01. Boas práticas na API

Date	@30/01/2023
Categoria	Java API
Curso	Spring Boot 3: aplique boas práticas e proteja uma API Rest

Tópicos

- Apresentação
- Preparando o ambiente - Projeto incial
- Padronizando retornos da API
- Devolvendo o código HTTP 201
- Header Location
- Para saber mais: códigos do protocolo HTTP
- Detalhando dados na API
- Faça como eu fiz: ResponseEntity
- O que aprendemos?

Apresentação

Objetivos

- Boas práticas na API
- Tratamento de erros
- Autenticação/ Autorização
- Tokens JWT

Header Location

Você se depara com o seguinte método, em uma classe Controller:

```
@PostMapping
@Transactional
public ResponseEntity cadastrar(@RequestBody @Valid DadosCadastroProduto dados) {
    var produto = new Produto(dados);
    repository.save(produto);

    var uri = new URI("/produtos/{id}");

    return ResponseEntity.ok(new DadosDetalhamentoProduto(produto));
}
```

Levando em consideração as boas práticas de retorno em uma requisição HTTP do tipo **POST**, escolha as alternativas que indicam os problemas do código anterior:

- Não será devolvida uma representação do recurso criado na API.
    - O código anterior devolve um DTO no corpo da resposta, que representa os detalhes do recurso recém criado na API.
- O código anterior não devolverá o código 201 como resposta.
    - Será devolvido o código 200.
- O cabeçalho *Location* será criado de maneira incorreta.
    - No código anterior, não foi utilizada a classe `UriComponentsBuilder`, do Spring, para a criação da URI; em vez disso, a URI foi criada manualmente e de maneira incorreta. Além disso, a URI nem foi adicionada na resposta a ser devolvida pela API.
- A validação das informações não será executada.
    - O código anterior não apresenta problemas quanto à realização das validações com o Bean Validation.

Para saber mais: códigos do protocolo HTTP

O **protocolo HTTP** (*Hypertext Transfer Protocol*, RFC 2616) é o protocolo responsável por fazer a comunicação entre o cliente, que normalmente é um *browser*, e o servidor. Dessa forma, a cada "requisição" feita pelo cliente, o servidor responde se ele obteve sucesso ou não. Se não obtiver sucesso, na maioria das vezes, a resposta do servidor será uma sequência numérica acompanhada por uma mensagem. Se não soubermos o que significa o código de resposta, dificilmente saberemos qual o problema que está acontecendo, por esse motivo é muito importante saber quais são os códigos HTTP e o que significam.

Categoria de códigos

Os códigos HTTP (ou HTTPS) possuem três dígitos, sendo que o primeiro dígito significa a classificação dentro das possíveis cinco categorias.

1XX: *Informativo* – a solicitação foi aceita ou o processo continua em andamento;

2XX: *Confirmação* – a ação foi concluída ou entendida;

3XX: *Redirecionamento* – indica que algo mais precisa ser feito ou precisou ser feito para completar a solicitação;

4XX: *Erro do cliente* – indica que a solicitação não pode ser concluída ou contém a sintaxe incorreta;

5XX: *Erro no servidor* – o servidor falhou ao concluir a solicitação.

Principais códigos de erro

Como dito anteriormente, conhecer os principais códigos de erro HTTP vai te ajudar a identificar problemas em suas aplicações, além de permitir que você entenda melhor a comunicação do seu navegador com o servidor da aplicação que está tentando acessar.

Error 403

O código 403 é o erro "Proibido". Significa que o servidor entendeu a requisição do cliente, mas se recusa a processá-la, pois o cliente não possui autorização para isso.

Error 404

Quando você digita uma URL e recebe a mensagem *Error 404*, significa que essa URL não te levou a lugar nenhum. Pode ser que a aplicação não exista mais, a URL

mudou ou você digitou a URL errada.

Error 500

É um erro menos comum, mas de vez em quando ele aparece. Esse erro significa que há um problema com alguma das bases que faz uma aplicação rodar. Esse erro pode ser, basicamente, no servidor que mantém a aplicação no ar ou na comunicação com o sistema de arquivos, que fornece a infraestrutura para a aplicação.

Error 503

O erro 503 significa que o serviço acessado está temporariamente indisponível. Causas comuns são um servidor em manutenção ou sobrecarregado. Ataques maliciosos, como o DDoS, causam bastante esse problema.

Uma dica final: dificilmente conseguimos guardar em nossa cabeça o que cada código significa, portanto, existem sites na internet que possuem todos os códigos e os significados para que possamos consultar quando necessário. Existem dois sites bem conhecidos e utilizados por pessoas desenvolvedoras, um para cada preferência: se você gosta de gatos, pode utilizar o HTTP Cats; já, se prefere cachorros, utilize o HTTP Dogs.

O que aprendemos?

Nessa aula, você aprendeu como:

- Utilizar a classe `ResponseEntity`, do Spring, para personalizar os retornos dos métodos de uma classe Controller;
- Modificar o código HTTP devolvido nas respostas da API;
- Adicionar cabeçalhos nas respostas da API;
- Utilizar os códigos HTTP mais apropriados para cada operação realizada na API.
