SPRING BOOT EMPLOYEE MANAGEMENT REST API

(Internship Task Submission - Alfido Tech)

----

1. PROJECT OVERVIEW

This project is a Spring Boot based RESTful web application developed as part of my internship task.
The application provides a complete Employee Management system that supports CRUD (Create, Read, Update, Delete) operations. Employee data is stored in a MySQL database and the APIs are tested and documented using Swagger UI.

----

2. TECHNOLOGIES USED

- Java 17
- Spring Boot
- Spring Data JPA
- MySQL Database
- Maven
- Swagger UI (OpenAPI)
- VS Code

----

3. FEATURES IMPLEMENTED

- Create Employee using POST API
- Fetch All Employees using GET API
- Update Employee using PUT API
- Delete Employee using DELETE API
- RESTful API design
- Swagger UI integration for API testing and documentation

----

4. API ENDPOINTS

GET	/employees
POST	/employees
PUT	/employees/{id}
DELETE	/employees/{id}

----

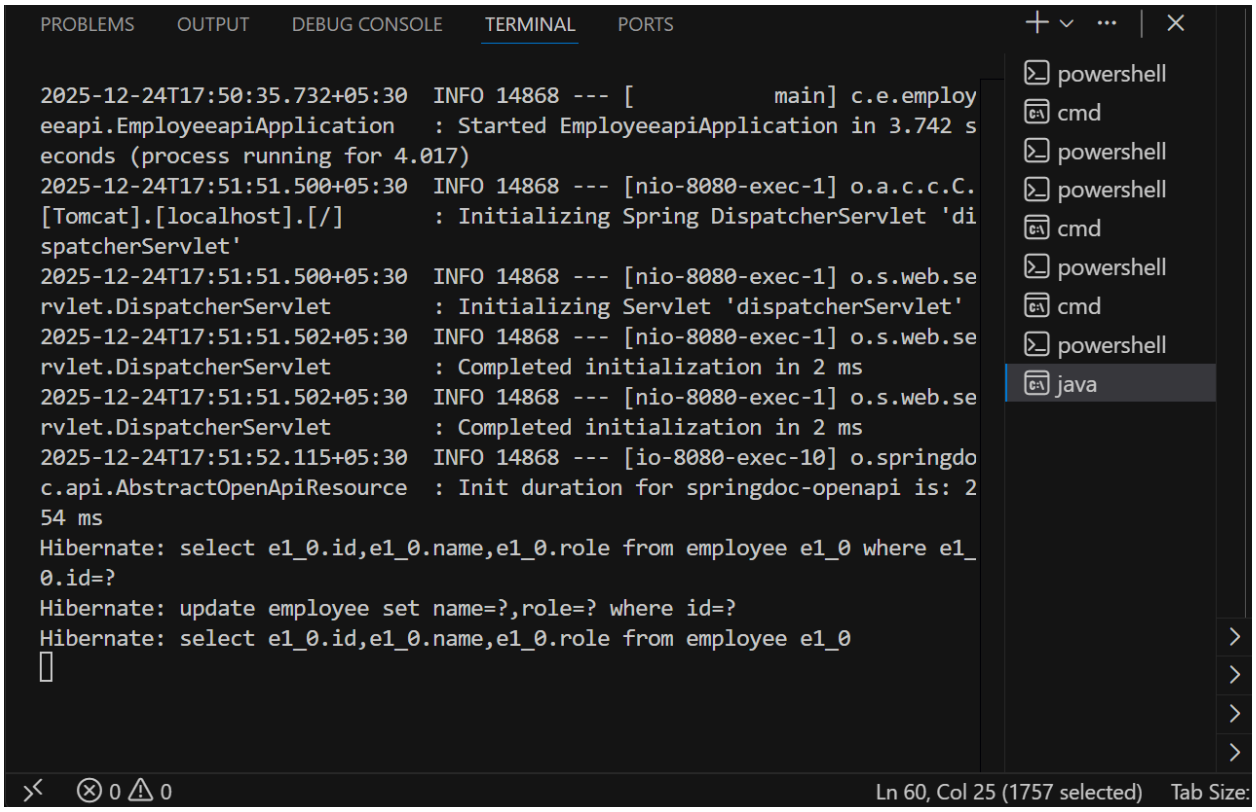
5. PROJECT WORKFLOW

1. Client sends HTTP request using Swagger UI or Thunder Client.
2. Controller layer receives the request.
3. Service layer processes business logic.
4. Repository layer interacts with MySQL database.
5. Response is returned to the client in JSON format.

----

6. SCREENSHOTS ATTACHED

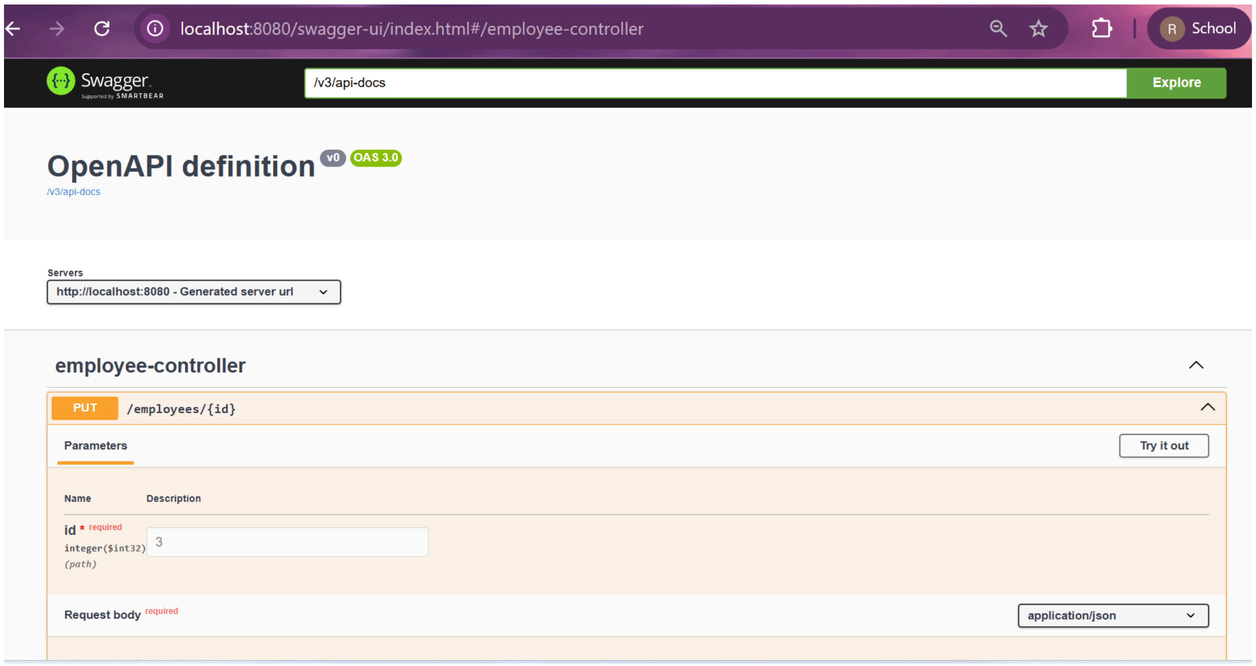
- Application running in VS Code (Tomcat started)



```
2025-12-24T17:50:35.732+05:30  INFO 14868 --- [           main] c.e.employ
eeapi.EmployeeapiApplication   : Started EmployeeapiApplication in 3.742 s
econds (process running for 4.017)
2025-12-24T17:51:51.500+05:30  INFO 14868 --- [nio-8080-exec-1] o.a.c.c.C.
[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'di
spatcherServlet'
2025-12-24T17:51:51.500+05:30  INFO 14868 --- [nio-8080-exec-1] o.s.web.se
rvlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2025-12-24T17:51:51.502+05:30  INFO 14868 --- [nio-8080-exec-1] o.s.web.se
rvlet.DispatcherServlet        : Completed initialization in 2 ms
2025-12-24T17:51:51.502+05:30  INFO 14868 --- [nio-8080-exec-1] o.s.web.se
rvlet.DispatcherServlet        : Completed initialization in 2 ms
2025-12-24T17:51:52.115+05:30  INFO 14868 --- [io-8080-exec-10] o.springdo
c.api.AbstractOpenApiResource  : Init duration for springdoc-openapi is: 2
54 ms
Hibernate: select e1_0.id,e1_0.name,e1_0.role from employee e1_0 where e1_
0.id=?
Hibernate: update employee set name=?,role=? where id=?
Hibernate: select e1_0.id,e1_0.name,e1_0.role from employee e1_0
```

Figure 1: Spring Boot application started successfully with embedded Tomcat server and MySQL database integration

- Swagger UI showing all Employee API


localhost:8080/swagger-ui/index.html#/employee-controller

/v3/api-docs

OpenAPI definition v0 OAS 3.0

/v3/api-docs

Servers

http://localhost:8080 - Generated server url

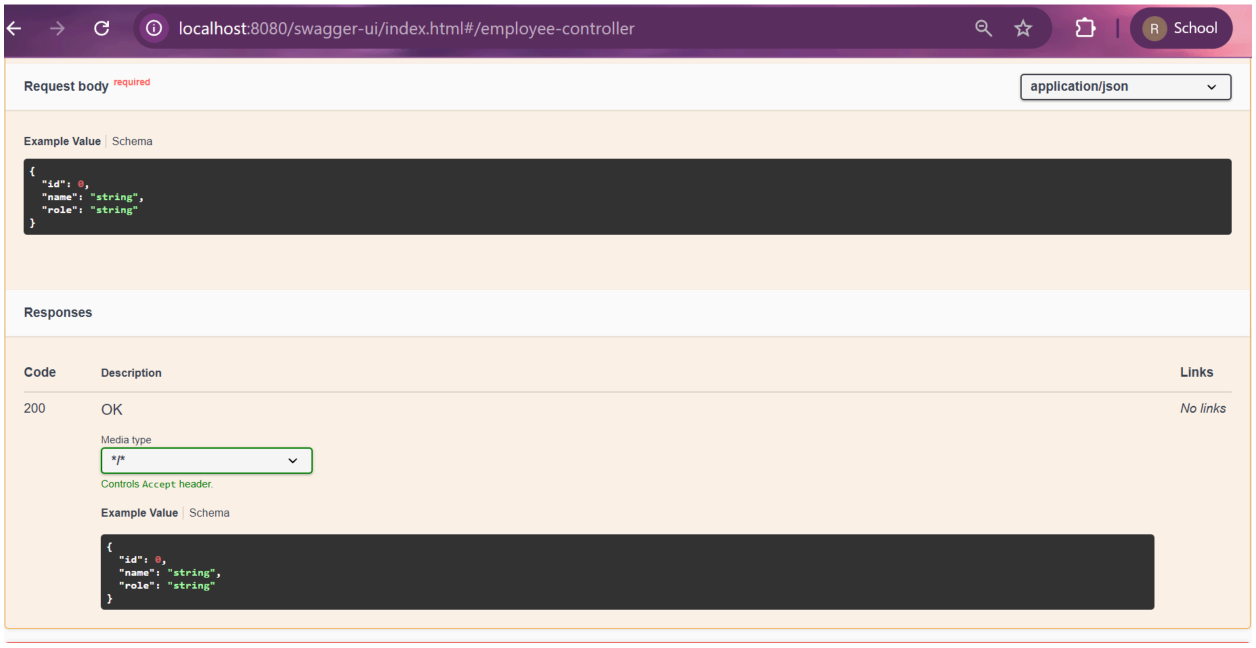
employee-controller

PUT /employees/{id}

Parameters

Name	Description
id * required integer($int32) (path)	3

Request body required application/json


localhost:8080/swagger-ui/index.html#/employee-controller

Request body required application/json

Example Value | Schema

```
{
  "id": 0,
  "name": "string",
  "role": "string"
}
```

Responses

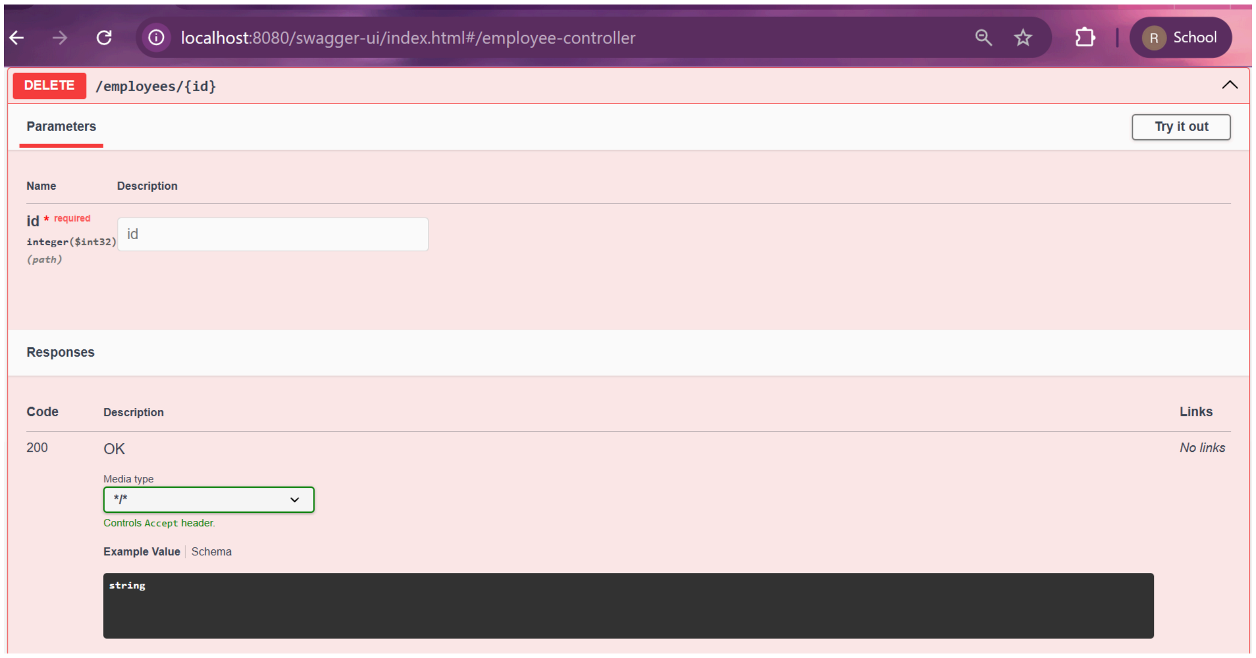
Code	Description	Links
200	OK	No links

Media type */*

Controls Accept header.

Example Value | Schema

```
{
  "id": 0,
  "name": "string",
  "role": "string"
}
```


localhost:8080/swagger-ui/index.html#/employee-controller

DELETE /employees/{id}

Parameters

Name	Description
id * required integer($int32) (path)	id

Responses

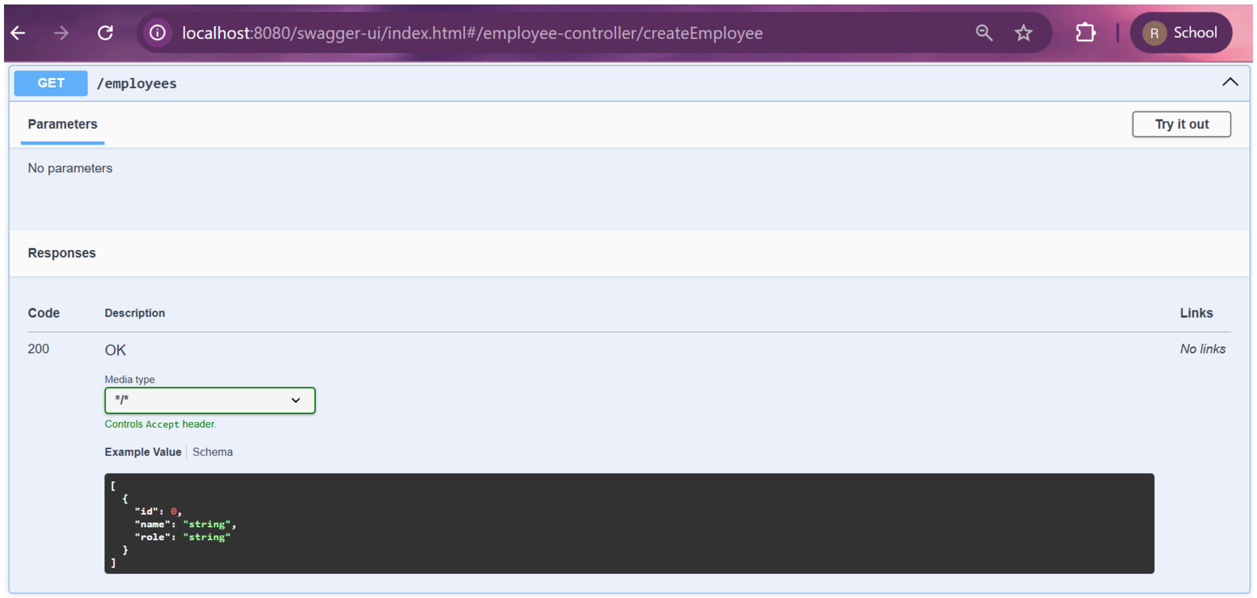
Code	Description	Links
200	OK	No links

Media type: */*

Controls Accept header.

Example Value | Schema

```
string
```


localhost:8080/swagger-ui/index.html#/employee-controller/createEmployee

GET /employees

Parameters

No parameters

Responses

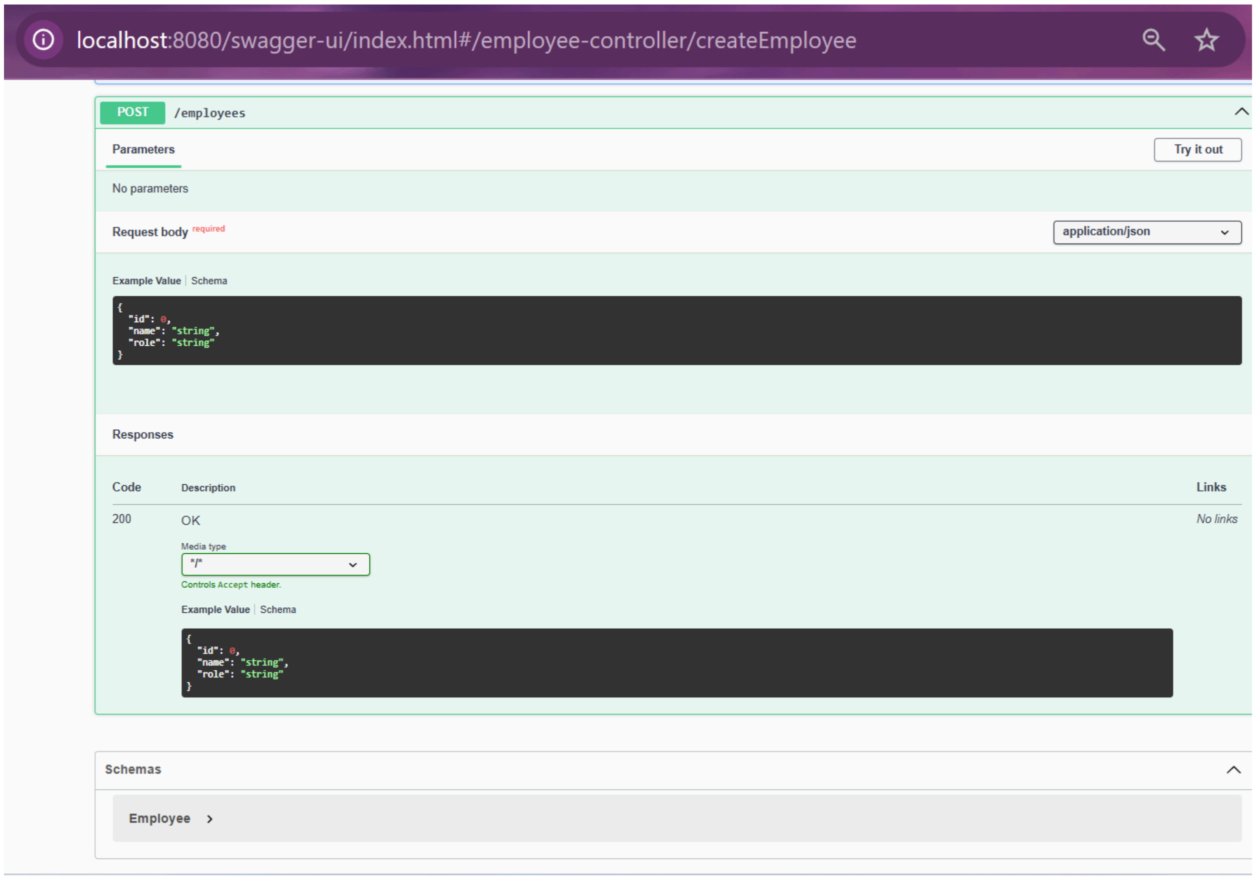
Code	Description	Links
200	OK	No links

Media type: */*

Controls Accept header.

Example Value | Schema

```
[
  {
    "id": 0,
    "name": "string",
    "role": "string"
  }
]
```


localhost:8080/swagger-ui/index.html#/employee-controller/createEmployee

No parameters

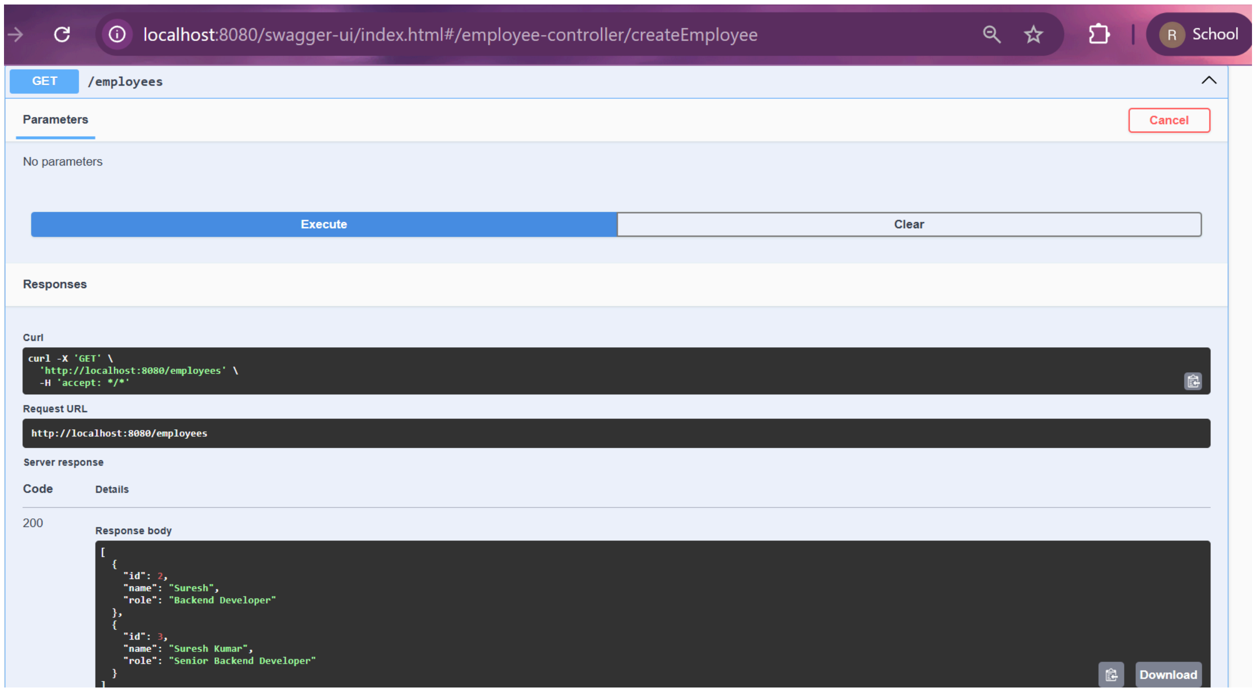
```
{
  "id": 0,
  "name": "string",
  "role": "string"
}
```

Code	Description	Links
200	OK	No links

```
{
  "id": 0,
  "name": "string",
  "role": "string"
}
```

Figure 2: Swagger UI displaying the OpenAPI documentation and available Employee Management REST APIs.

- GET employees API response


localhost:8080/swagger-ui/index.html#/employee-controller/createEmployee
School
GET
Parameters
Cancel
Execute
Clear
Download

/employees

No parameters

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8080/employees' \
  -H 'accept: */*'
```

Request URL

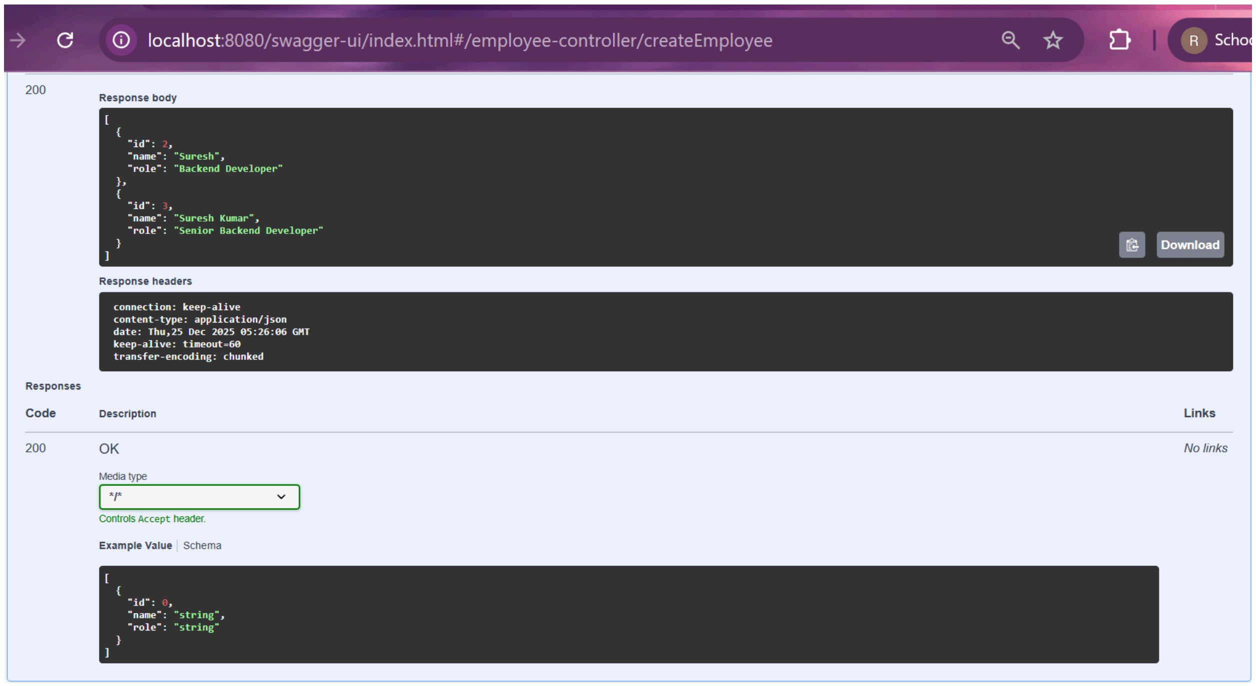
```
http://localhost:8080/employees
```

Server response

Code Details

200 Response body

```
[
  {
    "id": 2,
    "name": "Suresh",
    "role": "Backend Developer"
  },
  {
    "id": 3,
    "name": "Suresh Kumar",
    "role": "Senior Backend Developer"
  }
```


localhost:8080/swagger-ui/index.html#/employee-controller/createEmployee
Download

200 Response body

```
[
  {
    "id": 2,
    "name": "Suresh",
    "role": "Backend Developer"
  },
  {
    "id": 3,
    "name": "Suresh Kumar",
    "role": "Senior Backend Developer"
  }
]
```

Response headers

```
connection: keep-alive
content-type: application/json
date: Thu,25 Dec 2025 05:26:06 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	OK	*No links*

Media type

/

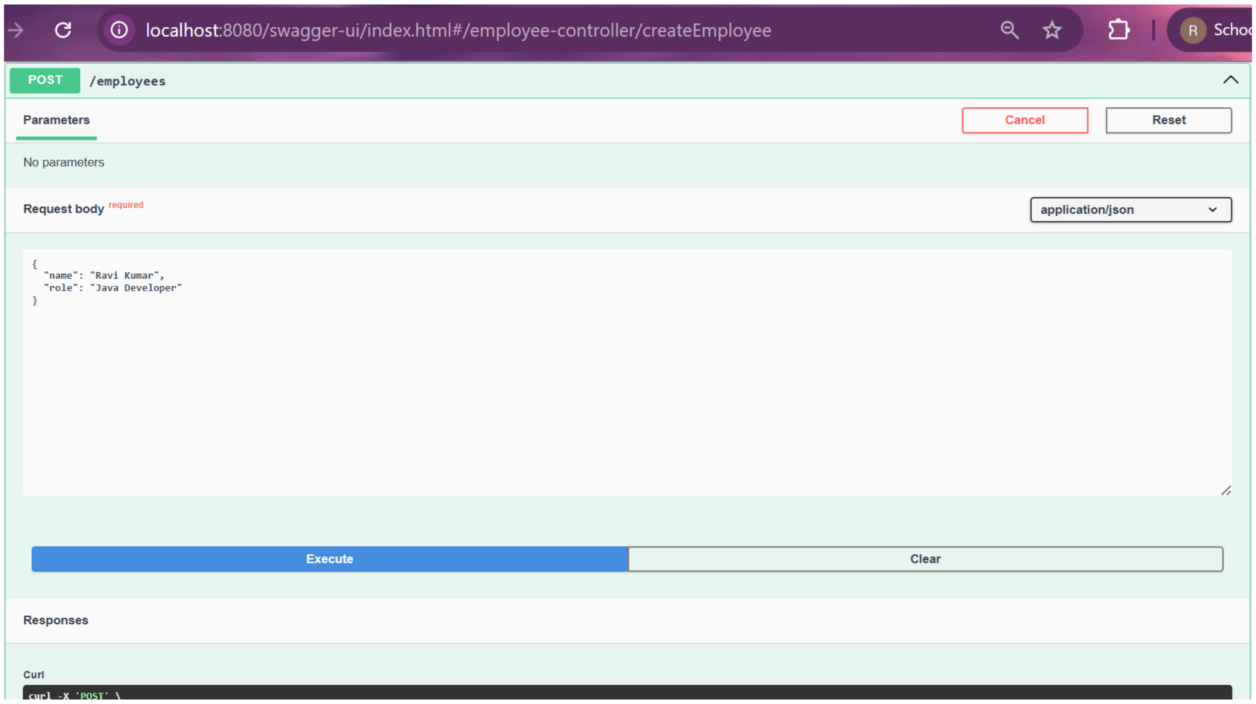
Controls Accept header.

Example Value | Schema

```
[
  {
    "id": 0,
    "name": "string",
    "role": "string"
  }
]
```

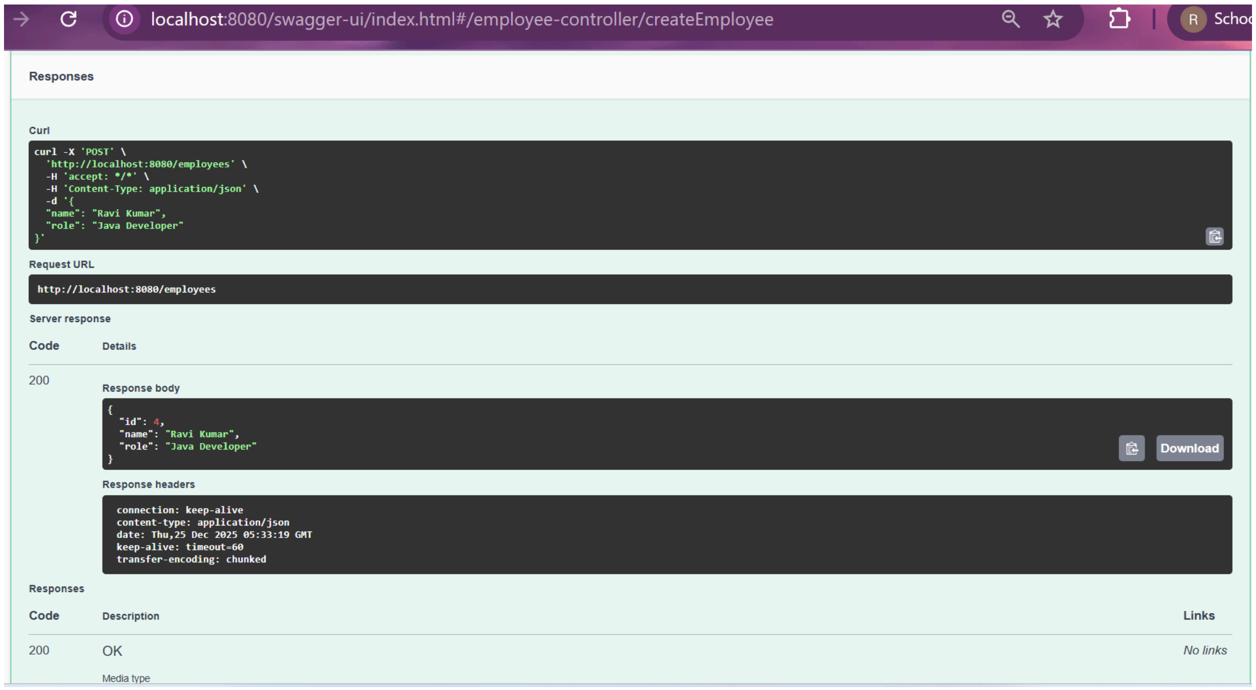
Figure 3: GET /employees API response showing the list of all employees retrieved from the MySQL database in JSON format.

- POST employee API response



```
{
  "name": "Ravi Kumar",
  "role": "Java Developer"
}
```

```
curl -X 'POST' \
```


localhost:8080/swagger-ui/index.html#/employee-controller/createEmployee
Responses
Curl

```
curl -X 'POST' \
  'http://localhost:8080/employees' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "name": "Ravi Kumar",
  "role": "Java Developer"
}'
```

Request URL

```
http://localhost:8080/employees
```

Server response

Code	Details
200	Response body

```
{
  "id": 4,
  "name": "Ravi Kumar",
  "role": "Java Developer"
}
```

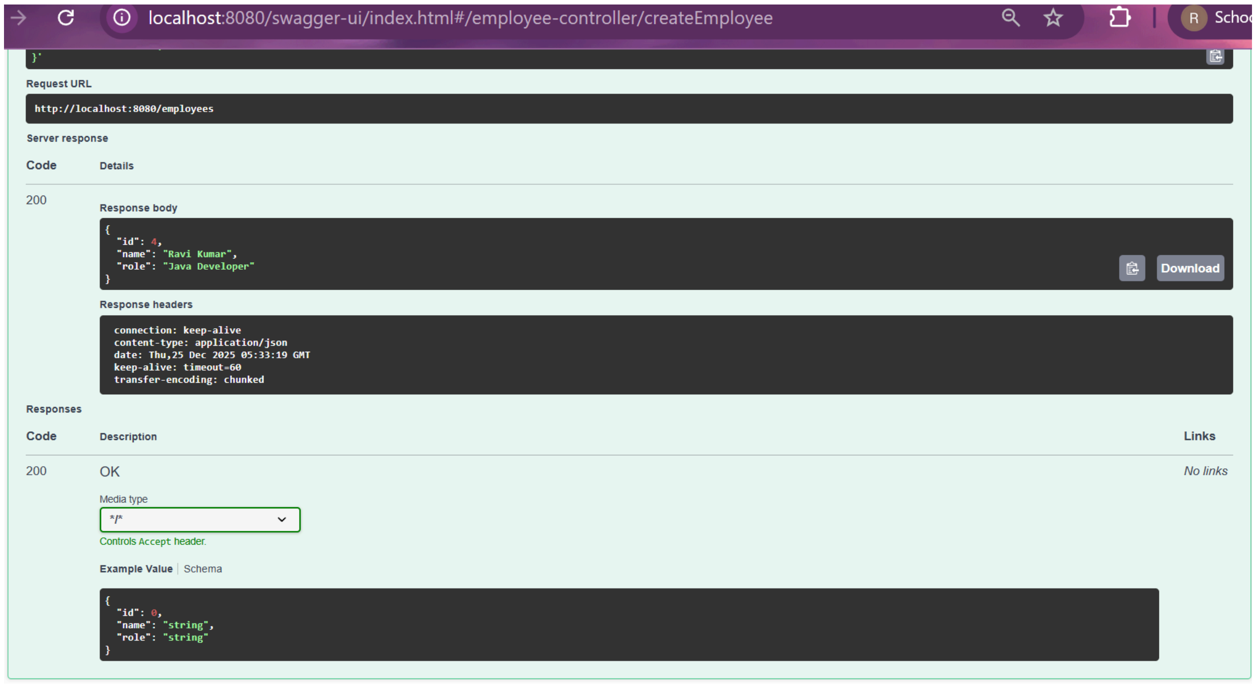
Response headers

```
 connection: keep-alive
 content-type: application/json
 date: Thu,25 Dec 2025 05:33:19 GMT
 keep-alive: timeout=60
 transfer-encoding: chunked
```

Responses

Code	Description	Links
200	OK	*No links*

Media type


localhost:8080/swagger-ui/index.html#/employee-controller/createEmployee
Download

}'

Request URL

```
http://localhost:8080/employees
```

Server response

Code | Details

200

Response body

```
{
  "id": 4,
  "name": "Ravi Kumar",
  "role": "Java Developer"
}
```

Response headers

```
 connection: keep-alive
 content-type: application/json
 date: Thu,25 Dec 2025 05:33:19 GMT
 keep-alive: timeout=60
 transfer-encoding: chunked
```

Responses

Code	Description	Links
200	OK	*No links*

Media type: */*

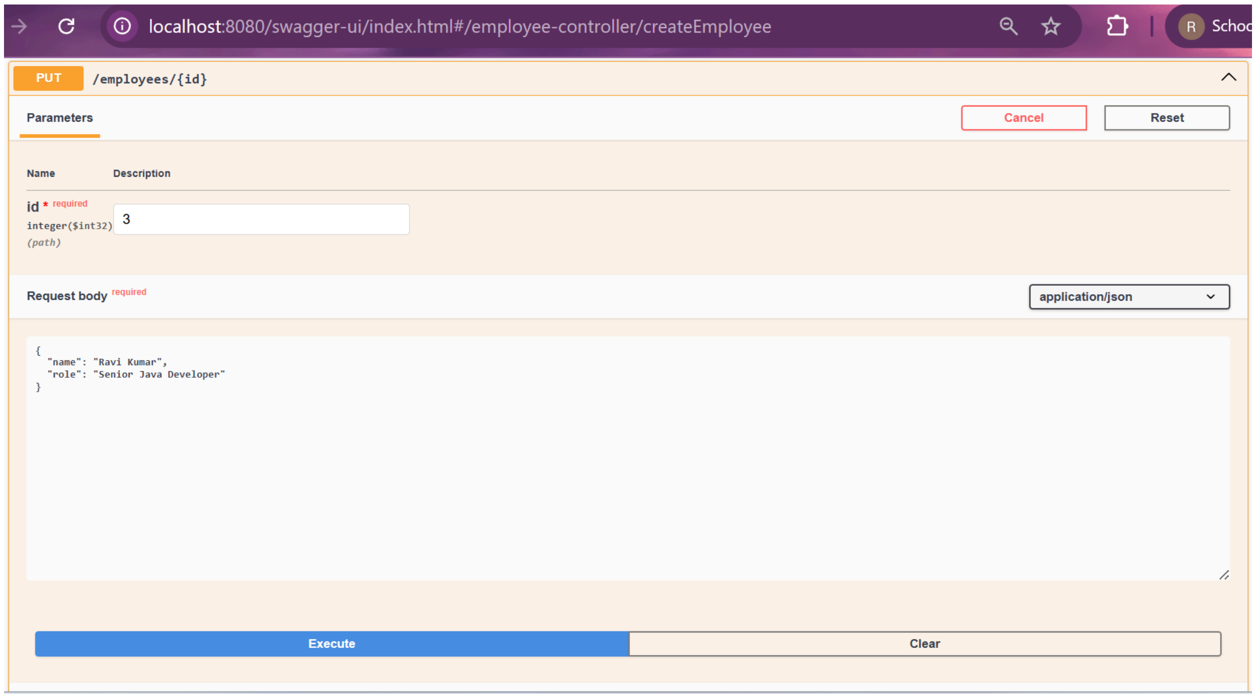
Controls Accept header.

Example Value | Schema

```
{
  "id": 0,
  "name": "string",
  "role": "string"
}
```

Figure 4: POST /employees API request and response showing successful creation of a new employee record.

- PUT employee API response


localhost:8080/swagger-ui/index.html#/employee-controller/createEmployee
PUT
Cancel
Reset
Execute
Clear

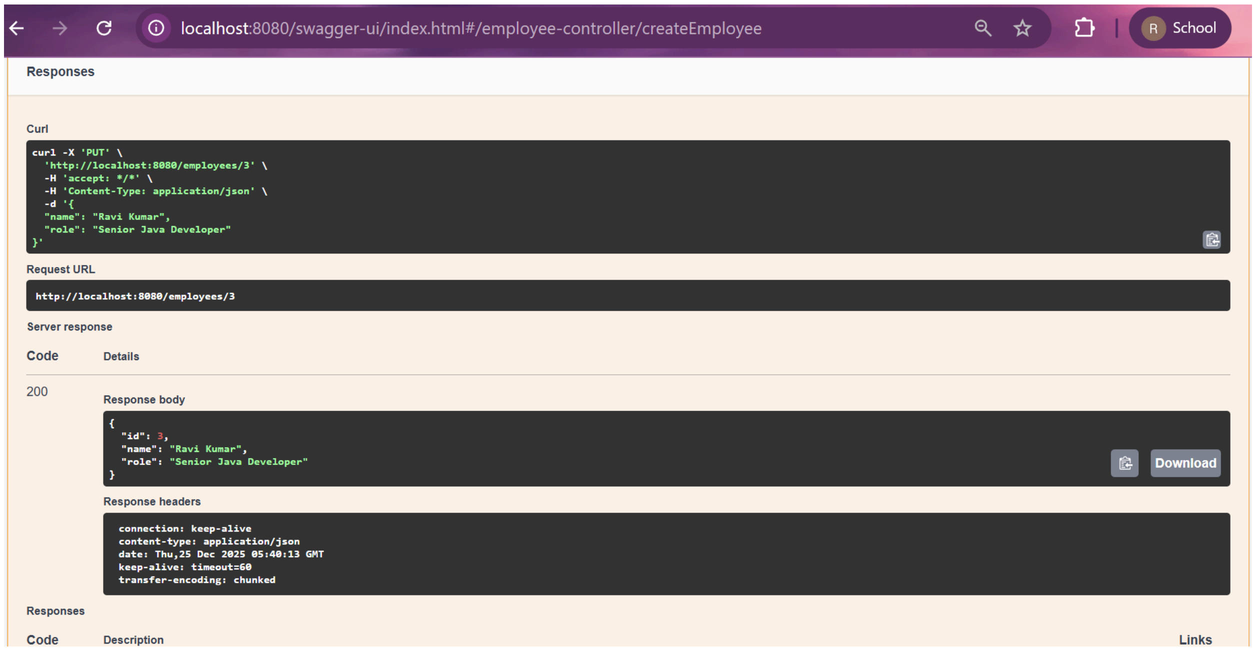
/employees/{id}

Parameters

Name	Description
id * required integer($int32) *(path)*	3

Request body required — application/json

```
{
  "name": "Ravi Kumar",
  "role": "Senior Java Developer"
}
```


localhost:8080/swagger-ui/index.html#/employee-controller/createEmployee

Responses

Curl

```
curl -X 'PUT' \
  'http://localhost:8080/employees/3' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "name": "Ravi Kumar",
  "role": "Senior Java Developer"
}'
```

Request URL

```
http://localhost:8080/employees/3
```

Server response

Code Details

200

Response body

```
{
  "id": 3,
  "name": "Ravi Kumar",
  "role": "Senior Java Developer"
}
```

Response headers

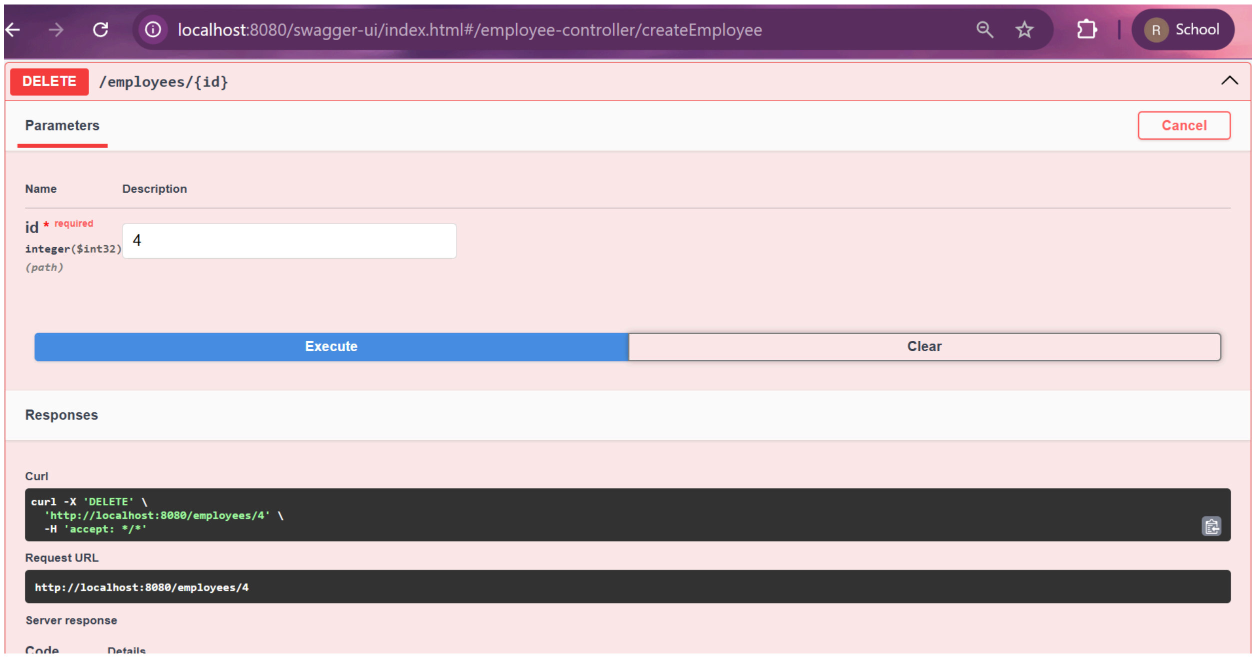
```
 connection: keep-alive
 content-type: application/json
 date: Thu,25 Dec 2025 05:40:13 GMT
 keep-alive: timeout=60
 transfer-encoding: chunked
```

Responses

Code Description Links

Figure 5: PUT /employees/{id} API request and response showing successful update of employee details.

- DELETE employee API response


localhost:8080/swagger-ui/index.html#/employee-controller/createEmployee

DELETE /employees/{id}

Parameters

Name Description

id * required integer($int32) (path): 4

Responses

Curl

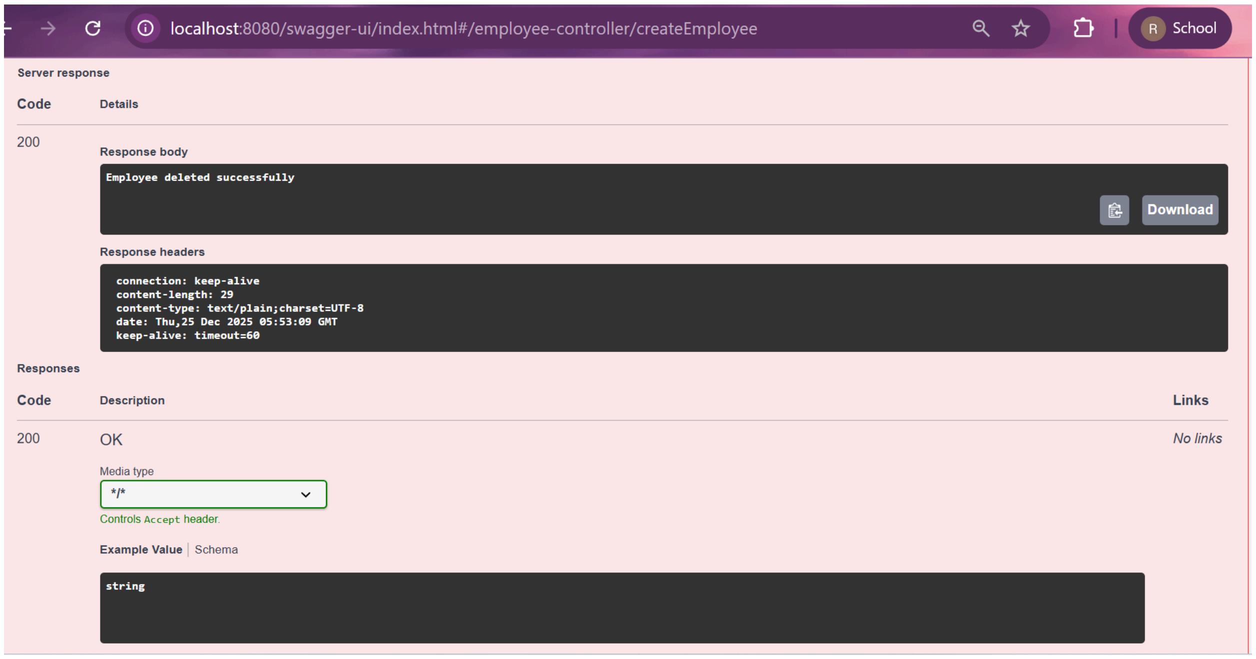
```
curl -X 'DELETE' \
  'http://localhost:8080/employees/4' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:8080/employees/4
```

Server response

Code Details


localhost:8080/swagger-ui/index.html#/employee-controller/createEmployee
School

Server response

Code Details

200

Response body

```
Employee deleted successfully
```

Response headers

```
 connection: keep-alive
 content-length: 29
 content-type: text/plain;charset=UTF-8
 date: Thu,25 Dec 2025 05:53:09 GMT
 keep-alive: timeout=60
```

Responses

Code	Description	Links
200	OK	*No links*

Media type

/

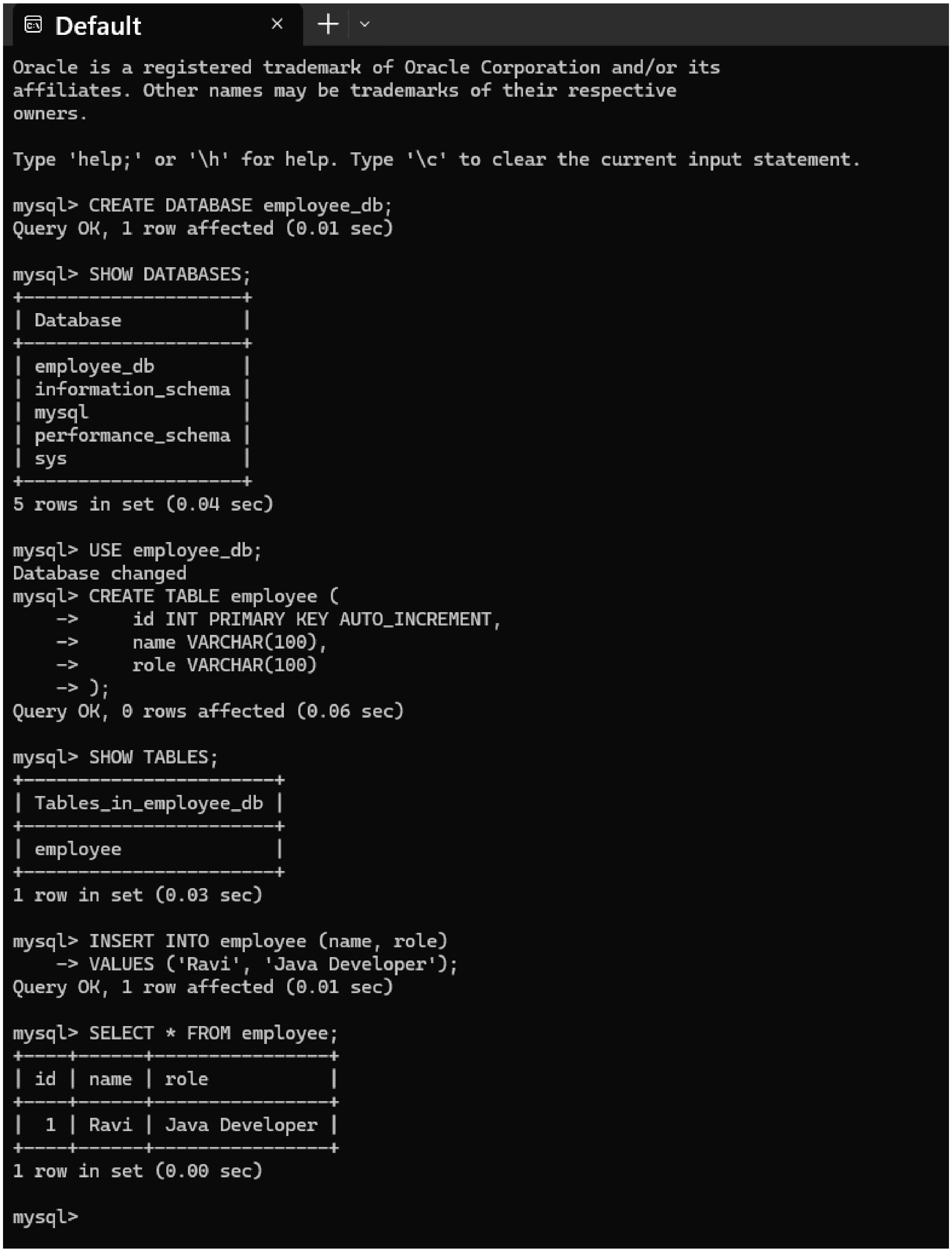
Controls `Accept` header.

Example Value | Schema

```
string
```

Figure 6: DELETE /employees/{id} API execution showing successful removal of an employee record from the MySQL database.

- MySQL employee table data



```
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE employee_db;
Query OK, 1 row affected (0.01 sec)

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| employee_db        |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.04 sec)

mysql> USE employee_db;
Database changed
mysql> CREATE TABLE employee (
    ->     id INT PRIMARY KEY AUTO_INCREMENT,
    ->     name VARCHAR(100),
    ->     role VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.06 sec)

mysql> SHOW TABLES;
+-----------------------+
| Tables_in_employee_db |
+-----------------------+
| employee              |
+-----------------------+
1 row in set (0.03 sec)

mysql> INSERT INTO employee (name, role)
    -> VALUES ('Ravi', 'Java Developer');
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM employee;
+----+------+----------------+
| id | name | role           |
+----+------+----------------+
|  1 | Ravi | Java Developer |
+----+------+----------------+
1 row in set (0.00 sec)

mysql>
```

Figure 7: MySQL database setup showing employee_db, creation of employee table, and inserted employee records used by the Spring Boot application.

----

7. CODE SNIPPETS

- EmployeeController.java

Java
```java
package com.example.employeeapi.controller;

import com.example.employeeapi.entity.Employee;
import com.example.employeeapi.service.EmployeeService;
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 EmployeeService employeeService;

    //  GET ALL
    @GetMapping
    public List<Employee> getAllEmployees() {
        return employeeService.getAllEmployees();
    }

    //  POST
    @PostMapping
    public Employee createEmployee(@RequestBody Employee employee) {
        return employeeService.createEmployee(employee);
    }

    //  PUT (THIS FIXES YOUR ERROR)
    @PutMapping("/{id}")
    public Employee updateEmployee(
            @PathVariable int id,
            @RequestBody Employee employee) {
        return employeeService.updateEmployee(id, employee);
    }

    //  DELETE
    @DeleteMapping("/{id}")
    public String deleteEmployee(@PathVariable int id) {
        employeeService.deleteEmployee(id);
        return "Employee deleted successfully";
    }
}
```

-EmployeeService.java

Java
```java
package com.example.employeeapi.service;

import com.example.employeeapi.entity.Employee;
import java.util.List;

public interface EmployeeService {
    List<Employee> getAllEmployees();

    Employee createEmployee(Employee employee);

    Employee updateEmployee(int id, Employee employee);

    void deleteEmployee(int id);
}
```

-EmployeeServiceImpl.java

Java
```
package com.example.employeeapi.service;

import com.example.employeeapi.entity.Employee;
import com.example.employeeapi.repository.EmployeeRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class EmployeeServiceImpl implements EmployeeService {

    @Autowired
    private EmployeeRepository employeeRepository;

    //  GET ALL
    @Override
```

```java
    public List<Employee> getAllEmployees() {
        return employeeRepository.findAll();
    }

    //  CREATE
    @Override
    public Employee createEmployee(Employee employee) {
        return employeeRepository.save(employee);
    }

    //  UPDATE
    @Override
    public Employee updateEmployee(int id, Employee employee) {
        Employee existingEmployee = employeeRepository.findById(id)
                .orElseThrow(() -> new RuntimeException("Employee
not found"));

        existingEmployee.setName(employee.getName());
        existingEmployee.setRole(employee.getRole());

        return employeeRepository.save(existingEmployee);
    }

    //  DELETE
    @Override
    public void deleteEmployee(int id) {
        employeeRepository.deleteById(id);
    }
}
```

-Employee.java

Java
```java
package com.example.employeeapi.entity;

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

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

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

    private String name;
    private String role;

    public Employee() {
    }

    public Employee(Long id, String name, String role) {
        this.id = id;
        this.name = name;
        this.role = role;
    }

    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 getRole() {
        return role;
    }

    public void setRole(String role) {
        this.role = role;
    }
}
```

—--

8.COMMANDS USED

The following commands were used to build and run the Spring Boot application:

1.

```shell
 mvn clean install
```

This command downloads dependencies and builds the project.

2.

```shell
mvn spring-boot:run
```

This command starts the Spring Boot application on the embedded Tomcat server.

3. http://localhost:8080/employees
   Used to test GET API in browser.

4. http://localhost:8080/swagger-ui/index.html
   Used to access Swagger UI for API testing.

—---

9. PROJECT LINKS

GitHub Repository:
https://github.com/Revathi2006-op/Alfido-Tech-Internship

Swagger UI URL:
http://localhost:8080/swagger-ui/index.html

----

10. CONCLUSION

This project helped me gain hands-on experience in developing RESTful APIs using Spring Boot,
integrating MySQL database using Spring Data JPA, and documenting APIs using Swagger UI.
It improved my understanding of backend development and layered application architecture.

Submitted by:
Name: REVATHI S