Aula 6: Tratamento de Erros com Código 404

September 18, 2025

1 Introdução

Na Aula 5, configuramos o `application.properties` para remover o stack trace de respostas de erro 500 (Internal Server Error) no endpoint de detalhamento. No entanto, quando um ID inexistente é passado (ex.: `GET http://localhost:8080/medicos/6999`), o código 500 ainda é retornado devido à `EntityNotFoundException`. Nesta aula, personalizaremos o tratamento de erros para retornar o código 404 (Not Found) usando uma classe com `@RestControllerAdvice`. As alterações serão feitas no `Visual Studio Code` ou sua IDE preferida, com testes no Insomnia.

2 Revisão do Problema no Endpoint de Detalhamento

O endpoint de detalhamento no `MedicoController` é:

```
@GetMapping("/{id}")
public ResponseEntity detalhar(@PathVariable Long id) {
    var medico = repository.getReferenceById(id);
    return ResponseEntity.ok(new DadosDetalhamentoMedico(medico));
}
```

Quando testado com `GET http://localhost:8080/medicos/6`, retorna 200 (OK) com os dados do médico:

```
{
    "id": 6,
    "nome": "Nome do Médico",
    "email": "medico@voll.med",
    "crm": "233444",
    "telefone": "61999998888",
    "especialidade": "ORTOPEDIA",
    "endereco": {
        "logradouro": "rua 1",
        "bairro": "bairro",
        "cep": "12345678",
        "numero": null,
        "complemento": null,
        "cidade": "Brasil",
        "uf": "DF"
    }
}
```

Porém, para um ID inexistente (`GET http://localhost:8080/medicos/6999`), o método `repository.getReferenceById(id)` lança uma `EntityNotFoundException`, resultando em um erro 500 com a mensagem:

```
{
    "timestamp": "2022-10-21T18:06:01.320+00:00",
    "status": 500,
    "error": "Internal Server Error",
```

```
    "message": "Unable to find med.voll.api.medico.Medico with id 6999",
    "path": "/medicos/6999"
}
```

O código 500 é inadequado, pois o erro ocorre devido à ausência do recurso, e o código correto seria 404 (Not Found).

3 Reorganizando a Estrutura de Pacotes

Antes de criar o tratamento de erros, organizaremos os pacotes para separar o domínio da infraestrutura. No pacote `med.voll.api`, movemos os subpacotes `endereco`, `medico` e `paciente` para um novo pacote `domain`. A nova estrutura é:

- `med.voll.api`
  - `controller`
  - `domain`
    - `endereco`
    - `medico`
    - `paciente`

Após a refatoração, ajustamos os imports em `MedicoController` e `PacienteController`:

```
import med.voll.api.domain.medico.*;
import med.voll.api.domain.paciente.*;
```

4 Criando a Classe de Tratamento de Erros

Para tratar a `EntityNotFoundException` globalmente, criaremos uma classe no novo pacote `infra` chamado `TratadorDeErros`. O código é:

```
package med.voll.api.infra;

import jakarta.persistence.EntityNotFoundException;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.RestControllerAdvice;

@RestControllerAdvice
public class TratadorDeErros {

    @ExceptionHandler(EntityNotFoundException.class)
    public ResponseEntity tratarErro404() {
        return ResponseEntity.notFound().build();
    }
}
```

4.1 Explicação do Código

- `@RestControllerAdvice`: Define a classe como um manipulador global de exceções para todos os controladores REST.
- `@ExceptionHandler(EntityNotFoundException.class)`: Associa o método `tratarErro404` à `EntityNotFoundException`.
- `ResponseEntity.notFound().build()`: Retorna o código 404 (Not Found) sem corpo na resposta.

Essa abordagem evita a necessidade de `try-catch` no `MedicoController`, mantendo o código enxuto e centralizando o tratamento de erros.

5 Testando no Insomnia

Após salvar a classe `TratadorDeErros`, o Spring DevTools reinicia o projeto. No Insomnia, testamos o endpoint de detalhamento com um ID inexistente:

- **Requisição**: `GET http://localhost:8080/medicos/6999`
- **Resultado**: Código 404 (Not Found), sem corpo na resposta.

Para um ID existente (`GET http://localhost:8080/medicos/6`), o comportamento permanece inalterado, retornando 200 (OK) com os dados do médico.

6 Próximos Passos

Com o tratamento do código 404 implementado, a próxima aula abordará o tratamento do erro 400 (Bad Request) para falhas de validação em requisições de cadastro de médicos ou pacientes. Continue praticando no `Visual Studio Code` ou sua IDE preferida!

7 Dica do Professor

- **Aprofunde-se em tratamento de exceções**: Consulte a documentação do Spring Boot sobre `@RestControllerAdvice` (`https://docs.spring.io/spring-framework/docs/current/reference/html/web.html#mvc-ann-restcontrolleradvice`) para explorar outras estratégias de manipulação de erros.
- **Comunidade no X**: Siga perfis como @SpringGuru e @APIDev no X para dicas sobre tratamento de exceções e boas práticas no Spring Boot.
- **Pratique**: Adicione tratamento de erros semelhante ao `PacienteController` para o endpoint `GET /pacientes/id`, testando no Insomnia com IDs válidos e inválidos.