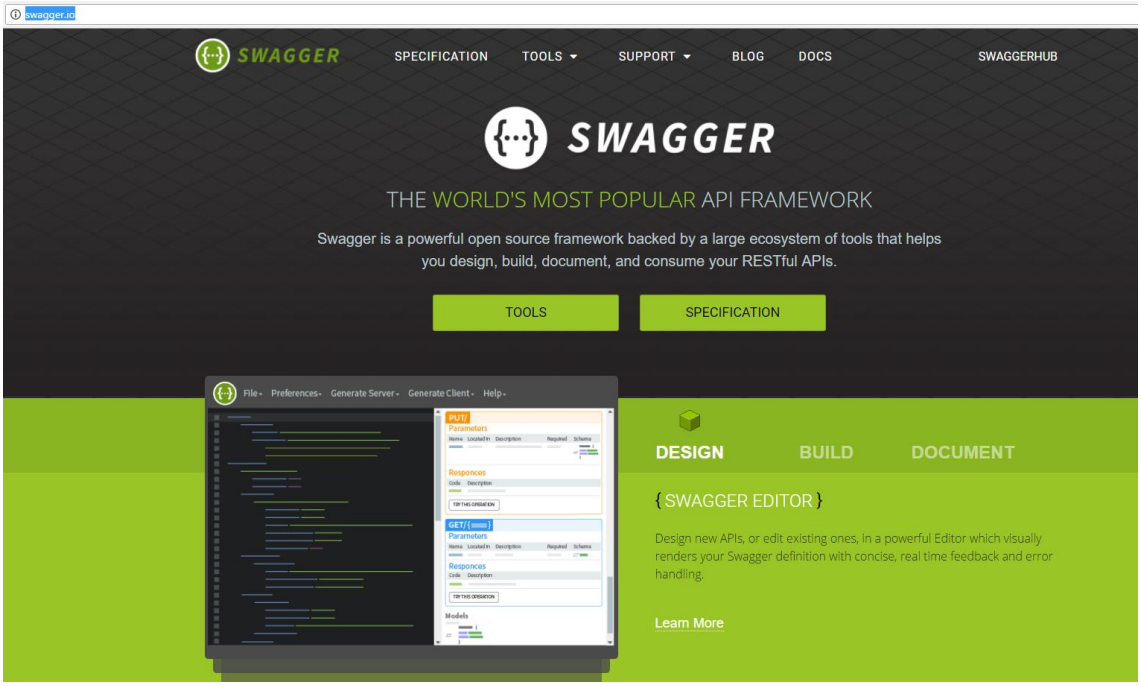
Framework Swagger



Qué es?

Es un framework, es decir un marco de trabajo que incluye tanto la especificación como la implementación que permite trabajar con servicios REST.

- ✓ DESCRIBIR
- ✓ PRODUCIR
- ✓ CONSUMIR
- ✓ VISUALIZAR

Función

Documentar los servicios Rest a medida que se actualiza el servicio, de manera que las apis estén siempre sincronizadas.

Incluye:

- ✓ Métodos
- ✓ Parámetros
- ✓ Modelo

Por ser una especificación es independiente del lenguaje.

Implementaciones que ofrece:

- ✓ Java
- ✓ HTML5
- ✓ Scala

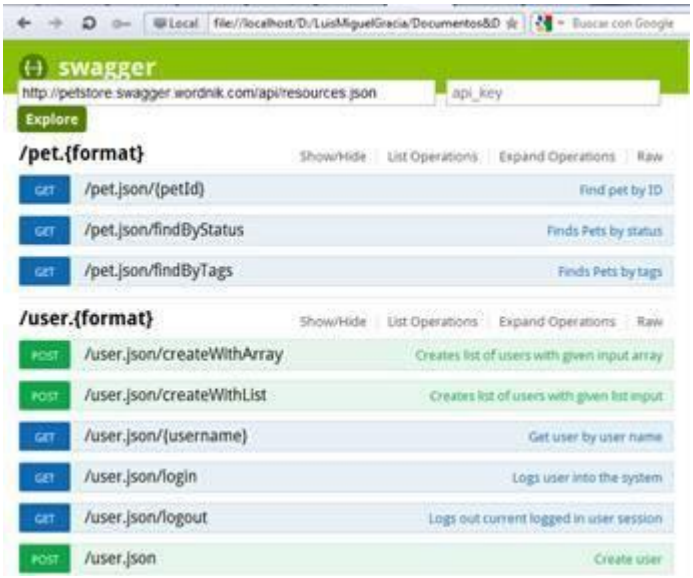
Soporta: JSON y XML.

Módulos

-Swagger UI: una aplicación HTML (+JS+CSS) que permite generar documentación de un API Swagger.

Es una consola web HTML que permite:

- Listar operaciones:



- Ejecutar operaciones

- **Swagger Core**: define las anotaciones Java y la lógica requerida para generar un cliente o servidor Swagger. Incluye ejemplos en Java, Scala, con Play2 framework:

-**Swagger node.js**: Servidor Swagger stand-alone escrito en Javascript con node.js

- **Swagger Java Sample App**: Servidor Swagger stand-alone escrito en Java que demuestra como habilitar Swagger en tu API.

Ejemplo

Si revisamos el código de la clase **PetResource** vemos que Swagger ofrece anotaciones para documentar las operaciones (@ApiOperation), los errores (@ApiError) y los parámetros (@ApiParam)

Anotaciones

– @**Api** documenta el servicio REST en si. Va a ser la descripción que salga en el listado, entre otras cosas.
– @**ApiOperation** documenta cada método del servicio.

– @**ApiResponses** documenta las posibles respuestas del método, con mensaje explicativo.
Existen otras anotaciones para los parámetros,

```
public class PetResource extends JavaHelp {
        static PetData petData = new PetData();
        static JavaRestResourceUtil ru = new JavaRestResourceUtil();

        @GET
        @Path("/{petId}")
        @ApiOperation(value = "Find pet by ID", notes = "Returns a pet when ID < 10. "
                        + "ID > 10 or nonintegers will simulate API error conditions", responseClass = "com.wordnik.swagger.sample.model.Pet")
        @ApiErrors(value = { @ApiError(code = 400, reason = "Invalid ID supplied"),
                        @ApiError(code = 404, reason = "Pet not found") })
        public Response getPetById (
                        @ApiParam(value = "ID of pet that needs to be fetched", allowableValues = "range[1,5]", required = true) @PathParam("petId") String petId)
        throws NotFoundException {
                Pet pet = petData.getPetById(ru.getLong(0, 100000, 0, petId));
                if (null != pet) {
                        return Response.ok().entity(pet).build();
                } else {
                        throw new NotFoundException(404, "Pet not found");
                }
        }
```

Como se ven en la consola Web

Y cuando se produce error:

· **Swagger CodeGen** ofrece un motor de plantillas para generar código cliente en diferentes lenguajes parseando la declaración de recursos.

Ejemplo PetStore

http://petstore.swagger.io/

Links de Interes

http://swagger.io/
https://github.com/wordnik/swagger-spec
https://github.com/wordnik/swagger-ui
https://github.com/martypitt/swagger-springmvc

https://bitbucket.org/somospnt/demo-swagger-springfox

Editor Swagger

-Editor Swagger Web

http://editor.swagger.io/#/

-Editor swagger local

https://github.com/swagger-api/swagger-editor

Decargar y abrir en el navegador el archivo index.html

Importar un archivo:

File - Import File

Yaml

http://yaml.org/

https://learnxinyminutes.com/docs/es-es/yaml-es/

YAML es un formato para guardar objetos de datos con estructura de árbol. Sus siglas significan YAML Ain't Markup Language (YAML no es otro lenguaje de marcado).

Este lenguaje es muy legible para las personas, más legible que JSON y sobretodo que XML.

Normalmente se utiliza para definir archivos de configuración, aunque también es posible serializar objetos, es decir, escribir la estructura de un objeto en modo cadena de texto para posteriormente poderlo recuperar. Sin embargo, para este propósito es bastante más lento que utilizar JSON.

Tiene como objetivo facilitar el mapeo de estructuras de datos más complejas (como listas y arreglos asociativos) en un documento de texto plano legible para un ser humano. Si bien es un formato joven, sus características le han hecho ganarse un lugar importante en la web, junto con XML y JSON.

Hay librerías para todos los lenguajes de programación, desde Ruby, PHP, Java hasta C# entre otros.

La estructura del archivo es de tipo clave-valor. Además es importante la indentación, para definir la estructura de árbol padre-hijo.