

28-02-2025

## What is Swagger API?

**Swagger** is an **open-source framework** used for **designing, documenting, and consuming** RESTful APIs. It helps developers to **describe, produce, and visualize APIs** in a structured way.

## Why Use Swagger?

- **API Documentation** – Generates interactive documentation.
  - **Testing & Debugging** – Allows developers to test APIs without external tools like Postman.
  - **Client Code Generation** – Generates API client libraries for different languages.
  - **Standardization** – Uses **OpenAPI Specification (OAS)** for consistency.
- 

## Swagger Components

Swagger consists of the following main tools:

1. **Swagger UI** – Generates a web-based, interactive API documentation.
2. **Swagger Editor** – Online editor for designing API specifications.
3. **Swagger Codegen** – Auto-generates client SDKs and server stubs.
4. **Swagger Inspector** – Used for testing APIs.

This is the dependency we need to add in the pom.xml

```
67
68     <dependency>
69         <groupId>org.springdoc</groupId>
70         <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
71         <version>2.8.5</version>
72     </dependency>
--
```

This is the dependency code we need to add...rest of the code is same

```
1 package com.wipro;
2
3+ import org.springframework.boot.SpringApplication;
4
5
6 @SpringBootApplication
7 public class SpringdatajpademoApplication {
8-     public static void main(String[] args) {
9         SpringApplication.run(SpringdatajpademoApplication.class, args);
10    }
11 }
12 |
```

```

1 package com.wipro.controller;
2
3 import java.util.List;
4
23
24 @RestController
25 @RequestMapping("/employee")
26 public class EmployeeController {
27
28     @Autowired
29     private EmployeeService service;
30
31     // Create Employee
32     //create Employee
33     @PostMapping
34     public ResponseEntity<EmployeeDTO> createEmployee(@Valid @RequestBody EmployeeDTO dto) {
35         return new ResponseEntity<EmployeeDTO>(service.addEmployee(dto), HttpStatus.CREATED);
36     }
37
38
39     // Get All Employees
40     @GetMapping
41     public List<Employee> getAllEmployees() {
42         return service.getEmployees();
43     }
44
45     // Get Employee by ID
46     @GetMapping("/{id}")
47     public Employee getEmployeeById(@PathVariable Long id) throws ResourceNotFoundException {
48         return service.getEmployeeById(id);
49     }
50
51     // Update Employee
52     @PutMapping("/{id}")
53     public Employee updateEmployee(@PathVariable Long id, @RequestBody Employee updatedEmployee) {
54         return service.updateEmployee(id, updatedEmployee);
55     }
56
57     // Delete Employee
58     @DeleteMapping("/{id}")
59     public String deleteEmployee(@PathVariable Long id) {
60         return service.deleteEmployeeById(id);
61     }
62
63     // Get Employees by Name
64     @GetMapping("/name/{name}")
65     public List<Employee> getEmployeesByName(@PathVariable String name) {
66         return service.getEmployeesByName(name);
67     }
68
69     // Get Employees by Department
70     @GetMapping("/department/{department}")
71     public List<Employee> getEmployeesByDepartment(@PathVariable String department) {
72         return service.getEmployeesByDepartment(department);
73     }
74 }
75

```

Springdatajp...

EmployeeCon...

Employee.java

```
1 package com.wipro.dto;
2
3 public class EmployeeDTO
4 {
5     private Long id;
6     private String name;
7     private String department;
8     public Long getId() {
9         return id;
10    }
11    public void setId(Long id) {
12        this.id = id;
13    }
14    public String getName() {
15        return name;
16    }
17    public void setName(String name) {
18        this.name = name;
19    }
20    public String getDepartment() {
21        return department;
22    }
23    public void setDepartment(String department) {
24        this.department = department;
25    }
26    public EmployeeDTO(Long id, String name, String department) {
27        super();
28        this.id = id;
29        this.name = name;
30        this.department = department;
31    }
32
33    public EmployeeDTO()
34    {
35
36    }
37
38
39
40 }
```

```
1 package com.wipro.entity;
2
3+ import jakarta.persistence.*;
4
5
6 @Entity
7 public class Employee {
8
9-     @Id
10     @GeneratedValue(strategy = GenerationType.AUTO)
11     private Long id;
12
13-     @NotBlank(message = "Name is required")
14     @Size(min = 2, max = 50, message = "Name must be between 2 and 50 characters")
15     private String name;
16
17-     @NotBlank(message = "Department Name is required")
18     private String department;
19
20     // Default constructor
21     public Employee() {}
22
23     // Parameterized constructor
24-     public Employee(String name, String department) {
25         this.name = name;
26         this.department = department;
27     }
28
29     // Getters and Setters
30-     public Long getId() {
31         return id;
32     }
33
34-     public void setId(Long id) {
35         this.id = id;
36     }
37
38-     public String getName() {
39         return name;
40     }
41
42-     public void setName(String name) {
43         this.name = name;
44     }
45
46-     public String getDepartment() {
47         return department;
48     }
49
50-     public void setDepartment(String department) {
51         this.department = department;
52     }
53 }
54
```

```

1 package com.wipro.exception;
2
3 import java.util.ArrayList;
4
13 @RestControllerAdvice
14 public class GlobalExceptionHandler {
15
16     @ExceptionHandler(MethodArgumentNotValidException.class)
17     public ResponseEntity<Map<String, Object>> handleValidationException(MethodArgumentNotValidException ex) {
18         Map<String, Object> response = new HashMap<>();
19         response.put("status", HttpStatus.BAD_REQUEST.value());
20         response.put("error", "Validation Failed");
21
22         List<Map<String, String>> errorList = new ArrayList<>();
23         ex.getBindingResult().getFieldErrors().forEach(error -> {
24             Map<String, String> errorMap = new HashMap<>();
25             errorMap.put("field", error.getField());
26             errorMap.put("message", error.getDefaultMessage());
27             errorList.add(errorMap);
28         });
29
30         response.put("errors", errorList);
31         return new ResponseEntity<>(response, HttpStatus.BAD_REQUEST);
32     }
33
34     @ExceptionHandler(ResourceNotFoundException.class)
35     public ResponseEntity<Map<String, Object>> handleResourceNotFound(ResourceNotFoundException ex) {
36         Map<String, Object> response = new HashMap<>();
37         response.put("status", HttpStatus.NOT_FOUND.value());
38         response.put("error", "Resource not found");
39         response.put("message", ex.getMessage());
40         return new ResponseEntity<>(response, HttpStatus.NOT_FOUND);
41     }
42 }
43
44

```

```

1 package com.wipro.exception;
2
3 public class NameNotFoundException extends Exception {
4
5     public NameNotFoundException(String message) {
6         super(message); //now define the name not found exception in the service class
7     }
8 }
9
10
11
12

```

```
Springdatajp... EmployeeCon... Employee.java
1 package com.wipro.exception;
2
3 public class ResourceNotFoundException extends Exception {
4
5     public ResourceNotFoundException(String message) {
6         super(message);
7     }
8 }
9
```

```
Employee.java GlobalExcep... ResourceNot... NotFoundException
1 package com.wipro.repository;
2
3 import java.util.List;
4
5 @Repository
6 public interface EmployeeRepository extends JpaRepository<Employee, Long> {
7     List<Employee> findByName(String name);
8     List<Employee> findByDepartment(String department);
9 }
10
11
12
13
```

```

Employee.java  GlobalExcep...  ResourceNotF...  EmployeeServ...  NameNotFound...  Empic
1  package com.wipro.service;
2
3  import java.util.List;
4
5  import org.springframework.beans.factory.annotation.Autowired;
6  import org.springframework.stereotype.Service;
7
8  import com.wipro.dto.EmployeeDTO;
9  import com.wipro.entity.Employee;
10 import com.wipro.exception.ResourceNotFoundException;
11 import com.wipro.repository.EmployeeRepository;
12
13 @Service
14 public class EmployeeService {
15
16     @Autowired
17     private EmployeeRepository repository;
18
19     // Add Employee
20     public EmployeeDTO addEmployee(EmployeeDTO dto) {
21         //convert dto to entity
22         Employee employee=new Employee(dto.getName(),dto.getDepartment());
23         Employee savedEmployee=repository.save(employee);//entity class
24
25         //convert entity to dto object again
26
27         EmployeeDTO employeeDTO=new EmployeeDTO(savedEmployee.getId(),savedEmployee.getName(),savedEmployee.getDepartment());
28         return employeeDTO;
29     }
30
31
32     // Get All Employees
33     public List<Employee> getEmployees() {
34         return repository.findAll();
35     }
36
37     // Get Employee by ID
38     public Employee getEmployeeById(Long id) throws ResourceNotFoundException {
39         return repository.findById(id)
40             .orElseThrow(() -> new ResourceNotFoundException("Employee with given id " + id + " is not present"));
41     }
42
43     // Update Employee
44     public Employee updateEmployee(Long id, Employee updatedEmployee) {
45         return repository.findById(id).map(employee -> {
46             employee.setName(updatedEmployee.getName());
47             employee.setDepartment(updatedEmployee.getDepartment());
48             return repository.save(employee);
49         }).orElse(null);
50     }
51
52     // Delete Employee
53     public String deleteEmployeeById(Long id) {
54         if (repository.existsById(id)) {
55             repository.deleteById(id);
56             return "Employee with ID " + id + " deleted successfully.";
57         } else {
58             return "Employee not found.";
59         }
60     }
61 }

```



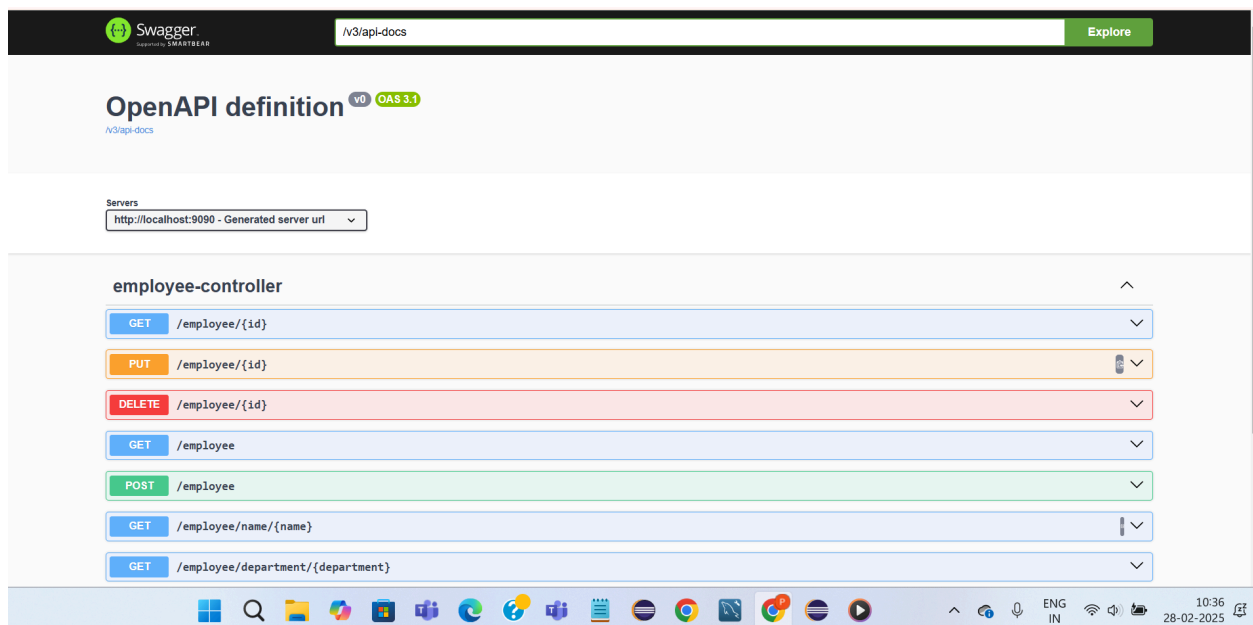
```

51
52 // Delete Employee
53 public String deleteEmployeeById(Long id) {
54     if (repository.existsById(id)) {
55         repository.deleteById(id);
56         return "Employee with ID " + id + " deleted successfully.";
57     } else {
58         return "Employee not found.";
59     }
60 }
61
62 // Get Employees by Name
63 public List<Employee> getEmployeesByName(String name) {
64     return repository.findByName(name);
65 }
66
67 // Get Employees by Department
68 public List<Employee> getEmployeesByDepartment(String department) {
69     return repository.findByDepartment(department);
70 }
71 }
72

```

Run the application and check by this code

<http://localhost:9090/swagger-ui/index.html>



Now we r going to add the annotations to my code and check what changes we r going to get

@Tag(name = "Employee Controller", description = "Employee Management APIs")

**ADDS A GROUP NAME TO THE SWAGGER**

Add this one into the employeecontroller class

```
@RequestMapping("/employee/")
@Tag(name = "Employee Controller", description = "Operation related to employees")
public class EmployeeController {

    @Autowired
    private EmployeeService service;
```

Now check whether the changes are going to be added into the swagger or not

Servers

http://localhost:9090 - Generated server url

## Employee Controller

Operation related to employees

```
@GetMapping
@Operation(summary = "Get All Employees", description = "Retrieves a list of all employees")
public List<Employee> getAllEmployees() {
    return service.getEmployees();
}

// Get Employee by ID
```

Operation: **Describes each API end point**

Check now

## Employee Controller Operation related to employees

**GET** /employee/{id}

**PUT** /employee/{id}

**DELETE** /employee/{id}

**GET** /employee Get All Employees

**POST** /employee

**GET** /employee/name/{name}

**GET** /employee/department/{department}

### 1. API Documentation Annotations

Annotation	Purpose
@OpenAPIDefinition	Defines global OpenAPI metadata, including servers, security, and tags.
@Info	Provides metadata about the API, such as title, version, and description.
@Tag	Groups related endpoints together in Swagger UI.
@Server	Specifies the server URLs where the API is hosted.

EmployeeCon... × Employee.java GlobalExcep... ResourceNotF... EmployeeServ... NameNotFound

```
3 import java.util.List;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.http.HttpStatus;
7 import org.springframework.http.ResponseEntity;
8 import org.springframework.web.bind.annotation.DeleteMapping;
9 import org.springframework.web.bind.annotation.GetMapping;
10 import org.springframework.web.bind.annotation.PathVariable;
11 import org.springframework.web.bind.annotation.PostMapping;
12 import org.springframework.web.bind.annotation.PutMapping;
13 import org.springframework.web.bind.annotation.RequestBody;
14 import org.springframework.web.bind.annotation.RequestMapping;
15 import org.springframework.web.bind.annotation.RestController;
16
17 import com.wipro.dto.EmployeeDTO;
18 import com.wipro.entity.Employee;
19 import com.wipro.exception.ResourceNotFoundException;
20 import com.wipro.service.EmployeeService;
21
22 import io.swagger.v3.oas.annotations.Operation;
23 import io.swagger.v3.oas.annotations.Parameter;
24 import io.swagger.v3.oas.annotations.responses.ApiResponse;
25 import io.swagger.v3.oas.annotations.tags.Tag;
26 import jakarta.validation.Valid;
27
28 @RestController
29 @RequestMapping("/employee")
30 @Tag(name = "Employee Controller", description = "Operations related to employees")
31 public class EmployeeController {
32
33     @Autowired
34     private EmployeeService service;
35
36     // Create Employee
37     @PostMapping
38     @Operation(summary = "Create Employee", description = "Adds a new employee to the system")
39     @ApiResponse(responseCode = "201", description = "Employee created successfully")
40     @ApiResponse(responseCode = "400", description = "Invalid input data")
41     public ResponseEntity<EmployeeDTO> createEmployee(@Valid @RequestBody EmployeeDTO dto) {
42         return new ResponseEntity<>(service.addEmployee(dto), HttpStatus.CREATED);
43     }
44
45     // Get All Employees
46     @GetMapping
47     @Operation(summary = "Get All Employees", description = "Retrieves a list of all employees")
48     @ApiResponse(responseCode = "200", description = "List of employees retrieved successfully")
49     public List<Employee> getAllEmployees() {
50         return service.getEmployees();
51     }
52
53     // Get Employee by ID
54     @GetMapping("/{id}")
55     @Operation(summary = "Get Employee by ID", description = "Retrieves an employee by their ID")
56     @ApiResponse(responseCode = "200", description = "Employee found")
57     @ApiResponse(responseCode = "404", description = "Employee not found")
58     public Employee getEmployeeById(
59         @Parameter(description = "Employee ID", example = "101") @PathVariable Long id) throws ResourceNotFoundException {
60         return service.getEmployeeById(id);
61     }
62 }
```

```

50 public Employee getEmployeeById(
51     @Parameter(description = "Employee ID", example = "101") @PathVariable Long id) throws ResourceNotFoundException {
52     return service.getEmployeeById(id);
53 }
54
55 // Update Employee
56 @PutMapping("/{id}")
57 @Operation(summary = "Update Employee", description = "Updates an existing employee's details")
58 @ApiResponse(responseCode = "200", description = "Employee updated successfully")
59 @ApiResponse(responseCode = "404", description = "Employee not found")
60 public Employee updateEmployee(
61     @Parameter(description = "Employee ID", example = "101") @PathVariable Long id,
62     @RequestBody Employee updatedEmployee) {
63     return service.updateEmployee(id, updatedEmployee);
64 }
65
66 // Delete Employee
67 @DeleteMapping("/{id}")
68 @Operation(summary = "Delete Employee", description = "Deletes an employee by their ID")
69 @ApiResponse(responseCode = "200", description = "Employee deleted successfully")
70 @ApiResponse(responseCode = "404", description = "Employee not found")
71 public String deleteEmployee(
72     @Parameter(description = "Employee ID", example = "101") @PathVariable Long id) {
73     return service.deleteEmployeeById(id);
74 }
75
76 // Get Employees by Name
77 @GetMapping("/name/{name}")
78 @Operation(summary = "Get Employees by Name", description = "Retrieves employees by their name")
79 @ApiResponse(responseCode = "200", description = "Employees retrieved successfully")
80 @ApiResponse(responseCode = "404", description = "No employees found with the given name")
81 public List<Employee> getEmployeesByName(
82     @Parameter(description = "Employee Name", example = "John Doe") @PathVariable String name) {
83     return service.getEmployeesByName(name);
84 }
85
86 // Get Employees by Department
87 @GetMapping("/department/{department}")
88 @Operation(summary = "Get Employees by Department", description = "Retrieves employees by department")
89 @ApiResponse(responseCode = "200", description = "Employees retrieved successfully")
90 @ApiResponse(responseCode = "404", description = "No employees found in the given department")
91 public List<Employee> getEmployeesByDepartment(
92     @Parameter(description = "Department Name", example = "HR") @PathVariable String department) {
93     return service.getEmployeesByDepartment(department);
94 }
95 }
96

```

## OpenAPI definition v0 OAS 3.1

[V3/api-docs](#)

### Servers

<http://localhost:9090> - Generated server url

### Employee Controller Operations related to employees

GET	/employee/{id}	Get Employee by ID	⌵
PUT	/employee/{id}	Update Employee	⌵
DELETE	/employee/{id}	Delete Employee	⌵
GET	/employee	Get All Employees	⌵
POST	/employee	Create Employee	⌵
GET	/employee/name/{name}	Get Employees by Name	⌵
GET	/employee/department/{department}	Get Employees by Department	⌵

Checking whether responseapi is working or not

Name	Description
<b>id</b> * required integer(\$int64) (path)	Employee ID

Execute

Clear

### Responses

Curl

```
curl -X 'GET' \
  'http://localhost:9090/employee/101' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:9090/employee/101
```

Server response

Code	Details
404	Error: response status is 404

Response body

```
{
  "error": "Resource not found",
  "message": "Employee with given id 101 is not present",
  "status": 404
}
```

Response headers

```
connection: keep-alive
content-type: application/json
date: Fri, 28 Feb 2025 05:44:39 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Name	Description
<b>id</b> * required integer(\$int64) (path)	Employee ID

Execute

Clear

### Responses

Curl

```
curl -X 'GET' \
  'http://localhost:9090/employee/52' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:9090/employee/52
```

Server response

Code	Details
200	

Response body

```
{
  "id": 52,
  "name": "kalyan",
  "department": "IT"
}
```

Response headers

```
connection: keep-alive
content-type: application/json
date: Fri, 28 Feb 2025 05:46:24 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Now we will add few more annotations in the dto class

**@Schema(name = "EmployeeDTO", description = "Data Transfer Object for Employee")**

**@Schema(description = "Department of the employee", example = "IT")**

is used to enhance API documentation in Swagger. Let's break it down:

#### ♦ Purpose of @Schema

- It describes a field in a DTO or Entity class.
- It helps Swagger UI generate clear documentation.
- It provides an example value ("IT") for reference.

```
EmployeeCon... ResourceNotr... Employeeserv... NameNotFound... Employ
1 package com.wipro.dto;
2
3 import io.swagger.v3.oas.annotations.media.Schema;
4
5 @Schema(name = "EmployeeDTO", description = "Data Transfer Object for Employee")
6
7 public class EmployeeDTO
8 {
9     @Schema(description = "Unique identifier of the employee", example = "101")
10
11     private Long id;
12     @Schema(description = "Name of the employee", example = "Pavan")
13
14     private String name;
15     @Schema(description = "Department of the employee", example = "IT")
16
17     private String department;
18     public Long getId() {
19         return id;
20     }
21     public void setId(Long id) {
22         this.id = id;
23     }
24     public String getName() {
25         return name;
26     }
27 }
```

**POST** /employee Create Employee

Adds a new employee to the system

**Parameters** Try it out

No parameters

**Request body** required application/json

Example Value | Schema

```
{
  "id": 101,
  "name": "Pavan",
  "department": "IT"
}
```

**Responses**

Code	Description	Links
201	Employee created successfully	No links

Media type: \*/

Controls Accept header.

Example Value | Schema

## Micro Services

Microservices, or **microservices architecture**, is a software development approach where an application is built as a collection of small, independent services that communicate with each other. Each microservice is responsible for a specific business functionality and runs independently, making the application more scalable, flexible, and maintainable.

### Key Features of Microservices:

1. **Independent Deployment:** Each service can be developed, deployed, and updated separately without affecting the entire system.
2. **Single Responsibility:** Each microservice is responsible for a single function or feature.
3. **Scalability:** Services can be scaled independently based on demand.
4. **Technology Diversity:** Each microservice can be developed using different programming languages, frameworks, or databases.
5. **Fault Isolation:** If one service fails, it does not necessarily bring down the entire application.
6. **Lightweight Communication:** Microservices communicate using APIs (usually RESTful APIs or messaging queues like Kafka).

### How Microservices Work?

- Each microservice has its **own database** and business logic.
- They communicate with each other through **HTTP (REST API), gRPC, or message brokers**.



- Microservices are usually deployed in **containers** (e.g., Docker) and managed using **orchestration tools** like Kubernetes.

## Example of Microservices in Action:

A **shopping website** may have the following microservices:

1. **User Service** – Manages user authentication and profiles.
2. **Product Service** – Handles product details and inventory.
3. **Order Service** – Manages orders and payments.
4. **Cart Service** – Handles shopping cart functionality.
5. **Notification Service** – Sends emails and messages.

## Benefits of Microservices:

- ✓ **Faster Development:** Teams can work on different services simultaneously.
- ✓ **Easy Maintenance:** Each service is small and manageable.
- ✓ **Better Performance:** Services can be optimized and scaled individually.
- ✓ **Resilience:** A failure in one service doesn't crash the entire system.

## Challenges of Microservices:

- ⚠ **Complexity:** Requires handling multiple services and inter-service communication.
- ⚠ **Data Management:** Each service has its own database, requiring careful synchronization.
- ⚠ **Security:** More APIs mean more potential security vulnerabilities.
- ⚠ **Deployment Overhead:** More services require better orchestration and monitoring tools.

## Popular Tools for Microservices:

- **Frameworks:** Spring Boot, Micronaut, Quarkus, Node.js
- **API Communication:** REST, gRPC, GraphQL
- **Databases:** MySQL, PostgreSQL, MongoDB, Redis
- **Containerization:** Docker, Kubernetes
- **Monitoring:** Prometheus, Grafana, ELK Stack

Micro Services

SpringbootDepartmentApplication.java

```
1 package com.wipro;
2
3 import org.modelmapper.ModelMapper;
4
5 @SpringBootApplication
6 public class SpringbootDepartmentApplication {
7
8     public static void main(String[] args) {
9         SpringApplication.run(SpringbootDepartmentApplication.class, args);
10    }
11
12    @Bean
13    public ModelMapper getModelMapper() {
14        return new ModelMapper();
15    }
16 }
17
18
```

Department.java

package com.wipro.controller;

```
1
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
5 @RestController
6 @RequestMapping("/departments")
7 public class DepartmentController {
8
9     @Autowired
10     DepartmentService service;
11
12     @PostMapping
13     public DepartmentDto saveDepartment(@RequestBody DepartmentDto deptdto) {
14         return service.saveDepartment(deptdto);
15     }
16
17     @GetMapping("/{code}")
18     public DepartmentDto getDepartmentByCode(@PathVariable String code) {
19         return service.getDepartmentByCode(code);
20     }
21 }
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
```

```

1 package com.wipro.dto;
2
3 import lombok.AllArgsConstructor;
4
5
6
7 @AllArgsConstructor
8 @NoArgsConstructor
9 @Data
10 public class DepartmentDto {
11
12     private Long deptId;
13     private String department;
14     public DepartmentDto(Long deptId, String department, String description, String deptcode) {
15         super();
16         this.deptId = deptId;
17         this.department = department;
18         this.description = description;
19         this.deptcode = deptcode;
20     }
21     public Long getDeptId() {
22         return deptId;
23     }
24     public void setDeptId(Long deptId) {
25         this.deptId = deptId;
26     }
27     public String getDepartment() {
28         return department;
29     }
30     public void setDepartment(String department) {
31         this.department = department;
32     }
33     public DepartmentDto() {
34         super();
35     }
36     public String getDescription() {
37         return description;
38     }
39     public void setDescription(String description) {
40         this.description = description;
41     }
42     public String getDeptcode() {
43         return deptcode;
44     }
45     public void setDeptcode(String deptcode) {
46         this.deptcode = deptcode;
47     }
48     private String description;
49     private String deptcode;
50
51 }
52

```

```

1 package com.wipro.entity;
2
3 import jakarta.persistence.Entity;
4
5
6
7
8
9
10
11
12 @Entity
13 @Data
14 @NoArgsConstructor
15 @AllArgsConstructor
16 @Table(name = "departments")
17 public class Department {
18
19     @Id
20     @GeneratedValue(strategy = GenerationType.AUTO)
21     private Long deptId;
22     private String department;
23     private String description;
24     private String deptcode;
25     public Long getDeptId() {
26         return deptId;
27     }
28     public void setDeptId(Long deptId) {
29         this.deptId = deptId;
30     }
31     public String getDepartment() {
32         return department;
33     }
34     public void setDepartment(String department) {
35         this.department = department;
36     }
37     public String getDescription() {
38         return description;
39     }
40     public void setDescription(String description) {
41         this.description = description;
42     }
43     public String getDeptcode() {
44         return deptcode;
45     }
46     public void setDeptcode(String deptcode) {
47         this.deptcode = deptcode;
48     }
49     public Department(Long deptId, String department, String description, String deptcode) {
50         super();
51         this.deptId = deptId;
52         this.department = department;
53         this.description = description;
54         this.deptcode = deptcode;
55     }
56     public Department() {
57         super();
58     }
59
60
61 }
62

```

```

1 package com.wipro.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4 @Repository
5 public interface DepartmentRepo extends JpaRepository<Department, Long> {
6
7     Department findByDeptcode(String deptcode); // Return Department entity, not DTO
8 }
9
10
11
12
13
14

```

```

1 package com.wipro.service;
2
3 import org.modelmapper.ModelMapper;
4
5 @Service
6 public class DepartmentService {
7
8     @Autowired
9     private DepartmentRepo repo;
10
11     @Autowired
12     private ModelMapper mapper;
13
14     public DepartmentDto saveDepartment(DepartmentDto deptdto) {
15         Department dept = mapper.map(deptdto, Department.class);
16         Department newdept = repo.save(dept);
17         return mapper.map(newdept, DepartmentDto.class);
18     }
19
20     public DepartmentDto getDepartmentByCode(String code) {
21         Department department = repo.findByDeptcode(code); // Fetch entity
22         return department != null ? mapper.map(department, DepartmentDto.class) : null;
23     }
24 }
25
26
27
28
29
30
31
32

```

```

1 spring.application.name=Springboot-Department
2 server.port=9090
3 # MySQL Configuration
4 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
5 spring.datasource.url=jdbc:mysql://localhost:3306/wipro
6 spring.datasource.username=root
7 spring.datasource.password=#Mahadev7
8
9 # JPA & Hibernate
10 spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
11 spring.jpa.hibernate.ddl-auto=update
12 spring.jpa.show-sql=true

```

```

https://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation)
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
4   <modelVersion>4.0.0</modelVersion>
5   <parent>
6     <groupId>org.springframework.boot</groupId>
7     <artifactId>spring-boot-starter-parent</artifactId>
8     <version>3.4.3</version>
9     <relativePath/> <!-- lookup parent from repository -->
10  </parent>
11  <groupId>com.wipro</groupId>
12  <artifactId>Springboot-Department</artifactId>
13  <version>0.0.1-SNAPSHOT</version>
14  <name>Springboot-Department</name>
15  <description>Demo project for Spring Boot microservice</description>
16  <url/>
17  <licenses>
18    <license/>
19  </licenses>
20  <developers>
21    <developer/>
22  </developers>
23  <scm>
24    <connection/>
25    <developerConnection/>
26    <tag/>
27    <url/>
28  </scm>
29  <properties>
30    <java.version>17</java.version>
31  </properties>
32  <dependencies> Add Spring Boot Starters...
33    <dependency>
34      <groupId>org.springframework.boot</groupId>
35      <artifactId>spring-boot-starter-data-jpa</artifactId>
36    </dependency>
37    <dependency>
38      <groupId>org.springframework.boot</groupId>
39      <artifactId>spring-boot-starter-web</artifactId>
40    </dependency>
41
42    <dependency>
43      <groupId>org.springframework.boot</groupId>
44      <artifactId>spring-boot-devtools</artifactId>
45      <scope>runtime</scope>
46      <optional>true</optional>
47    </dependency>
48    <dependency>
49      <groupId>org.modelmapper</groupId>
50      <artifactId>modelmapper</artifactId>
51      <version>3.2.0</version>
52    </dependency>
53
54    <dependency>
55      <groupId>com.mysql</groupId>
56      <artifactId>mysql-connector-j</artifactId>

```

```

52 </dependency>
53
54- <dependency>
55     <groupId>com.mysql</groupId>
56     <artifactId>mysql-connector-j</artifactId>
57     <scope>runtime</scope>
58 </dependency>
59- <dependency>
60     <groupId>org.projectlombok</groupId>
61     <artifactId>lombok</artifactId>
62     <optional>true</optional>
63 </dependency>
64- <dependency>
65     <groupId>org.springframework.boot</groupId>
66     <artifactId>spring-boot-starter-test</artifactId>
67     <scope>test</scope>
68 </dependency>
69 </dependencies>
70
71- <build>
72-     <plugins>
73-         <plugin>
74             <groupId>org.apache.maven.plugins</groupId>
75             <artifactId>maven-compiler-plugin</artifactId>
76             <configuration>
77                 <annotationProcessorPaths>
78                     <path>
79                         <groupId>org.projectlombok</groupId>
80                         <artifactId>lombok</artifactId>
81                     </path>
82                 </annotationProcessorPaths>
83             </configuration>
84         </plugin>
85-         <plugin>
86             <groupId>org.springframework.boot</groupId>
87             <artifactId>spring-boot-maven-plugin</artifactId>
88             <configuration>
89                 <excludes>
90                     <exclude>
91                         <groupId>org.projectlombok</groupId>
92                         <artifactId>lombok</artifactId>
93                     </exclude>
94                 </excludes>
95             </configuration>
96         </plugin>
97     </plugins>
98 </build>
99
100 </project>
101

```

POST



localhost:9090/departments

Send



Params

Auth

Headers (9)

Body



Scripts

Settings

Cookies

raw



JSON



Beautify

```
1  {
2
3
4    "department": "ece",
5    "description": "electronics and communication engineering",
6    "deptcode": "ece-101"
7  }
8  }
```

Body



200 OK

• 125 ms • 275 B •



{ } JSON



▶ Preview



Visualize



```
1  {
2    "deptId": 52,
3    "department": "ece",
4    "description": "electronics and communication
5                  engineering",
6    "deptcode": "ece-101"
7  }
```



POST

localhost:9090/departments

Send

ParamsAuthHeaders (9)BodyScriptsSettingsCookies

rawJSONBeautify

```
1  {
2
3
4    "department": "cse",
5    "description": "computer science and engineering",
6    "deptcode": "cse-101"
7  }
8
```

Body200 OK • 14 ms • 266 B

{ } JSON

PreviewVisualize

```
1  {
2    "deptId": 53,
3    "department": "cse",
4    "description": "computer science and engineering",
5    "deptcode": "cse-101"
6  }
```

Result Grid

Filter Rows:

Edit:Export

	dept_id	department	deptcode	description
▶	52	ece	ece-101	electronics and communication engineering
	53	cse	cse-101	computer science and engineering
✱	NULL	NULL	NULL	NULL

GET

localhost:9090/departments/cse-101

Send

ParamsAuthHeaders (9)Body●ScriptsSettingsCookies

rawJSONBeautify

```
1  {
2
3
4    "department": "cse",
5    "description": "computer science and engineering",
6    "deptcode": "cse-101"
7  }
8
```





























Body200 OK • 226 ms • 266 B

{ } JSON

PreviewVisualize

```
1  {
2    "deptId": 53,
3    "department": "cse",
4    "description": "computer science and engineering",
5    "deptcode": "cse-101"
6  }
```

ConsolePostbotRunnerVault

- ▼  Springboot-Department [boot] [devtools]
  - ▼  src/main/java
    - ▼  com.wipro
      - >  SpringbootDepartmentApplication.java
    - ▼  com.wipro.controller
      - >  DepartmentController.java
    - ▼  com.wipro.dto
      - >  DepartmentDto.java
    - ▼  com.wipro.entity
      - >  Department.java
    - ▼  com.wipro.repository
      - >  DepartmentRepo.java
    - ▼  com.wipro.service
      - >  DepartmentService.java
  - ▼  src/main/resources
    -  static
    -  templates
    -  application.properties
  - >  src/test/java
  - >  JRE System Library [JavaSE-17]
  - >  Maven Dependencies
  -  target/generated-test-sources/test-annotations
  -  target/generated-sources/annotations
  - >  src
  - >  target
  -  HELP.md
  -  mvnw
  -  mvnw.cmd

Now employee project

SpringbootEmployeesApplication.java ×

```
1 package com.wipro;
2
3 import org.springframework.boot.SpringApplication;
4
5
6
7
8 @SpringBootApplication
9 public class SpringbootEmployeesApplication {
10
11     public static void main(String[] args) {
12         SpringApplication.run(SpringbootEmployeesApplication.class, args);
13     }
14
15     /*
16      * @Bean public ModelMapper getModelMapper() { return new ModelMapper(); }
17      */
18
19     @Bean
20     public RestTemplate getRestTemplate() {
21         return new RestTemplate();
22     }
23
24 }
25
```

SpringbootEmployeesApplication.java

EmployeeController.java ×

```
1 package com.wipro.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
5
6
7
8
9
10
11
12
13 @RestController
14 @RequestMapping("/employees")
15 public class EmployeeController {
16
17     @Autowired
18     EmployeeService service;
19
20     @PostMapping
21     public EmployeeDto saveEmployee(@RequestBody EmployeeDto empdto) {
22         return service.saveEmployee(empdto);
23     }
24
25     @GetMapping("/{id}")
26     public EmployeeDepartmentDto getEmployeeById(@PathVariable Long id) {
27         return service.getEmployeeById(id);
28     }
29
30 }
31
32
33
```

```
1 package com.wipro.dto;
2
3 import com.wipro.entity.Employee;
4
5 public class EmployeeDepartmentDto {
6
7     private Employee empDto;
8     private DepartmentDto depDto;
9     public Employee getEmpDto() {
10         return empDto;
11     }
12     public void setEmpDto(Employee empDto) {
13         this.empDto = empDto;
14     }
15     public DepartmentDto getDepDto() {
16         return depDto;
17     }
18     public void setDepDto(DepartmentDto depDto) {
19         this.depDto = depDto;
20     }
21     public EmployeeDepartmentDto(Employee employee, DepartmentDto depDto) {
22         super();
23         this.empDto = employee;
24         this.depDto = depDto;
25     }
26     public EmployeeDepartmentDto() {
27         super();
28     }
29
30
31 }
32
```

```
1 package com.wipro.dto;
2
3 public class EmployeeDto {
4     private Long empId;
5     private String empName;
6     private String email;
7     private String deptCode;
8     public Long getEmpId() {
9         return empId;
10    }
11    public void setEmpId(Long empId) {
12        this.empId = empId;
13    }
14    public String getEmpName() {
15        return empName;
16    }
17    public EmployeeDto() {
18        super();
19    }
20    public EmployeeDto(Long empId, String empName, String email, String deptCode) {
21        super();
22        this.empId = empId;
23        this.empName = empName;
24        this.email = email;
25        this.deptCode = deptCode;
26    }
27    public void setEmpName(String empName) {
28        this.empName = empName;
29    }
30    public String getEmail() {
31        return email;
32    }
33    public void setEmail(String email) {
34        this.email = email;
35    }
36    public String getDeptCode() {
37        return deptCode;
38    }
39    public void setDeptCode(String deptCode) {
40        this.deptCode = deptCode;
41    }
42
43 }
44
45 }
```

```

1 package com.wipro.entity;
2
3 import jakarta.persistence.Entity;
4
5
6
7
8 @Entity
9 public class Employee {
10
11     @Id
12     @GeneratedValue(strategy=GenerationType.AUTO)
13     private Long empId;
14     private String empName;
15     private String email;
16     private String deptCode;
17
18     public Employee(Long empId, String empName, String email, String deptCode) {
19         super();
20         this.empId = empId;
21         this.empName = empName;
22         this.email = email;
23         this.deptCode = deptCode;
24     }
25     public Long getEmpId() {
26         return empId;
27     }
28     public void setEmpId(Long empId) {
29         this.empId = empId;
30     }
31     public String getEmpName() {
32         return empName;
33     }
34     public void setEmpName(String empName) {
35         this.empName = empName;
36     }
37     public String getEmail() {
38         return email;
39     }
40     public void setEmail(String email) {
41         this.email = email;
42     }
43     public String getDeptCode() {
44         return deptCode;
45     }
46     public void setDeptCode(String deptCode) {
47         this.deptCode = deptCode;
48     }
49     public Employee() {
50         super();
51     }
52
53
54
55
56
57
58 }
59

```

```

1 package com.wipro.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5
6
7
8 @Repository
9 public interface EmployeeRepo extends JpaRepository<Employee, Long> {
10
11 }
12

```

```

1 package com.wipro.service;
2
3 import com.wipro.dto.EmployeeDepartmentDto;
4
5
6 public interface EmployeeService {
7
8     public EmployeeDto saveEmployee(EmployeeDto empdto);
9
10    public EmployeeDepartmentDto getEmployeeById(Long id);
11
12 }
13

```

```

1 package com.wipro.service;
2
3 import org.modelmapper.ModelMapper;
4
5 @Service
6 public class EmployeeServiceImpl implements EmployeeService {
7
8     @Autowired
9     EmployeeRepo repo;
10
11     @Autowired
12     ModelMapper mapper;
13
14     @Autowired
15     RestTemplate restTemplate;
16
17     @Override
18     public EmployeeDto saveEmployee(EmployeeDto empdto) {
19         Employee emp= mapper.map(empdto, Employee.class);
20         Employee newEmp=repo.save(emp);
21         EmployeeDto dto= mapper.map(newEmp, EmployeeDto.class);
22         return dto;
23     }
24
25     @Override
26     public EmployeeDepartmentDto getEmployeeById(Long id) {
27
28         /*
29          * Optional<Employee> emp=repo.findById(id); EmployeeDto dto= mapper.map(emp,
30          * EmployeeDto.class); return dto;
31          */
32
33         Employee employee = repo.findById(id).get();
34
35         ResponseEntity<DepartmentDto> responseEntity = restTemplate.getForEntity("http://localhost:8080/departments/"+employee.getDeptCode(), DepartmentDto.class);
36
37         DepartmentDto departmentDto = responseEntity.getBody();
38
39         EmployeeDepartmentDto empDptDto=new EmployeeDepartmentDto(employee,departmentDto);
40         return empDptDto;
41     }
42
43 }
44
45
46
47
48
49
50
51
52
53
54
55

```



```
1 spring.application.name=Springboot-Employees
2
3
4 # MySQL Configuration
5 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
6 spring.datasource.url=jdbc:mysql://localhost:3306/wipro
7 spring.datasource.username=root
8 spring.datasource.password=#Mahadev7
9
10 # JPA & Hibernate
11 server.port=9091
12 spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
13 spring.jpa.hibernate.ddl-auto=update
14 spring.jpa.show-sql=true
15
16
```

```
https://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation)
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
4   <modelVersion>4.0.0</modelVersion>
5   <parent>
6     <groupId>org.springframework.boot</groupId>
7     <artifactId>spring-boot-starter-parent</artifactId>
8     <version>3.4.3</version>
9     <relativePath/> <!-- lookup parent from repository -->
10  </parent>
11  <groupId>com.wipro</groupId>
12  <artifactId>Springboot-Employees</artifactId>
13  <version>0.0.1-SNAPSHOT</version>
14  <name>Springboot-Employees</name>
15  <description>Demo project for Spring Boot microservice</description>
16  <url/>
17  <licenses>
18    <license/>
19  </licenses>
20  <developers>
21    <developer/>
22  </developers>
23  <scm>
24    <connection/>
25    <developerConnection/>
26    <tag/>
27    <url/>
28  </scm>
29  <properties>
30    <java.version>17</java.version>
31  </properties>
32  <dependencies>    Add Spring Boot Starters...
33    <dependency>
34      <groupId>org.springframework.boot</groupId>
35      <artifactId>spring-boot-starter-data-jpa</artifactId>
36    </dependency>
37    <dependency>
38      <groupId>org.springframework.boot</groupId>
39      <artifactId>spring-boot-starter-web</artifactId>
40    </dependency>
41  </dependencies>
42  <dependency>
43    <groupId>org.modelmapper</groupId>
44    <artifactId>modelmapper</artifactId>
45    <version>3.2.0</version>
46  </dependency>
47  <dependency>
48    <groupId>org.springframework.boot</groupId>
49    <artifactId>spring-boot-devtools</artifactId>
50    <scope>runtime</scope>
51    <optional>true</optional>
52  </dependency>
53  <dependency>
54    <groupId>com.mysql</groupId>
55    <artifactId>mysql-connector-j</artifactId>
56    <scope>runtime</scope>
57  </dependency>
```

```

58         <groupId>org.projectlombok</groupId>
59         <artifactId>lombok</artifactId>
60         <optional>true</optional>
61     </dependency>
62     <dependency>
63         <groupId>org.springframework.boot</groupId>
64         <artifactId>spring-boot-starter-test</artifactId>
65         <scope>test</scope>
66     </dependency>
67     <dependency>
68         <groupId>com.wipro</groupId>
69         <artifactId>Springboot-Department</artifactId>
70         <version>0.0.1-SNAPSHOT</version>
71     </dependency>
72 </dependencies>
73
74 <build>
75     <plugins>
76         <plugin>
77             <groupId>org.apache.maven.plugins</groupId>
78             <artifactId>maven-compiler-plugin</artifactId>
79             <configuration>
80                 <annotationProcessorPaths>
81                     <path>
82                         <groupId>org.projectlombok</groupId>
83                         <artifactId>lombok</artifactId>
84                     </path>
85                 </annotationProcessorPaths>
86             </configuration>
87         </plugin>
88         <plugin>
89             <groupId>org.springframework.boot</groupId>
90             <artifactId>spring-boot-maven-plugin</artifactId>
91             <configuration>
92                 <excludes>
93                     <exclude>
94                         <groupId>org.projectlombok</groupId>
95                         <artifactId>lombok</artifactId>
96                     </exclude>
97                 </excludes>
98             </configuration>
99         </plugin>
100     </plugins>
101 </build>
102
103 </project>
104

```

POST

localhost:9091/employees

Send

ParamsAuthHeaders (9)BodyScriptsSettingsCookies

rawJSONBeautify

```
1  {
2
3
4    "empName": "Pavan",
5    "email": "pavan@gmail.com",
6    "deptCode": "cse-101"
7
8  }
```

Body200 OK • 16 ms • 242 B •

{ } JSONPreviewVisualize

```
1  {
2    "empId": 253,
3    "empName": "Pavan",
4    "email": "pavan@gmail.com",
5    "deptCode": "cse-101"
6  }
```

ConsolePostbotRunnerVault

	emp_id	dept_code	email	emp_name
▶	252	NULL	pavan@gmail.com	Pavan
	253	cse-101	pavan@gmail.com	Pavan
✱	NULL	NULL	NULL	NULL