¿Cómo incluir Swagger en nuestro proyecto?

Desarrollo de un API REST con Spring Boot

Swagger + SpringFox

- SpringFox es un conjunto de librerías que nos permite generar automáticamente la documentación de nuestra API.
- Es capaz de generar esta documentación en formato Swagger.
- La ventaja es que no tenemos que generar, manualmente, el fichero *swagger.json*.
- Disponemos de clases y anotaciones para poder afinar la configuración.

Nuevas dependencias en ***pom.xml***

```
<dependency>
	<groupId>io.springfox</groupId>
		<artifactId>springfox-swagger2</artifactId>
		<version>2.9.2</version>
</dependency>
<dependency>
		<groupId>io.springfox</groupId>
		<artifactId>springfox-swagger-ui</artifactId>
		<version>2.9.2</version>
</dependency>
```

Configuración

- Anotación *@EnableSwagger2* en una clase *@Configuration*
- Necesitamos un bean de tipo *Docket* que incluya la configuración para generar la documentación.

Configuración básica

```
@Configuration @EnableSwagger2
public class SwaggerConfig {

        @Bean
        public Docket api() {
                return new Docket(DocumentationType.SWAGGER_2)
                        .select()
                        .apis(
                RequestHandlerSelectors
                 .basePackage("com.openwebinars.rest.controller"))
                        .paths(PathSelectors.any())
                        .build();
        }
}
```

Personalización

- Podemos personalizar (a través de anotaciones y programáticamente) el resultado de la documentación.
- Podemos incluir una ApiInfo acorde a nuestro proyecto.

```
@Bean
public Docket api() {
     return new Docket(DocumentationType.SWAGGER_2)
               .select()
               .apis(RequestHandlerSelectors
          .basePackage("com.openwebinars.rest.controller"))
               .paths(PathSelectors.any())
               .build().apiInfo(apiInfo());
     }
```

Anotaciones

- Disponemos de algunas anotaciones que nos permiten personalizar determinados aspectos
- A nivel de método de controlador
  - @ApiOperation: describe qué hace el método del controlador
  - @ApiResponse/s: describen las diferentes respuestas que puede dar dicho método
  - @ApiParam: describen el parámetro que recibe el método

Anotaciones

- A nivel de objetos POJO
  - @ApiModelProperty: nos permite personalizar la información que aparece de cada propiedad del modelo
    - Nombre
    - Tipo de dato
    - Valor de ejemplo
    - Posición.

Reto

- Finaliza la documentación del resto de métodos del controlador.
- En la documentación adicional tienes el enlace a la lista completa de anotaciones, por si quieres investigar alguna adicional.