Introdução a Web Services REST com Spring Framework

Parte 5 → Documentação com Swagger

Prof. Me. Jorge Luís Gregório

www.jlgregorio.com.br

Agenda

- Sobre o Swagger Open API
- Configurando a aplicação para gerar a documentação
  - Dependências do Swagger;
  - Gerando a documentação;
  - Testando
- Personalizando a documentação

O que é Swagger?

Swagger Open API

- O Swagger é um framework open-soure usado para a documentação de APIs abertas.
- Ele possui a capacidade de gerar uma página HTML contendo a documentação das classes da API, além de permitir realizar testes de requisição;
- Isso é possível graças a anotações padrão do Spring e/ou do próprio Swagger;
- A documentação geralmente descreve os recursos disponíveis agrupando-os por ***controllers*** e ***actions***, além de models;
- Os parâmetros de entrada e os tipos de retorno também são documentados;
- Site oficial: https://swagger.io/

Configurando

Declarando as dependências

- O Swagger precisa de duas dependências:
  - **Swager** → gera a documentação "pura" em formato JSON;
  - **Swagger UI** → cria a página HTML com a documentação;
- Para incluir as dependências necessárias, use o seguinte código no arquivo ***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>
```

Ajustando o arquivo application.properties

- Para usar o Swagger UI, ou seja, a documentação formatada em HTML, é recomendável adicionar a seguinte configuração no arquivo ***application.properties***:

*spring.mvc.pathmatch.matching-strategy=**ant-path-matcher***

Essa configuração aplica a sintaxe ***ant-path-matcher*** para mapear classes e arquivos de configuração.

Para mais detalhes, consulte a documentação:
https://spring.io/blog/2020/06/30/url-matching-with-pathpattern-in-spring-mvc

Adicionando um arquivo de configuração

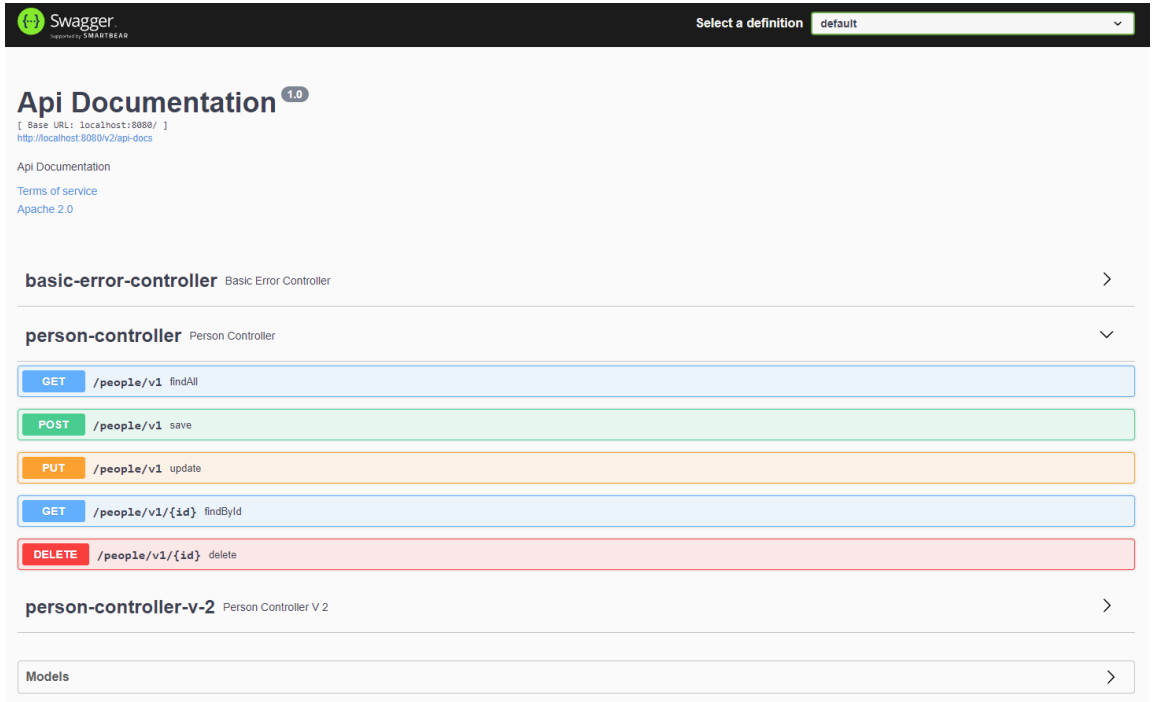
- No pacote principal da aplicação, crie um pacote chamado ***config***;
- Adicione uma classe java chamada ***SwaggerConfig***;
- Ela deve ter o seguinte Código:

```java
package br.com.jlgregorio.springcrud.config;
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;

@Configuration //it is a class configuration - read it from boot!
@EnableSwagger2 // enables the Swagger
public class SwaggerConfig {

    @Bean
    public Docket api(){
        // a docket is an instance of docs
        return new Docket(DocumentationType.SWAGGER_2)
                .select()
                .apis(RequestHandlerSelectors.any())
                .paths(PathSelectors.any())
                .build();
    }

}
```

Verificando a documentação

- Para a documentação “pura”, basta fazer a requisição para a url: http://localhost:8080/v2/api-docs
- Isso deve retornar apenas em formato JSON;

- Para a página da documentação, acesse a URL: http://localhost:8080/swagger-ui.html



Personalizando

Como personalizar o Swagger?

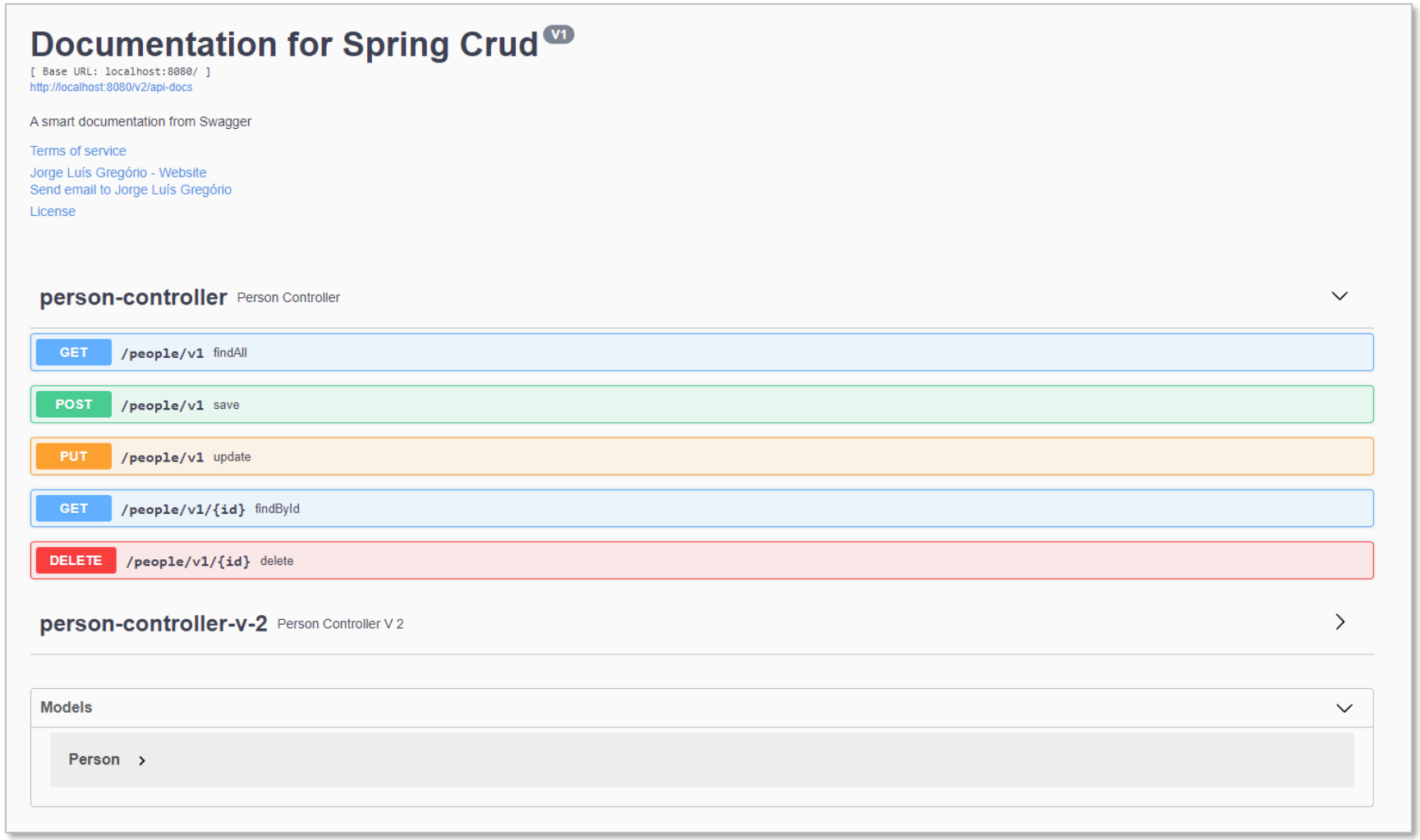
- A estrutura básica (padrão ou default) gera uma documentação útil, mas é importante fazer algumas personalizações, como por exemplo:
  - Alterar a descrição
  - Adicionar uma URL par termos de uso;
  - Especificar melhor os endpoints;
  - Etc...

Alterando a classede configurações

- Abra a classe ***SwaggerConfig.java*** e faça as seguintes alterações.
- Note que no médoto ***api(),*** colocamos o pacote-base que será considerado na criação da documentação;
- Também adicionamos um objeto **ApiInfo**, que contém informações sobre a API.

```
@Bean
public Docket api(){
    // a docket is an instance of docs
    return new Docket(DocumentationType.SWAGGER_2)
            .select()
            .apis(RequestHandlerSelectors.basePackage("br.com.jlgregorio.springcrud"))
            .paths(PathSelectors.any())
            .build()
            .apiInfo(apiInfo());
}

private ApiInfo apiInfo() {
    return new ApiInfo( title: "Documentation for Spring Crud",
            description: "A smart documentation from Swagger", version: "V1",
            termsOfServiceUrl: "http://www.jlgregorio.com.br",
            new Contact( name: "Jorge Luís Gregório", url: "http://www.jlgregorio.com.br",
                    email: "jorge.gregorio@fatec.sp.gov.br"),
            license: "License", licenseUrl: "http://www.jlgregorio.com.br",
            Collections.emptyList());
}
```

Verificando as mudanças



Documentando operações

- Para documentar melhor os controllers, use a anotação **@Api**:

```
@Api(value = "Person Endpoint V2", tags = {"Person", "Model", "Endpoint"})
@RestController
@RequestMapping("/people/v2")
public class PersonControllerV2 {
```

- Para documentar melhor as operações, use a anotação **@ApiOperation**:

```
@ApiOperation(value = "Persist an Person", response = Person.class)
@PostMapping
public Person save(@RequestBody Person person) { return service.save(person); }
```

Documentando parâmetros

- Para documentar os parâmetros, use a anotação **@ApiParam**.
- No exemplo a seguir usamos a anotação **@ApiOperation** para documentar o método e seu retorno, e a anotação **@ApiParam** para documentar o parâmetro de entrada do método

```
@ApiOperation(value = "Find a person by id.", response = Person.class)
@GetMapping("/{id}")
public Person findById(@ApiParam(name = "id", value = "An integer value", required = true)
                       @PathVariable("id") long id) throws Exception{
    return service.findById(id);
}
```

Desafios

- Documente as demais operações de sua API
- Como posso documentar as classes *model*? 🤔
- Será possível criar a UI do Swagger separado por versões? 🤔
- Envie seu projeto para o repositório Github.

Conclusão

- O Swagger oferece uma maneira simples e rápida de documentar suas APIs;
- É importante sempre documentar, pois os clientes precisam estar cientes de como acessar o seus serviços;
- Repositório
  - O repositório com o Código de teste está em https://github.com/prof-jlgregorio

Sobre mim

JORGE LUÍS GREGÓRIO

- Professor da Faculdade de Tecnologia “Prof. José Camargo” – Fatec Jales, e da Escola Técnica Estadual Dr. José Luiz Viana Coutinho – Etec Jales;
- Articulista do Jornal de Jales – Coluna “Fatecnologia”;
- Apresentador do Tech Trends, podcast oficial da Fatec Jales;
- Bacharel em Sistemas de Informação; Especialista em Desenvolvimento de Software para Web e Mestre em Ciência da Computação.
- Trabalha com tecnologia desde 1998, tendo atuado como analista de suporte; administrador de redes de computadores; desenvolvedor de software, *webdesigner* e professor.

- Site oficial: www.jlgregorio.com.br
- Perfil do LinkedIn: www.linkedin.com/in/jlgregorio81
- Currículo Lattes: http://lattes.cnpq.br/3776799279256689