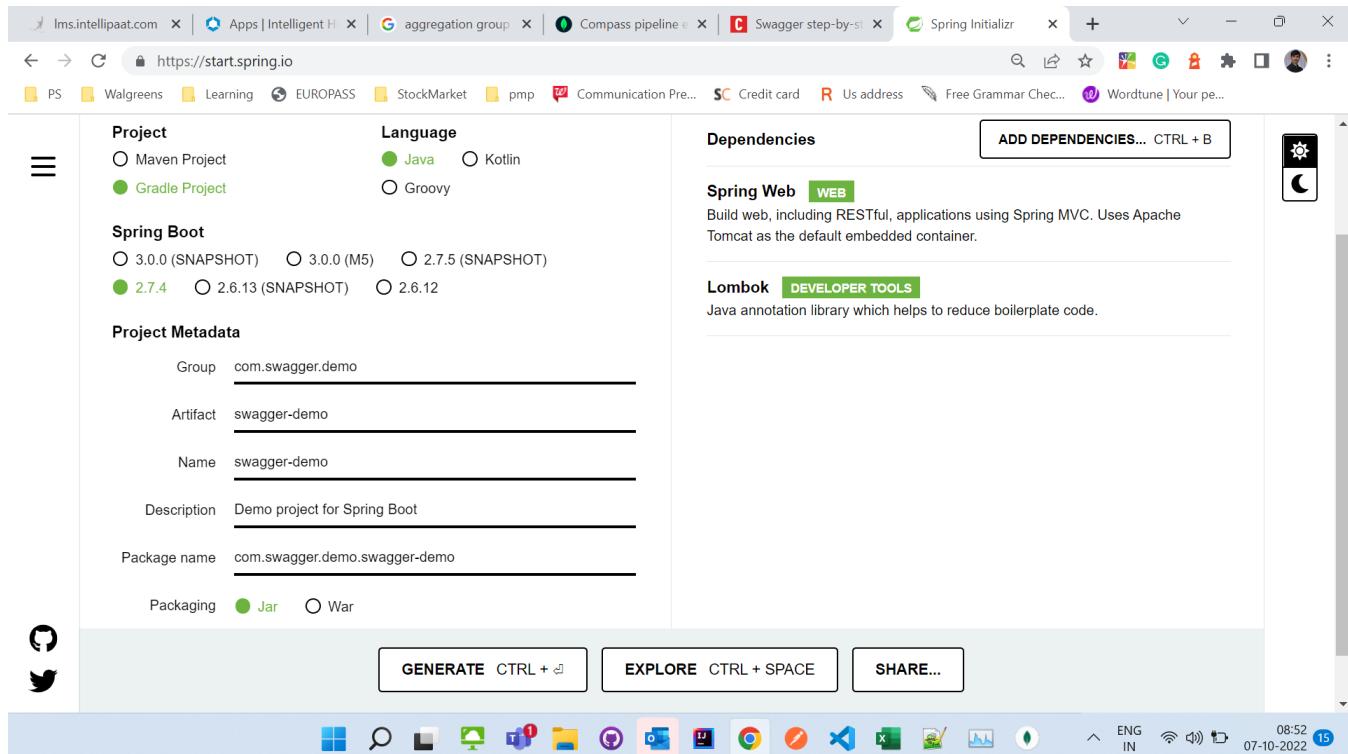


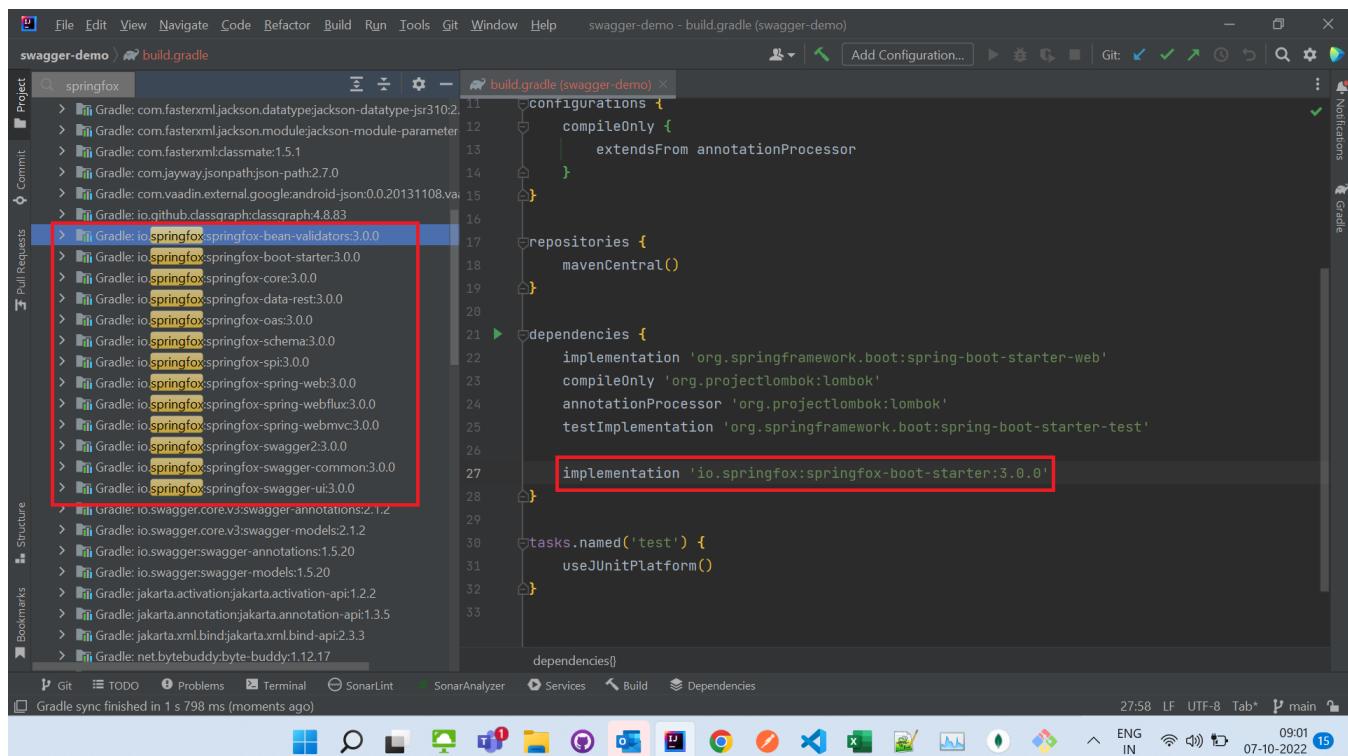
Swagger step-by-step integration guide with SpringBoot

Step 1: Let's create a new project from <https://start.spring.io/>. in case you already have a SpringBoot project well and good.



Step 2: Let's integrate the swagger into our project. Add the below dependency to our project **build.gradle** file and build the project.

implementation 'io.springfox:springfox-boot-starter:3.0.0'



Step 3: Let's add the Swagger configuration class

```
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;

@Configuration
@EnableWebMvc
public class SwaggerConfiguration {

}
```

Step 4: Build the project and run the project (**Remember we haven't exposed any API's in our code**). Visit <http://localhost:8080/v3/api-docs> and <http://localhost:8080/swagger-ui/index.html> page for yaml and swagger UI document.

http://localhost:8080/v3/api-docs/

```
{
  openapi: "3.0.3",
  info: {
    title: "Api Documentation",
    description: "Api Documentation",
    termsOfService: "urn:tos",
    contact: {},
    license: {
      name: "Apache 2.0",
      url: "http://www.apache.org/licenses/LICENSE-2.0"
    },
    version: "1.0"
  },
  servers: [
    {
      url: "http://localhost:8080",
      description: "Inferred Url"
    }
  ],
  tags: [
    {
      name: "basic-error-controller",
      description: "Basic Error Controller"
    }
  ],
  paths: {
    /error: {
      get: {
        tags: [
          "basic-error-controller"
        ],
        summary: "error",
        operationId: "errorUsingGET",
        responses: {
          "200": {
            description: "Successful operation"
          }
        }
      }
    }
  }
}
```

http://localhost:8080/swagger-ui/index.html#/basic-error-controller

The screenshot shows the Swagger UI interface for a Spring application. At the top, there's a navigation bar with links like 'PS', 'Walgreens', 'Learning', 'EUROPASS', 'StockMarket', 'pmp', 'Communication Pre...', 'Credit card', 'Us address', 'Free Grammar Chec...', 'Wordtune | Your pe...', and a 'view source' button. Below the navigation is the JSON API definition. The main content area displays the 'basic-error-controller' documentation with its methods: GET /error, PUT /error, POST /error, DELETE /error, OPTIONS /error, and HEAD /error. Each method has a detailed description and examples. The bottom of the screen shows the Windows taskbar with various pinned icons and system status.

NOTE: You see that basic-error-controller API's exposed from spring framework.

Step 5: Let's expose some of the APIs in our project and verify the swagger UI. (exposed 2 APIs in sample project as shown below)

```
File Edit View Navigate Code Refactor Build Run Tools Git Window Help swagger-demo - APIController.java [swagger-demo.main]
swagger-demo > src > main > java > com > swagger > demo > swaggerdemo > APIController
build.gradle (swagger-demo) < SwaggerDemoApplication.java < SwaggerConfiguration.java < APIController.java < Employee.java < BasicErrorHandler.java < EmployeeNotFoundException.java < Notifications
Project Commit Pull Requests Structure Bookmarks
10
11 @RestController
12 public class APIController {
13
14     2 usages
15     private final List<Employee> employeeList = new ArrayList<>();
16
17     @PostMapping("/employee")
18     public ResponseEntity<Employee> insertEmployee(@RequestBody Employee employee) {
19         employee.setId(UUID.randomUUID().toString());
20         employeeList.add(employee);
21         return new ResponseEntity<>(employee, HttpStatus.CREATED);
22     }
23
24     @GetMapping("/employee/{employeeId}")
25     public ResponseEntity<Employee> findEmployeeById(@PathVariable("employeeId") String employeeId) {
26         final Employee employee = employeeList.stream().filter(emp -> emp.getId().equals(employeeId)).findFirst().orElseThrow(
27             () -> new EmployeeNotFoundException(employeeId)
28         );
29         return ResponseEntity.ok(employee);
30     }
31
32 }
```

```
File Edit View Navigate Code Refactor Build Run Tools Git Window Help swagger-demo - Employee.java [swagger-demo.main]
swagger-demo > src > main > java > com > swagger > demo > swaggerdemo > Employee > salary
Employee.java < SwaggerConfiguration.java < APIController.java < SwaggerDemoExceptionHandler.java < ErrorMessage.java < Employee.java < BasicErrorHandler.java < EmployeeNotFoundException.java < Notifications
Project Commit Pull Requests Structure Bookmarks
1 package com.swagger.demo.swaggerdemo;
2
3 import lombok.Data;
4
5 @Data
6 public class Employee {
7     private String id;
8     private String name;
9     private int age;
10    private double salary;
11 }
12
```

Let's see how our api-doc and swagger-ui looks like.

```

{
  openapi: "3.0.3",
  info: {
    title: "API Documentation",
    description: "API Documentation",
    termsOfService: "urn:nos",
    contact: {},
    license: {
      name: "Apache 2.0",
      url: "http://www.apache.org/licenses/LICENSE-2.0"
    },
    version: "1.0"
  },
  servers: [
    {
      url: "http://localhost:8888",
      description: "Inferred URL"
    }
  ],
  tags: [
    {
      name: "api-controller",
      description: "API Controller"
    },
    {
      name: "basic-error-controller",
      description: "Basic Error Controller"
    }
  ],
  paths: {
    "/employee": {
      post: {
        tags: [
          "api-controller"
        ],
        summary: "InsertEmployee",
        operationId: "InsertEmployeeUsingPOST",
        requestBody: {
          content: {
            "application/json": {
              schema: {
                $ref: "#/components/schemas/Employee"
              }
            }
          }
        },
        responses: {
          "200": {
            description: "OK",
            content: {
              "*/*": {}
            }
          }
        }
      }
    }
  }
}

```

api-controller API Controller

POST /employee InsertEmployee

GET /employee/{employeeId} findEmployeeById

basic-error-controller Basic Error Controller

Schemas

```

Employee {
  age           integer($int32)
  id            string
  name          string
  salary        number($double)
}

```

NOTE: We haven't written a single line of code for the swagger document till now. The springfox library has done all the work till now.

Step 6: If you want to expose only controller-specific APIs then you need to explicitly create a **Docket** bean for the controller as below.

```
import org.springframework.context.annotation.Bean;
```

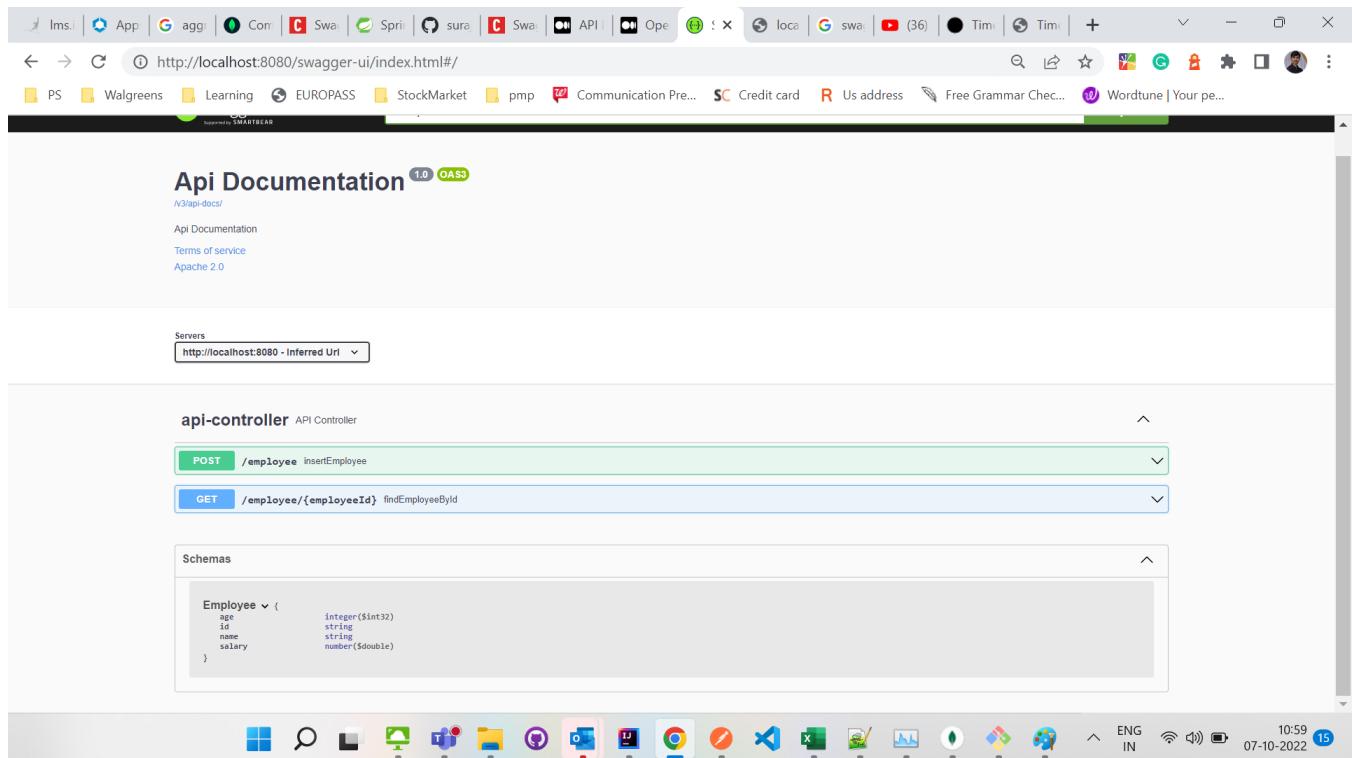
```

import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;

@Configuration
@EnableWebMvc
public class SwaggerConfiguration {

    @Bean
    public Docket api() {
        return new Docket(DocumentationType.OAS_30)
            .select()
            .apis(RequestHandlerSelectors.basePackage( "com.swagger.demo.swaggerdemo" ))
            .apis(RequestHandlerSelectors.any())
            .paths(PathSelectors.any())
            .build();
    }
}

```



The screenshot shows a browser window displaying an API documentation page at <http://localhost:8080/v3/api-docs/>. The page contains a JSON-like schema for an API definition. Several sections of the schema are highlighted with red boxes:

- A red box highlights the `tags` section under `paths`, which contains a single entry for the `api-controller`.
- A larger red box highlights the `paths` section under `paths`, specifically the `/employee` endpoint.

```
{  
  openapi: "3.0.3",  
  info: {  
    title: "API Documentation",  
    description: "API Documentation",  
    termsOfService: "urn:tos",  
    contact: {},  
    license: {  
      name: "Apache 2.0",  
      url: "https://www.apache.org/licenses/LICENSE-2.0"  
    },  
    version: "1.0"  
  },  
  servers: [  
    {  
      url: "http://localhost:8080",  
      description: "Inferred URL"  
    }  
  ],  
  tags: [  
    {  
      name: "api-controller",  
      description: "API Controller"  
    }  
  ],  
  paths: {  
    /employee: {  
      post: {  
        tags: [  
          "api-controller"  
        ],  
        summary: "InsertEmployee",  
        operationId: "InsertEmployeeUsingPOST",  
        requestBody: {  
          content: {  
            application/json: {  
              schema: {  
                $ref: "#/components/schemas/Employee"  
              }  
            }  
          }  
        },  
        responses: {  
          200: {  
            description: "OK",  
            content: {  
              */*: {  
                schema: {  
                  $ref: "#/components/schemas/Employee"  
                }  
              }  
            }  
          }  
        }  
      }  
    }  
  }  
}  
paths["/employee"].post
```

Step 7: Now let's test and verify each API.

Let's see the documentation for `insertEmployee` API. The documentation shows how the `requestBody` should look like. What should be the response status and response body for the API.

The screenshot shows the Swagger UI interface for an API controller. At the top, the URL is <http://localhost:8080/swagger-ui/index.html#/api-controller/insertEmployeeUsingPOST>. The main area displays the following information:

Parameters: No parameters.

Request body: application/json

Example Value | Schema:

```
{
  "age": 0,
  "id": "string",
  "name": "string",
  "salary": 0
}
```

Responses

Code	Description	Links
200	OK	No links
201	Created	No links
401	Unauthorized	No links
403	Forbidden	No links
404	Not Found	No links

Let's test the API from the swagger-ui. Use **Try it out** Option.

api-controller API Controller

POST /employee insertEmployee

Parameters

No parameters

Request body

```
{
  "age": 26,
  "name": "Suraj Panduranga Jannu",
  "salary": 100.00
}
```

Cancel Reset

Execute Clear

Responses

curl -X 'POST' \
 http://localhost:8080/employee' \
 -H 'accept: */*' \
 -H 'Content-Type: application/json' \
 -d '{
 "age": 26,
 "name": "Suraj Panduranga Jannu",
 "salary": 100.00
 }'

Request URL

http://localhost:8080/employee

Server response

Code	Details
201	<p>Response body</p> <pre>{ "id": "89fcfb46-354a-4734-b42e-a2be67a61ab2", "name": "Suraj Panduranga Jannu", "age": 26, "salary": 100 }</pre> <p>Download</p> <p>Response headers</p> <pre>connection: keep-alive content-type: application/json date: Fri, 07 Oct 2022 05:31:46 GMT keep-alive: timeout=60 transfer-encoding: chunked</pre>

Responses

Code	Description	Links
200		No links

NOTE: You can see that swagger has made a CURL API call to our service and got the response (response body, response status code, and response headers).

Similarly, let's verify the `findEmployeeById` API. (Copy the employeeId from the `insertEmployee` API responseBody)

The screenshot shows two views of the Swagger UI interface. The top view displays a 'Not Found' error page with a status code of 404. The bottom view shows the 'Responses' section for the `GET /employee/{employeeId}` endpoint. It includes a 'Curl' command, a 'Request URL' (set to `http://localhost:8080/employee/89fcfb46-354a-4734-b42e-a2be67a61ab2`), and a 'Server response' section. The 'Server response' section contains a JSON response body with a red box highlighting it. The JSON response is:

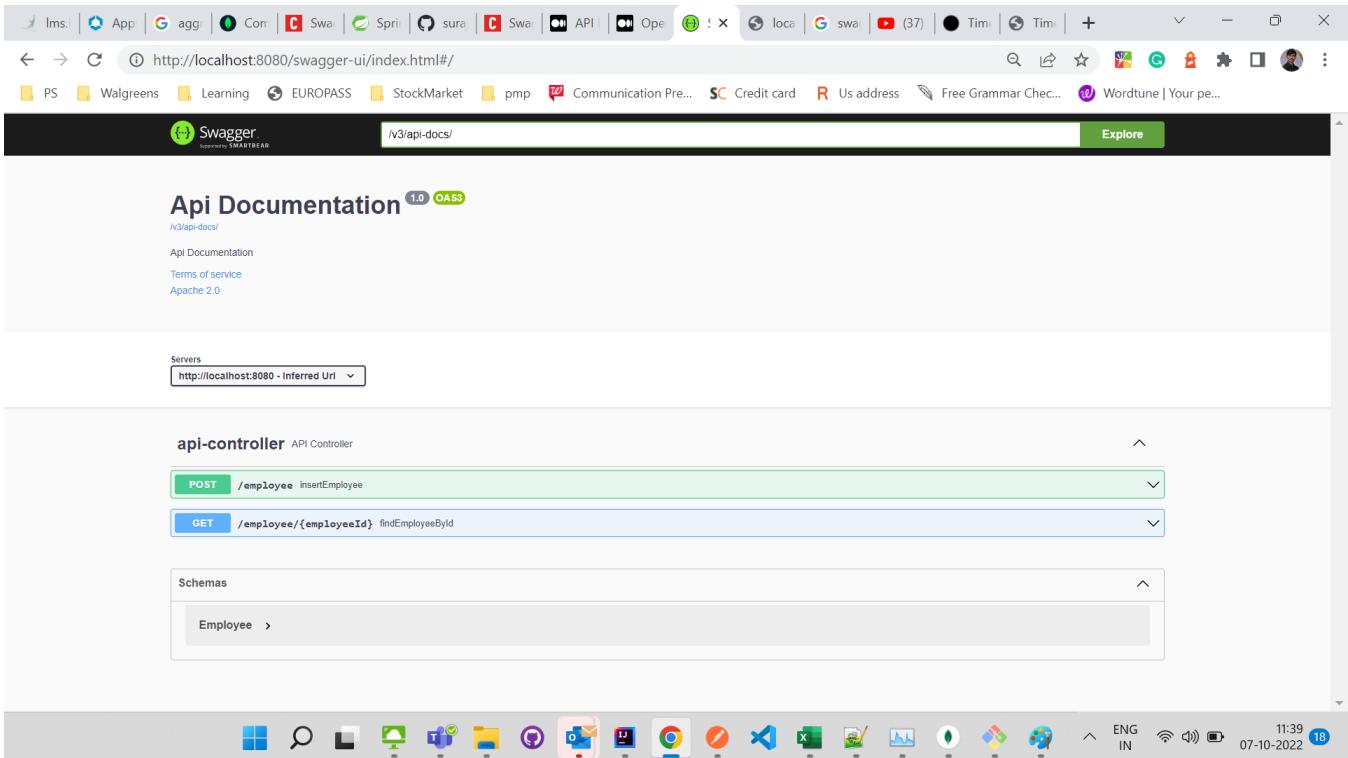
```
{ "id": "89fcfb46-354a-4734-b42e-a2be67a61ab2", "name": "Suraj Panduranga Janmu", "age": 26, "salary": 100 }
```

The bottom view also shows a 'Code' tab with a 'Details' sub-tab selected, displaying the response body and headers. The 'Responses' tab is also visible.

Step 8: We have learned what swagger documentation looks like and how to test from swagger UI. Now let's look at how can we customize the swagger documentation based on our needs.

Let's capture the Before and after screenshots to see the difference.

Before:



Step 9: You can change the **application-related information** in the documentation to give a clear understanding of the application/service. Adding the below piece of code in the configuration file will add details about the application in the swagger documentation.

```

@Bean
public Docket api() {
    return new Docket(DocumentationType.OAS_3_0)
        .apiInfo(apiInfo())
        .select()
        .apis(RequestHandlerSelectors.basePackage("com.swagger.demo.swaggerdemo"))
        .apis(RequestHandlerSelectors.any())
        .paths(PathSelectors.any())
        .build();
}

private ApiInfo apiInfo() {
    return new ApiInfoBuilder().title("Swagger Demo Application")
        .description("Swagger demo application to demonstrate the functionality and documentation | New line documentation")
        .termsOfServiceUrl("https://www.apache.org/licenses/LICENSE-2.0")
        .contact(new Contact("Suraj Jannu", "https://jannusuraj.wixsite.com/resume", "jannusuraj@gmail.com"))
        .license("Apache 2.0")
        .licenseUrl("https://www.apache.org/licenses/LICENSE-2.0")
        .version("0.0.1")
        .build();
}

```

You can validate the changes in swagger-ui as shown below.

The screenshot shows the Swagger UI interface for a 'Swagger Demo Application'. At the top, there's a navigation bar with various links like PS, Walgreens, Learning, EUROPASS, StockMarket, pmp, Communication Pre..., Credit card, Us address, Free Grammar Check..., Wordtune, and Your pe... Below the navigation is the main content area.

Swagger Demo Application (version 0.0.1, OAS3)

Swagger demo application to demonstrate the functionality and documentation | New line documentation

Terms of service

Suraj Jannu - Web developer
Send email to Suraj Jannu
Apache 2.0

Servers

http://localhost:8080 - Inferred Url

api-controller API Controller

- POST /employee insertEmployee**
- GET /employee/{employeeId} findEmployeeById**

Schemas

Bottom right corner shows system status: ENG IN, 12:02, 07-10-2022, 19 notifications.

Step 10: Let's make changes to our API documentation

1. `@Api` annotation on our `Controller` class to describe our API.

```
@Api(value = "Employee Controller", description = "Employee Controller")
```

2. `@ApiOperation` annotation to describe the endpoint and its response type.

```
@ApiOperation(value = "Create Employee", response = Employee.class)
```

3. For mediaType support defined the produces property for API as shown below for JSON.

```
@PostMapping(value = "/employee", produces = MediaType.APPLICATION_JSON_VALUE)
@GetMapping(value = "/employee/{employeeId}", produces = MediaType.APPLICATION_JSON_VALUE)
```

4. You can use the `@ApiModelProperty` annotation to describe the properties of the model. With `@ApiModelProperty` you can also document property as required.

```
@ApiModelProperty(notes = "Employee name", required = true)
```

The screenshot shows two screenshots of a Swagger UI interface for a REST API.

Screenshot 1: POST /employee

This section details the creation of an employee record. It includes:

- Method:** POST
- Path:** /employee
- Description:** Create Employee
- Parameters:** None
- Request body:** application/json
- Example Value:** Schema (JSON object with fields age, id, name, salary)
- Responses:**
 - Code:** 200, **Description:** Employee record created, **Media type:** application/json (highlighted with a red box)
 - Code:** 401, **Description:** Unauthorized
 - Code:** 403, **Description:** Forbidden
 - Code:** 404, **Description:** Employee record not found

Screenshot 2: Schemas

This section shows the schema definition for the Employee entity:

```

Employee {
    age: integer($int32)
    id: string
    name*: string
    salary: number($double)
}
  
```

The fields are annotated with descriptions:

- age: Employee age
- id: Unique employee ID
- name*: Employee name
- salary: Employee salary

5. In order to remove the unwanted default response code use the **useDefaultResponseMessages** method.

```

@Bean
public Docket api() {
    return new Docket(DocumentationType.OAS_3_0)
        .apiInfo(apiInfo())
        .useDefaultResponseMessages(false)
        .select()
        .apis(RequestHandlerSelectors.basePackage("com.swagger.demo.swaggerdemo"))
        .apis(RequestHandlerSelectors.any())
        .paths(PathSelectors.any())
        .build();
}

```

The screenshot shows the Swagger UI interface for a REST API. The URL in the address bar is <http://localhost:8080/swagger-ui/index.html#/api-controller/insertEmployeeUsingPOST>. The main content area is divided into sections: 'Request body' (containing a schema example) and 'Responses' (listing a 200 status code with its description and media type). A 'GET /employee/{employeeId}' button is visible at the bottom. The bottom of the screen features a taskbar with various application icons.

Step 11: Swagger-related code in the project

File Edit View Navigate Code Refactor Build Run Tools Git Window Help swagger-demo - SwaggerConfiguration.java [swagger-demo.main]

swagger-demo > src > main > java > com > swagger > demo > swaggerdemo > SwaggerConfiguration.java

```
17 package com.swagger.demo.swaggerdemo;
18
19 import io.swagger.v3.oas.annotations.Bean;
20 import io.swagger.v3.oas.annotations.Docket;
21 import io.swagger.v3.oas.annotations.apiInfo.ApiInfo;
22 import io.swagger.v3.oas.annotations.selectors.RequestHandlerSelectors;
23 import io.swagger.v3.oas.annotations.selectors.PathSelectors;
24 import io.swagger.v3.oas.annotations.selectors.ApiSelectorBuilder;
25 import io.swagger.v3.oas.annotations.apis.Apis;
26 import io.swagger.v3.oas.annotations.paths.Paths;
27 import io.swagger.v3.oas.annotations.builds.Build;
28
29
30 @Bean
31 public Docket api() {
32     return new Docket(DocumentationType.OAS_3_0)
33         .apiInfo(apiInfo())
34         .useDefaultResponseMessages(false)
35         .select(ApiSelectorBuilder.basePackage("com.swagger.demo.swaggerdemo"))
36         .apis(RequestHandlerSelectors.any())
37         .paths(PathSelectors.any())
38         .build();
39 }
40
41
42 private ApiInfo apiInfo() {
43     return new ApiInfoBuilder().title("Swagger Demo Application")
44         .description("Swagger demo application to demonstrate the functionality and documentation | New line documentation")
45         .termsOfServiceUrl("https://www.apache.org/licenses/LICENSE-2.0")
46         .contact(new Contact("Suraj Jannu", "https://jannusuraj.wixsite.com/resume", "jannusuraj@gmail.com"))
47         .license("Apache 2.0")
48         .licenseUrl("https://www.apache.org/licenses/LICENSE-2.0")
49         .version("0.0.1")
50         .build();
51 }
```

Project Commit Pull Requests Structure Bookmarks Bookmarks Notifications Grade

Git Run TODO Problems Terminal SonarLint SonarAnalyzer Services Build Dependencies

Gradle sync finished in 23 s 667 ms (26 minutes ago)

17:5 CRLF UTF-8 4 spaces main 14:36 20 07-10-2022

File Edit View Navigate Code Refactor Build Run Tools Git Window Help swagger-demo - APIController.java [swagger-demo.main]

swagger-demo > src > main > java > com > swagger > demo > swaggerdemo > APIController > findEmployeeById > SwaggerDemoApplication

```
16 package com.swagger.demo.swaggerdemo;
17
18 import io.swagger.v3.oas.annotations.RestController;
19 import io.swagger.v3.oas.annotations.Api;
20 import io.swagger.v3.oas.annotations.Operation;
21 import io.swagger.v3.oas.annotations.responses.ApiResponses;
22 import io.swagger.v3.oas.annotations.responses.ApiResponse;
23 import io.swagger.v3.oas.annotations.parameters.RequestBody;
24 import io.swagger.v3.oas.annotations.parameters.PathVariable;
25 import io.swagger.v3.oas.annotations.parameters.Query;
26 import io.swagger.v3.oas.annotations.parameters.Header;
27 import io.swagger.v3.oas.annotations.parameters.Cookie;
28
29
30 @RestController
31 @Api(value = "Employee Controller", description = "Employee Controller")
32 public class APIController {
33
34     2 usages
35     private final List<Employee> employeeList = new ArrayList<>();
36
37     @PostMapping(value = "/employee", produces = MediaType.APPLICATION_JSON_VALUE)
38     @ApiOperation(value = "Create Employee", response = Employee.class)
39     @ApiResponses(value = {
40         @ApiResponse(code = 200, message = "Employee record created")
41     })
42     public ResponseEntity<Employee> insertEmployee(@RequestBody Employee employee) { ... }
43
44     @suraj jannu
45     @GetMapping(value = "/employee/{employeeId}", produces = MediaType.APPLICATION_JSON_VALUE)
46     @ApiOperation(value = "Find Employee by Id", response = Employee.class)
47     @ApiResponses(value = {
48         @ApiResponse(code = 200, message = "Employee record found"),
49         @ApiResponse(code = 404, message = "Employee record not found")
50     })
51     public ResponseEntity<Employee> findEmployeeById(@PathVariable("employeeId") String employeeId) throws EmployeeNotFoundException { ... }
52 }
```

Project Commit Pull Requests Structure Bookmarks Bookmarks Notifications Grade

Git Run TODO Problems Terminal SonarLint SonarAnalyzer Services Build Dependencies

Gradle sync finished in 23 s 667 ms (27 minutes ago)

33:21 CRLF UTF-8 4 spaces main 14:37 20 07-10-2022

```
File Edit View Navigate Code Refactor Build Run Tools Git Window Help swagger-demo - Employee.java [swagger-demo.main]
swagger-demo src main java com swagger demo swaggerdemo Employee
APIController.java Employee.java SwaggerConfiguration.java

1 package com.swagger.demo.swaggerdemo;
2
3 import io.swagger.annotations.ApiModelProperty;
4 import lombok.Data;
5
6 7 usages surajjannu
7 @Data
8 public class Employee {
9     @ApiModelProperty(notes = "Unique employee ID")
10    private String id;
11    @ApiModelProperty(notes = "Employee name", required = true)
12    private String name;
13    @ApiModelProperty(notes = "Employee age")
14    private int age;
15    @ApiModelProperty(notes = "Employee salary")
16    private double salary;
17 }
```

Project Commit Pull Requests Structure Bookmarks

File Edit View Navigate Code Refactor Build Run Tools Git Window Help swagger-demo - Employee.java [swagger-demo.main]

swagger-demo src main java com swagger demo swaggerdemo Employee

APIController.java Employee.java SwaggerConfiguration.java

1 package com.swagger.demo.swaggerdemo;

2

3 import io.swagger.annotations.ApiModelProperty;

4 import lombok.Data;

5

7 usages surajjannu

7 @Data

8 public class Employee {

9 @ApiModelProperty(notes = "Unique employee ID")

10 private String id;

11 @ApiModelProperty(notes = "Employee name", required = true)

12 private String name;

13 @ApiModelProperty(notes = "Employee age")

14 private int age;

15 @ApiModelProperty(notes = "Employee salary")

16 private double salary;

17 }

Notifications Grade

Git Run TODO Problems Terminal SonarLint SonarAnalyzer Services Build Dependencies

Gradle sync finished in 23 s 667 ms (28 minutes ago)

7:14 CRLF UTF-8 4 spaces main 14:38 07-10-2022 ENG IN

File Edit View Navigate Code Refactor Build Run Tools Git Window Help swagger-demo - Employee.java [swagger-demo.main]

swagger-demo src main java com swagger demo swaggerdemo Employee

APIController.java Employee.java SwaggerConfiguration.java

1 package com.swagger.demo.swaggerdemo;

2

3 import io.swagger.annotations.ApiModelProperty;

4 import lombok.Data;

5

7 usages surajjannu

7 @Data

8 public class Employee {

9 @ApiModelProperty(notes = "Unique employee ID")

10 private String id;

11 @ApiModelProperty(notes = "Employee name", required = true)

12 private String name;

13 @ApiModelProperty(notes = "Employee age")

14 private int age;

15 @ApiModelProperty(notes = "Employee salary")

16 private double salary;

17 }

Notifications Grade

Git Run TODO Problems Terminal SonarLint SonarAnalyzer Services Build Dependencies

Gradle sync finished in 23 s 667 ms (28 minutes ago)

7:14 CRLF UTF-8 4 spaces main 14:38 07-10-2022 ENG IN