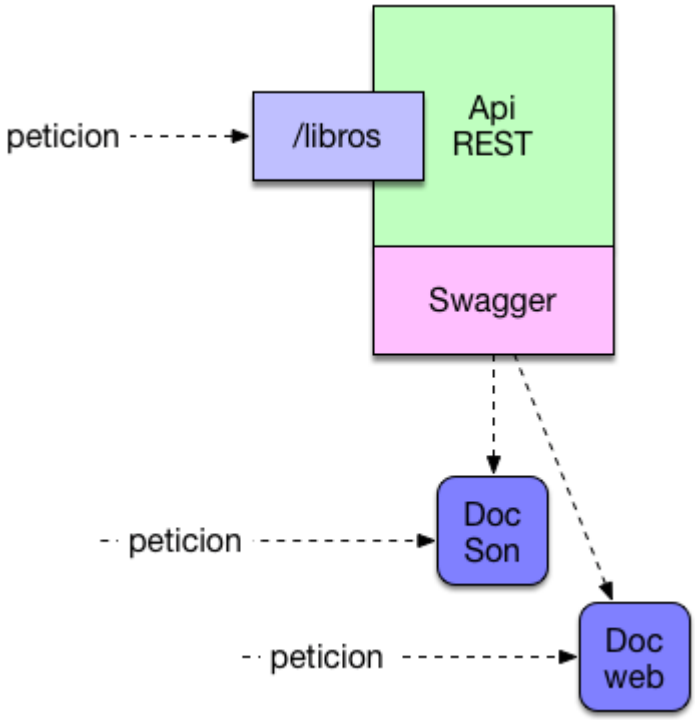
Acabamos de ver como usar Java equals y hashcode

Swagger es una herramienta que nos permite documentar de una forma sencilla nuestras APIs REST hoy por hoy se ha convertido en uno de los standares de facto a la hora de trabajar . Cada día publicamos un mayor número de APIs REST y lo lógico es que publiquemos una documentación clara de como trabajar con ellas. Swagger nos permite hacerlo de una forma muy rápida y directa.



Vamos a ver un ejemplo, para ello partiremos de un proyecto de Spring boot que tiene un servicio que devuelve una lista de personas. Las dependencias de Maven son:

```
<dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
```

```
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger2</artifactId>
            <version>2.6.1</version>
        </dependency>

        <!--
https://mvnrepository.com/artifact/io.springfox/springfox-swagger-ui -
->
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger-ui</artifactId>
            <version>2.6.1</version>
        </dependency>
    </dependencies>
```

Spring @RestController

En ellas podemos ver como hemos incluido las nuevas librerías. El siguiente paso es definir un servicio REST con la anotación @RestController.

```
package com.arquitecturajava.rest;

import java.util.ArrayList;
import java.util.List;
```

```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class ControladorPersona {

	@GetMapping("/personas")
	public List&amp;amp;lt;Persona&amp;amp;gt; findAll() {

		List&amp;lt;Persona&amp;gt; lista= new
ArrayLis&amp;lt;Persona&amp;gt;();
		lista.add(new Persona("pepe","perez",25));
		lista.add(new Persona("juan","sanchez",35));
		lista.add(new Persona("ana","gomez",25));
		return lista;
	}

}
```

Configuración de Spring

Es momento de configurar la herramienta para que nos genere una documentación del API REST.Para ello añadimos un nuevo fichero de configuración de Spring.

```
package com.arquitecturajava.rest1;

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

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

import static springfox.documentation.builders.PathSelectors.regex;
@Configuration
@EnableSwagger2
public class SwaggerConfiguration {

         @Bean
            public Docket productApi() {
                return new Docket(DocumentationType.SWAGGER_2)
                                    .select()
.apis(RequestHandlerSelectors.basePackage("com.arquitecturajava.rest")
)
                        .paths(regex("/personas.*"))
                        .build();
            }

}
```

Aquí configuramos swagger para que publique documentación para la url que publicamos en este caso nuestra url de /personas. Nos queda por ver el fichero de configuración principal de Spring framework que importará el de Swagger.

```
package com.arquitecturajava.rest;
```

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Import;
import
org.springframework.web.servlet.config.annotation.ResourceHandlerRegis
try;
import
org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@SpringBootApplication
@Import(SwaggerConfiguration.class)
public class RestApplication implements WebMvcConfigurer{

        public static void main(String[] args) {
                SpringApplication.run(RestApplication.class, args);
        }

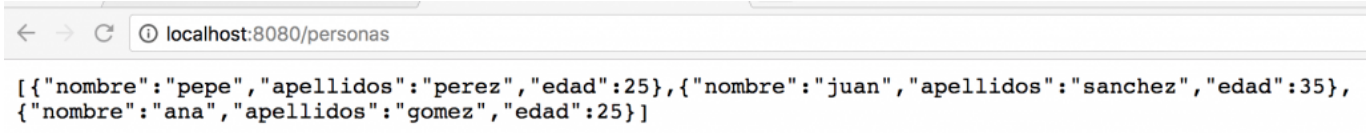
        @Override
        public void addResourceHandlers(ResourceHandlerRegistry
registry) {

                registry.addResourceHandler("swagger-ui.html")
                        .addResourceLocations("classpath:/META-
INF/resources/");

        }
}
```

Swagger y documentación

Arrancamos la aplicación con Spring Boot y accederemos de forma directa a la url de personas que nos mostrará la lista de personas.



Hasta aquí todo es lo habitual , sin embargo si miramos los ficheros de log de Spring boot podremos ver que se han publicado más urls.

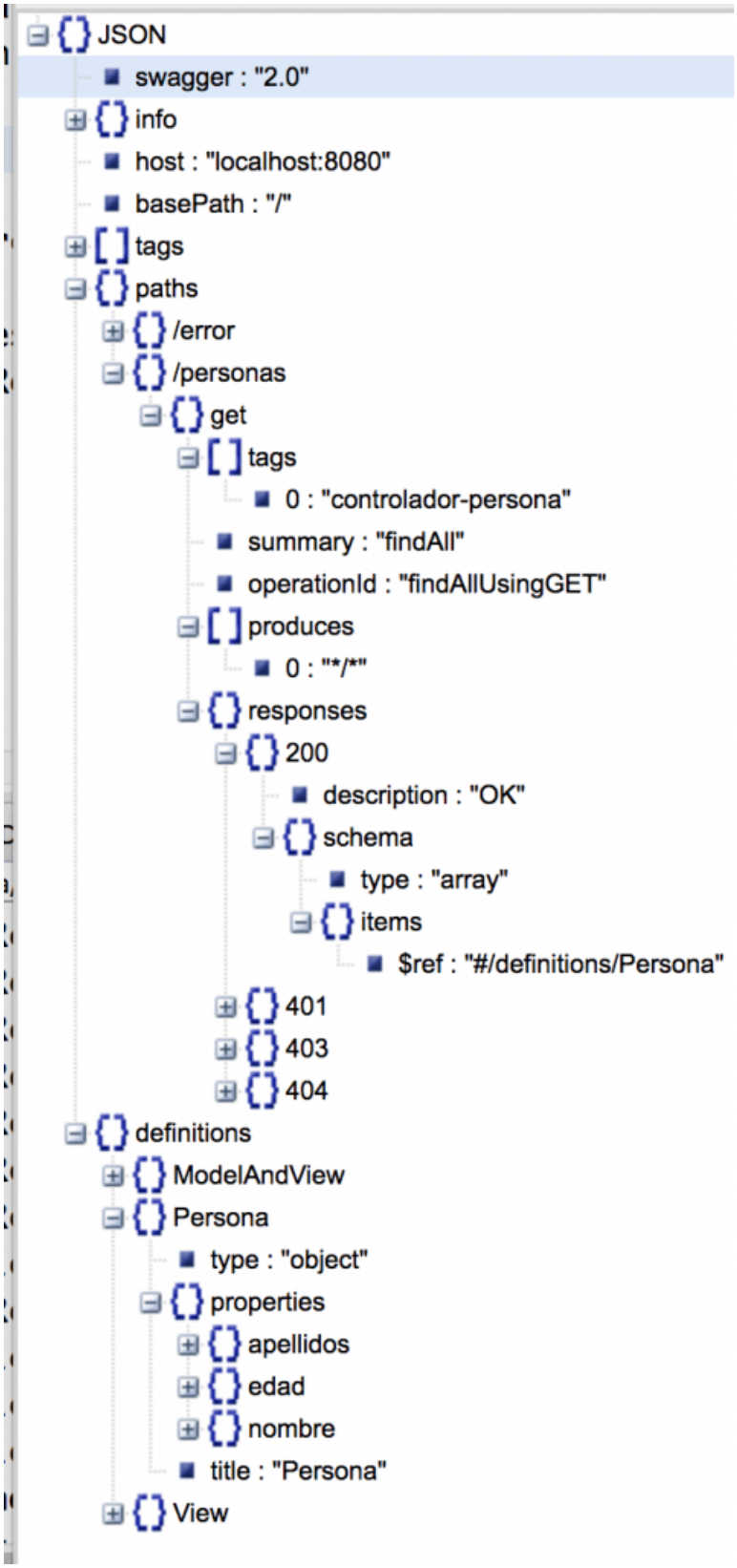


Dos de las urls más importantes de swagger son /v2/api-docs que publica la documentación en formato json . Si echamos un vistazo a la url recibiremos algo como :


```
{"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":"basic-error-controller","description":"Basic Error Controller"},{"name":"controlador-persona","description":"Controlador
Persona"}],"paths":{"/error":{"get":{"tags":["basic-error-controller"],"summary":"error","operationId":"errorUsingGET","produces":
["*/*"],"responses":{"200":{"description":"OK","schema":{"type":"object","additionalProperties":{"type":"object"}}},"401":
{"description":"Unauthorized"},"403":{"description":"Forbidden"},"404":{"description":"Not Found"}}},"head":{"tags":["basic-error-
controller"],"summary":"error","operationId":"errorUsingHEAD","consumes":["application/json"],"produces":["*/*"],"responses":{"200":
{"description":"OK","schema":{"type":"object","additionalProperties":{"type":"object"}}},"204":{"description":"No Content"},"401":
{"description":"Unauthorized"},"403":{"description":"Forbidden"}}},"post":{"tags":["basic-error-
controller"],"summary":"error","operationId":"errorUsingPOST","consumes":["application/json"],"produces":["*/*"],"responses":{"200":
{"description":"OK","schema":{"type":"object","additionalProperties":{"type":"object"}}},"201":{"description":"Created"},"401":
{"description":"Unauthorized"},"403":{"description":"Forbidden"},"404":{"description":"Not Found"}}},"put":{"tags":["basic-error-
controller"],"summary":"error","operationId":"errorUsingPUT","consumes":["application/json"],"produces":["*/*"],"responses":{"200":
{"description":"OK","schema":{"type":"object","additionalProperties":{"type":"object"}}},"201":{"description":"Created"},"401":
{"description":"Unauthorized"},"403":{"description":"Forbidden"},"404":{"description":"Not Found"}}},"delete":{"tags":["basic-error-
controller"],"summary":"error","operationId":"errorUsingDELETE","produces":["*/*"],"responses":{"200":{"description":"OK","schema":
{"type":"object","additionalProperties":{"type":"object"}}},"204":{"description":"No Content"},"401":{"description":"Unauthorized"},"403":
{"description":"Forbidden"}}},"options":{"tags":["basic-error-controller"],"summary":"error","operationId":"errorUsingOPTIONS","consumes":
["application/json"],"produces":["*/*"],"responses":{"200":{"description":"OK","schema":{"type":"object","additionalProperties":
{"type":"object"}}},"204":{"description":"No Content"},"401":{"description":"Unauthorized"},"403":{"description":"Forbidden"}}},"patch":{"tags":
["basic-error-controller"],"summary":"error","operationId":"errorUsingPATCH","consumes":["application/json"],"produces":["*/*"],"responses":
{"200":{"description":"OK","schema":{"type":"object","additionalProperties":{"type":"object"}}},"204":{"description":"No Content"},"401":
{"description":"Unauthorized"},"403":{"description":"Forbidden"}}}},"/personas":{"get":{"tags":["controlador-
persona"],"summary":"findAll","operationId":"findAllUsingGET","produces":["*/*"],"responses":{"200":{"description":"OK","schema":
{"type":"array","items":{"$ref":"#/definitions/Persona"}}},"401":{"description":"Unauthorized"},"403":{"description":"Forbidden"},"404":
{"description":"Not Found"}}}}},"definitions":{"ModelAndView":{"type":"object","properties":{"empty":{"type":"boolean"},"model":
{"type":"object"},"modelMap":{"type":"object","additionalProperties":{"type":"object"}},"reference":{"type":"boolean"},"status":
{"type":"string","enum":
["100","101","102","103","200","201","202","203","204","205","206","207","208","226","300","301","302","303","304","305","307","308","400","401"
,"402","403","404","405","406","407","408","409","410","411","412","413","414","415","416","417","418","419","420","421","422","423","424","426"
,"428","429","431","451","500","501","502","503","504","505","506","507","508","509","510","511"]},"view":
{"$ref":"#/definitions/View"},"viewName":{"type":"string"}},"title":"ModelAndView"},"Persona":{"type":"object","properties":{"apellidos":
{"type":"string"},"edad":{"type":"integer","format":"int32"},"nombre":{"type":"string"}},"title":"Persona"},"View":
{"type":"object","properties":{"contentType":{"type":"string"}},"title":"View"}}}
```


En principio es una información que nos puede parecer extraña. Es suficiente abrirla con un

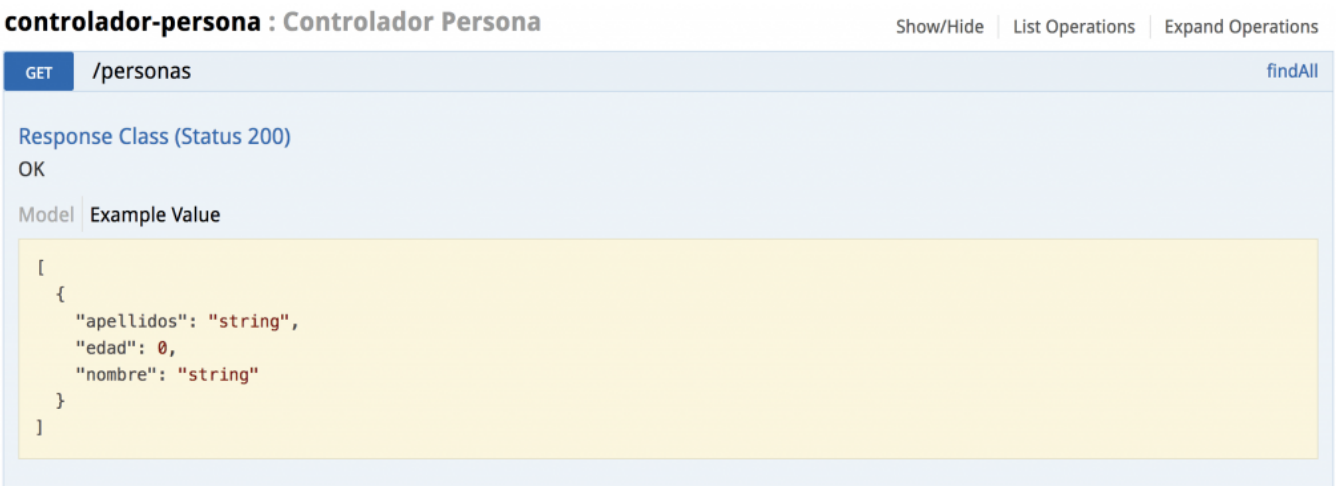
visor de JSON para que la información nos quede mucho más clara:


JSON
swagger : "2.0"
info
host : "localhost:8080"
basePath : "/"
tags
paths
/error
/personas
get
tags
0 : "controlador-persona"
summary : "findAll"
operationId : "findAllUsingGET"
produces
0 : "*/*"
responses
200
description : "OK"
schema
type : "array"
items
$ref : "#/definitions/Persona"
401
403
404
definitions
ModelAndView
Persona
type : "object"
properties
apellidos
edad
nombre
title : "Persona"
View

Podemos ver claramente la definición de la operación y el tipo de objeto que devuelve, así como información adicional. Otra de las opciones que tenemos es acceder a una vista web de la documentación mucho más amable utilizando la url /swagger-ui.html.



Entramos a controlador persona y vemos los métodos que tiene soportados:



Acabamos de documentar nuestro api REST con swagger.

Otros artículos relacionados:

1. REST HTTP return codes y sus curiosidades
2. Spring REST Service con @RestController
3. Arquitecturas REST y sus niveles
4. Spring Boot