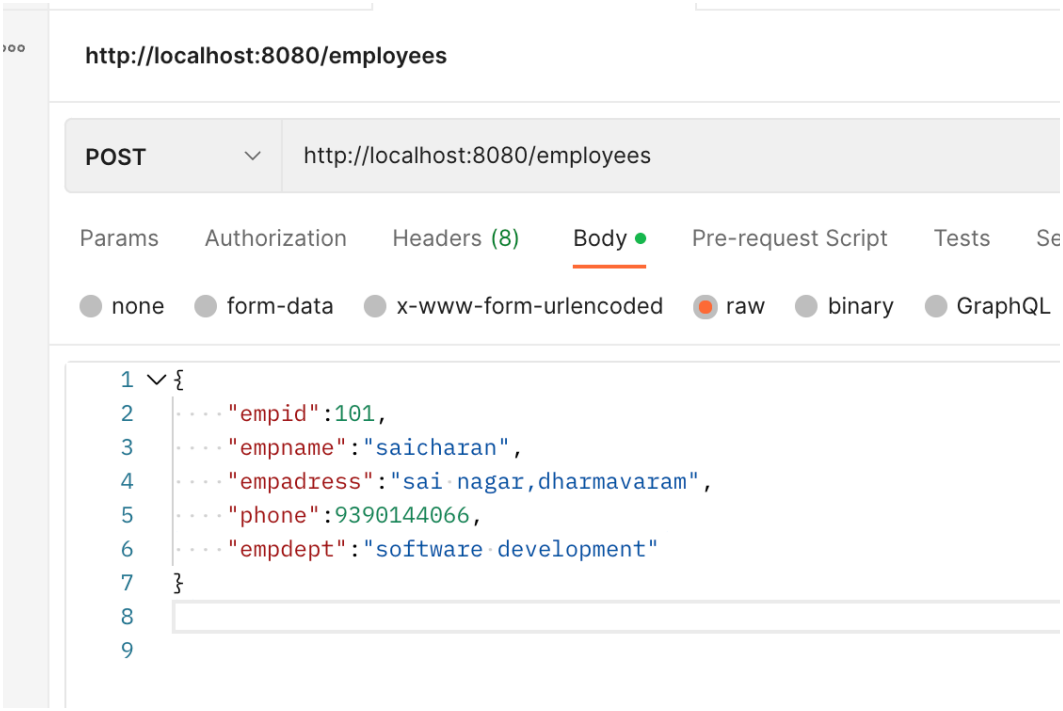
Let's open H2 console with url: http://localhost:8080/h2-ui

```
User name: charan
Password:password
```

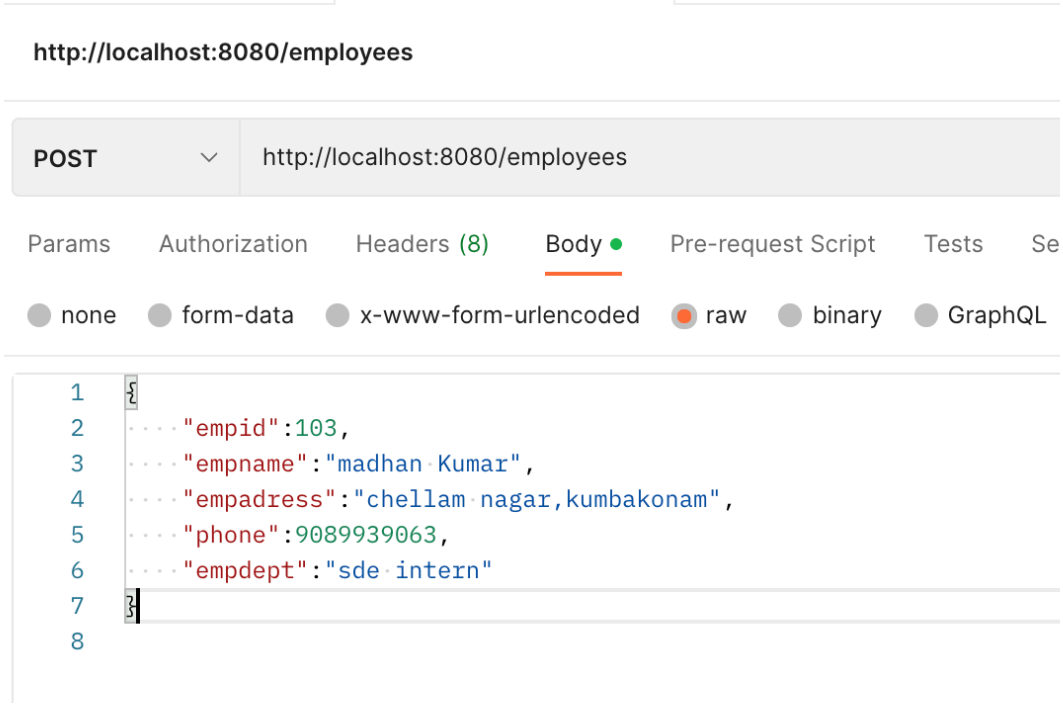



H2database connection

Let Create Some Employee details Using **POST:**

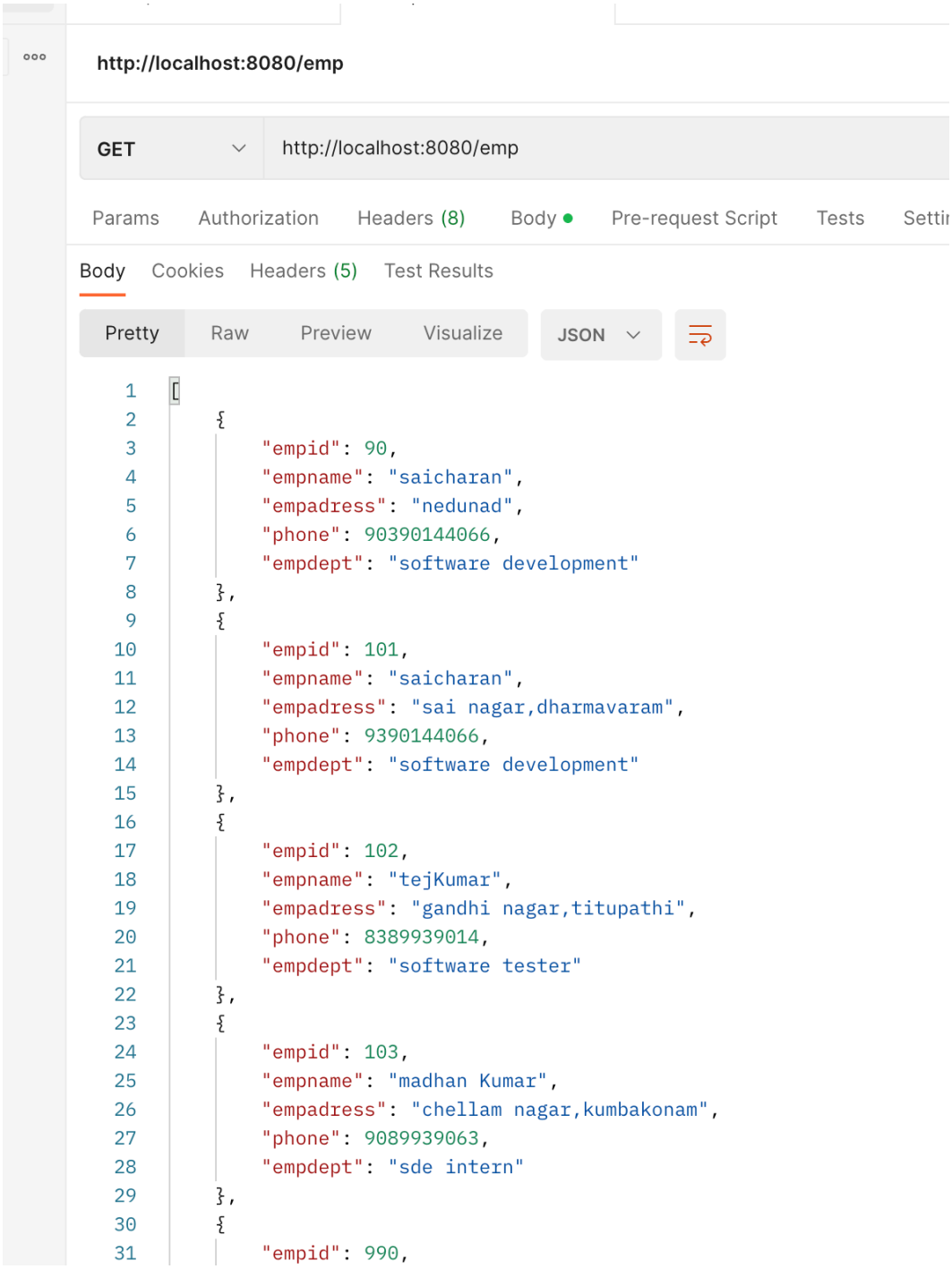



POST:inserting employees

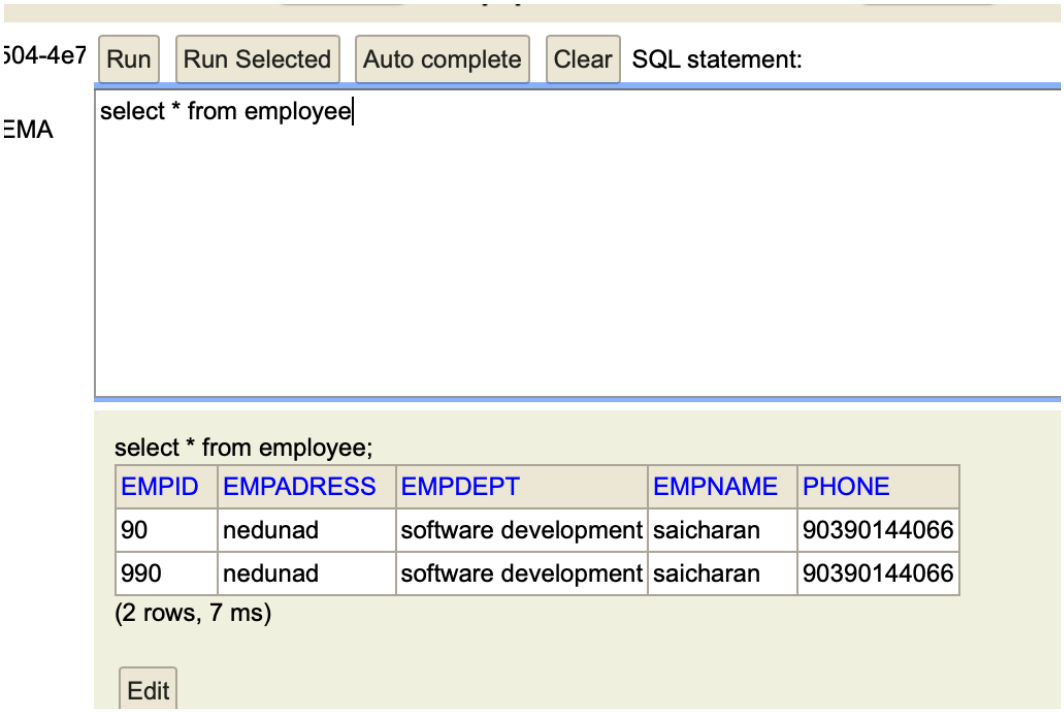



POST: Inserting Employee

Let Get Some Employee details Using **GET:**

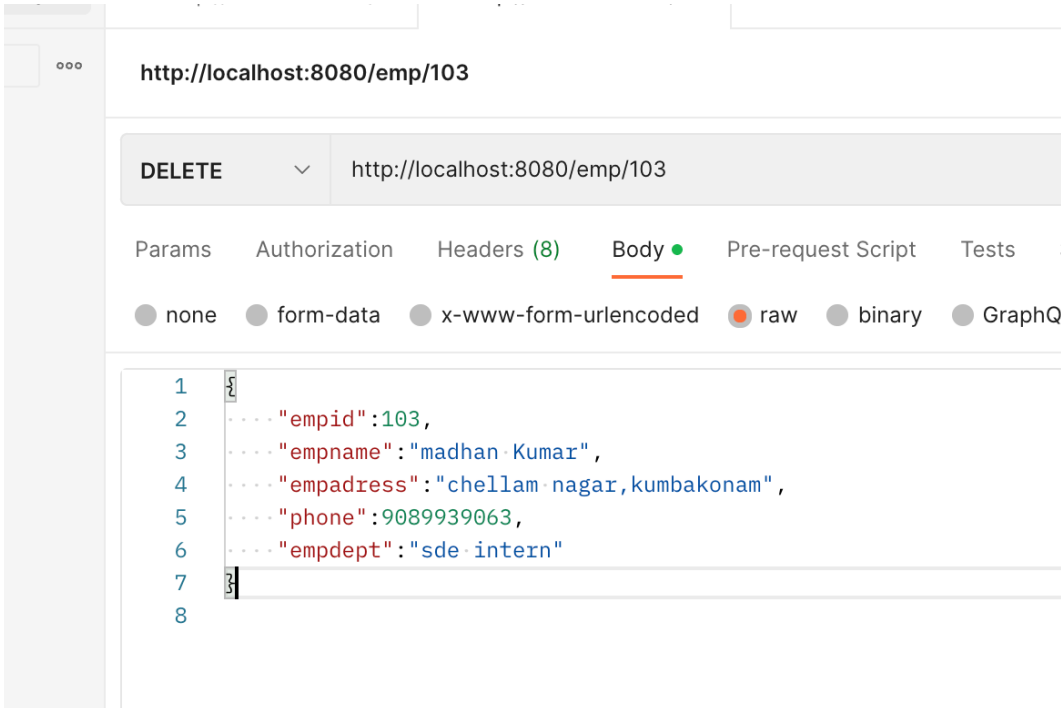



Retrieving employee Details by GET method




From the local host

Let DeleteSome Employee details Using **DELETE:**




Deleting the employee