CRUD CON SWAGGER

EDUARD LARA

INDICE

1. DESPLIEGUE DEL PROYECTO

Paso 1) Descargamos el proyecto springBootInitialDemo

IT-Academy-BCN / springBootInitialDemo Public

Notifications | Star 0 | Fork 7

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights

master | 1 branch | 0 tags | Go to file | Code

jonatanvicente Adding some explanations a13fd5a on 28 Jul 4 commits

gradle/wrapper	Initial project import	2 years ago
images	Adding some explanations	4 months ago
src/main/java/springBootInitialDemo	Initial project import	2 years ago
README.md	Adding some explanations	4 months ago
build.gradle	Initial project import	2 years ago
gradlew	Initial project import	2 years ago
gradlew.bat	Initial project import	2 years ago

About

No description, website, or topics provided.

Readme

Releases

No releases published

Packages

No packages published

Compartir con | Nueva carpeta

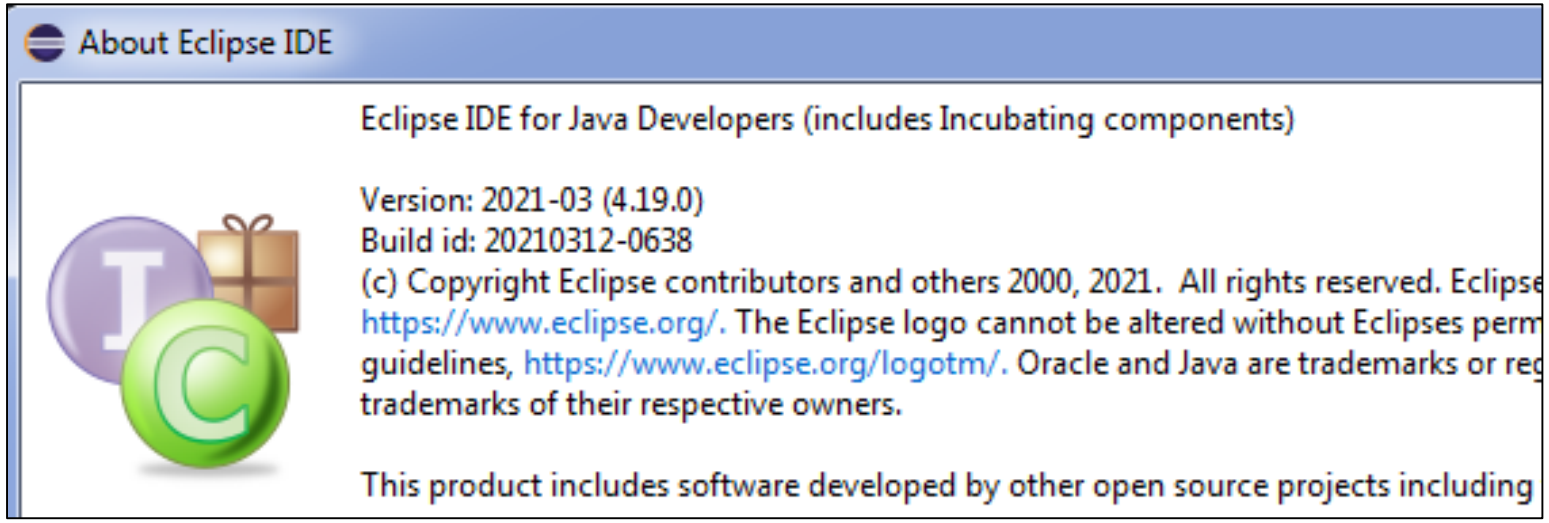
Nombre

springBootInitialDemo-master

springBootInitialDemo-master.zip

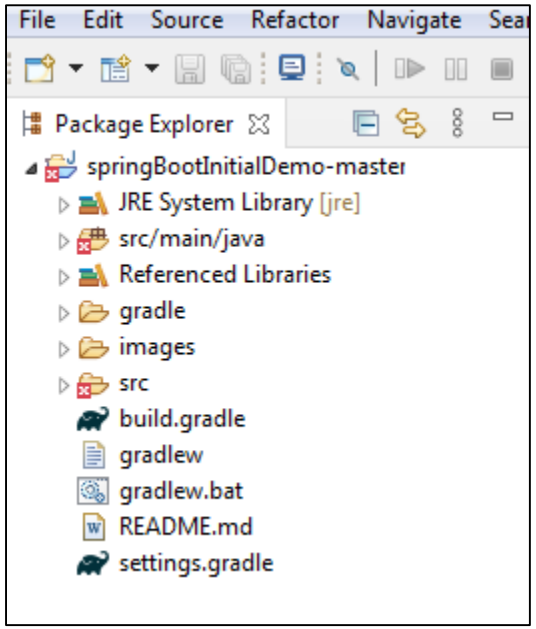
1. DESPLIEGUE DEL PROYECTO

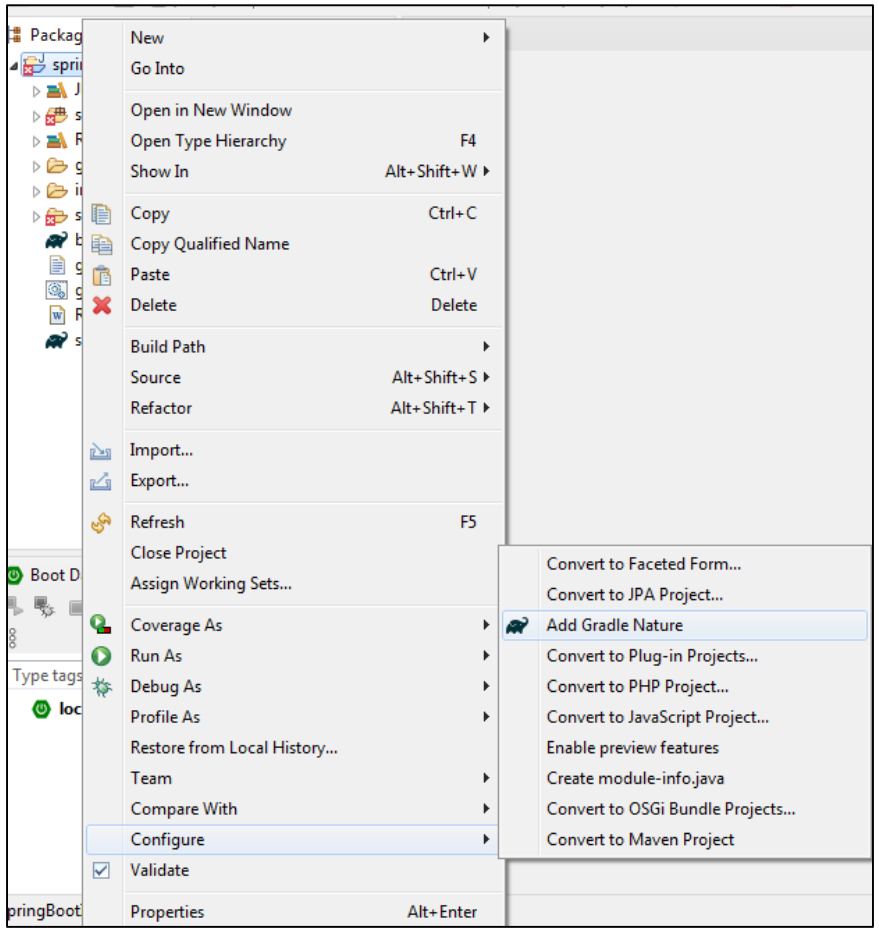
Paso 2) La compilación y ejecución de springBootInitialDemo no presenta problemas con IntelliJ, pero si con eclipse. Parece que las versiones de eclipse de 06/2021 y 09/2021 no entienden el gradle del proyecto. Además se constata que eclipse anda peleado en general con el repositorio gradle. En todo caso si se desea compilar este proyecto con eclipse se debe de utilizar la versión 03/2021.

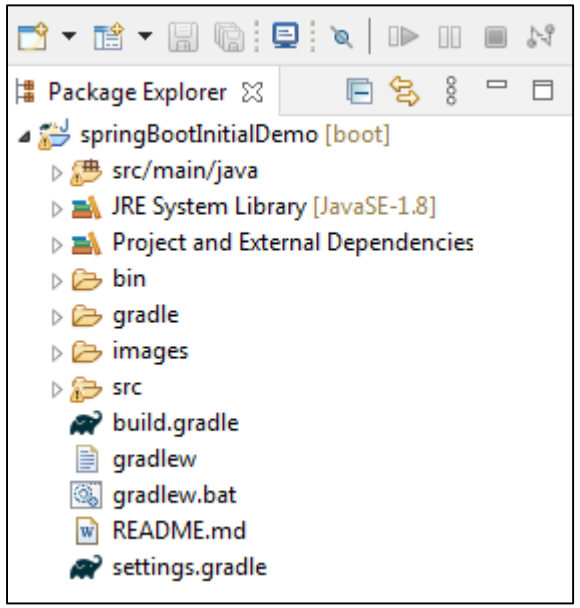


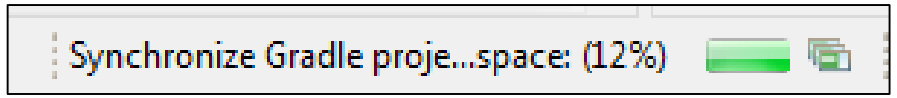
1. DESPLIEGUE DEL PROYECTO

Paso 3) Al abrir el proyecto observamos que no aparece la clausula [boot] al lado del nombre del proyecto. Debemos ir a Configure/Add Gradle Nature para forzar que eclipse reconozca el proyecto como Spring Boot.





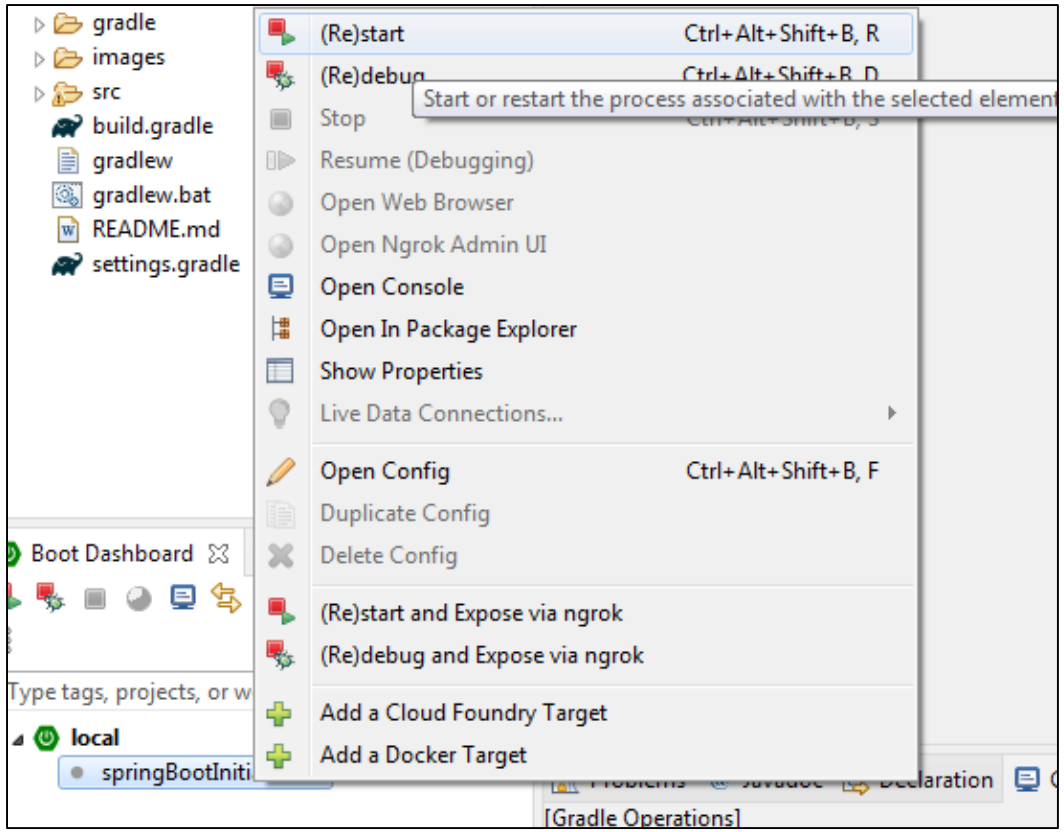


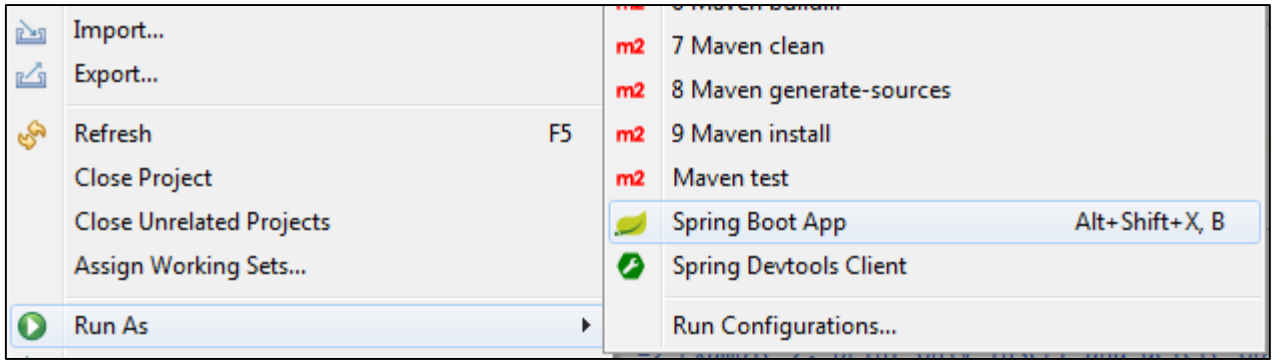


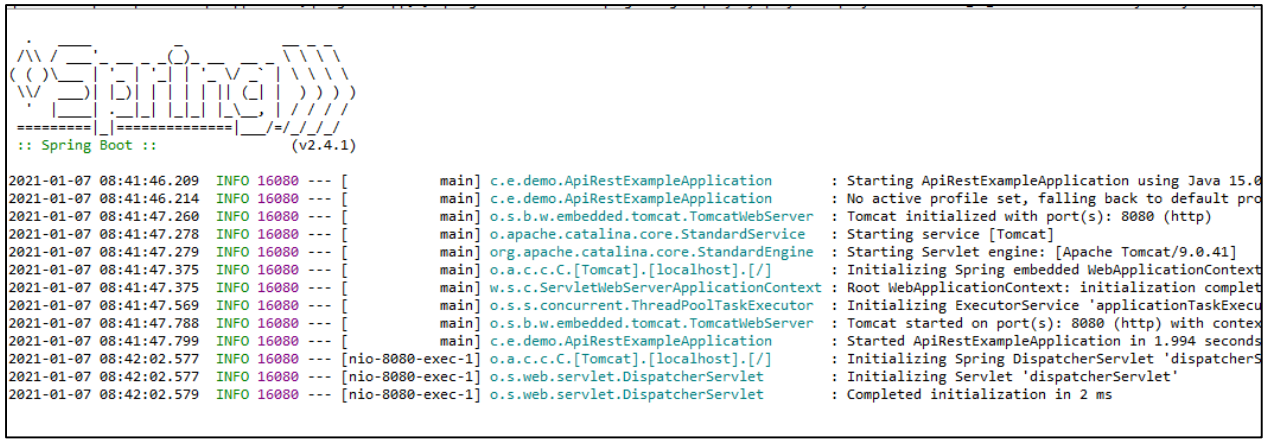
1. DESPLIEGUE DEL PROYECTO

Paso 4) Probamos de ejecutar el proyecto, para ello tenemos dos maneras:

- Levantamos el servidor Tomcat haciendo Run As/Spring Boot App.
- Haciendo click botón derecho sobre el proyecto en Dashboard + ReStart.







2. TESTING

Paso 1) Una vez vemos que ha arrancado correctamente el servidor, vamos a un navegador y ponemos **localhost:8080/v1/test**

InitialController.java

```java
1  package springBootInitialDemo.controller;
2
3  import org.springframework.beans.factory.annotation.Autowired;
9
10 @RestController
11 @RequestMapping("/v1")
12 public class InitialController {
13
14     private final IUserService userService;
15
16     @Autowired
17     public InitialController(IUserService userService){
18         this.userService = userService;
19     }
20
21     @GetMapping("/test")
22     public String helloGradle() {
23         return "Hello Gradle!";
24     }
```

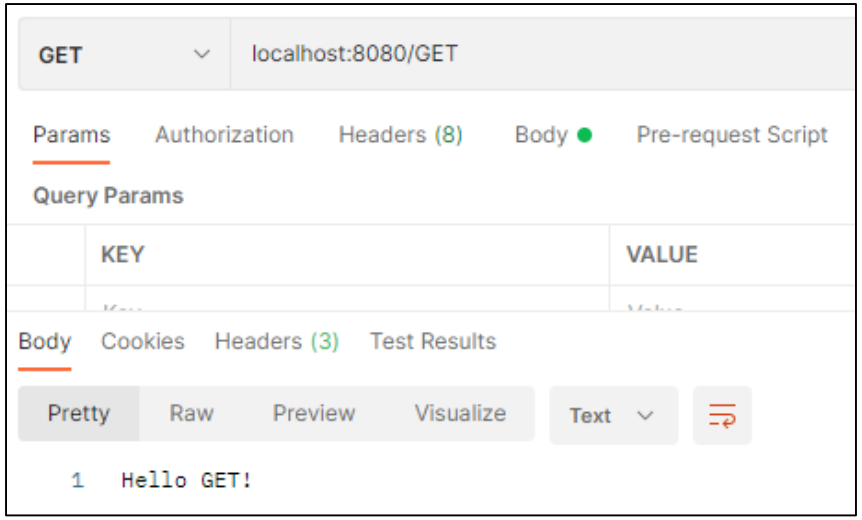
2. TESTING

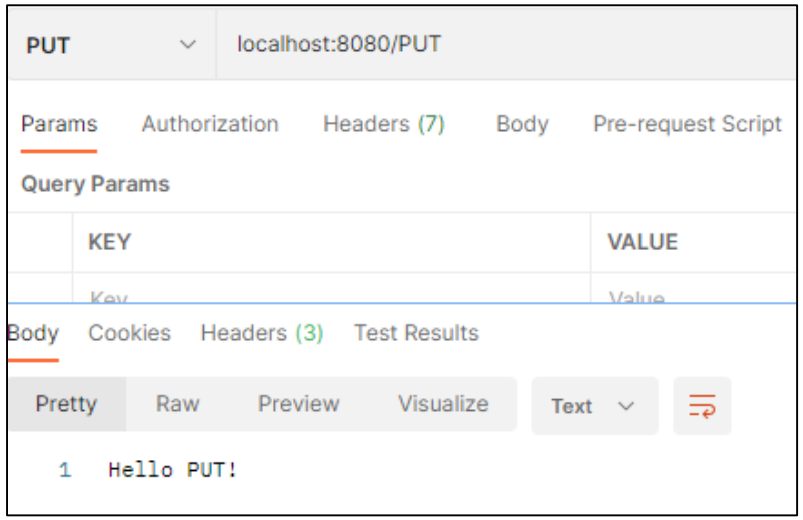
Paso 2) Añadiremos 3 endpoints REST: PUT, POST DELETE. Redefiniremos la url de inicio del controlador para simplificar:

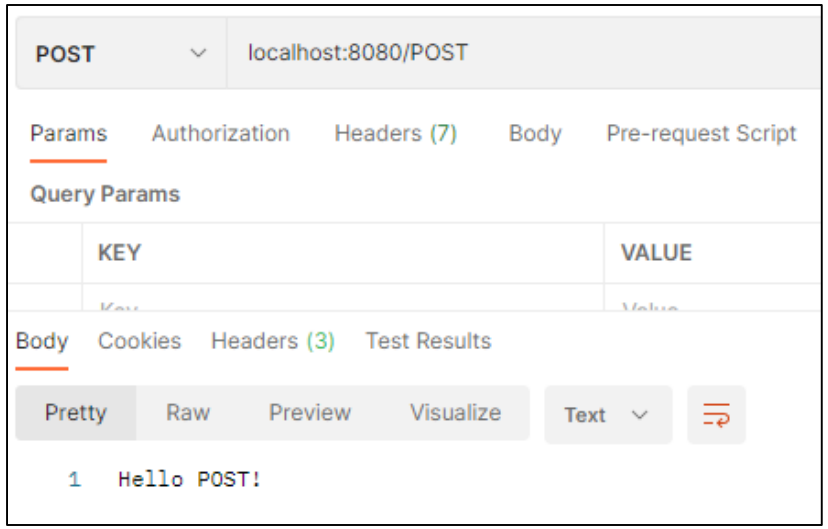
```
@RestController
@RequestMapping("/")
public class InitialController {
    private final IUserService userService;

    @Autowired
    public InitialController(IUserService userService){
        this.userService = userService;
    }
    @GetMapping("/GET")
    public String helloGET() {
        return "Hello GET!";
    }
    @PutMapping("/PUT")
    public String helloPUT() {
        return "Hello PUT!";
    }
    @DeleteMapping("/DELETE")
    public String helloDELETE() {
        return "Hello DELETE!";
    }
    @PostMapping("/POST")
    public String helloPOST() {
        return "Hello POST!";
    }
```

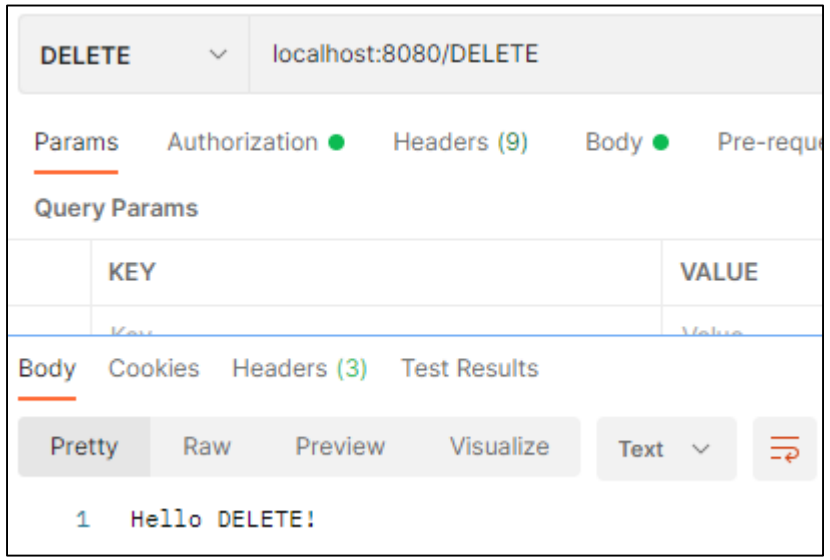
2. TESTING

Paso 3) Los probamos con Postman.









3. SWAGGER

Swagger es un conjunto de herramientas de software de código abierto para diseñar, construir, documentar, y utilizar servicios web RESTful.

Fue desarrollado por SmartBear Software e incluye documentación automatizada, generación de código, y generación de casos de prueba.

Las librerías de Swagger se utilizan para documentar de forma automática los microservicios a partir de los metadatos existentes en nuestro controlador: @DeleteMapping, @PutMapping, etc.

Permite documentar cada microservicio junto con los parámetros utilizados y su URL. De esta manera esta se puede informar correctamente a un desarrollador que no esta en nuestro equipo.

3. SWAGGER

Paso 1) Primero de todo debemos agregar las librerías de swagger al fichero build.gradle:

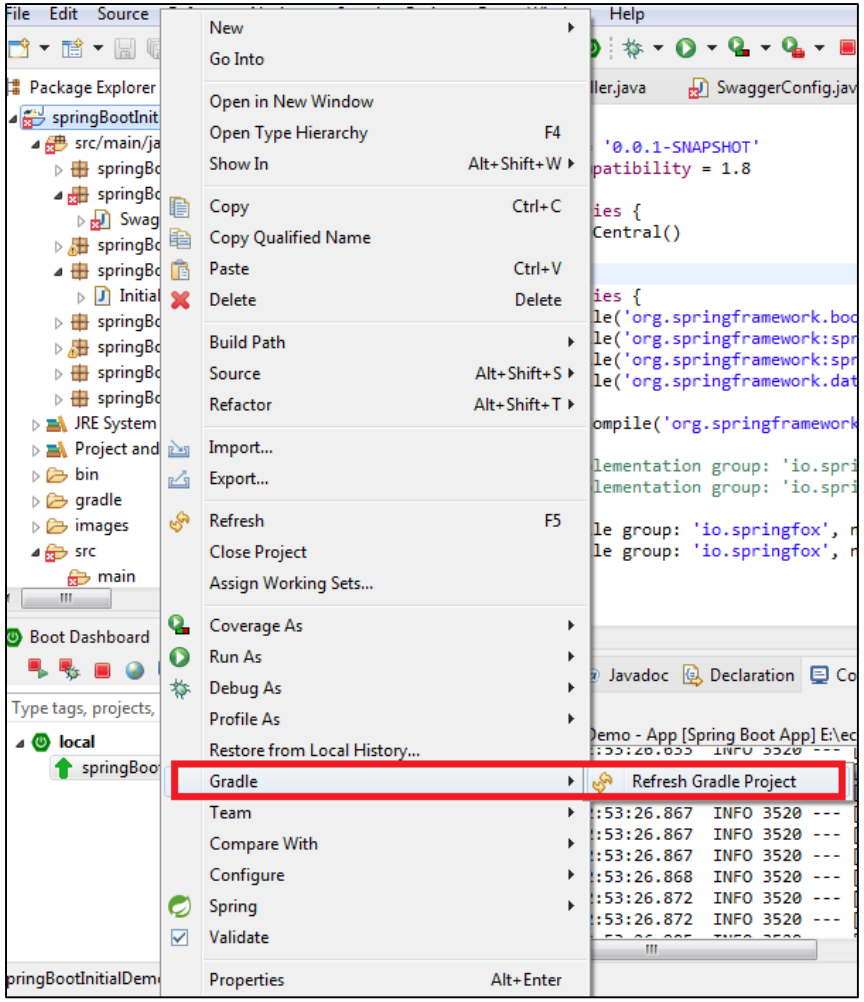
build.gradle

```
28
29 dependencies {
30     compile('org.springframework.boot:spring-boot-starter-web')
31     compile('org.springframework:spring-tx')
32     compile('org.springframework:spring-webmvc')
33     compile('org.springframework.data:spring-data-jpa')
34
35     //io.springfox >= 2.X
36     compile group: 'io.springfox', name: 'springfox-swagger2', version: '2.9.2'
37     compile group: 'io.springfox', name: 'springfox-swagger-ui', version: '2.9.2'
38
39     //io.springfox >= 3.X
40     //compile group: 'io.springfox', name: 'springfox-boot-starter', version: '3.0.0'
41
42     testCompile('org.springframework.boot:spring-boot-starter-test')
43 }
44
```

Cuidado con las versiones!!! SpringBootInitialDemo solo funciona con las versiones de Swagger inferiores a 3.0.0. Eso es debido a que utiliza la versión de Spring 2.0.2

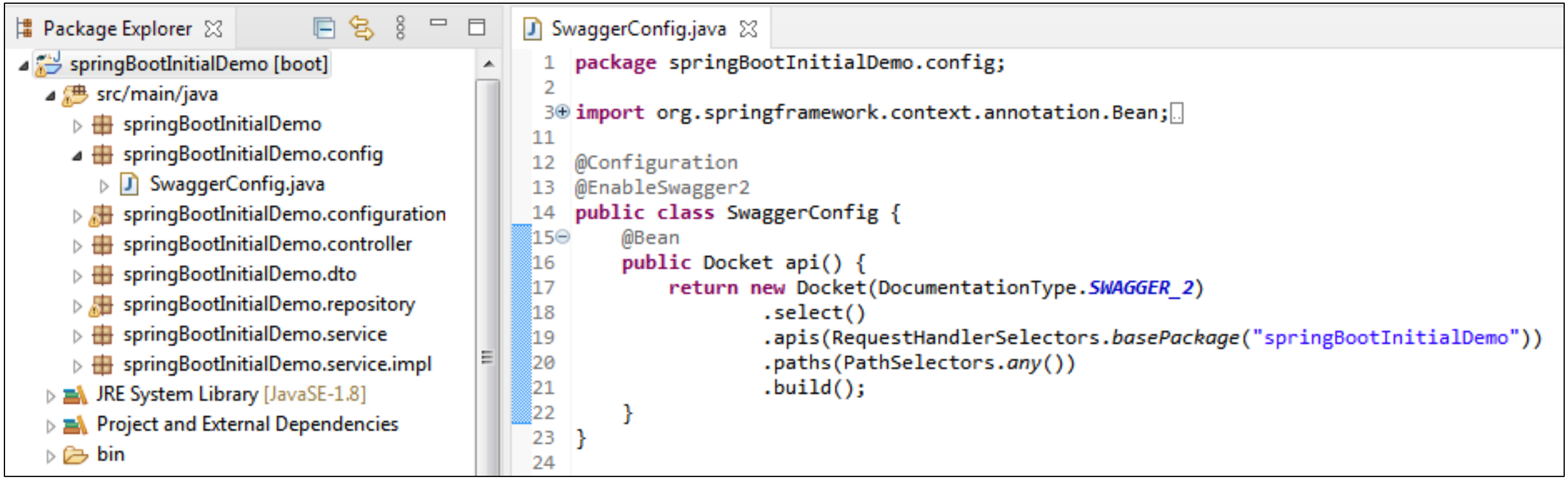
3. SWAGGER

Paso 2) Debemos refrescar las librerías gradle para forzar la descarga a nuestro equipo local. Hacemos click botón derecho encima del proyecto, y seleccionamos Gradle + Refresh Gradle Project:



3. SWAGGER

Paso 3) Creamos el package springBootInitialDemo.config y dentro el fichero SwaggerConfig con la configuración para activar Swagger:



3. SWAGGER

Paso 4) Levantamos la aplicación. Nos indica que en **localhost:8080/v2/api-docs** se ha generado la documentación de los endpoints:

localhost:8080/v2/api-docs


```
{"swagger":"2.0","info":{"description":"Api Documentation","version":"1.0","title":"Api Documentation","termsOfService":"urn:tos","contact":{},"license":{"name":"Apache
2.0","url":"http://www.apache.org/licenses/LICENSE-2.0"}},"host":"localhost:8080","basePath":"/","tags":[{"name":"initial-controller","description":"Initial Controller"}],"paths":
{"/DELETE":{"delete":{"tags":["initial-controller"],"summary":"helloDELETE","operationId":"helloDELETEUsingDELETE","produces":["*/*"],"responses":{"200":{"description":"OK","schema"
{"type":"string"}},"204":{"description":"No Content"},"401":{"description":"Unauthorized"},"403":{"description":"Forbidden"}},"deprecated":false}},"/GET":{"get":{"tags":["initial-
controller"],"summary":"helloGET","operationId":"helloGETUsingGET","produces":["*/*"],"responses":{"200":{"description":"OK","schema":{"type":"string"}},"401":
{"description":"Unauthorized"},"403":{"description":"Forbidden"},"404":{"description":"Not Found"}},"deprecated":false}},"/POST":{"post":{"tags":["initial-
controller"],"summary":"helloPOST","operationId":"helloPOSTUsingPOST","consumes":["application/json"],"produces":["*/*"],"responses":{"200":{"description":"OK","schema":
{"type":"string"}},"201":{"description":"Created"},"401":{"description":"Unauthorized"},"403":{"description":"Forbidden"},"404":{"description":"Not
Found"}},"deprecated":false}},"/PUT":{"put":{"tags":["initial-controller"],"summary":"helloPUT","operationId":"helloPUTUsingPUT","consumes":["application/json"],"produces":
["*/*"],"responses":{"200":{"description":"OK","schema":{"type":"string"}},"201":{"description":"Created"},"401":{"description":"Unauthorized"},"403":
{"description":"Forbidden"},"404":{"description":"Not Found"}},"deprecated":false}},"/user/{uuid}":{"get":{"tags":["initial-
controller"],"summary":"updatePrescription","operationId":"updatePrescriptionUsingGET","produces":["*/*"],"parameters":
[{"name":"uuid","in":"path","description":"uuid","required":true,"type":"string"}],"responses":{"200":{"description":"OK","schema":{"$ref":"#/definitions/UserResponseDto"}},"401":
{"description":"Unauthorized"},"403":{"description":"Forbidden"},"404":{"description":"Not Found"}},"deprecated":false}}},"definitions":{"UserResponseDto":
{"type":"object","properties":{"dateOfBirth":{"type":"string"},"gender":{"type":"string"},"name":{"type":"string"},"surname":{"type":"string"}},"title":"UserResponseDto"}}}
```

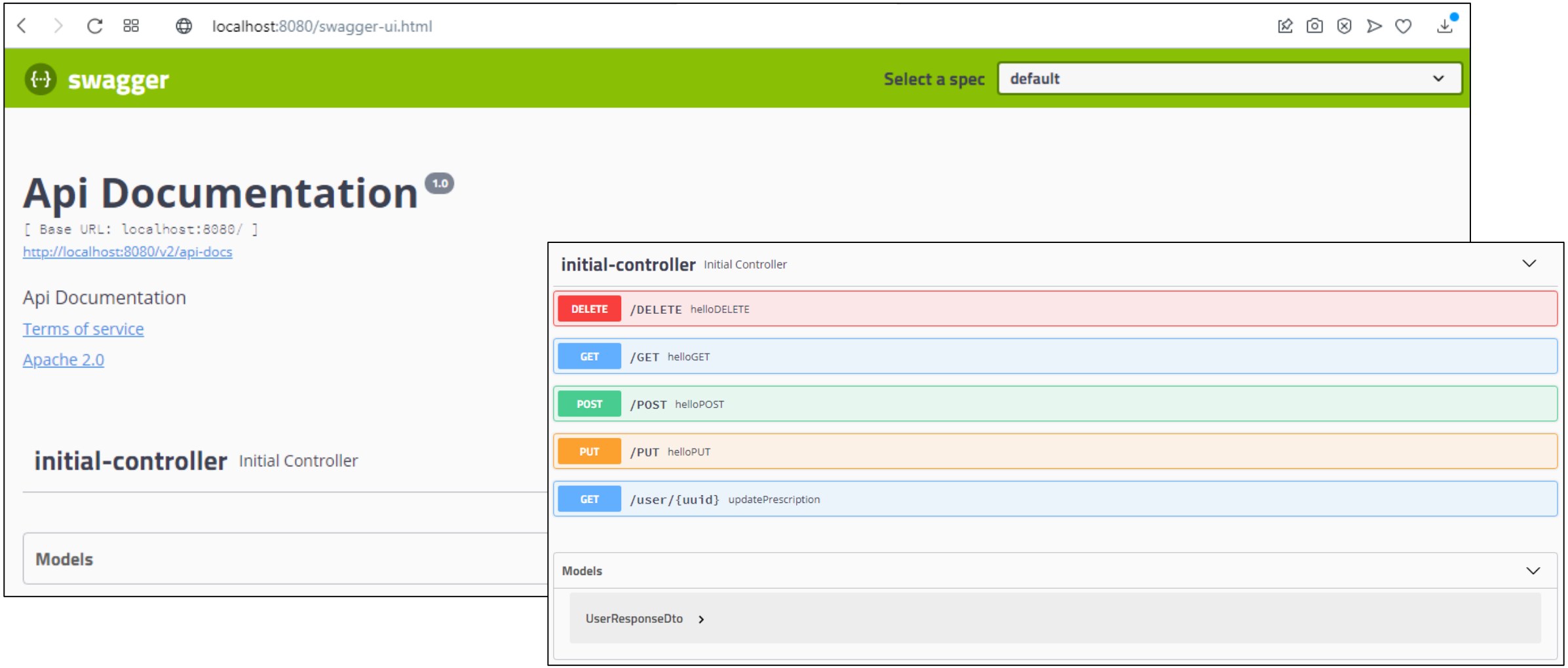

Problems | Javadoc | Declaration | Console

springBootInitialDemo - App [Spring Boot App] C:\Program Files\Java\jdk1.8.0_202\bin\javaw.exe (21 nov. 2021 17:54:56)

```
2021-11-21 17:55:03.691  INFO 7352 --- [          main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/user/{uuid}],methods=[GET]}" onto public org.springf
2021-11-21 17:55:03.700  INFO 7352 --- [          main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/swagger-resources/configuration/security]}" onto pub
2021-11-21 17:55:03.702  INFO 7352 --- [          main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/swagger-resources/configuration/ui]}" onto public org
2021-11-21 17:55:03.703  INFO 7352 --- [          main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/swagger-resources]}" onto public org.springframework
2021-11-21 17:55:03.711  INFO 7352 --- [          main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error]}" onto public org.springframework.http.Respon
2021-11-21 17:55:03.712  INFO 7352 --- [          main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error],produces=[text/html]}" onto public org.spring
2021-11-21 17:55:03.883  INFO 7352 --- [          main] pertySourcedRequestMappingHandlerMapping : Mapped URL path [/v2/api-docs] onto method [public org.springfr
2021-11-21 17:55:03.975  INFO 7352 --- [          main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/**] onto handler of type [class org.springfra
2021-11-21 17:55:04.190  INFO 7352 --- [          main] s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking for @ControllerAdvice: org.springframework.boot.web.ser
```

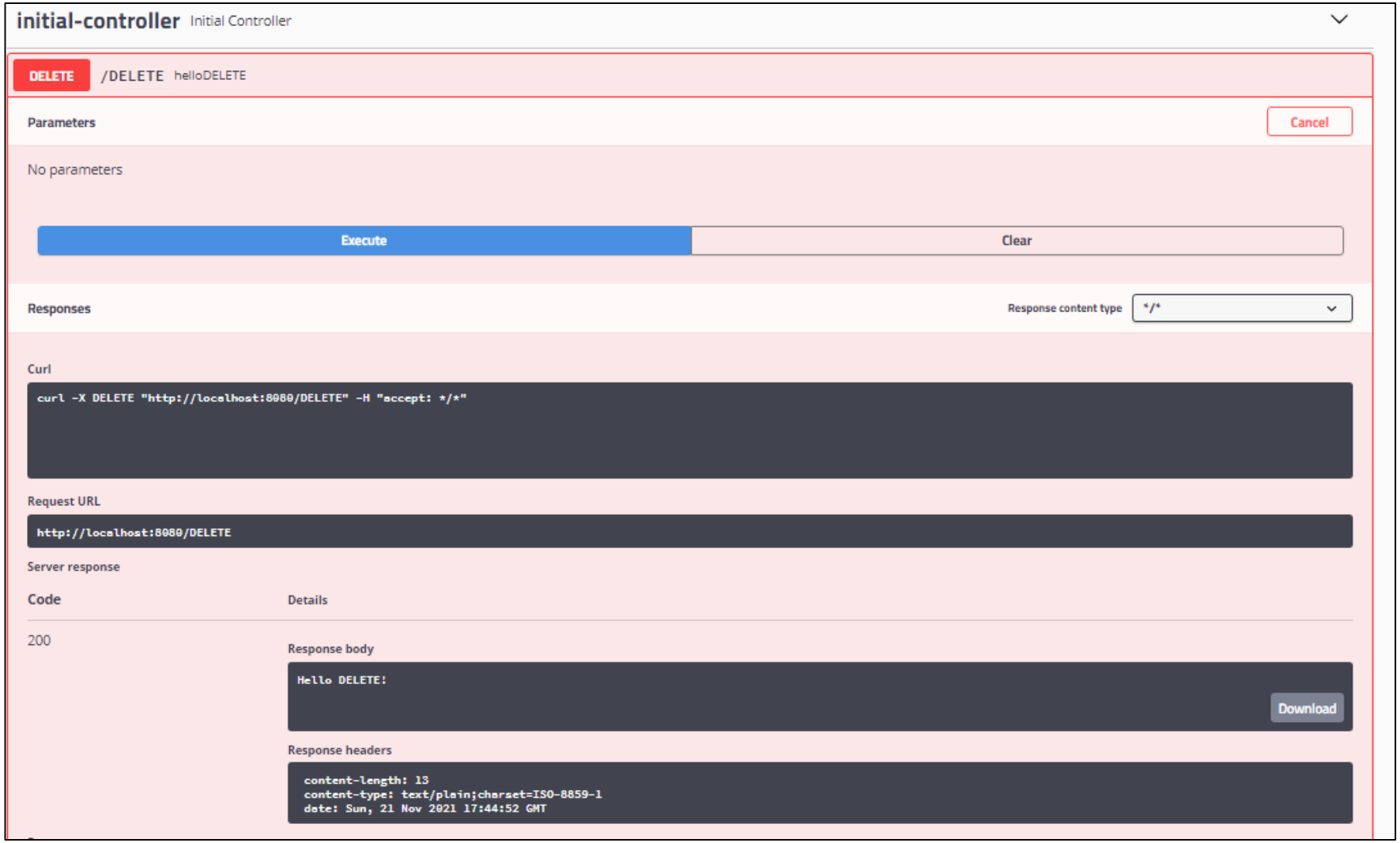
3. SWAGGER

Paso 5) En la url http://localhost:8080/swagger-ui.html tenemos una interfaz grafica para testear los endpoints:



3. SWAGGER

Paso 6) Nos permite ejecutar los endpoints:



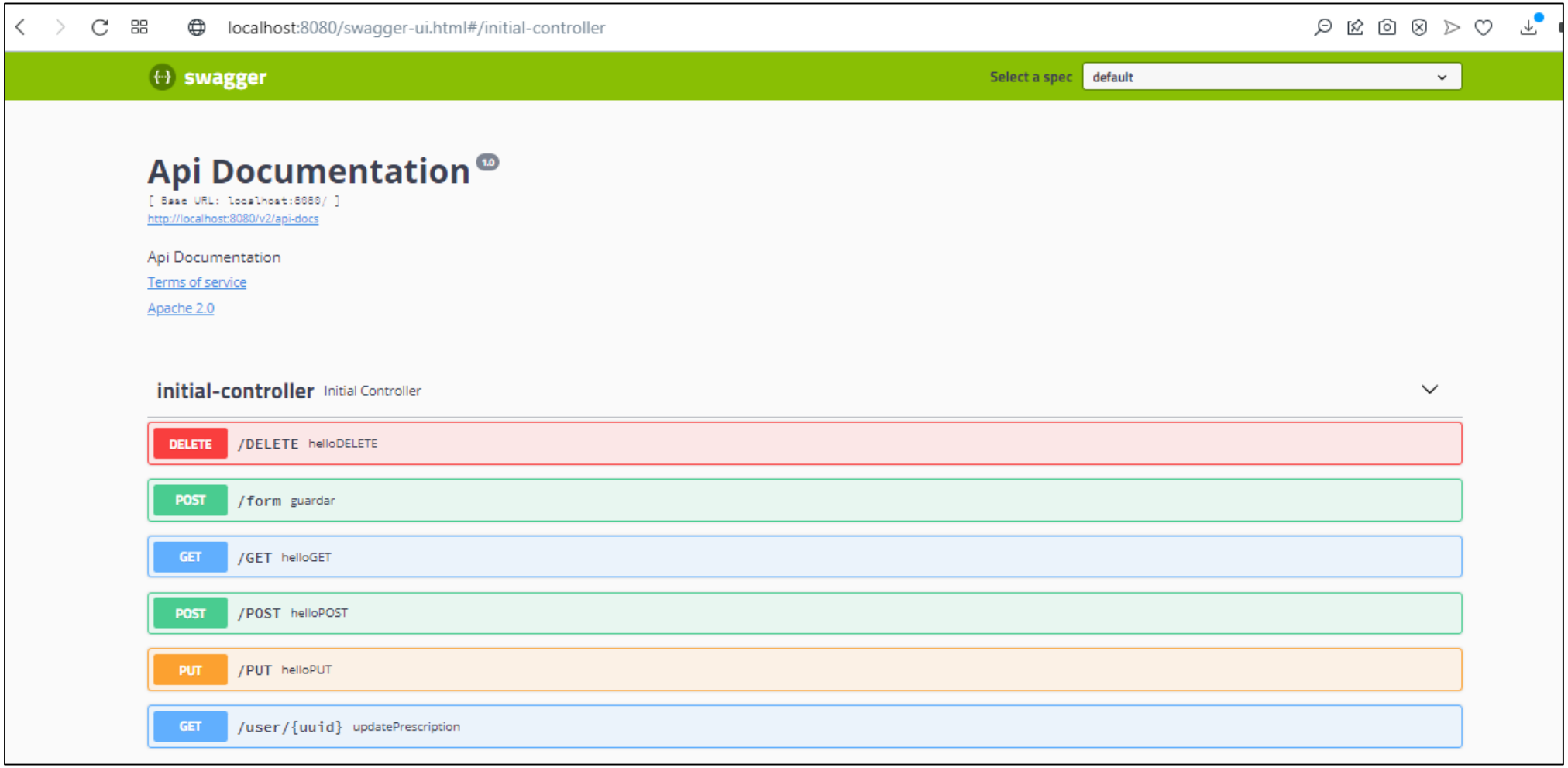
4. UPLOAD FILE

Paso 1) Agregaremos el siguiente endpoint al controlador:

```
@PostMapping(value = "/form")
public String guardar(@RequestParam("file") MultipartFile foto){
    String message=null;
    try {
        if (!foto.isEmpty()) {
            Path rootPath = Paths.get("uploads")
                            .resolve(foto.getOriginalFilename())
                            .toAbsolutePath();
            Files.copy(foto.getInputStream(), rootPath);
            message="Fichero subido :"+foto.getOriginalFilename();
        }else {
            message="Fichero vacio";
        }
    } catch (IOException e) {
        message=e.getMessage();
    }
    return message;
}
```

4. UPLOAD FILE

Paso 2) Comprobamos la subida con swagger en localhost:8080/swagger-iu.html



4. UPLOAD FILE

Paso 3) Hacemos un upload del fichero en una carpeta del proyecto:

POST /form guardar

Parameters

Cancel

Name	Description
file * required file (formData)	file Seleccionar archivo subir.txt

Execute | Clear

Responses

Response content type */*

Curl

```
curl -X POST "http://localhost:8080/form" -H "accept: */*" -H "Content-Type: multipart/form-data" -F "file=@subir.txt;type=text/plain"
```

Request URL

```
http://localhost:8080/form
```

Server response

Code	Details
200	Response body Fichero subido :subir.txt Download

springBootInitialDemo-master ▸ uploads

n biblioteca ▾ Compartir con ▾ Nueva ca

Nombre

subir

4. UPLOAD FILE

Paso 4) Agregaremos el siguiente endpoint al controlador de descarga:

```
@GetMapping(value = "/uploads/{filename:.+}")
public ResponseEntity<Resource> verFoto(@PathVariable String filename) {
    Resource recurso = null;
    Path pathFoto = null;
    try {
        pathFoto = Paths.get("uploads").resolve(filename).toAbsolutePath();
        recurso = new UrlResource(pathFoto.toUri());
        if (!recurso.exists() || !recurso.isReadable()) {
            throw new RuntimeException("Error: no se puede cargar la imagen: " + pathFoto.toString());
        }
    } catch (MalformedURLException e) {
        e.printStackTrace();
    }

    return ResponseEntity.ok()
            .header(HttpHeaders.CONTENT_DISPOSITION, "attachment; filename=\"" +
                    recurso.getFilename() + "\"")
            .body(recurso);
}
```

4. UPLOAD FILE

Paso 5) Realizamos la descarga del fichero desde el directorio del proyecto: