Swagger

Victor Herrero Cazurro

Contenidos

1. Introduccion

Swagger es una especificación para documentar servicios.

Por un lado proporciona un API para añadir meta-información a los servicios y por otro un cliente web que permite interpretar dicha meta-información ofreciendo la documentacion formateada.

El cliente web se puede descargar de aquí

Se puede arrancar en un contenedor Docker

```
docker pull swaggerapi/swagger-ui

docker run -p 80:8080 swaggerapi/swagger-ui
```

Tambien se puede emplear una que los fabricantes tienen publicada en http://petstore.swagger.io/

O incluso añadirla como dependencia al proyecto para proyectos Spring.

2. Soporte para Spring MVC

Se ha de añadir la dependencia con Maven

```
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger2</artifactId>
    <version>2.4.0</version>
</dependency>
```

A mayores se puede definir la dependencia con **swagger-ui** para que se genere el cliente

```
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger-ui</artifactId>
    <version>2.4.0</version>
</dependency>
```

Se ha de activar **Swagger**

Con XML

```
<swagger:enable-swagger2/>
```

Con JavaConfig

```
@Configuration
@EnableSwagger2
```

Si se ha incluido **swagger-ui**, se han de mapear los recursos estaticos siguientes empleando la herencia de la clase **WebMvcConfigurerAdapter**, ya que estos recursos residen en el jar **springfox-swagger-ui**

```
@Configuration
@EnableSwagger2
public class SpringConfig extends WebMvcConfigurerAdapter {
    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry.addResourceHandler("swagger-ui.html")
.addResourceLocations("classpath:/META-INF/resources/");
        registry.addResourceHandler("/webjars/**").
addResourceLocations("classpath:/META-INF/resources/webjars/");
    }
}
```

Tambien se han de definir los siguientes Bean que automatizan la lectura de los APIs y la documentacion general del API.

```
@Bean
public Docket newsApi(ApiInfo apiInfo) {
    return new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(apiInfo)
                .select()
                    .apis(RequestHandlerSelectors.any())
                    .paths(PathSelectors.any())
                    .build();
}

//Bean que permite definir Meta-información general del API

@Bean
public ApiInfo apiInfo() {
    return new ApiInfoBuilder()
            .title("API Rest con Spring MVC documentada con Swagger")
            .description("API Rest con Spring MVC documentada con
Swagger")
            .contact(new Contact("Victor",
"victorherrerocazurro.github.io", "victorherrerocazurro@gmail.com"))
            .license("Apache License Version 2.0")
            .build();
}
```

NOTE

No olvidar el Binder de Jackson para las transformaciones en JSON

```
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.5.3</version>
</dependency>
```

NOTE

Para acceder a la documentacion generada con **Swagger**, se ha de acceder a la siguiente ruta **http://<servername>:<port>/<context root>/v2/api-docs**

2.1. Anotaciones para personalizar la documentación

Se ofrecen las siguientes anotaciones en el paquete **io.swagger.annotations**

- **@Api**: Se emplea a nivel de clase de controlador, tipicamente anotada con **@RestController**, para que **Swagger** la tenga en cuenta a la hora de generar la documentación.

```
@Api(tags={"persona"}, description="API REST sobre la entidad persona")
```

- **@ApiOperation**: Se emplea a nivel de método de controlador, tipicamente anotado con **@RequestMapping** para documentar la operación que representa el método

```
@ApiOperation(value = "Consultar una persona por identificador")
```

- **@ApiParam**: Se emplea a nivel de parametro de método de controlador, tipicamente anotado con **@PathVariable** o **@RequestParam**, para documentar los parametros que recibe el método.

```
@ApiParam(name="personaId", value="El identificador de la persona",
required=true)
```