Swagger a una aplicació Canigó 3.1

A qui va dirigit

Aquest how-to va dirigit a tots aquells desenvolupadors/arquitectes que vulguin afegir Swagger a una aplicació Canigó 3.1 (REST).

Versió de Canigó

Els passos descrits en aquest document apliquen a la versió 3.1.x del Framework Canigó amb arquitectura REST+HTML5/JS.

Introducció

En aquest HowTo s'explica com afegir Swagger a una aplicació Canigó 3.1 amb arquitectura REST+HTML5/JS. Per a fer-ho desplegarem l'aplicació demo que genera el plugin de Canigó.

Swagger a una aplicació Canigó 3.1

Afegir Llibreries

Primer de tot afegim les llibreries necessàries de Swagger. Utilitzem les llibreries que proporciona Springfox per la integració amb Spring:

pom.xml

```
…
<!-- SWAGGER -->
<dependency>
	<groupId>io.springfox</groupId>
	<artifactId>springfox-swagger2</artifactId>
	<version>2.5.0</version>
</dependency>

<dependency>
	<groupId>io.springfox</groupId>
	<artifactId>springfox-swagger-ui</artifactId>
	<version>2.5.0</version>
</dependency>
…
```

Configurar Swagger

En cas de que l'aplicació incorpori Spring Boot no cal realitzar cap configuració per a que funcioni Swagger. En el cas contrari s'ha de realitzar la següent configuració:

Es crea una classe que hereti de WebMvcConfigurerAdapter i 's'afegeixen els resources handlers.

src/main/java/cat/gencat/config/WebAppConfig.java

```
package cat.gencat.canigorest.config;

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

@Configuration
@EnableWebMvc
public class WebAppConfig 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/");
    }

}
```

Swagger a una aplicació Canigó 3.1

Al fitxer app-custom-beans.xml afegim la ubicació del fitxer que hem creat per a que cerqui els beans:

app-custom-beans.xml

```
...
<context:component-scan base-package="cat.gencat.canigorest.config" />
...
```

Per a finalitzar la configuració, tant amb Spring Boot com sense, s'ha de configurar els paths on es troben les Api's que es vol que apareguin a la plana de Swagger.

A la carpeta config es crea una altre classe, SwaggerConfiguration per exemple, i al nostre exemple configurem el path "api".

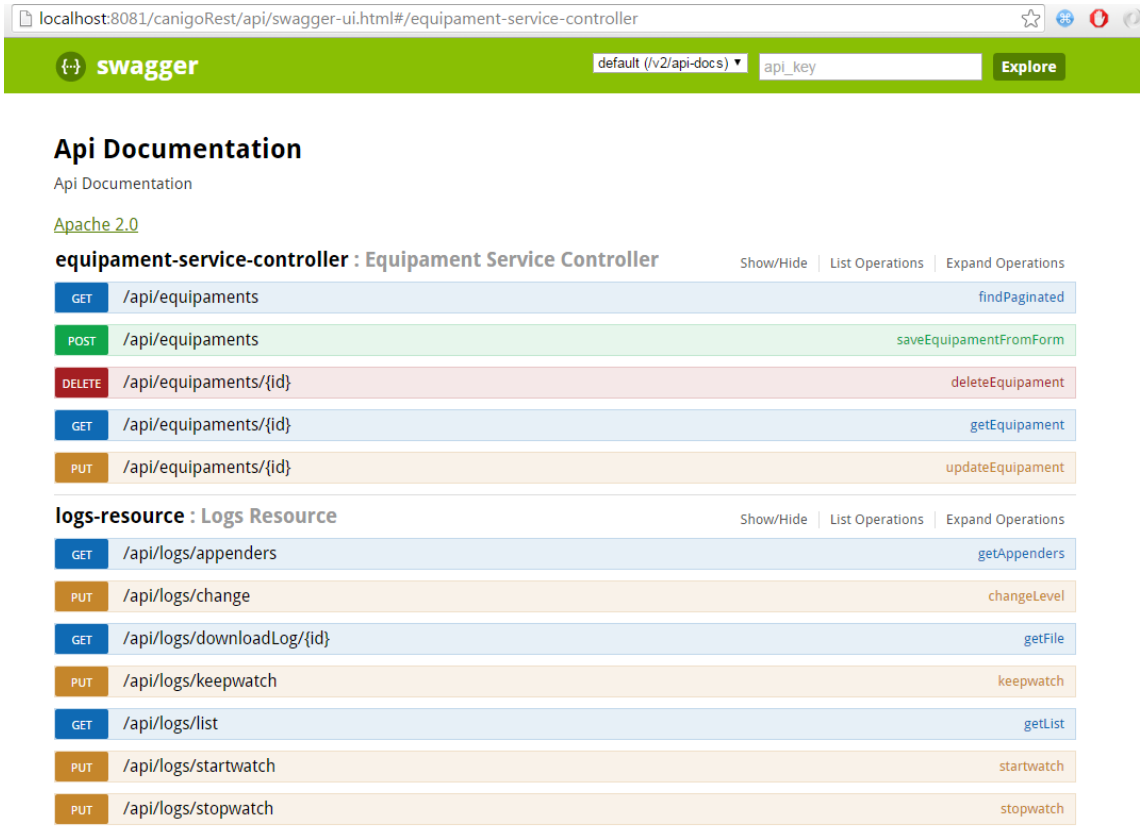
src/main/java/cat/gencat/config/SwaggerConfiguration.java

```
package cat.gencat.apprestswagger.config.apidoc;

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

import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

/**
 * Swagger configuration.
 *
 */
@Configuration
@EnableSwagger2
public class SwaggerConfiguration  {

       @Bean
    public Docket api() {
              Docket docket = new Docket(DocumentationType.SWAGGER_2)
                        .select()
                        .apis(RequestHandlerSelectors.any())
                        .paths(PathSelectors.any())
                        .build();
              docket.pathMapping("api");
         return docket;
    }

}
```

Swagger a una aplicació Canigó 3.1

Resultat

Per a accedir a Swagger, es fa a través del path "/api/swagger.ui.html"



Gràcies a Swagger és molt fàcil testejar les API's. Els desenvolupadors de backend defineixen els contractes i documenten les APIs, i d'aquesta manera els desenvolupadors de frontend poden provar de forma força intuïtiva les crides abans d'integrar-les a la capa de presentació.