Spring Web e Swagger

Interação de arquivos JSON para aplicação. Isso não se trata de Spring Web MVC.

API: faz a “ponte” de comunicações entre duas aplicações distintas.

- API REST (representational state transfer): é como um guia de boas práticas
- RESTful: capacidade de determinado sistema aplicar os princípios de REST.

Nível 1: Aplicação de Resources.

Nível 2: Implementação de verbos HTTP.

Nível 3: HATEOAS (Hypermedia as the Engine of Application State). Obs.: Conteúdo menos explorado, por isso a maioria das empresas não a utiliza.

Starter para o primeiro projeto: *Spring Web*.

Um controller é um recurso que disponibiliza as funcionalidades de negócio da aplicação através do protocolo HTTP.

Para se utilizar, cria-se por recomendação um pacote denominado “controller”.

Na classe, é denominado a anotação @RestController.

Ao utilizar um método, é feito como exemplo a anotação @GetMapping(“/”), que mapeia o método, dizendo que ele é um recurso HTTP – método GET.

Dentro deste método, ao retornar uma String, estamos desenvolvendo uma aplicação que retorna uma mensagem por meio do protocolo HTTP. Incrível!

Rest Controller é uma classe que disponibiliza recursos HTTP para nosso projeto.

É possível construir mais de um Controller. Mas essas classes não devem utilizar a mesma anotação de método com finalidades diferentes, pois o Spring não saberá qual deve ser executada. Por exemplo, ao invés de usar outro Controller que tenha @GetMapping(“/”), pode ser feito @GetMapping(“/exemplo”).

Para utilizar um parâmetro na anotação de método, é só colocar o nome do parâmetro entre chaves e defini-la no @PathVariable(). Por exemplo:

```
@GetMapping(“/exemplo/{nome}”)

public Usuario getOne (@PathVariable(“nome”) String nome){ ... }
```

Para deletar algo, utiliza-se a anotação @DeleteMapping(). E para o Post, utiliza-se @PostMapping().

@RequestBody serve para garantir que passe o corpo. O Spring interpreta que é para pegar o JSON e converter no objeto conforme programado. É utilizado em método POST.

@RequestMapping("/exemplo"), adicionado antes da classe, vai fazer com que todos os métodos HTTP desta classe irá se iniciar com o que está dentro desta anotação.

Swagger

É uma linguagem de descrição de interface para descrever APIs RESTful expressas usando JSON. O Swagger é usado junto com um conjunto de ferramentas de software de código aberto para projetar, construir, documentar e usar serviços da Web RESTful.

Para versões posteriores a 2.4.4 do Spring Boot, é utilizado o Swagger 3.0.

Este vídeo me ajudou a rodar a aplicação do Swagger de forma mais fácil com a nova versão: https://www.youtube.com/watch?v=f0aliuJNt4o

Do vídeo do YT

No arquivo pom, abaixo de java version, foi inserido o comando:

```
<springdoc.openapi.starter.webmvc.ui.version>2.0.2</springdoc.openapi.
starter.webmvc.ui.version>
```

E dentro de dependencies, foi inserido o comando:

```
<dependency>
   <groupId>org.springdoc</groupId>
   <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
   <version>${springdoc.openapi.starter.webmvc.ui.version}</version>
</dependency>
```

Do curso da DIO

Ao invés de utilizar a mesma versão de dependência que foi utilizado no curso, a mais recente que é compatível com a nova versão do spring boot é:

```
<dependency>
   <groupId>io.springfox</groupId>
   <artifactId>springfox-boot-starter</artifactId>
   <version>3.0.0</version>
</dependency>
<dependency>
   <groupId>io.springfox</groupId>
   <artifactId>springfox-swagger-ui</artifactId>
   <version>3.0.0</version>
</dependency>
```

Obs: A dependência certa é muito importante. As vezes podemos achar que o erro está na implementação do código, mas muitas vezes está na dependência.

Abrindo no localhost do Swagger está os métodos HTTP da classe Controller, onde é possível realizar testes facilmente e observar os resultados pelo back-end da aplicação, passando ou obtendo valores. Isso é incrível!

Exception Handlers

As aplicações precisam ser resilientes a possíveis comportamentos inesperados, realizando implementações que centralizam e gerenciam este tipo de tratamento de exceções.

Exception Handler: Um manipulador de exceção é o código que estipula o que um programa fará quando um evento anômalo interromper o fluxo normal das instruções desse programa.

Solução 1: Nível do Controller - @ExceptionHandler.

Solução 2: ResponseStatusExceptionResolver

RestControllerAdvice: nos permite consolidar nossos múltiplos ExceptionHandlers espalhados de antes em um único componente global de tratamento de erros.

GlobalExceptionHandler: tratamento de exceções de forma global.

Algumas das nossas exceções estão relacionadas ao domínio ou negócio da nossa aplicação, sendo assim, vamos criar uma classe de exceção que estende *RunTimeException*, que servirá como base para todas as outras exceções de negócio.