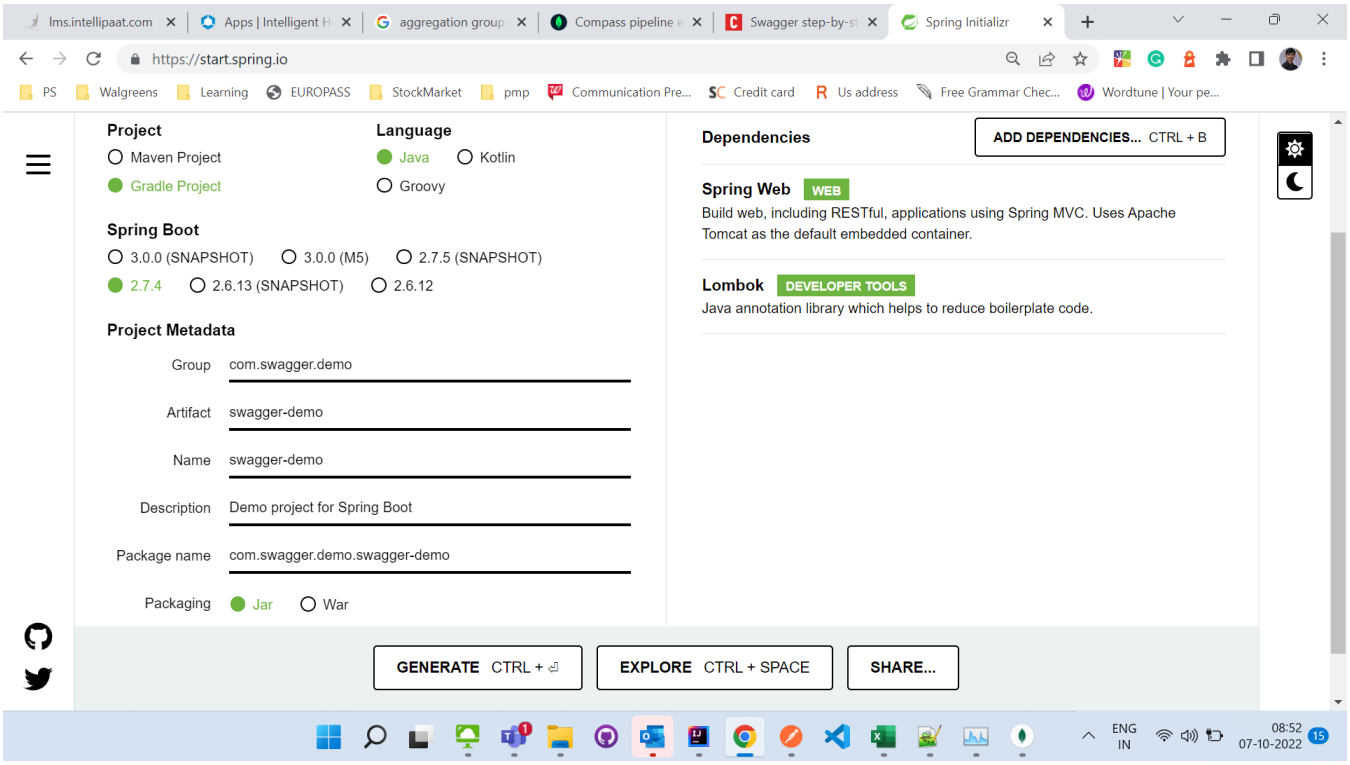
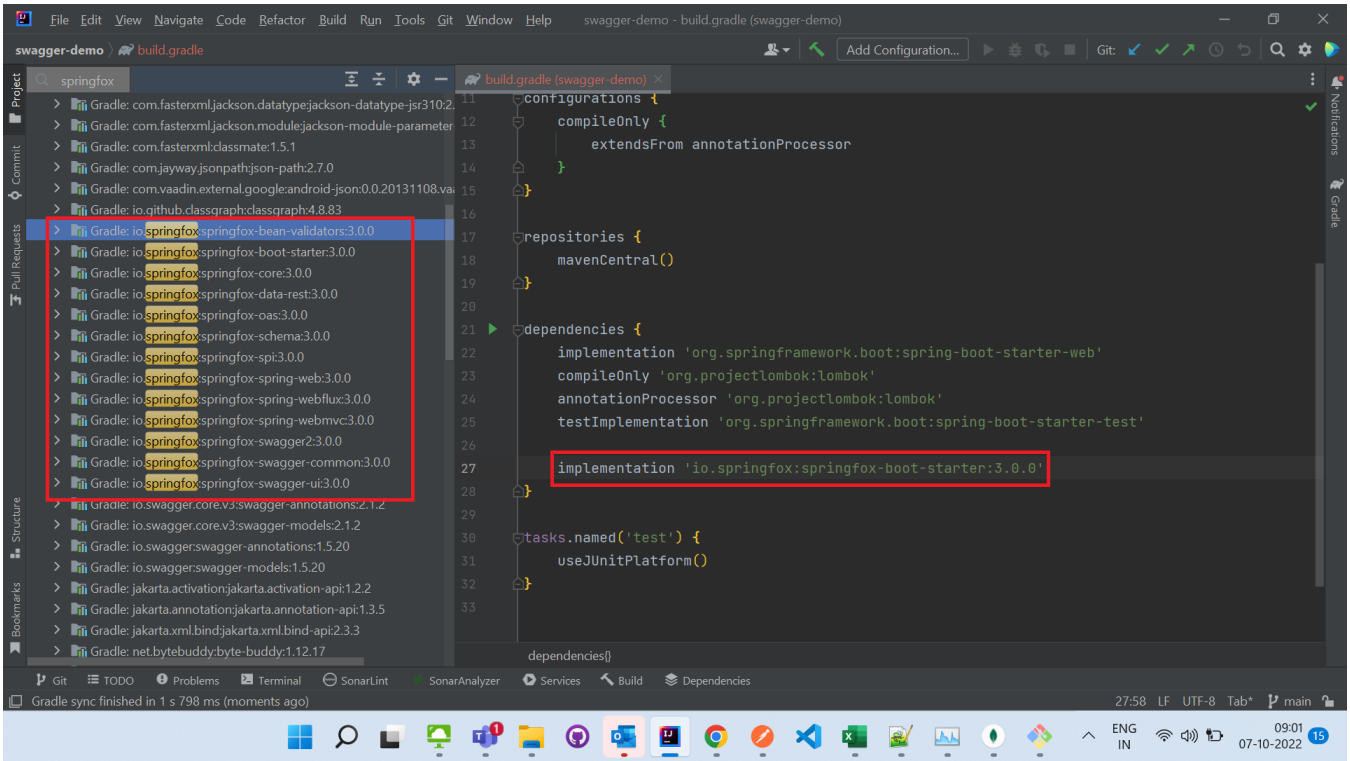
Swagger step-by-step integration guide with SpringBoot

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.

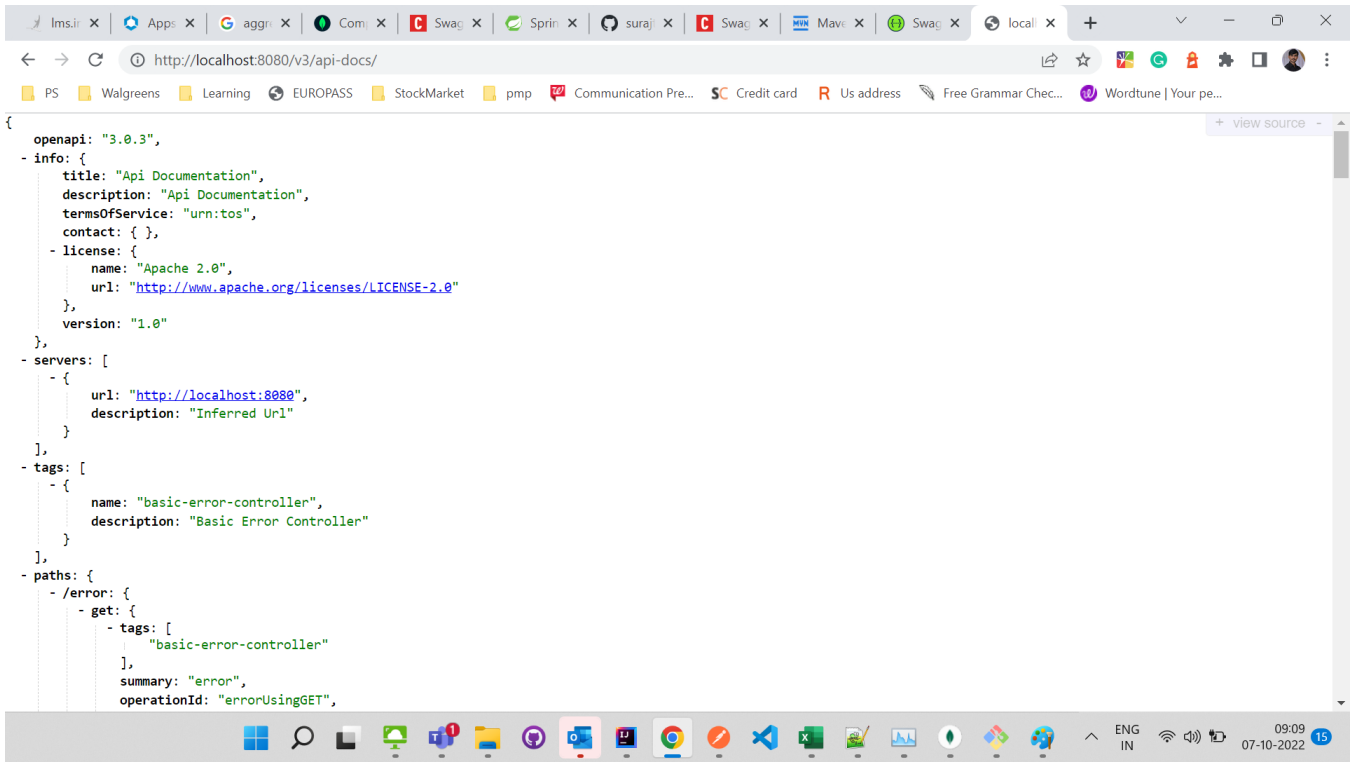
implementation 'io.springfox:springfox-boot-starter:3.0.0'

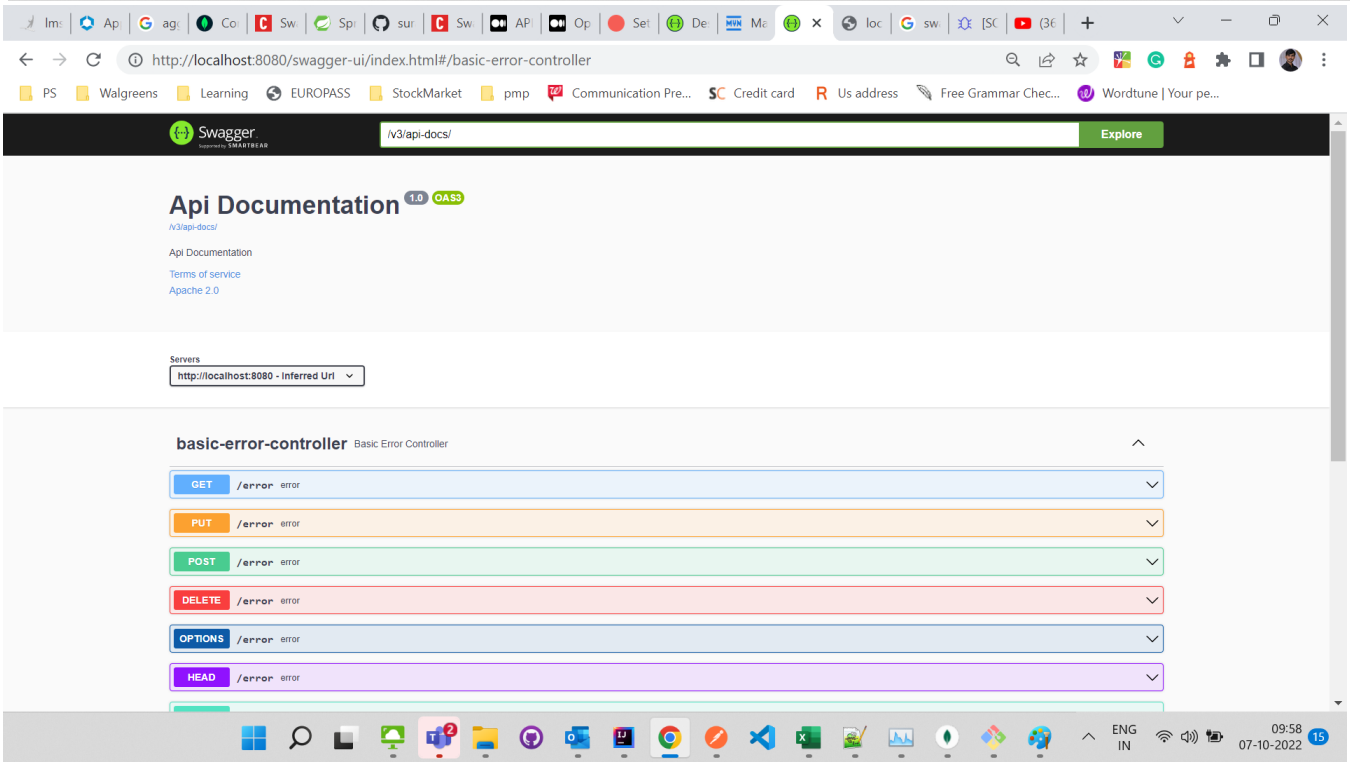


Step 3: Let's add the Swagger configuration class

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

@EnableWebMvc
@Configuration
public class SwaggerConfiguration {

}
```

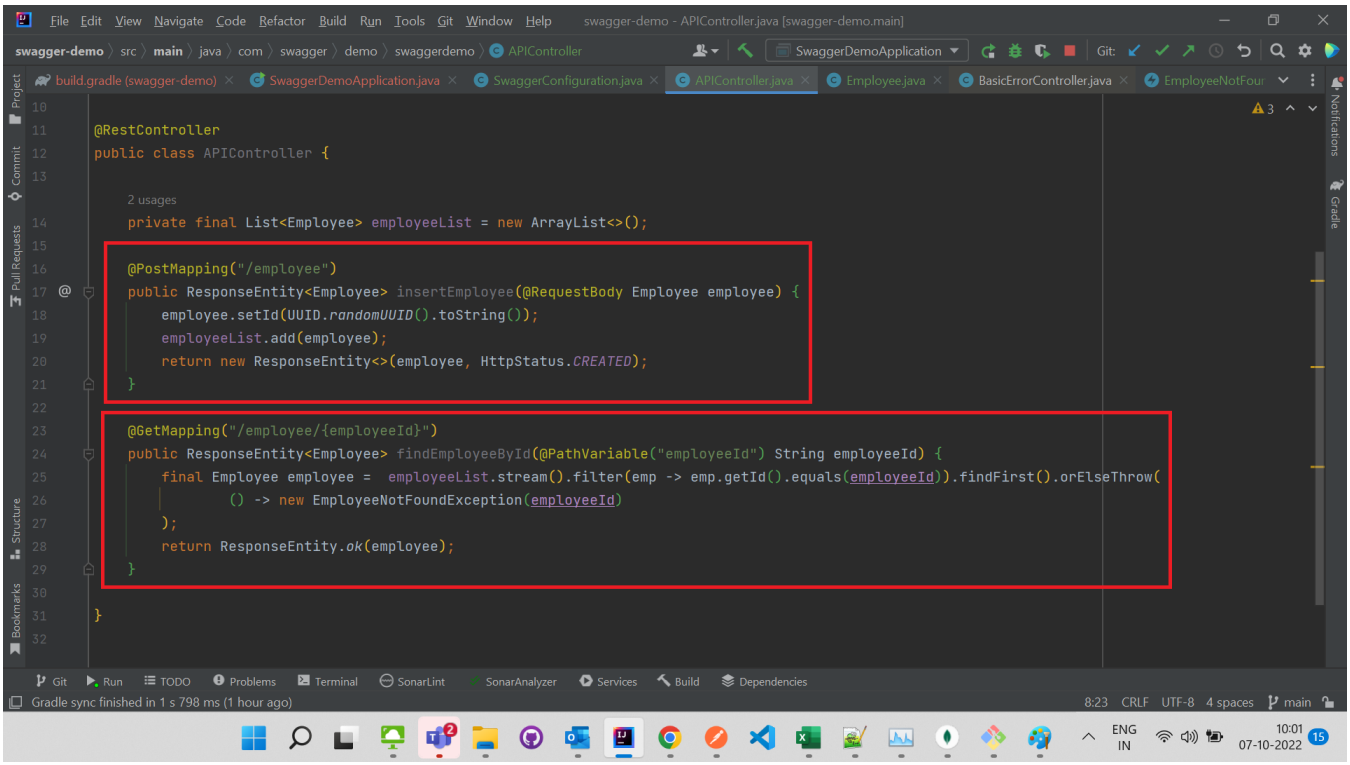
Step 4: Build the project and run the project (**Remember we haven't exposed any API's in our code**). Visit http://localhost:8080/v3/api-docs/ and http://localhost:8080/swagger-ui/index.html page for yaml and swagger UI document.

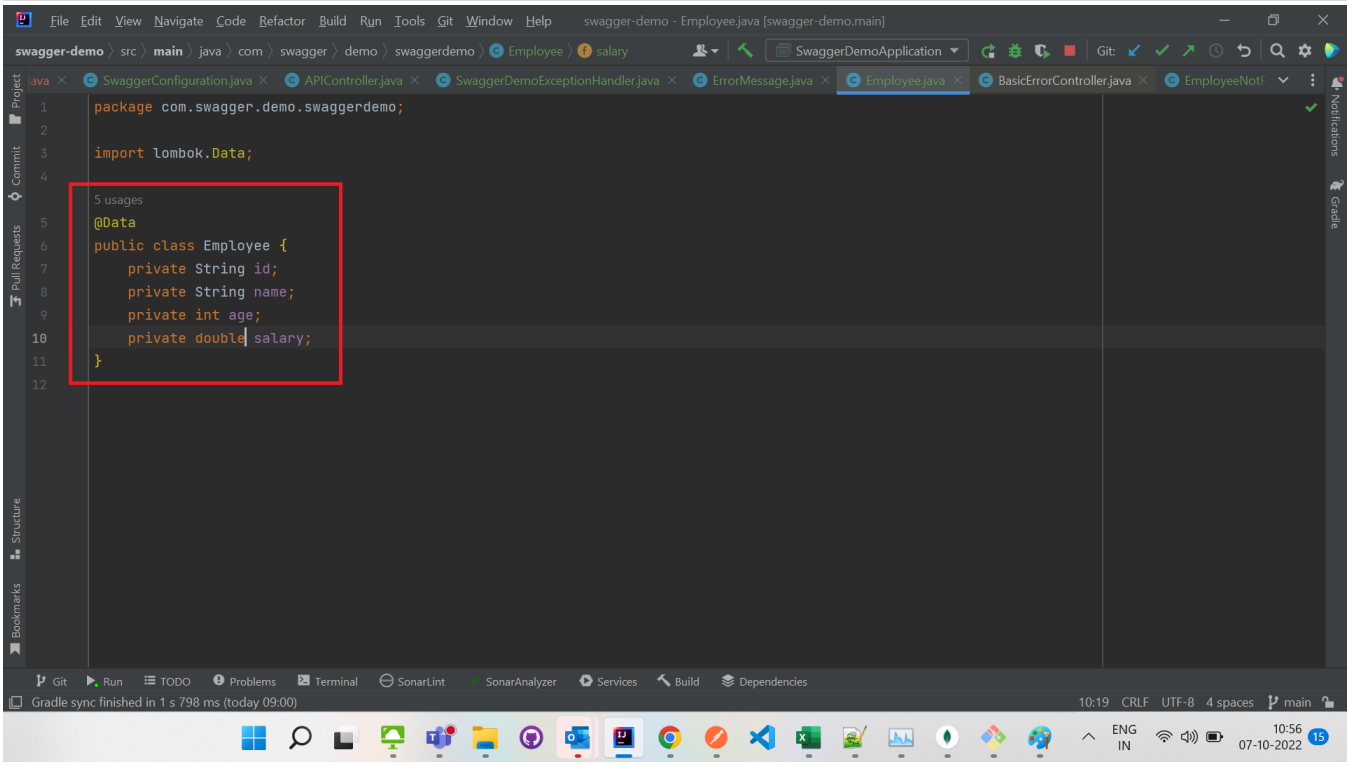




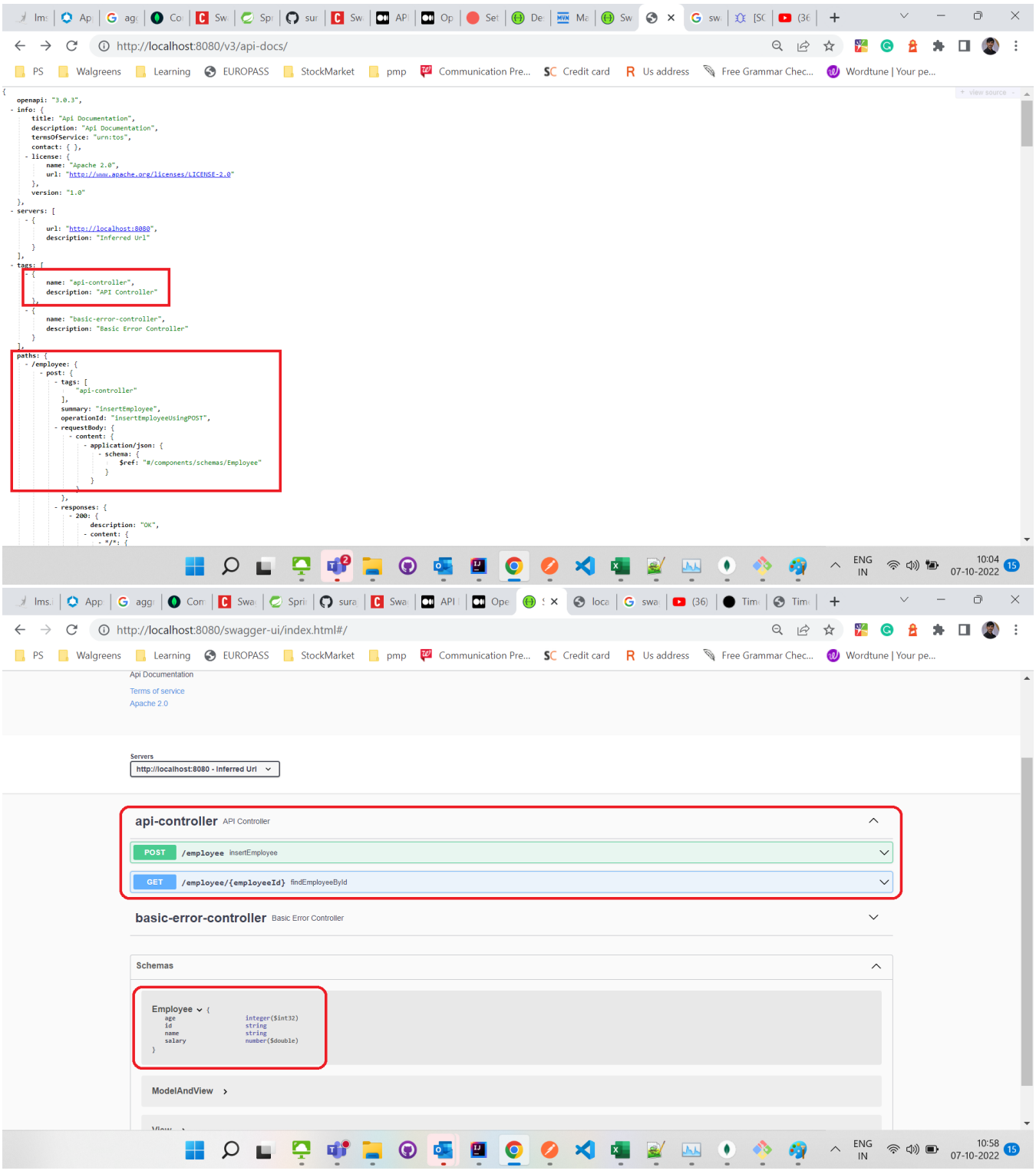
NOTE: You see that basic-error-controller API's exposed from spring framework.

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)





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

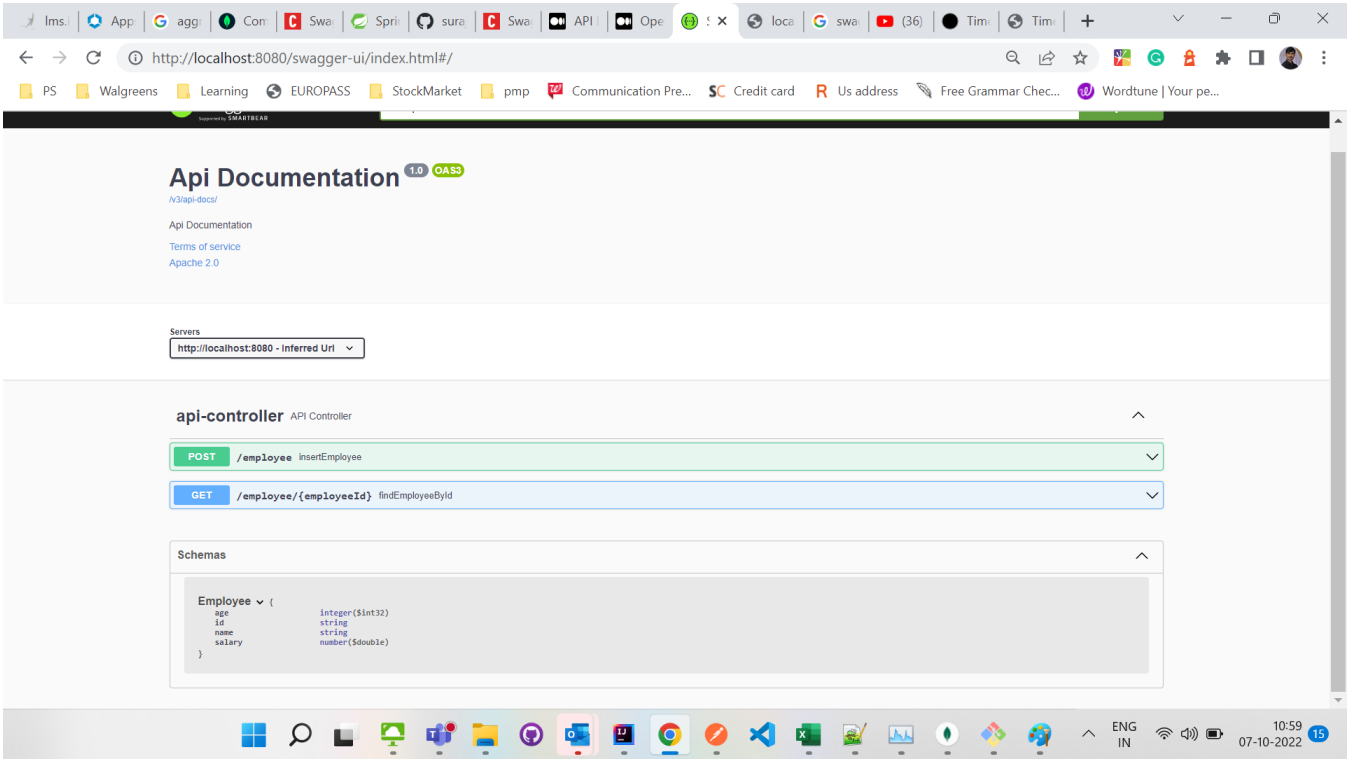


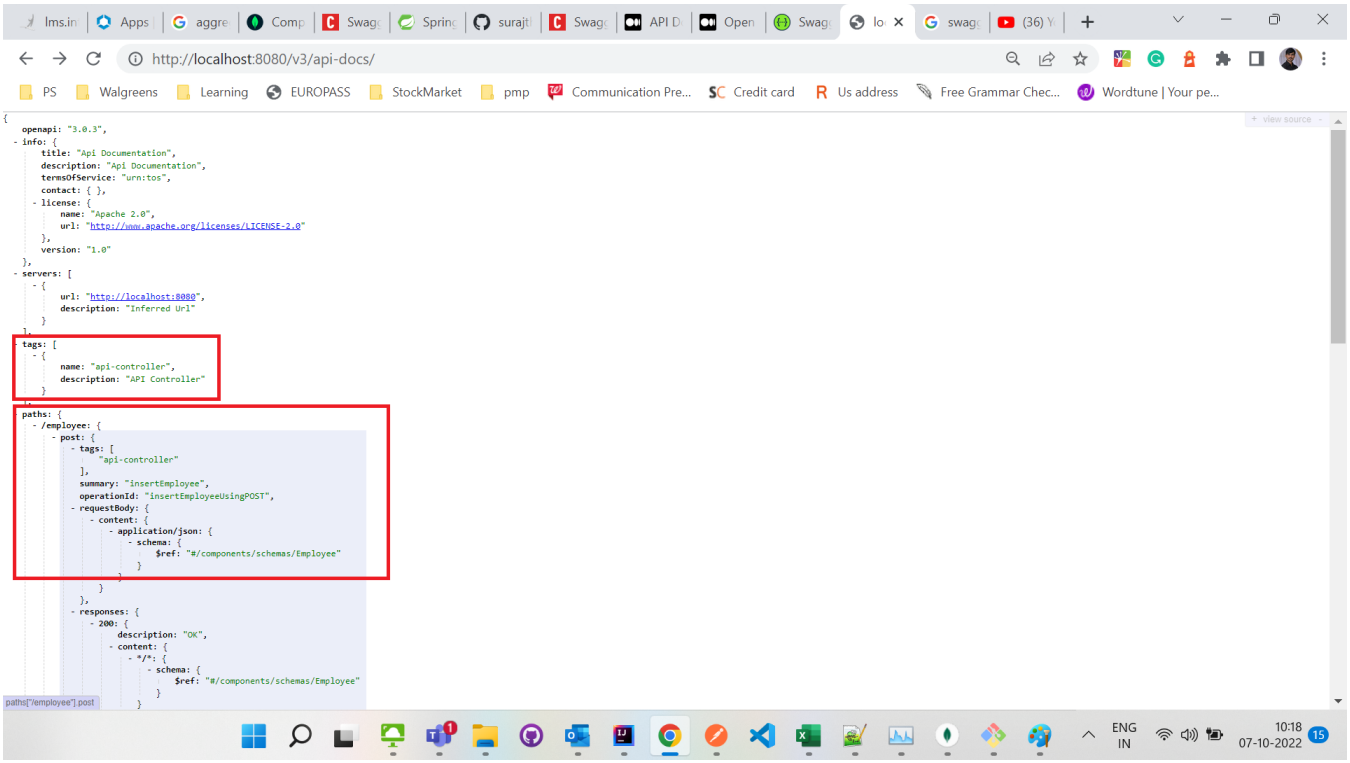
NOTE: We haven't written a single line of code for the swagger document till now. The springfox library has done all the work till now.

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;
```

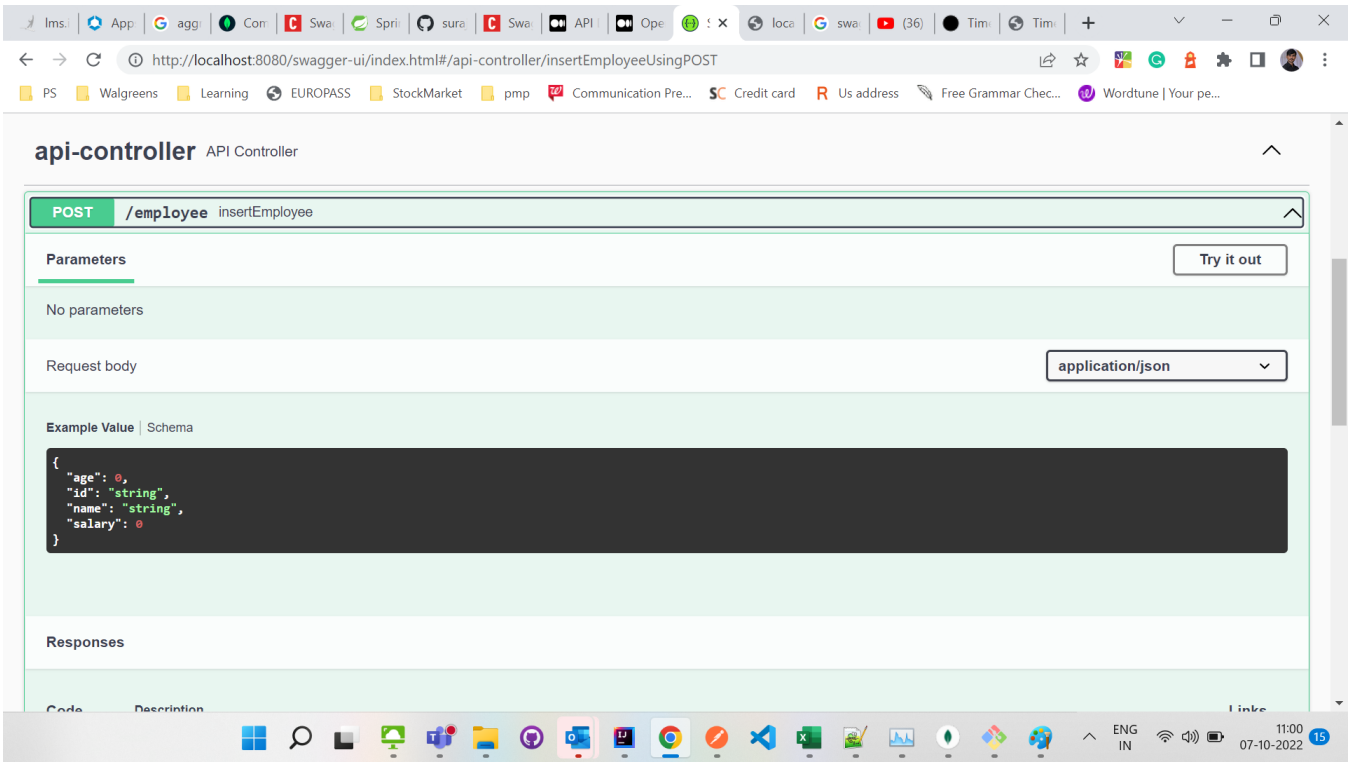
```java
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.spi.DocumentationType;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

@EnableWebMvc
@Configuration
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();
    }
}
```

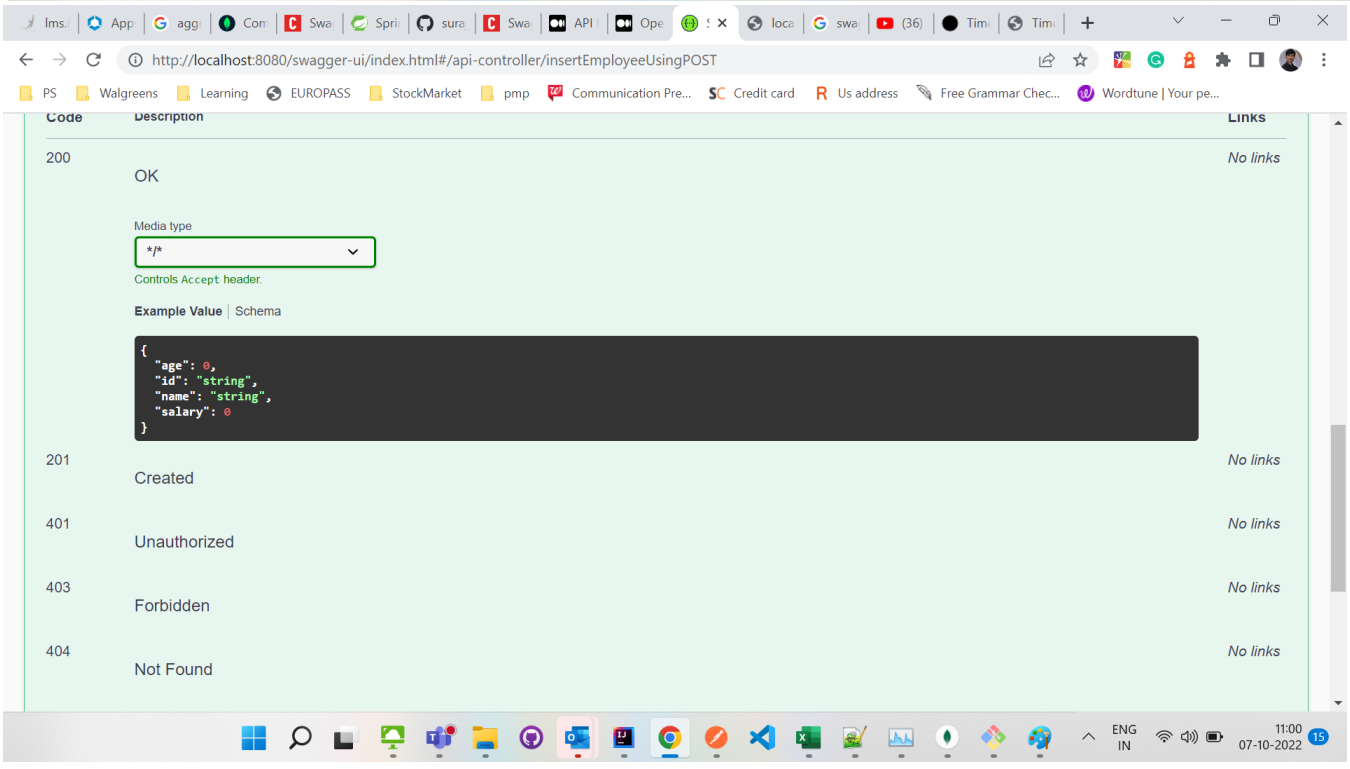




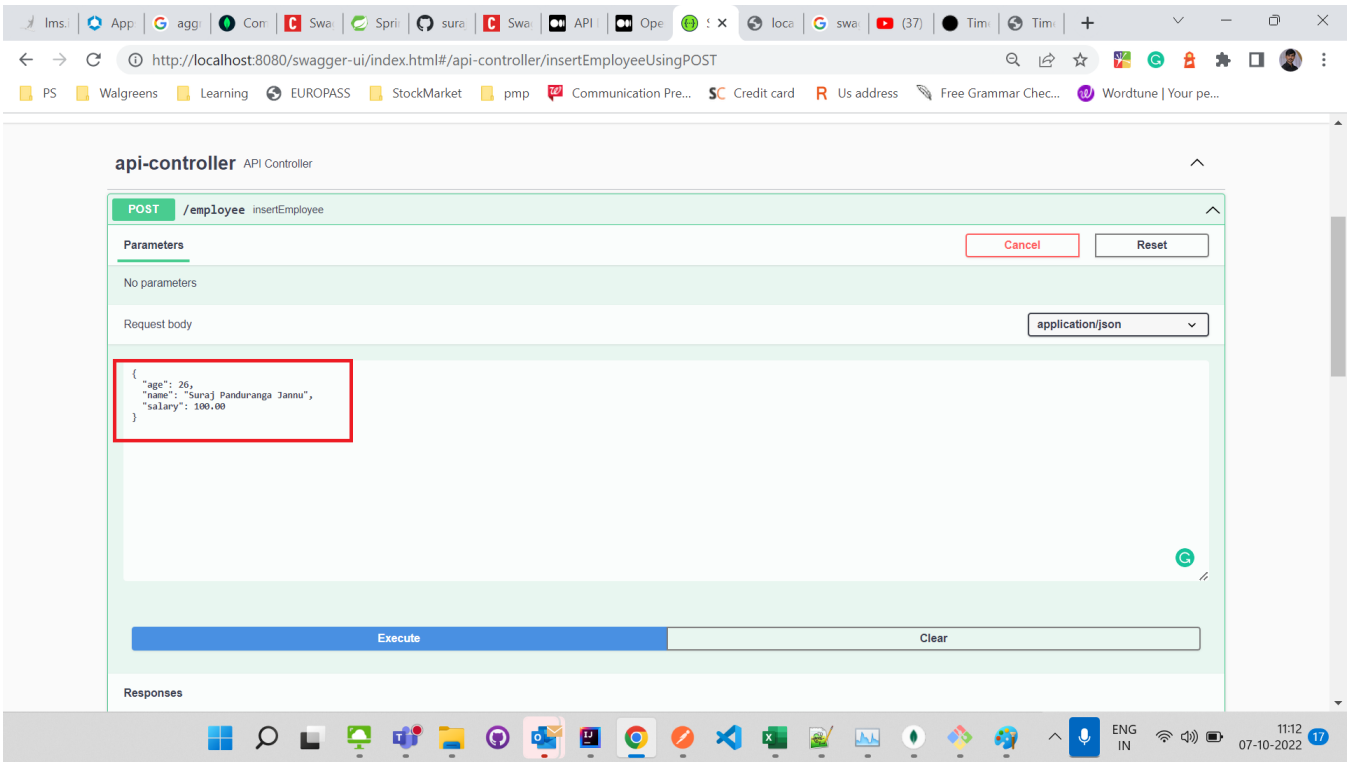
Step 7: Now let's test and verify each API.

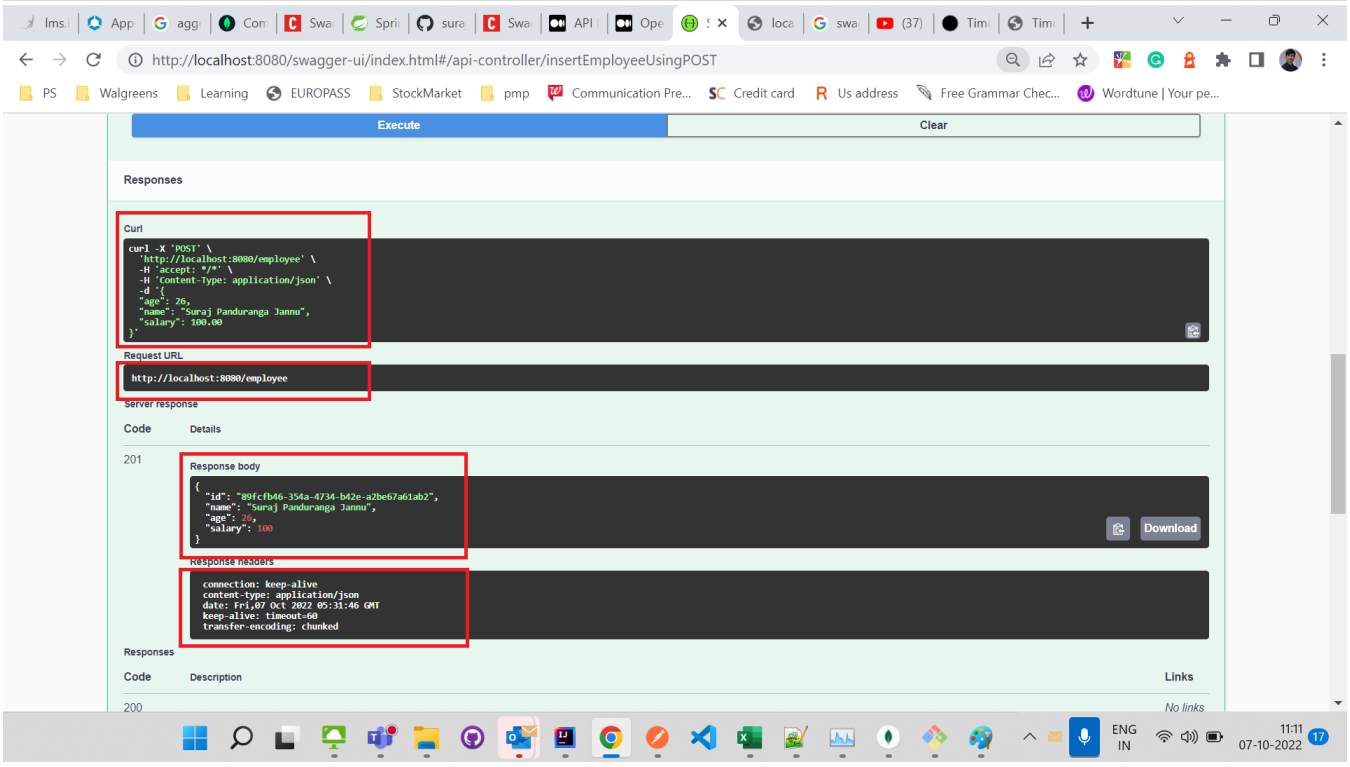
Let's see the documentation for insertEmployee API. The documentation shows how the requestBody should look like. What should be the response status and response body for the API.





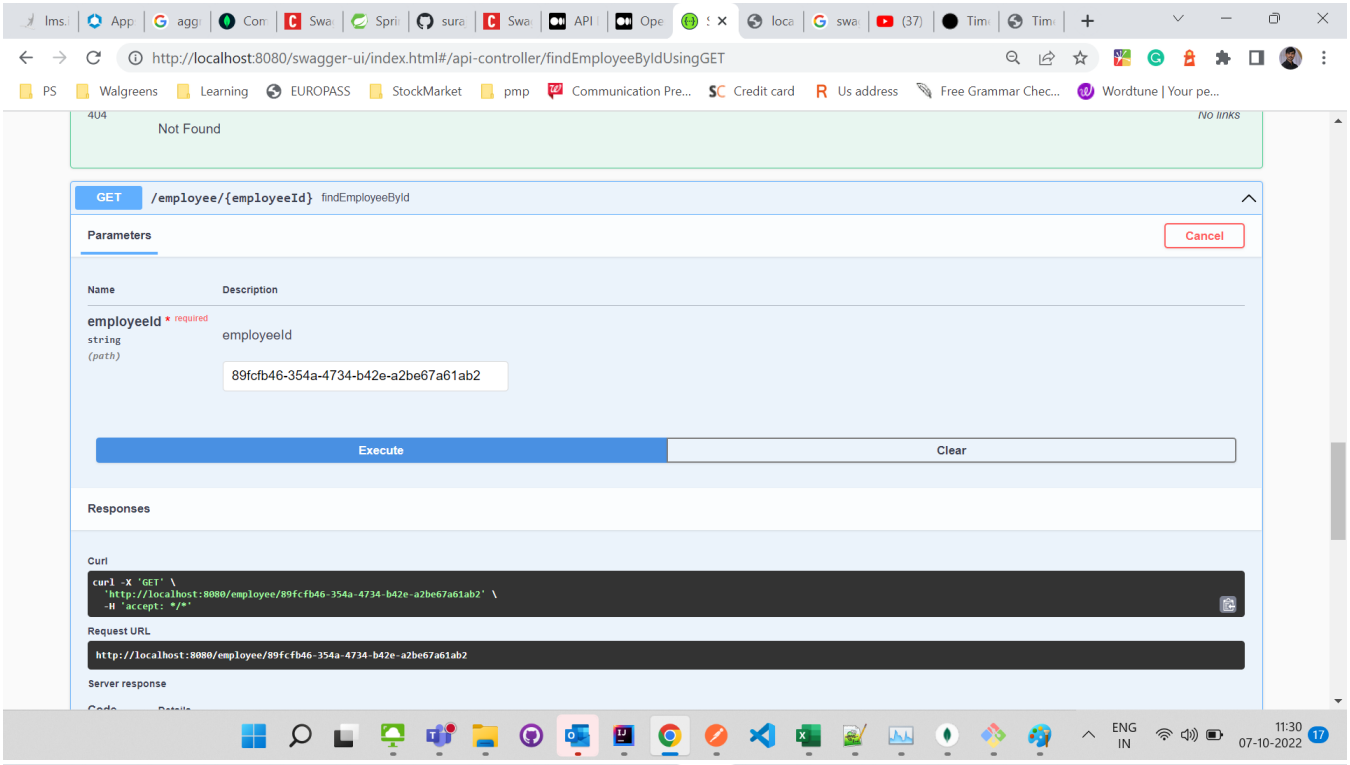
Let's test the API from the swagger-ui. Use **Try it out** Option.

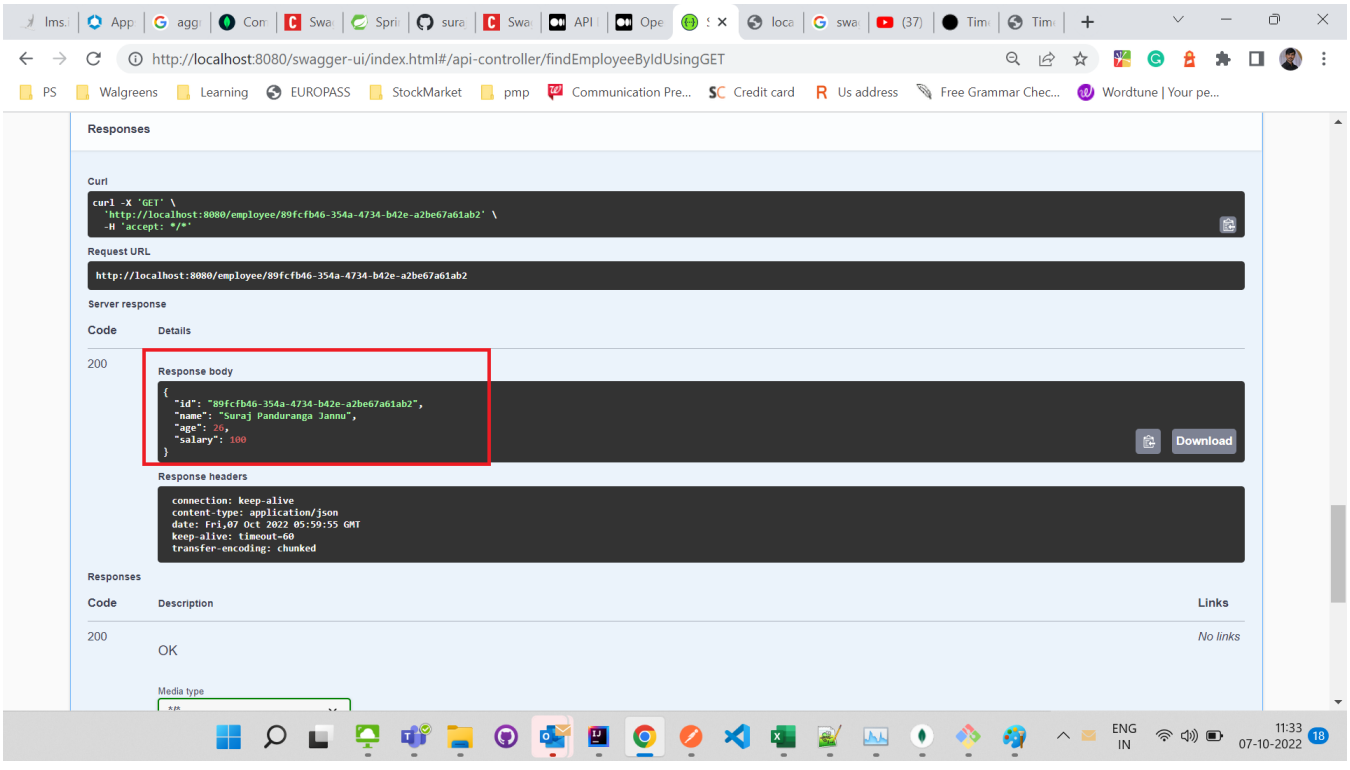




NOTE: You can see that swagger has made a CURL API call to our service and got the response (response body, response status code, and response headers).

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

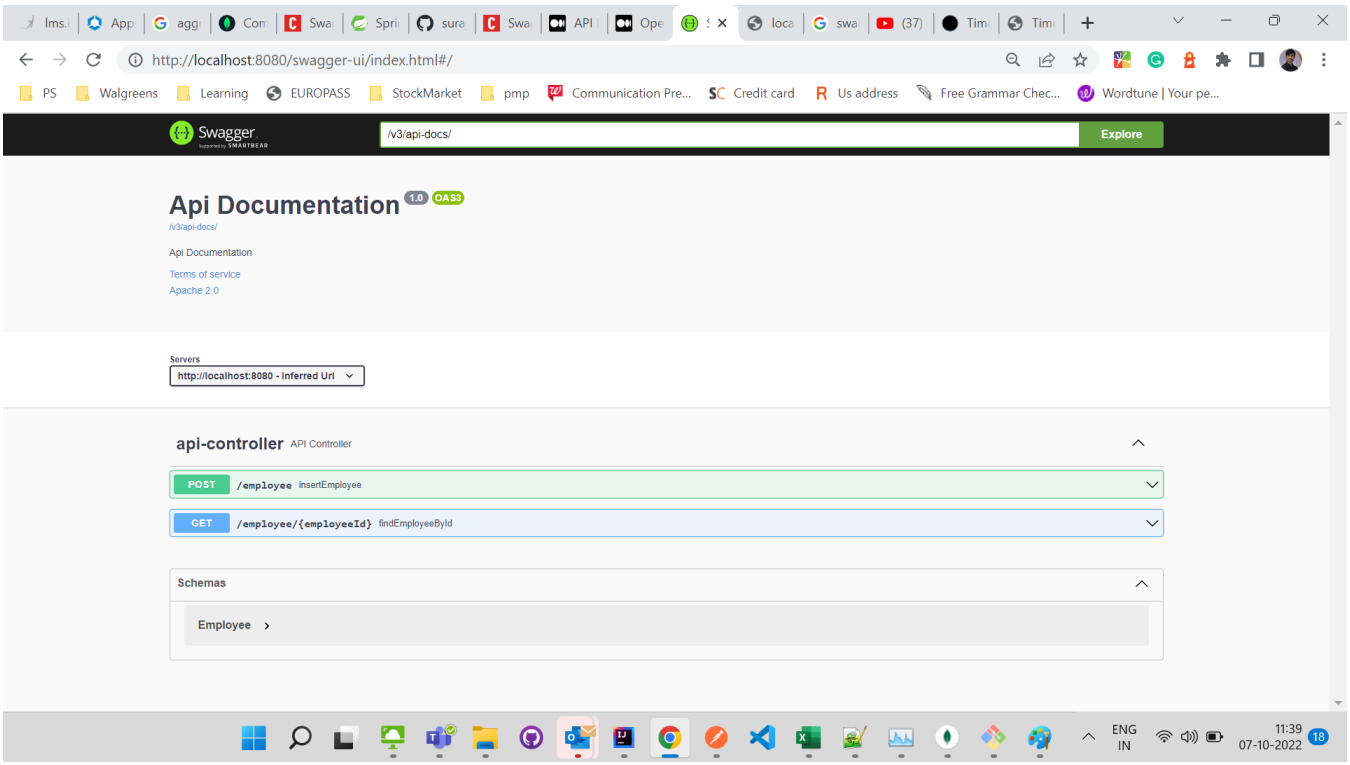




Step 8: We have learned what swagger documentation looks like and how to test from swagger UI. Now let's look at how can we customize the swagger documentation based on our needs.

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

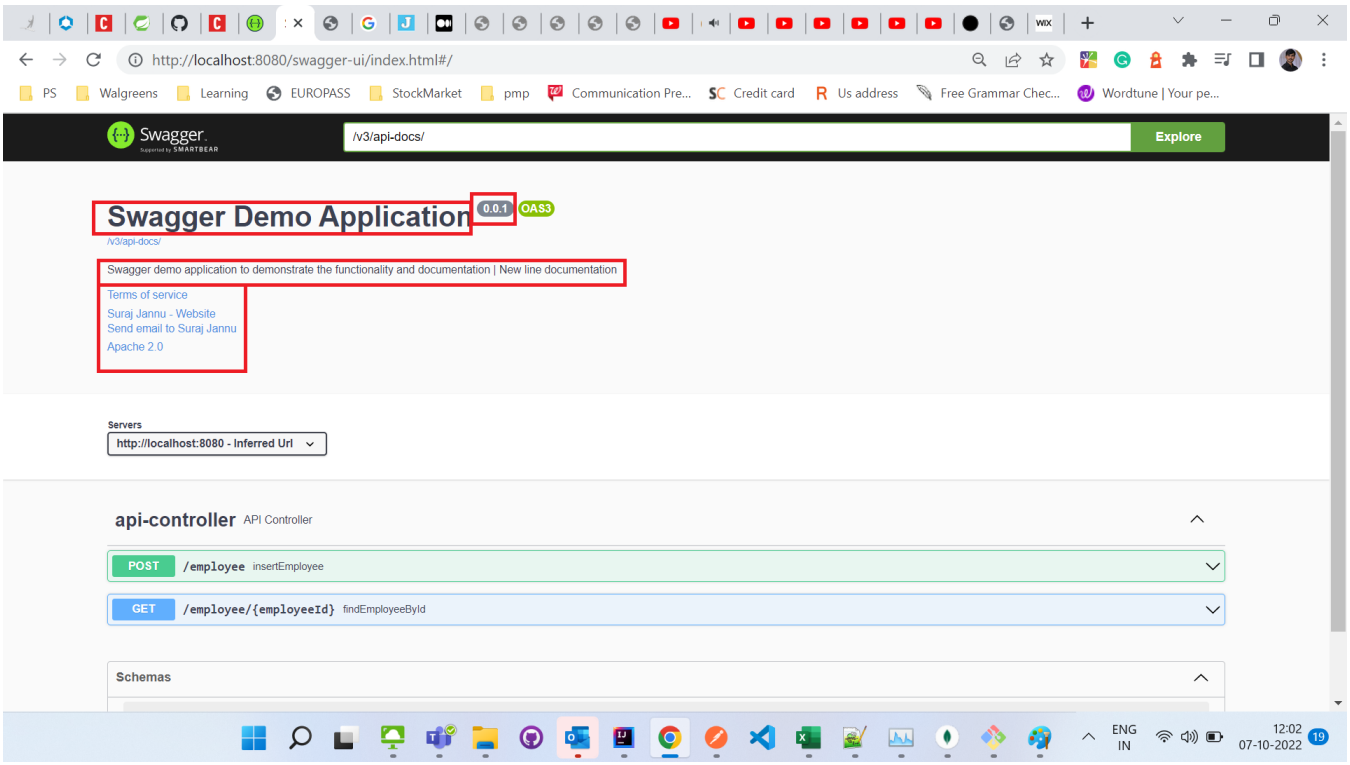
Before:



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_30)
            .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.



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.

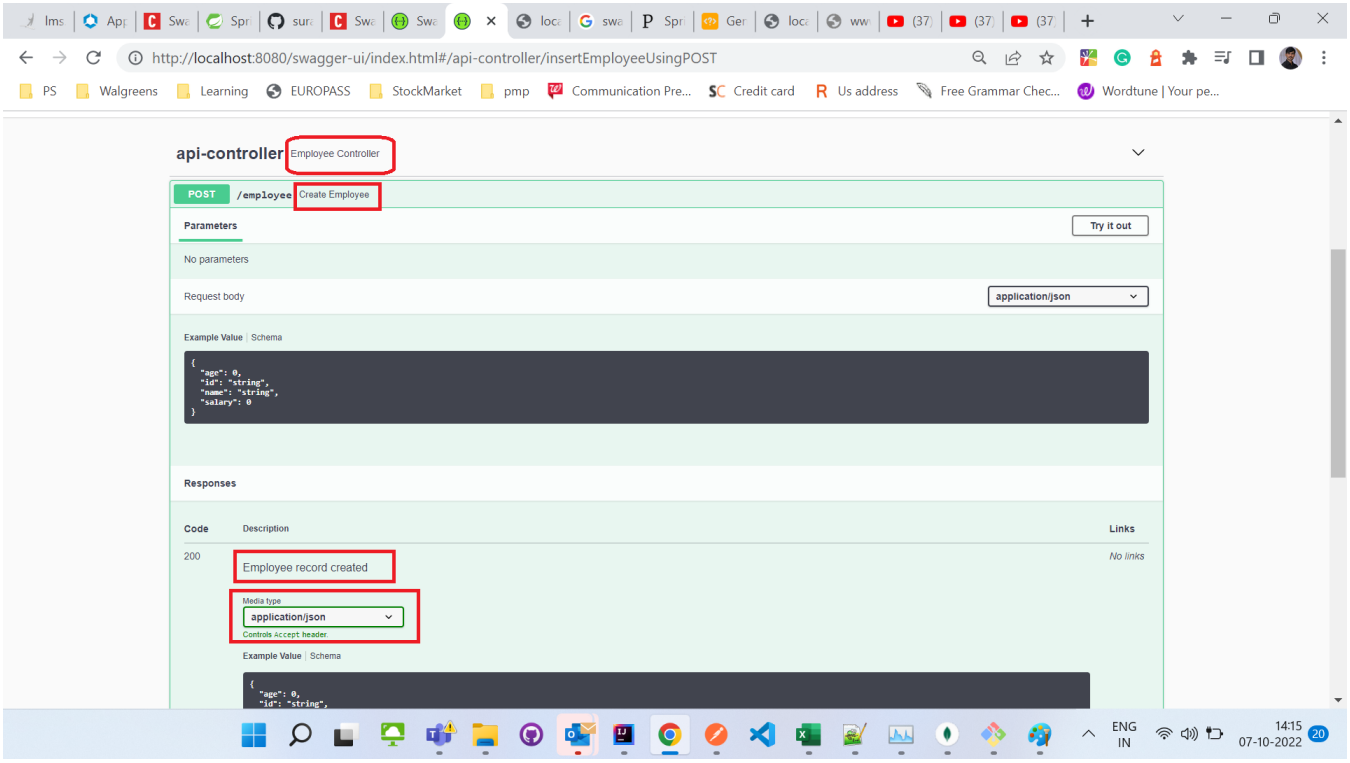
```
@ApiOperation(value = "Create Employee", response = Employee.class)
```

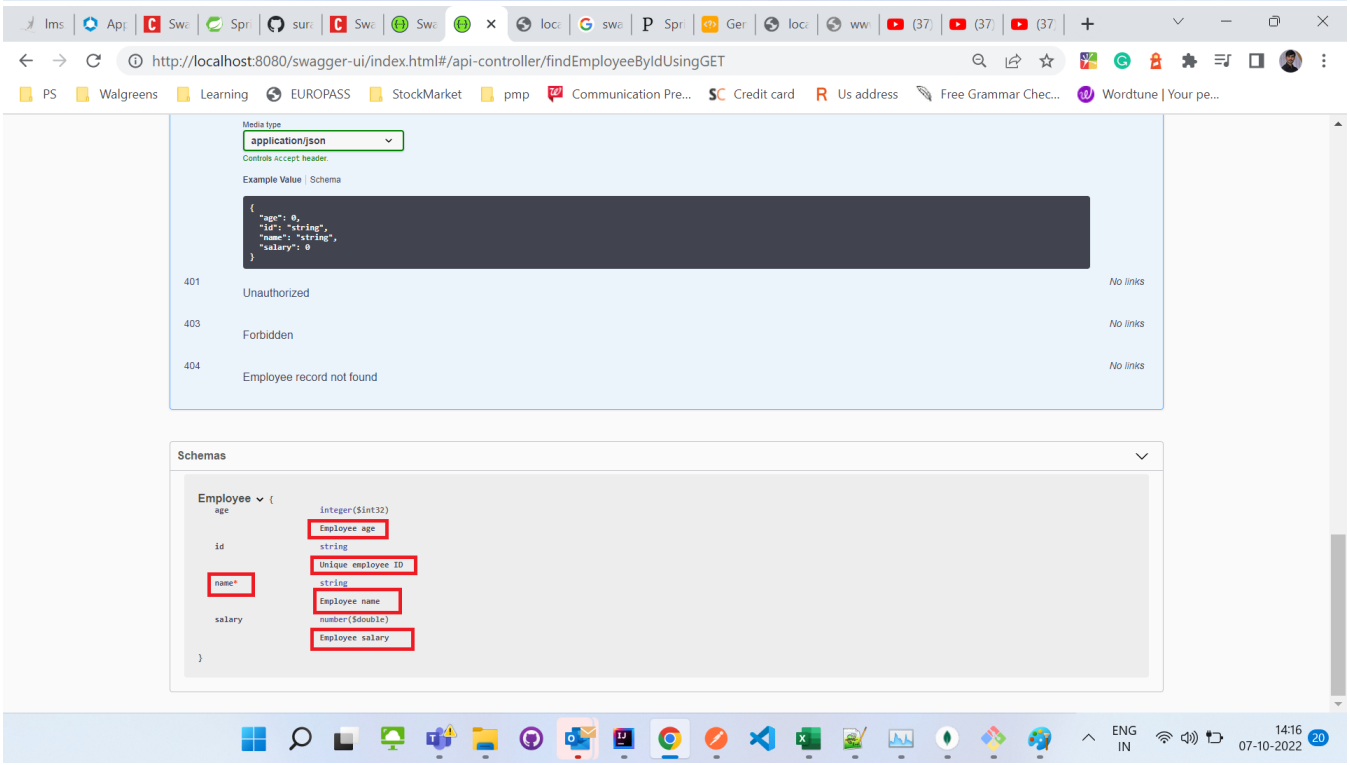
3. For mediaType support defined the produces property for API as shown below for JSON.

```
@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.

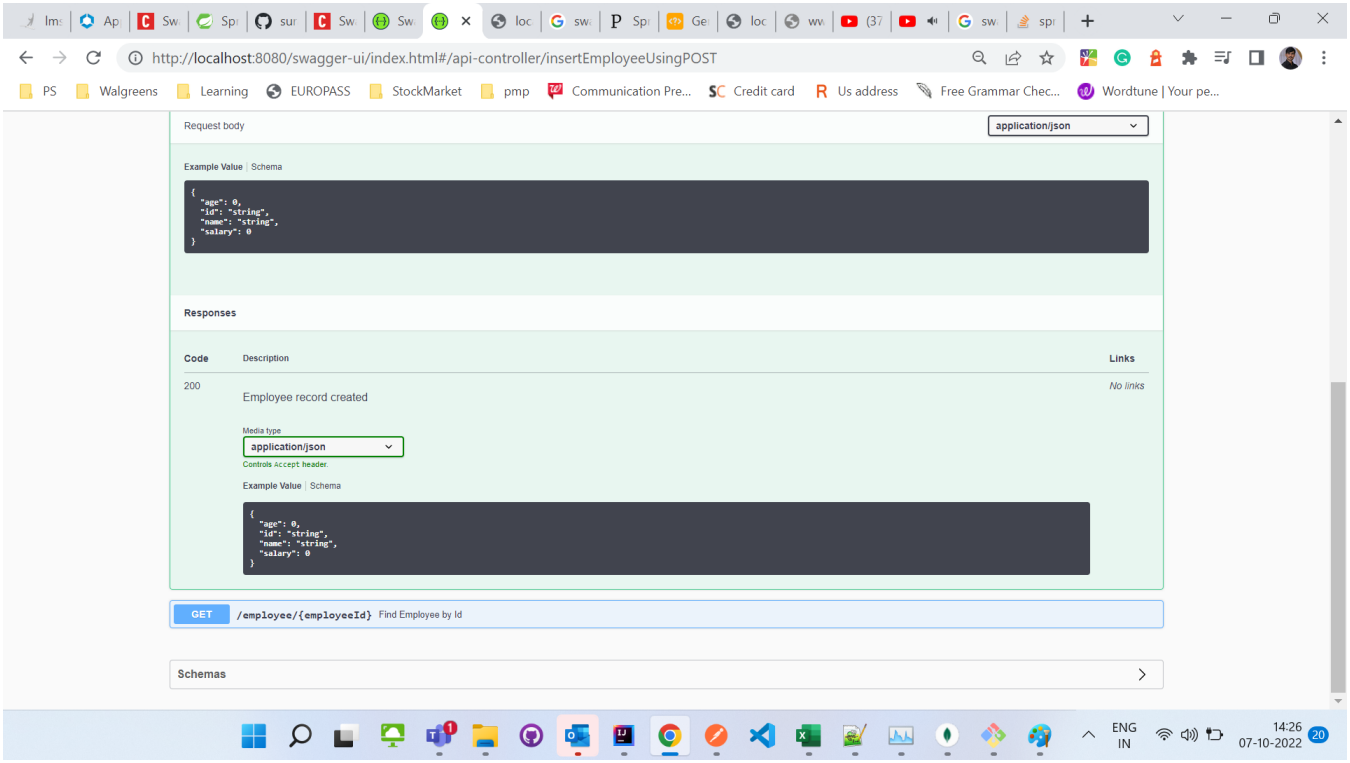
```
@ApiModelProperty(notes = "Employee name", required = true)
```



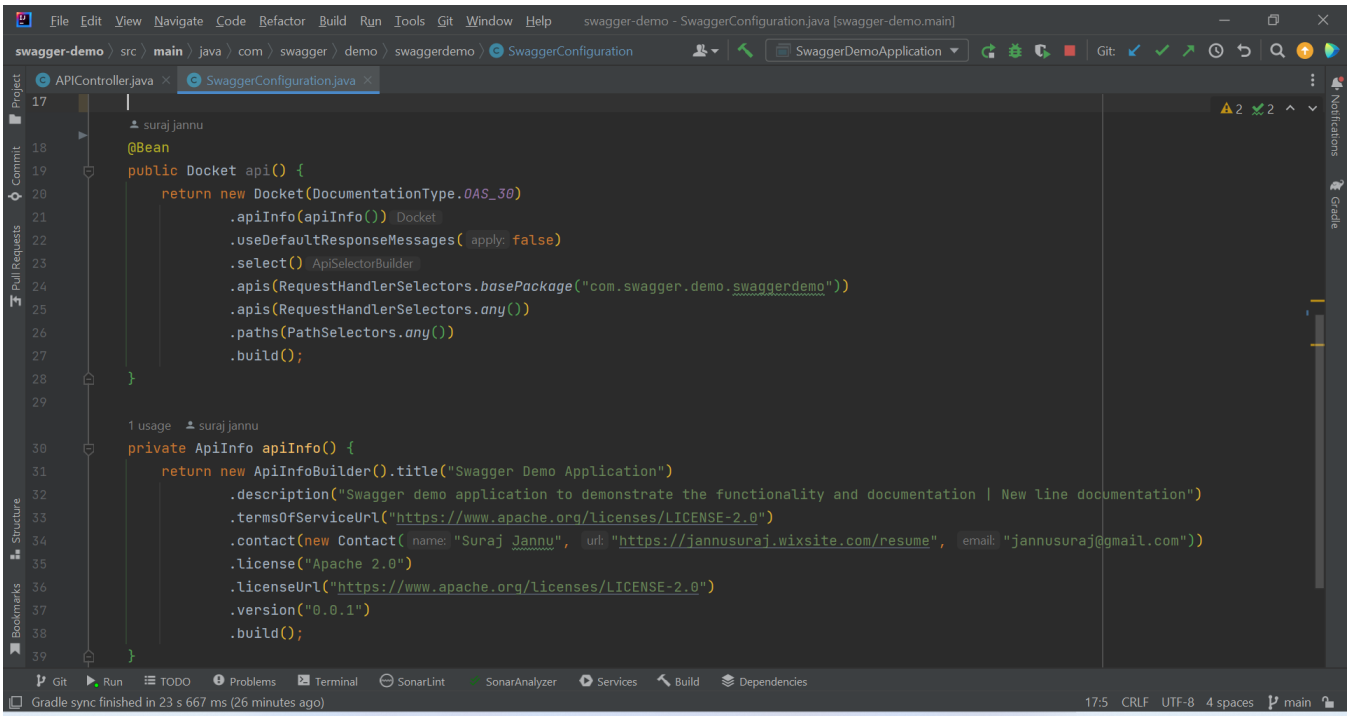


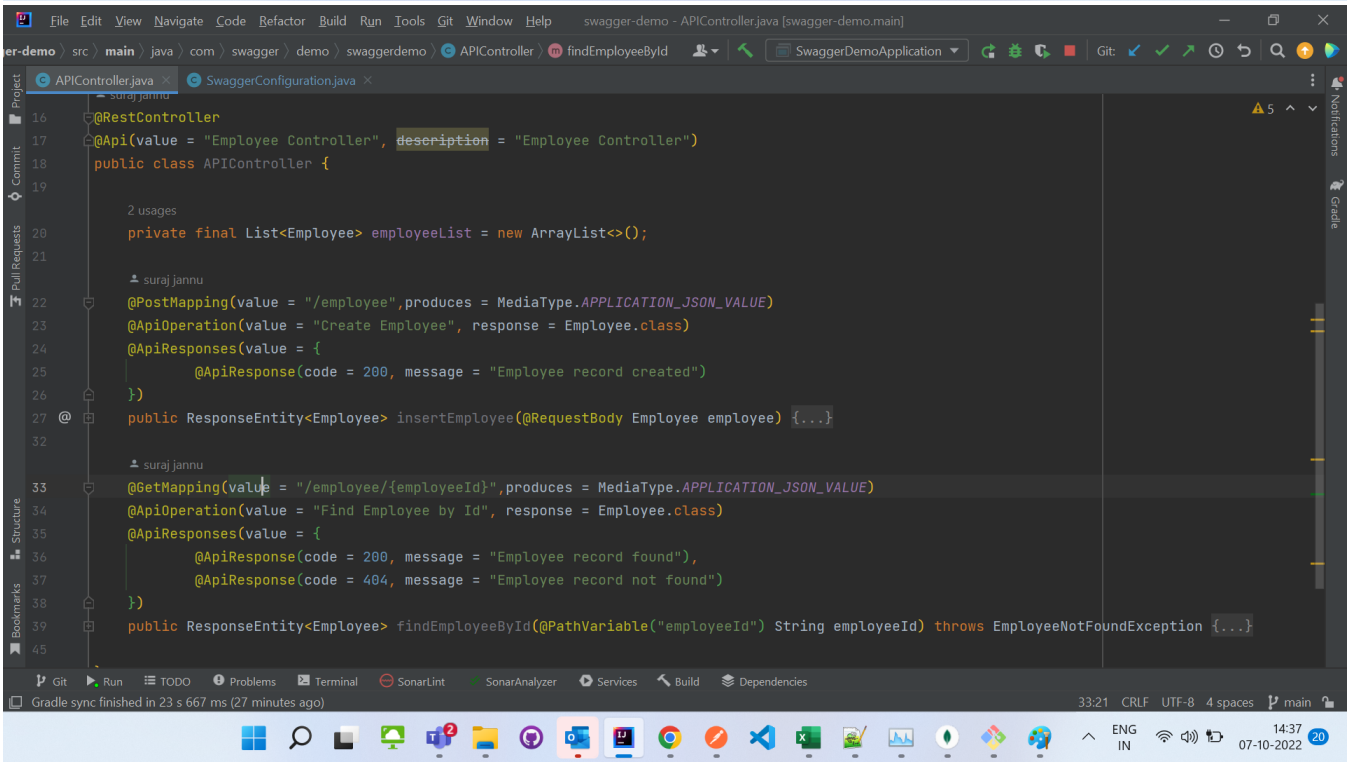
5. In order to remove the unwanted default response code use the **useDefaultResponseMessages** method.

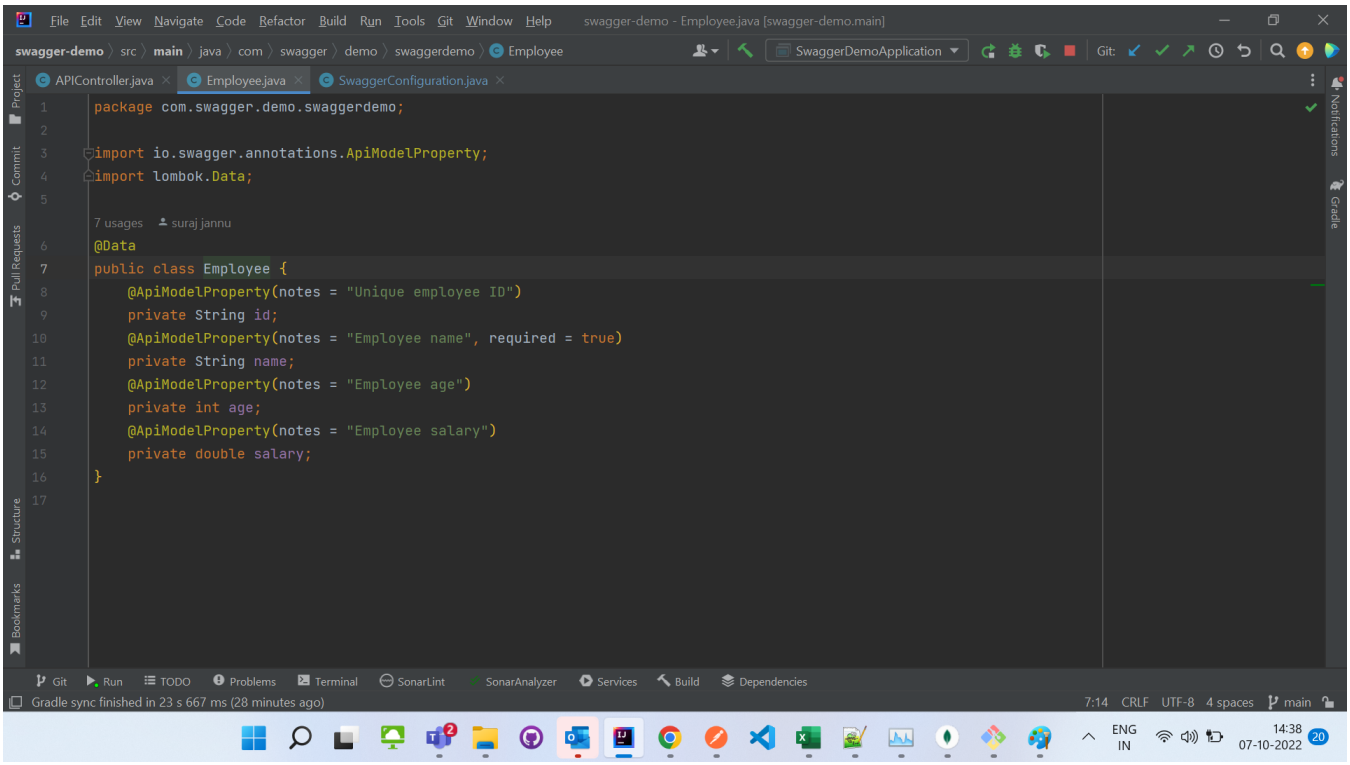
```java
@Bean
public Docket api() {
    return new Docket(DocumentationType.OAS_30)
            .apiInfo(apiInfo())
            .useDefaultResponseMessages(false)
            .select()
            .apis(RequestHandlerSelectors.basePackage("com.swagger.demo.swaggerdemo"))
            .apis(RequestHandlerSelectors.any())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



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
SwaggerDemoApplication
APIController.java
SwaggerConfiguration.java
suraj jannu
@Bean
public Docket api() {
return new Docket(DocumentationType.OAS_30)
.apiInfo(apiInfo())
.useDefaultResponseMessages(apply: false)
.select()
.apis(RequestHandlerSelectors.basePackage("com.swagger.demo.swaggerdemo"))
.apis(RequestHandlerSelectors.any())
.paths(PathSelectors.any())
.build();
}
1 usage suraj jannu
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(name: "Suraj Jannu", url: "https://jannusuraj.wixsite.com/resume", email: "jannusuraj@gmail.com"))
.license("Apache 2.0")
.licenseUrl("https://www.apache.org/licenses/LICENSE-2.0")
.version("0.0.1")
.build();
}
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
ENG IN
14:36 07-10-2022


File Edit View Navigate Code Refactor Build Run Tools Git Window Help
swagger-demo - APIController.java [swagger-demo.main]
er-demo › src › main › java › com › swagger › demo › swaggerdemo › APIController › findEmployeeById
SwaggerDemoApplication
APIController.java
SwaggerConfiguration.java
@RestController
@Api(value = "Employee Controller", description = "Employee Controller")
public class APIController {
2 usages
private final List<Employee> employeeList = new ArrayList<>();
suraj jannu
@PostMapping(value = "/employee",produces = MediaType.APPLICATION_JSON_VALUE)
@ApiOperation(value = "Create Employee", response = Employee.class)
@ApiResponses(value = {
@ApiResponse(code = 200, message = "Employee record created")
})
public ResponseEntity<Employee> insertEmployee(@RequestBody Employee employee) {...}
suraj jannu
@GetMapping(value = "/employee/{employeeId}",produces = MediaType.APPLICATION_JSON_VALUE)
@ApiOperation(value = "Find Employee by Id", response = Employee.class)
@ApiResponses(value = {
@ApiResponse(code = 200, message = "Employee record found"),
@ApiResponse(code = 404, message = "Employee record not found")
})
public ResponseEntity<Employee> findEmployeeById(@PathVariable("employeeId") String employeeId) throws EmployeeNotFoundException {...}
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
ENG IN
14:37 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
SwaggerDemoApplication
Git:
APIController.java
Employee.java
SwaggerConfiguration.java
package com.swagger.demo.swaggerdemo;
import io.swagger.annotations.ApiModelProperty;
import lombok.Data;
7 usages suraj jannu
@Data
public class Employee {
@ApiModelProperty(notes = "Unique employee ID")
private String id;
@ApiModelProperty(notes = "Employee name", required = true)
private String name;
@ApiModelProperty(notes = "Employee age")
private int age;
@ApiModelProperty(notes = "Employee salary")
private double salary;
}
Project
Commit
Pull Requests
Structure
Bookmarks
Notifications
Gradle
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
ENG IN
14:38
07-10-2022