



Spring Boot

Spring JPA – Evitando erro de serialização

- Quando criamos a relação de GrupoProduto em Produto (@ManyToOne) - idGrupoProduto chave estrangeira em Produto.



```
@ManyToOne 3 usages  
@JoinColumn(name="idgrupoproduto")  
private GrupoProduto grupoProduto;
```

- É padrão criarmos a relação inversa na classe GrupoProduto (relação @OneToMany) que armazenará uma lista de produtos que pertencem ao respectivo grupo de produtos.



```
@JsonIgnore 2 usages  
@OneToMany(mappedBy = "grupoProduto")  
private List<Produto> produtos = new ArrayList<>();
```

- Insira o atributo e gere o Get e Set deste atributo no domínio GrupoProduto.**

```
public List<Produto> getProdutos() { no usages new *  
    return produtos;  
}  
  
public void setProdutos(List<Produto> produtos) { no  
    this.produtos = produtos;  
}
```

- Aqui temos um atributo produtos em uma classe que está representando um relacionamento entre GrupoProduto e Produto em um contexto de persistência de dados com o JPA e controle de serialização com o Jackson (que é a biblioteca usada no Spring para converter objetos Java em JSON e vice-versa).
- **Atributo produtos:**
- O atributo produtos é uma **lista de objetos Produto** que pertence a uma classe, provavelmente GrupoProduto.
- Ele representa um relacionamento de **"um para muitos" entre GrupoProduto e Produto**, indicando que um grupo de produtos pode ter vários produtos associados.

```
@JsonIgnore 2 usages  
@OneToMany(mappedBy = "grupoProduto")  
private List<Produto> produtos = new ArrayList<>();
```

```
@JsonIgnore 2 usages  
@OneToMany(mappedBy = "grupoProduto")  
private List<Produto> produtos = new ArrayList<>();
```

- **2. Anotação @OneToMany:**

- A anotação **@OneToMany** define que um objeto da classe (no caso, GrupoProduto) pode estar associado a muitos objetos da outra classe (Produto).
- **mappedBy = "grupoProduto":** Este parâmetro indica que o lado dono do relacionamento está na classe Produto, no campo grupoProduto. Isso quer dizer que a classe Produto tem uma referência ao GrupoProduto, e essa chave estrangeira é usada para mapear o relacionamento.
- No JPA, isso é conhecido como o lado inverso do relacionamento. O GrupoProduto sabe quais são seus produtos por meio dessa lista, mas não é ele quem controla a relação diretamente no banco de dados (essa responsabilidade está na classe Produto).


```
@JsonIgnore 2 usages  
@OneToMany(mappedBy = "grupoProduto")  
private List<Produto> produtos = new ArrayList<>();
```

- **3. Anotação @JsonIgnore:**
- A anotação **@JsonIgnore** vem da biblioteca Jackson e é usada para **impedir que o atributo produtos seja serializado quando o objeto GrupoProduto for convertido em JSON.**
- **Sem essa anotação**, ao serializar GrupoProduto, o Spring tentaria incluir a lista produtos no JSON gerado, o que **pode causar problemas de performance ou mesmo problemas de serialização cíclica** (explicados abaixo).

```
@JsonIgnore 2 usages
@OneToMany(mappedBy = "grupoProduto")
private List<Produto> produtos = new ArrayList<>();
```

- **1. O problema da relação bidirecional:**
- Quando você tem um relacionamento bidirecional no JPA, como o relacionamento entre GrupoProduto e Produto, **a serialização JSON pode entrar em um loop infinito**. Isso ocorre quando:
 - A entidade GrupoProduto contém uma lista de Produto.
 - Cada Produto tem uma referência de volta ao GrupoProduto.
 - Isso cria um ciclo:
 - Quando você serializa um GrupoProduto, ele tenta serializar a lista de Produtos.
 - Cada Produto então tenta serializar o GrupoProduto ao qual pertence, o que, por sua vez, tenta novamente serializar os Produtos – e isso continua indefinidamente.

```
@JsonIgnore 2 usages
@OneToMany(mappedBy = "grupoProduto")
private List<Produto> produtos = new ArrayList<>();
```

- **2. Exemplo do problema de serialização:**
- Imagine as classes 
- Sem @JsonIgnore, ao tentar serializar um GrupoProduto em JSON:
 1. GrupoProduto é serializado.
 2. Ele contém uma lista de Produtos.
 3. Cada Produto tenta serializar seu campo grupoProduto.
 4. Esse grupoProduto tem uma lista de Produtos, que são serializados de novo.
 5. E assim o ciclo continua, resultando em um loop infinito.

```
@Entity
public class GrupoProduto {

    @OneToMany(mappedBy = "grupoProduto")
    private List<Produto> produtos = new ArrayList<>();

    // getters e setters
}


@Entity
public class Produto {

    @ManyToOne
    @JoinColumn(name = "id_grupo_produto")
    private GrupoProduto grupoProduto;

    // getters e setters
}
```



```
@JsonIgnore 2 usages
@OneToMany(mappedBy = "grupoProduto")
private List<Produto> produtos = new ArrayList<>();
```

- **2. Exemplo do problema de serialização:**
- Imagine as classes 
- Sem @JsonIgnore, ao tentar serializar um GrupoProduto em JSON:
 1. GrupoProduto é serializado.
 2. Ele contém uma lista de Produtos.
 3. Cada Produto tenta serializar seu campo grupoProduto.
 4. Esse grupoProduto tem uma lista de Produtos, que são serializados de novo.
 5. E assim o ciclo continua, resultando em um loop infinito.

```
@Entity
public class GrupoProduto {

    @OneToMany(mappedBy = "grupoProduto")
    private List<Produto> produtos = new ArrayList<>();

    // getters e setters
}

@Entity
public class Produto {

    @ManyToOne
    @JoinColumn(name = "id_grupo_produto")
    private GrupoProduto grupoProduto;

    // getters e setters
}
```


- **2. Exemplo do problema de serialização (Continuação)**
- Esse ciclo pode causar erros como **StackOverflowError** ou tornar a serialização extremamente lenta.
- **3. Como o @JsonIgnore resolve isso:**
- A anotação @JsonIgnore em GrupoProduto informa ao Jackson que ele deve ignorar a lista de produtos (produtos) ao serializar GrupoProduto para JSON. Isso evita que o ciclo de serialização ocorra, resolvendo o problema de loop infinito.
- Por exemplo:

```
@JsonIgnore 2 usages
@OneToMany(mappedBy = "grupoProduto")
private List<Produto> produtos = new ArrayList<>();
```

- Ao aplicar @JsonIgnore, a lista de Produtos não será incluída no JSON quando você serializar GrupoProduto. Isso quebra o ciclo e impede o loop de serialização.

```
@Entity
public class GrupoProduto {

    @OneToMany(mappedBy = "grupoProduto")
    private List<Produto> produtos = new ArrayList<>();

    // getters e setters
}

@Entity
public class Produto {

    @ManyToOne
    @JoinColumn(name = "id_grupo_produto")
    private GrupoProduto grupoProduto;

    // getters e setters
}
```

Exemplo

- **4. Alternativas ao @JsonIgnore:**
- Dependendo da necessidade do projeto, há outras soluções para resolver o problema de serialização cíclica além de @JsonIgnore.
- @JsonManagedReference e @JsonBackReference:
 - A. Outra abordagem é usar essas duas anotações para marcar explicitamente o lado que deve ser serializado (@JsonManagedReference) e o lado que deve ser ignorado na serialização (@JsonBackReference) (Veja o exemplo).
- **@JsonManagedReference:** Aplica-se no lado dono do relacionamento, que será serializado.
- **@JsonBackReference:** Aplica-se no lado inverso, que será ignorado durante a serialização.

```
public class GrupoProduto {
    @JsonManagedReference
    @OneToMany(mappedBy = "grupoProduto")
    private List<Produto> produtos = new ArrayList<>();
}

public class Produto {
    @JsonBackReference
    @ManyToOne
    @JoinColumn(name = "id_grupo_produto")
    private GrupoProduto grupoProduto;
}
```

- **4. Alternativas ao @JsonIgnore (Continuação)**
- **@JsonIdentityInfo:**
- Essa anotação pode ser usada para identificar entidades por um identificador (ID) e evitar a serialização completa de objetos repetidos, serializando-os como referências ao invés de repetir os dados (Veja o exemplo).

```
public class GrupoProduto {  
    @JsonManagedReference  
    @OneToMany(mappedBy = "grupoProduto")  
    private List<Produto> produtos = new ArrayList<>();  
}  
  
public class Produto {  
    @JsonBackReference  
    @ManyToOne  
    @JoinColumn(name = "id_grupo_produto")  
    private GrupoProduto grupoProduto;  
}
```

- **Resumo:**
- **@OneToMany(mappedBy = "grupoProduto"):** Define o relacionamento de um para muitos entre GrupoProduto e Produto, onde muitos produtos pertencem a um grupo.
- **@JsonIgnore:** Impede que a lista de produtos (produtos) seja serializada em JSON quando você serializar a entidade GrupoProduto. Isso é fundamental para evitar problemas de serialização cíclica, onde a serialização poderia entrar em um loop infinito.
- **Problema de serialização cíclica:** Quando você tem uma relação bidirecional (como GrupoProduto e Produto), a serialização de um lado pode causar um ciclo infinito, pois cada entidade faz referência à outra. O @JsonIgnore ou outras alternativas (como @JsonManagedReference e @JsonBackReference) ajudam a resolver esse problema.



Spring Boot

Criando o endpoint findById

PROF. ME. JEFFERSON PASSERINI

- Vamos implementar uma busca por id. Nosso id neste caso é do tipo UUID.
- Vamos implementar o **findById()** em nossas classes **GrupoProdutoService** e **ProdutoService**

```
1 package com.curso.services;
2
3 > import ...
4
13 @Service 2 usages jeffersonarpasserini *
14 public class GrupoProdutoService {
15
16     @Autowired 2 usages
17     private GrupoProdutoRepository grupoProdutoRepo;
18
19     public List<GrupoProdutoDTO> findAll(){ 1 usage jeffersonarpasserini
20         //retorna uma lista de ProdutoDTO
21         return grupoProdutoRepo.findAll().stream() Stream<GrupoProdu
22             .map(obj -> new GrupoProdutoDTO(obj)) Stream<GrupoPr
23             .collect(Collectors.toList());
24     }
25
26     public GrupoProduto findById(int id){ no usages new *
27         Optional<GrupoProduto> obj = grupoProdutoRepo.findById(id);
28         return obj.orElse( other: null);
29     }
30
31 }
```

```
1 package com.curso.services;
2
3 > import ...
4
14 @Service 2 usages jeffersonarpasserini *
15 public class ProdutoService {
16
17     @Autowired 2 usages
18     private ProdutoRepository produtoRepo;
19
20     public List<ProdutoDTO> findAll(){ 1 usage jeffersonarpasserini
21         //retorna uma lista de ProdutoDTO
22         return produtoRepo.findAll().stream() Stream<Produto
23             .map(obj -> new ProdutoDTO(obj)) Stream<Produto
24             .collect(Collectors.toList());
25     }
26
27     public Produto findById(Long id){ no usages new *
28         Optional<Produto> obj = produtoRepo.findById(id);
29         return obj.orElse( other: null);
30     }
31
32 }
```

- Como o objeto para retornar NULL da busca realizada então nosso retorno tem essa previsão.

- Agora na camada Resources implemente o **endpoint findById()**.
- A anotação @GetMapping agora recebe um parâmetro **{id}**
- Agora devemos informar o id no caminho de nosso endpoint (como no exemplo).
- Quando estamos desenvolvendo buscas por atributos de nosso domain podemos utilizar a junção de **findBy + o nome do atributo** que deseja buscar que no Repository nosso JPA cuidará automaticamente de busca.

```
1 package com.curso.resources;
2
3 > import ...
4
5
6
7
8
9
10
11
12
13
14
15
16 @RestController @jeffersonarpasserini *
17 @RequestMapping(value = "/produto")
18 public class ProdutoResource {
19
20     @Autowired
21     private ProdutoService produtoService;
22
23     @GetMapping //exemplo http://localhost:8080/produto @jeffersonarpasserini
24     public ResponseEntity<List<ProdutoDTO>> findAll(){
25         return ResponseEntity.ok().body(produtoService.findAll());
26     }
27
28     @GetMapping(value =("/{id}") //exemplo http://localhost:8080/produto/{id}
29     public ResponseEntity<ProdutoDTO> findById(@PathVariable Long id){
30         Produto obj = this.produtoService.findById(id);
31         return ResponseEntity.ok().body(new ProdutoDTO(obj));
32     }
33 }
```


- Agora na camada Resources implemente o **endpoint findById() - GrupoProdutoResource**.

```
1 package com.curso.resources;
2
3 > import ...
4
14
15 @RestController @jeffersonarpasserini *
16 @RequestMapping(value = "/grupoproduto")
17 public class GrupoProdutoResource {
18
19     @Autowired
20     private GrupoProdutoService grupoProdutoService;
21
22     @GetMapping //exemplo http://localhost:8080/produto @jeffersonarpasserini
23     public ResponseEntity<List<GrupoProdutoDTO>> findAll(){
24         return ResponseEntity.ok().body(grupoProdutoService.findAll());
25     }
26
27     @GetMapping(value =("/{id}") //exemplo http://localhost:8080/grupoproduto/1
28     public ResponseEntity<GrupoProdutoDTO> findById(@PathVariable Integer id){
29         GrupoProduto obj = this.grupoProdutoService.findById(id);
30         return ResponseEntity.ok().body(new GrupoProdutoDTO(obj));
31     }
32 }
```

- Testando nossos endpoints Produto e GrupoProduto.
- Primeiro faça uma busca (listar) no endpoint findAll() para pegar um id para nosso teste.

HTTP SuporteOS2024 / produto Save Share

GET http://localhost:8080/produto Send

Params Auth Headers (6) Body Scripts Settings

Query Params

Key	Value	Descri...	Bulk Edit
Key	Value	Description	

Body 200 OK • 25 ms • 992 B

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "idProduto": 1,
4     "descricao": "Coca-Cola",
5     "saldoEstoque": 100.000,
6     "valorUnitario": 3.500,
7     "valorEstoque": 350.00,
8     "dataCadastro": "03/10/2024",
9     "grupoProduto": 2,
10    "descricaoGrupoProduto": "Alimenticio",
11    "status": 1
12  },
13  {
14    "idProduto": 2,
15    "descricao": "Guarana Antartica",
16    "saldoEstoque": 200.000,
17    "valorUnitario": 3.000,
```

HTTP SuporteOS2024 / grupoproduto Save Share

GET http://localhost:8080/grupoproduto Send

Params Auth Headers (6) Body Scripts Settings

Query Params

Key	Value	Descri...	Bulk Edit
Key	Value	Description	

Body 200 OK • 234 ms • 253 B

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": 1,
4     "descricao": "Limpeza",
5     "status": 1
6   },
7   {
8     "id": 2,
9     "descricao": "Alimenticio",
10    "status": 1
11  }
12 ]
```

Criando Projeto Spring Boot – – findById()

- Com o id passe junto ao endpoint.
- `http://localhost:8080/grupoproduto/1`
- O id será capturado por nosso endpoint `findById()`.

```
@GetMapping(value = "{id}") //exemplo http://localhost:8080/grupoproduto/1
public ResponseEntity<GrupoProdutoDTO> findById(@PathVariable Integer id){
    GrupoProduto obj = this.grupoProdutoService.findById(id);
    return ResponseEntity.ok().body(new GrupoProdutoDTO(obj));
}
```

- Faça o mesmo teste para produto por Id.

The screenshot shows a REST client interface. At the top, the URL bar displays 'http://localhost:8080/grupoproduto/1' with a 'GET' method selected. Below the URL bar, the 'Query Params' section is empty. The response body is shown in JSON format, indicating a successful 200 OK status. The JSON data represents a product with id 1, description 'Limpeza', and status 1.

Key	Value	Descri...
Key	Value	Description

Body 200 OK • 16 ms • 205 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 1,
3   "descricao": "Limpeza",
4   "status": 1
5 }
```



Spring Boot

Criando o endpoint findByCodigoBarra() para Produto

PROF. ME. JEFFERSON PASSERINI

- Altere a model de Produto e insira o atributo codigoBarra. *Criando Projeto Spring Boot – findByCodigoBarra()*
- **Atualize o construtor da classe e seus gets e sets.**

```
1 package com.curso.domains;
2
3 > import ...
4
13
14 @Entity  jeffersonarpasserini +1 * 4 related problems
15 @Table(name = "produto")
16 public class Produto {
17
18     @Id
19     @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "seq_produto")
20     private long idProduto;
21
22     @NotBlank @NotNull 3 usages
23     private String codigoBarra;
24
25     @NotBlank @NotNull 6 usages
26     private String descricao;
27
28     @NotNull 4 usages
29     @Digits(integer = 15, fraction = 3)
30     private BigDecimal saldoEstoque;
31
32     @NotNull 4 usages
33     @Digits(integer = 15, fraction = 3)
34     private BigDecimal valorUnitario;
35 }
```

- Altere a model de Produto e insira o atributo codigoBarra. *Criando Projeto Spring Boot – findByCodigoBarra()*
- **Atualize o construtor da classe e seus gets e sets.**

```
public Produto(long idProduto, String codigoBarra, String descricao, BigDecimal saldoEstoque, BigDecimal valorUnitario,
               LocalDate dataCadastro, GrupoProduto grupoProduto, Status status) {
    this.idProduto = idProduto;
    this.codigoBarra = codigoBarra;
    this.descricao = descricao;
    //this.saldoEstoque = saldoEstoque;
    this.valorUnitario = valorUnitario;
    this.dataCadastro = dataCadastro;
    this.grupoProduto = grupoProduto;
    this.status = status;

    this.saldoEstoque = saldoEstoque != null ? saldoEstoque : BigDecimal.ZERO;
    this.valorEstoque = saldoEstoque != null ? saldoEstoque.multiply(valorUnitario) : BigDecimal.ZERO;
}
```

```
public long getIdProduto() { return idProduto; }
```

```
public void setIdProduto(long idProduto) { this.idProduto = idProduto; }
```

```
public String getCodigoBarra() { return codigoBarra; } no usages new *
```

```
public void setCodigoBarra(String codigoBarra) { this.codigoBarra = codigoBarra; } no usages jeffersonarpasserini *
```

- Altere a ProdutoDTO e insira o atributo codigoBarra.

Criando Projeto Spring Boot – findByCodigoBarra()

```
ProdutoDTO.java x
1 package com.curso.domains.dtos;
2
3 > import ...
4
15 public class ProdutoDTO { 7 usages 1 jeffersonarpasserini +1
16
17     private long idProduto; 3 usages
18
19     @NotBlank(message = "O campo código de barras não pode estar vazio") 3 us
20     @NotNull(message = "O campo código de barras não pode ser nulo")
21     private String codigoBarra;
22
23     @NotNull(message = "O campo descrição não pode ser nulo") 3 usages
24     @NotBlank(message = "O campo descrição não pode estar vazio")
25     private String descricao;
26
27     @NotNull(message = "O campo saldoEstoque não pode ser nulo") 3 usages
28     @Digits(integer = 15, fraction = 3)
29     private BigDecimal saldoEstoque;
30
31     @NotNull(message = "O campo valorUnitario não pode ser nulo") 3 usages
32     @Digits(integer = 15, fraction = 3)
33     private BigDecimal valorUnitario;
34
```

```
public ProdutoDTO(Produto produto) { 2 usages 1 jeffersonarpasserini +1
    this.idProduto = produto.getIdProduto();
    this.codigoBarra = produto.getCodigoBarra();
    this.descricao = produto.getDescricao();
    this.valorUnitario = produto.getValorUnitario();
    this.saldoEstoque = produto.getSaldoEstoque();
    this.valorEstoque = produto.getValorEstoque();
    this.dataCadastro = produto.getDataCadastro();
    this.grupoProduto = produto.getGrupoProduto().getId();
    this.descricaoGrupoProduto = produto.getGrupoProduto().getDescricao();
    this.status = produto.getStatus().getId();
}

public long getIdProduto() { return idProduto; }

public void setIdProduto(long idProduto) { this.idProduto = idProduto; }

public @NotBlank(message = "O campo código de barras não pode estar vazio") no usages 1 jeffersonarpasserini *
    @NotNull(message = "O campo código de barras não pode ser nulo") String getCodigoBarra() {
    return codigoBarra;
}

public void setCodigoBarra(@NotBlank(message = "O campo código de barras não pode estar vazio") no usages 1 jefferson
    @NotNull(message = "O campo código de barras não pode ser nulo") String codigoBarra) {
    this.codigoBarra = codigoBarra;
}
```


- Altere a DBService para a carga inicial de dados do produto
- ## Criando Projeto Spring Boot – findByCodigoBarra()

```
1 package com.curso.services;
2
3 > import ...
4
13
14 @Service 4 usages 1 jeffersonarpasserini *
15 public class DBService {
16
17     @Autowired
18     private GrupoProdutoRepository grupoProdutoRepo;
19
20     @Autowired
21     private ProdutoRepository produtoRepo;
22
23     public void initDB(){ 2 usages 1 jeffersonarpasserini *
24
25         GrupoProduto grupo01 = new GrupoProduto( id: 0, descricao: "Limpeza", Status.ATIVO);
26         GrupoProduto grupo02 = new GrupoProduto( id: 0, descricao: "Alimenticio", Status.ATIVO);
27
28         Produto produto01 = new Produto( idProduto: 0, codigoBarra: "1111", descricao: "Coca Cola", new BigDecimal( val: "100"), new BigDecimal( val: "3.5"),
29             LocalDate.now(), grupo02, Status.ATIVO);
30         Produto produto02 = new Produto( idProduto: 0, codigoBarra: "2222", descricao: "Guarana Antartica", new BigDecimal( val: "200"), new BigDecimal( val: "3.0"),
31             LocalDate.now(), grupo02, Status.ATIVO);
32         Produto produto03 = new Produto( idProduto: 0, codigoBarra: "3333", descricao: "Detergente Limpol", new BigDecimal( val: "300"), new BigDecimal( val: "4.0"),
33             LocalDate.now(), grupo01, Status.ATIVO);
34         Produto produto04 = new Produto( idProduto: 0, codigoBarra: "4444", descricao: "Sabão em Pó OMO", new BigDecimal( val: "400"), new BigDecimal( val: "15.5"),
35             LocalDate.now(), grupo02, Status.ATIVO);
36
37         grupoProdutoRepo.save(grupo01);
38         grupoProdutoRepo.save(grupo02);
39         produtoRepo.save(produto01);
40         produtoRepo.save(produto02);
41         produtoRepo.save(produto03);
42         produtoRepo.save(produto04);
43
44     }
45 }
```

- Vamos implementar uma busca pelo código de barras do produto.
- Para isso temos que declarar a assinatura do método findByCodigoBarra() na classe ProdutoRepository.
- O JPA cuidará da sua operacionalização temos apenas que declarar o método.
- O retorno é do tipo Optional<> pois pode-se ou não encontrar um técnico na busca.

```
1 package com.curso.repositories;
2
3 > import ...
4
5 @Repository 4 usages  jeffersonarpasserini *
6 public interface ProdutoRepository extends JpaRepository<Produto, Long> {
7
8     Optional<Produto> findByCodigoBarra(String codigoBarra); 1 usage  new *
9
10 }
11
12
13
14
```

Criando Projeto Spring Boot – findByCodigoBarra()

```
1 package com.curso.services;
2
3 > import ...
4
15 @Service 2 usages 1 jeffersonarpasserini +1 *
16 public class ProdutoService {
17
18     @Autowired
19     private ProdutoRepository produtoRepo;
20
21     public List<ProdutoDTO> findAll(){ 1 usage 1 jeffersonarpasserini
22         //retorna uma lista de ProdutoDTO
23         return produtoRepo.findAll().stream() Stream<Produto>
24             .map(obj -> new ProdutoDTO(obj)) Stream<ProdutoDTO>
25             .collect(Collectors.toList());
26     }
27
28     public Produto findById(Long id){ 1 usage 1 jeffersonarpasserini
29         Optional<Produto> obj = produtoRepo.findById(id);
30         return obj.orElse(other: null);
31     }
32
33     public Produto findByCodigoBarra(String codigoBarra){ 1 usage new *
34         Optional<Produto> obj = produtoRepo.findByCodigoBarra(codigoBarra);
35         return obj.orElse(other: null);
36     }
37 }
```

- Agora na camada Service desenvolva o código como demonstrado.

Criando Projeto Spring Boot – findByCodigoBarra()

- Agora na camada Resources desenvolva o código do endpoint **findByCodigoBarra()**.

- Neste endpoint temos:

- **http://localhost:8080/produto/codigobarra/1111**

- Em **vermelho** o caminho para o serviço, em **laranja** a definição de nosso endpoint para o atributo código de barra e em **verde** a informação a ser buscada.

```
1 package com.curso.resources;
2
3 > import ...
4
5
6 @RestController
7 @RequestMapping(value = "/produto")
8 public class ProdutoResource {
9
10     @Autowired
11     private ProdutoService produtoService;
12
13
14     @GetMapping //exemplo http://localhost:8080/produto
15     public ResponseEntity<List<ProdutoDTO>> findAll() { return ResponseEntity.ok().body(produtoService.findAll()); }
16
17
18     @GetMapping(value =("/{id}") //exemplo http://localhost:8080/produto/1
19     public ResponseEntity<ProdutoDTO> findById(@PathVariable Long id){
20         Produto obj = this.produtoService.findById(id);
21         return ResponseEntity.ok().body(new ProdutoDTO(obj));
22     }
23
24
25     @GetMapping(value = "/codigobarra/{codigoBarra}") //exemplo http://localhost:8080/produto/codigobarra/1
26     public ResponseEntity<ProdutoDTO> findById(@PathVariable String codigoBarra){
27         Produto obj = this.produtoService.findByCodigoBarra(codigoBarra);
28         return ResponseEntity.ok().body(new ProdutoDTO(obj));
29     }
30 }
```

Criando Projeto Spring Boot – findByCodigoBarra()

HTTP SuporteOS2024 / produto/codigobarra Save Share

GET http://localhost:8080/produto/codigobarra/1111 Send

Params Auth Headers (6) Body Scripts Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body 200 OK • 29 ms • 385 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "idProduto": 1,
3   "codigoBarra": "1111",
4   "descricao": "Coca Cola",
5   "saldoEstoque": 100.000,
6   "valorUnitario": 3.500,
7   "valorEstoque": 350.00,
8   "dataCadastro": "03/10/2024",
9   "grupoProduto": 2,
10  "descricaoGrupoProduto": "Alimenticio",
11  "status": 1
12 }
```

- Realize os testes com o Postman.
- No exemplo estamos buscando pelo Código de barras: **1111**
- **Como podemos verificar ainda temos uma requisição do tipo GET**
→ @GetMapping

Criando Projeto Spring Boot – findByIdCodigoBarra()

HTTP SuporteOS2024 / produto/codigobarra

GET http://localhost:8080/produto/codigobarra/555

Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (4) Test Results

500 Internal Server Error 157 ms 5.45 KB

Pretty Raw Preview Visualize JSON

```
{
  "timestamp": "2024-10-03T20:30:26.399+00:00",
  "status": 500,
  "error": "Internal Server Error",
  "trace": "java.lang.NullPointerException: Cannot invoke \"com.curso.domains.Produto.getIdProduto()\" because\n  \"produto\" is null\\r\\n\\tat com.curso.domains.dtos.ProdutoDTO.<init>(ProdutoDTO.java:51)\\r\\n\\tat com.curso.\n  resources.ProdutoResource.findById(ProdutoResource.java:37)\\r\\n\\tat java.base/jdk.internal.reflect.\n  NativeMethodAccessorImpl.invoke0(Native Method)\\r\\n\\tat java.base/jdk.internal.reflect.NativeMethodAccessorImpl.\n  invoke(NativeMethodAccessorImpl.java:77)\\r\\n\\tat java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke\n  (DelegatingMethodAccessorImpl.java:43)\\r\\n\\tat java.base/java.lang.reflect.Method.invoke(Method.java:568)\\r\\n\\tat\n  org.springframework.web.method.support.InvocableHandlerMethod.doInvoke(InvocableHandlerMethod.java:259)\\r\\n\\tat\n  org.springframework.web.method.support.InvocableHandlerMethod.invokeForRequest(InvocableHandlerMethod.java:192)\n  \\r\\n\\tat org.springframework.web.servlet.mvc.method.annotation.ServletInvocableHandlerMethod.invokeAndHandle\n  (ServletInvocableHandlerMethod.java:118)\\r\\n\\tat org.springframework.web.servlet.mvc.method.annotation.\n  RequestMappingHandlerAdapter.invokeHandlerMethod(RequestMappingHandlerAdapter.java:920)\\r\\n\\tat org.\n  springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.handleInternal\n  (RequestMappingHandlerAdapter.java:830)\\r\\n\\tat org.springframework.web.servlet.mvc.method.
```

• E se realizarmos a busca por um Código de Barras ou Id **INEXISTENTE**.

• Nossas API's (endpoint) findById() ou findByCpf() irá retornar um erro.

• **Erro: 500 – Internal Server Error - Java.lang.NullPointerException.**

• Pois como nosso Service não encontrou o técnico na busca retornou NULL o que gera o erro no endpoint.

• Mas esse erro precisa ser tratado corretamente para o retorno do endpoint.



Spring Boot

Tratamento de Erros Objeto não Encontrado

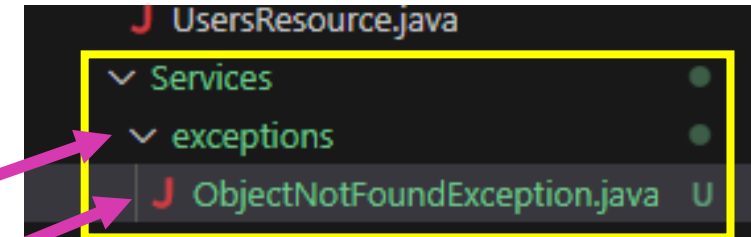
PROF. ME. JEFFERSON PASSERINI


- Para tratarmos nossa exceção de objeto não encontrado vamos:

Crie um sub-pacote denominado **exceptions** dentro de **services**.

Crie uma nova classe:

ObjectNotFoundException



```
1 package com.curso.services.exceptions;  
2   
3 public class ObjectNotFoundException {  
4  
5 }  
6
```

- Nossa classe **ObjectNotFoundException** irá estender a classe **RuntimeException** do Java.

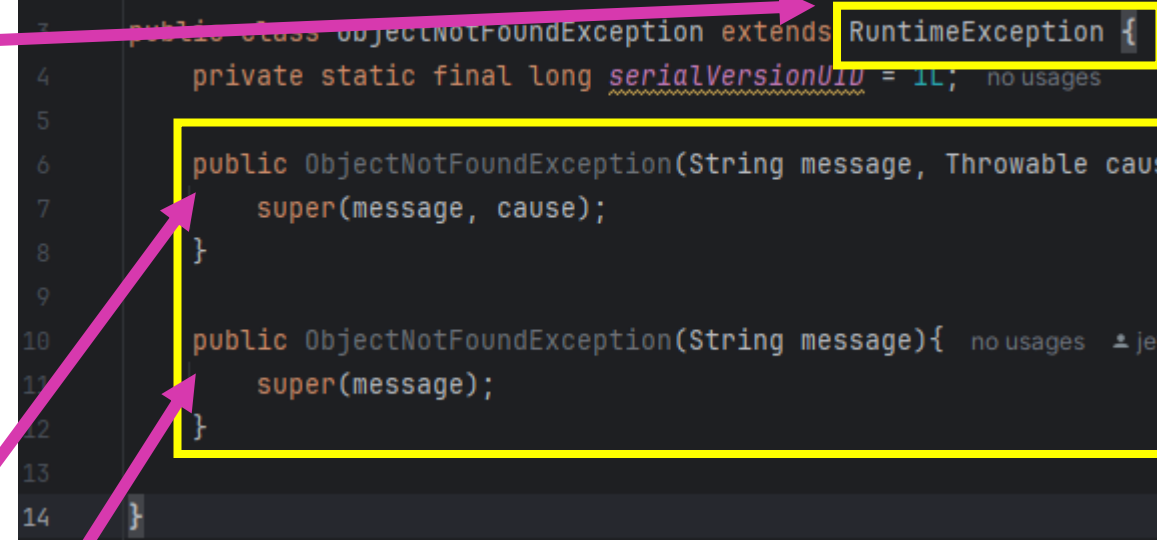
Crie os construtores de nossa classe
acionando os construtores da superclasse
`RuntimeException`.

Construtores:

`ObjectNotFoundException(String message,
Throwable cause)`

`ObjectNotFoundException(String message)`

```
1 package com.curso.services.exceptions;
2
3 public class ObjectNotFoundException extends RuntimeException {
4     private static final long serialVersionUID = 1L;
5
6     public ObjectNotFoundException(String message, Throwable cause){
7         super(message, cause);
8     }
9
10    public ObjectNotFoundException(String message){
11        super(message);
12    }
13
14 }
```



- Agora em nosso ProdutoService podemos alterar os retorno dos métodos findById() e findByCpf().
- Atualmente nossos métodos retornam o objeto encontrado ou o valor NULL.

```
public Produto findById(Long id){ 1 usage 1 jeffersonarpasserini
    Optional<Produto> obj = produtoRepo.findById(id);
    return obj.orElse( other: null);
}

public Produto findByCodigoBarra(String codigoBarra){ 1 usage new *
    Optional<Produto> obj = produtoRepo.findByCodigoBarra(codigoBarra);
    return obj.orElse( other: null);
}
```

```
1 package com.curso.services;
2
3 > import ...
14
15 @Service 2 usages 1 jeffersonarpasserini +1 *
16 public class ProdutoService {
17
18     @Autowired
19     private ProdutoRepository produtoRepo;
20
21     public List<ProdutoDTO> findAll(){ 1 usage 1 jeffersonarpasserini
22         //retorna uma lista de ProdutoDTO
23         return produtoRepo.findAll().stream() Stream<Produto>
24             .map(obj -> new ProdutoDTO(obj)) Stream<ProdutoDTO>
25             .collect(Collectors.toList());
26     }
27
28     public Produto findById(Long id){ 1 usage 1 jeffersonarpasserini *
29         Optional<Produto> obj = produtoRepo.findById(id);
30         return obj.orElseThrow (() -> new ObjectNotFoundException("Produto não encontrado! Id: "+id));
31     }
32
33     public Produto findByCodigoBarra(String codigoBarra){ 1 usage new *
34         Optional<Produto> obj = produtoRepo.findByCodigoBarra(codigoBarra);
35         return obj.orElseThrow(
36             () -> new ObjectNotFoundException("Produto não encontrado! CodigoBarra: "+codigoBarra)
37         );
38     }
39 }
```

- Agora nossos métodos se não encontrarem o técnico irão retornar uma exceção.

GET http://localhost:8080/produto/codigobarra/555 Send

Params Authorization Headers (6) Body Scripts Settings

Cookies

Body Cookies Headers (4) Test Results

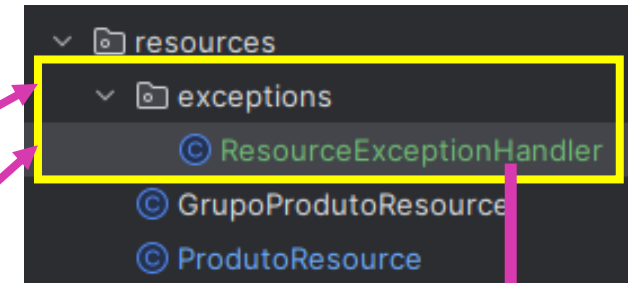
500 Internal Server Error • 159 ms • 5.56 KB • Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "timestamp": "2024-10-03T20:42:08.166+00:00",
3   "status": 500,
4   "error": "Internal Server Error",
5   "trace": "com.curso.services.exceptions.ObjectNotFoundException: Produto não encontrado! Codigobarra: 555\\r\\n\\tat com
        java:36)\\r\\n\\tat java.base/java.util.Optional.orElseThrow(Optional.java:403)\\r\\n\\tat com.curso.services.ProdutoSe
        resources.ProdutoResource.findById(ProdutoResource.java:36)\\r\\n\\tat java.base/jdk.internal.reflect.NativeMethodAc
        NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:77)\\r\\n\\tat java.base/jdk.internal.reflect.Delegati
        \\r\\n\\tat java.base/java.lang.reflect.Method.invoke(Method.java:568)\\r\\n\\tat org.springframework.web.method.support
        \\r\\n\\tat org.springframework.web.method.support.InvocableHandlerMethod.invokeForRequest(InvocableHandlerMethod.ja
        ServletInvocableHandlerMethod.invokeAndHandle(ServletInvocableHandlerMethod.java:118)\\r\\n\\tat org.springframework
        invokeHandlerMethod(RequestMappingHandlerAdapter.java:920)\\r\\n\\tat org.springframework.web.servlet.mvc.method.annotation
        (RequestMappingHandlerAdapter.java:830)\\r\\n\\tat org.springframework.web.servlet.mvc.method.AbstractHandlerMethodA
        springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:1089)\\r\\n\\tat org.springframework
        \\r\\n\\tat org.springframework.web.servlet.FrameworkServlet.processRequest(FrameworkServlet.java:1014)\\r\\n\\tat org.
        java:903)\\r\\n\\tat jakarta.servlet.http.HttpServlet.service(HttpServlet.java:564)\\r\\n\\tat org.springframework.web.servlet.FrameworkServlet.service(FrameworkServlet.java:885)\\r\\n\\tat
        jakarta.servlet.http.HttpServlet.service(HttpServlet.java:658)\\r\\n\\tat org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:206)\\r\\n\\tat org.
        apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:149)\\r\\n\\tat org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:51)\\r\\n\\tat org.apache.
        catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:174)\\r\\n\\tat org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:149)
        \\r\\n\\tat org.springframework.web.filter.RequestContextFilter.doFilterInternal(RequestContextFilter.java:100)\\r\\n\\tat org.springframework.web.filter.OncePerRequestFilter.doFilter
        (OncePerRequestFilter.java:116)\\r\\n\\tat org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:174)\\r\\n\\tat org.apache.catalina.core.
        ApplicationFilterChain.doFilter(ApplicationFilterChain.java:149)\\r\\n\\tat org.springframework.web.filter.CharacterEncodingFilter.
        doFilterInternal(CharacterEncodingFilter.java:201)\\r\\n\\tat org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:116)\\r\\n\\tat org.apache.catalina.core.
        ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:174)\\r\\n\\tat org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:149)\\r\\n\\tat org.
        apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:167)\\r\\n\\tat org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:90)\\r\\n\\tat org.
        apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:482)\\r\\n\\tat org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:115)\\r\\n\\tat org.
        apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:93)\\r\\n\\tat org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:74)\\r\\n\\tat org.apache.
        catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:344)\\r\\n\\tat org.apache.coyote.http11.Http11Processor.service(Http11Processor.java:391)\\r\\n\\tat org.apache.coyote.
        AbstractProcessorLight.process(AbstractProcessorLight.java:63)\\r\\n\\tat org.apache.coyote.AbstractProtocol$ConnectionHandler.process(AbstractProtocol.java:896)\\r\\n\\tat org.apache.tomcat.
        util.net.NioEndpoint$SocketProcessor.doRun(NioEndpoint.java:1744)\\r\\n\\tat org.apache.tomcat.util.net.SocketProcessorBase.run(SocketProcessorBase.java:52)\\r\\n\\tat org.apache.tomcat.util.
        threads.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1191)\\r\\n\\tat org.apache.tomcat.util.threads.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:659)\\r\\n\\tat org.apache.
        tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:63)\\r\\n\\tat java.base/java.lang.Thread.run(Thread.java:840)\\r\\n",
6   "message": "Produto não encontrado! Codigobarra: 555",
7   "path": "/produto/codigobarra/555"
8 }
```

Agora nossa mensagem de erro está padronizada, mas ainda continua causando a exceção em nossos endpoint's.

- Precisamos tratar o recebimento da exceção que geramos na camada Services em nossa camada Resources para isso vamos:
- Vamos criar um sub-pacote denominado **exceptions** no pacote **resources**.
- Dentro deste sub-pacote crie a classe **ResourceExceptionHandler**, que será responsável por tratar nossas exceções.

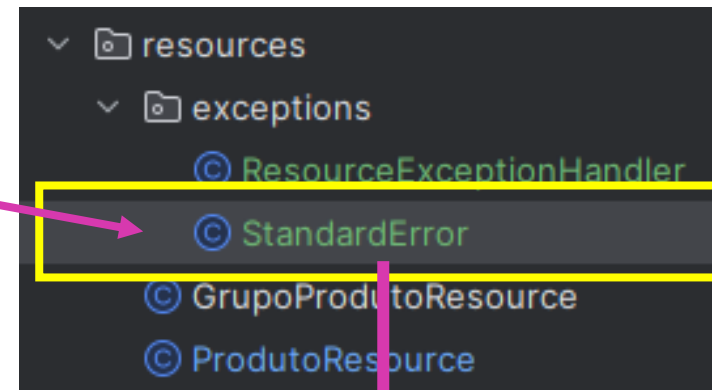


```
1 package com.curso.resources.exceptions;  
2  
3 public class ResourceExceptionHandler {  
4  
5 }  
6
```

- Dentro do sub-pacote denominado **exceptions** no pacote **resources**, vamos criar uma nova classe denominada **StandardError**.
- Na classe **StandardError** devemos **colocar os atributos de acordo com o retorno da exceção** gerada pelo **RuntimeException** do java.

```
{
  "timestamp": "2024-05-26T12:48:27.975+00:00",
  "status": 500,
  "error": "Internal Server Error",
  "trace": "com.curso.Services.exceptions.ObjectNo
  java.lang.IllegalArgumentException: CPF Inválido
  "message": "Objeto não encontrado! CPF:89308",
  "path": "/technician/cpf/89308"
}
```

Criando Projeto Spring Boot – Object Not Found



```
1 package com.curso.resources.exceptions;
2
3 public class StandardError {
4
5 }
6
```

Criando Projeto Spring Boot – Object Not Found

- Implemente a classe **StandardError** de acordo com o código ao lado.

```
1 package com.curso.resources.exceptions;
2
3 import java.io.Serializable;
4
5 public class StandardError implements Serializable { no usages new *
6
7     private static final long serialVersionUID = 1L; no usages
8
9     private Long timeStamp; 3 usages
10    private Integer status; 3 usages
11    private String error; 3 usages
12    private String message; 3 usages
13    private String path; 3 usages
14
15    public StandardError() { no usages new *
16        super();
17    }
18
19    public StandardError(Long timeStamp, Integer status, String error, String message, String path)
20        super();
21        this.timeStamp = timeStamp;
22        this.status = status;
23        this.error = error;
24        this.message = message;
25        this.path = path;
26    }
27
28    public Long getTimeStamp() { no usages new *
29        return timeStamp;
30    }
31
32    public void setTimeStamp(Long timeStamp) { no usages new *
33        this.timeStamp = timeStamp;
34    }
```

```
    "timestamp": "2024-05-26T12:48:27.975+00:00",
    "status": 500,
    "error": "Internal Server Error",
    "trace": "com.curso.Services.exceptions.ObjectNot
    java.lang.IllegalArgumentException: CPF inválido
    "message": "Objeto não encontrado! CPF:89308",
    "path": "/technician/cpf/89308"
```

```
35
36    public Integer getStatus() { no usages new *
37        return status;
38    }
39
40    public void setStatus(Integer status) { no usages
41        this.status = status;
42    }
43
44    public String getError() { no usages new *
45        return error;
46    }
47
48    public void setError(String error) { no usages
49        this.error = error;
50    }
51
52    public String getMessage() { no usages new *
53        return message;
54    }
55
56    public void setMessage(String message) { no usages
57        this.message = message;
58    }
59
60    public String getPath() { no usages new *
61        return path;
62    }
63
64    public void setPath(String path) { no usages new
65        this.path = path;
66    }
67
68 }
```



```
1 package com.curso.resources.exceptions;
2
3 import com.curso.services.exceptions.ObjectNotFoundException;
4 import jakarta.servlet.http.HttpServletRequest;
5 import org.springframework.http.HttpStatus;
6 import org.springframework.http.ResponseEntity;
7 import org.springframework.web.bind.annotation.ControllerAdvice;
8 import org.springframework.web.bind.annotation.ExceptionHandler;
9 import org.springframework.web.context.request.WebRequest;
10
11 @ControllerAdvice new *
12 public class ResourceExceptionHandler {
13
14     @ExceptionHandler new *
15     public ResponseEntity<StandardError> objectNotFoundException(ObjectNotFoundException ex, HttpServletRequest request)
16     {
17
18         StandardError error = new StandardError(System.currentTimeMillis(), HttpStatus.NOT_FOUND.value(),
19         error: "Object not found", ex.getMessage(), request.getRequestURI());
20
21         return ResponseEntity.status(HttpStatus.NOT_FOUND).body(error);
22     }
23 }
```

- **Altere a classe ResourceExceptionHandler em Resources → Exceptions.**
- **Implemente conforme a figura e não se esqueça das annotations.**

Criando Projeto Spring Boot – Object Not Found

```
1 package com.curso.resources.exceptions;
2
3 import org.springframework.http.HttpStatus;
4 import org.springframework.http.ResponseEntity;
5 import org.springframework.web.bind.annotation.ControllerAdvice;
6 import org.springframework.web.bind.annotation.ExceptionHandler;
7
8 import com.curso.Services.exceptions.ObjectNotFoundException;
9
10 import jakarta.servlet.http.HttpServletRequest;
11
12 @ControllerAdvice
13 public class ResourceExceptionHandler {
14     @ExceptionHandler(ObjectNotFoundException.class)
15     public ResponseEntity<StandardError> objectNotFoundException(Object
16
17         StandardError error = new StandardError(System.currentTimeMillis(),
18             ex.getMessage(), request
19
20         return ResponseEntity.status(HttpStatus.NOT_FOUND).body(error);
21     }
22 }
23
24
```

• @ControllerAdvice:

- **Descrição:** Esta anotação é usada para definir uma classe que vai tratar exceções em todo o aplicativo, ou seja, ela permite lidar com exceções lançadas por métodos anotados com @RequestMapping ou qualquer uma de suas variantes.
- **Função:** Indica que a classe ResourceExceptionHandler irá monitorar e fornecer conselhos (ou seja, tratar exceções) para todos os controladores no contexto da aplicação.

• @ExceptionHandler(ObjectNotFoundException.class):

- **Descrição:** Esta anotação é usada para definir um método que lida com uma exceção específica ou uma lista de exceções. Neste caso, a exceção específica é ObjectNotFoundException.
- **Função:** O método objectNotFoundException será invocado sempre que uma exceção do tipo ObjectNotFoundException for lançada em qualquer lugar do aplicativo.

```
StandardError error = new StandardError(System.currentTimeMillis(), HttpStatus.NOT_FOUND.value(), error: "Object not found",  
ex.getMessage(), request.getRequestURI());
```

A classe HttpStatus enumera todos os códigos de status HTTP padronizados, organizando-os em grupos baseados em suas categorias. Aqui estão alguns exemplos de constantes definidas na enumeração HttpStatus:

- **1xx Informational:**
 - HttpStatus.CONTINUE (100)
 - HttpStatus.SWITCHING_PROTOCOLS (101)
- **2xx Success:**
 - HttpStatus.OK (200)
 - HttpStatus.CREATED (201)
 - HttpStatus.ACCEPTED (202)
- **3xx Redirection:**
 - HttpStatus.MOVED_PERMANENTLY (301)
 - HttpStatus.FOUND (302)
 - HttpStatus.SEE_OTHER (303)
- **4xx Client Error:**
 - HttpStatus.BAD_REQUEST (400)
 - HttpStatus.UNAUTHORIZED (401)
 - HttpStatus.FORBIDDEN (403)
 - HttpStatus.NOT_FOUND (404)
- **5xx Server Error:**
 - HttpStatus.INTERNAL_SERVER_ERROR (500)
 - HttpStatus.NOT_IMPLEMENTED (501)
 - HttpStatus.BAD_GATEWAY (502)
 - HttpStatus.SERVICE_UNAVAILABLE (503)

Métodos comuns do HttpStatus:

```
StandardError error = new StandardError(System.currentTimeMillis(), HttpStatus.NOT_FOUND.value(), error:"Object not found",  
ex.getMessage(), request.getRequestURI());
```

- **value():** retorna o valor numérico do código de status.

```
int statusCode = HttpStatus.NOT_FOUND.value(); // Retorna 404
```

- **getReasonPhrase():** retorna o motivo associada ao código de status.

```
String reasonPhrase = HttpStatus.NOT_FOUND.getReasonPhrase(); // Retorna "Not Found"
```

- **Series():** retorna a série do código de status (INFORMATIONAL, SUCCESSFUL, REDIRECTION, CLIENT_ERROR, SERVER_ERROR)

```
// Retorna HttpStatus.Series.CLIENT_ERROR
```

```
HttpStatus.Series series = HttpStatus.NOT_FOUND.series();
```

- **is1xxInformational(), is2xxSuccessful(), is3xxRedirection(), is4xxClienterror(), is5xxServerError(),** métodos booleanos para verificar a categoria do código de status.

```
// Retorna true
```

```
boolean isClientError = HttpStatus.NOT_FOUND.is4xxClientError();
```

```
return ResponseEntity.status(HttpStatus.NOT_FOUND).body(error);
```

Em nossa resposta definimos como erro 404 não encontrado.

E enviamos no corpo da resposta nosso objeto de erro (StandardError) que contém os dados da exceção ocorrida.

The screenshot shows a web browser window with the URL `http://localhost:8080/produto/codigobarra/555`. The response status is **404 Not Found**, with a response time of 5 ms and a size of 326 B. The response body is displayed in JSON format:

```
{
  "timestamp": 1727989128562,
  "status": 404,
  "error": "Object not found",
  "message": "Produto não encontrado! CodigoBarra: 555",
  "path": "/produto/codigobarra/555"
}
```

Two pink arrows point from the text on the left to the code and the JSON response body.

- Agora nosso erro vem mais organizado.



Spring Boot

Tratamento de Erros Bad Request – Tipos diferentes findById()

PROF. ME. JEFFERSON PASSERINI

HTTP SuporteOS2024 / grupoprodutoById Save Share

GET http://localhost:8080/grupoproduto/X Send

Params Auth Headers (6) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body 400 Bad Request • 15 ms • 6.73 KB • e.g. ...

Pretty Raw Preview Visualize JSON

```
1 {
2   "timestamp": "2024-10-03T21:00:24.235+00:00",
3   "status": 400,
4   "error": "Bad Request",
5   "trace": "org.springframework.web.method.a
```

Criando Projeto Spring Boot – Bad Request

Nosso Id de Produto e GrupoProduto são dos tipos Long e Integer respectivamente.

Se passarmos para o findById() um valor não compatível como uma String "X" ocasionará um erro "Bad Request", como no exemplo.

Então vamos tratar essa exceção também!

Essa exceção não é gerada em nosso Services, como o erro anterior que é fruto de uma busca que retorna Null.

Assim temos apenas que tratar na camada Resources o erro de bad request por parâmetros incompatíveis.

Em exceptions de resources – altere a classe `ResourceExceptionHandler` e implemente um novo método para tratar a exceção de Bad Request, conforme destacado.

```
1 package com.curso.resources.exceptions;
2
3 import com.curso.services.exceptions.ObjectNotFoundException;
4 import jakarta.servlet.http.HttpServletRequest;
5 import org.springframework.http.HttpStatus;
6 import org.springframework.http.ResponseEntity;
7 import org.springframework.web.bind.MethodArgumentNotValidException;
8 import org.springframework.web.bind.annotation.ControllerAdvice;
9 import org.springframework.web.bind.annotation.ExceptionHandler;
10 import org.springframework.web.context.request.WebRequest;
11 import org.springframework.web.method.annotation.MethodArgumentTypeMismatchException;
12
13 @ControllerAdvice new *
14 public class ResourceExceptionHandler {
15
16     @ExceptionHandler new *
17     public ResponseEntity<StandardError> objectNotFoundException(ObjectNotFoundException ex, HttpServletRequest request)
18     {
19
20         StandardError error = new StandardError(System.currentTimeMillis(), HttpStatus.NOT_FOUND.value(),
21             error: "Object not found", ex.getMessage(), request.getRequestURI());
22
23         return ResponseEntity.status(HttpStatus.NOT_FOUND).body(error);
24     }
25
26     @ExceptionHandler(MethodArgumentTypeMismatchException.class) new *
27     public ResponseEntity<StandardError> handleMethodArgumentTypeMismatchException(
28         MethodArgumentTypeMismatchException ex, HttpServletRequest request)
29     {
30
31         StandardError error = new StandardError(System.currentTimeMillis(), HttpStatus.BAD_REQUEST.value(),
32             error: "Bad Request", ex.getMessage(), request.getRequestURI());
33
34         return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(error);
35     }
36 }
```


A classe

MethodArgumentTypeMismatchException

é do Spring então é só realizar a

importação.

```
1 package com.curso.resources.exceptions;
2
3 import com.curso.services.exceptions.ObjectNotFoundException;
4 import jakarta.servlet.http.HttpServletRequest;
5 import org.springframework.http.HttpStatus;
6 import org.springframework.http.ResponseEntity;
7 import org.springframework.web.bind.MethodArgumentNotValidException;
8 import org.springframework.web.bind.annotation.ControllerAdvice;
9 import org.springframework.web.bind.annotation.ExceptionHandler;
10 import org.springframework.web.context.request.WebRequest;
11 import org.springframework.web.method.annotation.MethodArgumentTypeMismatchException;
12
13 @ControllerAdvice new *
14 public class ResourceExceptionHandler {
15
16     @ExceptionHandler new *
17     public ResponseEntity<StandardError> objectNotFoundException(ObjectNotFoundException ex,
18     {
19
20         StandardError error = new StandardError(System.currentTimeMillis(), HttpStatus.NOT_FOUND.value(),
21         error: "Object not found", ex.getMessage(), request.getRequestURI());
22
23         return ResponseEntity.status(HttpStatus.NOT_FOUND).body(error);
24     }
25
26     @ExceptionHandler(MethodArgumentTypeMismatchException.class) new *
27     public ResponseEntity<StandardError> handleMethodArgumentTypeMismatchException(
28         MethodArgumentTypeMismatchException ex, HttpServletRequest request)
29     {
30
31         StandardError error = new StandardError(System.currentTimeMillis(), HttpStatus.BAD_REQUEST.value(),
32         error: "Bad Request", ex.getMessage(), request.getRequestURI());
33
34         return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(error);
35     }
36 }
```

Agora realize o teste novamente e o log de exceção estará formatado.

HTTP SuporteOS2024 / grupoprodutoById Save Share

GET Send http://localhost:8080/grupoproduto/X

Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body Cookies Headers (4) Test Results 400 Bad Request • 86 ms • 356 B • Save Response ...

Pretty Raw Preview Visualize JSON Copy Search

```
1 {
2   "timeStamp": 1727989714217,
3   "status": 400,
4   "error": "Bad Request",
5   "message": "Failed to convert value of type 'java.lang.String' to required type 'java.lang.Integer'; For input string:
6     \"X\",
7   "path": "/grupoproduto/X"
```


Atenção!

Os tratamentos de exceção valem para todos os domínios