Spring Boot: exceções, enums e opcionais

Prof. Pedro Pongelupe

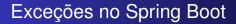


PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS Curso de Engenharia de Software

Sumário

- Exceções
 - Programação defensiva no Spring Boot
 - Criando as nossas próprias Exceptions
 - Tratando erros no Spring Boot
- Lançando Exceções com tipos opcionais
 - Tipos opcionais





Como funciona

As exceções no Spring Boot funcionam do mesmo jeito de quando fazemos desenvolvimentos com o Java SE. Ou seja, temos a utilização das mesma hierarquia de Throwable nas estruturas de *try*, *catch*, *finally*, *throw* e *throws*.

- Lançamento
- Propagação
- Captura e tratamento



Fluxo de execução de código

```
void metodo() {
    trv {
       bloco de codigo 1;
       throw new EX();
                          // -- excecao EX lancada --
       bloco de código 2; // não será executado.
     catch (EX e) {
       bloco de código 3; // irá capturar a exceção EX.
     finally {
       bloco de código 4; // será sempre executado.
    bloco de código 5; // não será executado, caso seja
    // lançada uma exceção inesperada
    // que não esteja sendo
    // tratada por um bloco catch.
```



Programação defensiva no Spring Boot

Relembrando...

"Se alguém fizer algo perigoso, você está preparado para evitar maiores consequências. (...) Você assume a responsabilidade pela sua saúde, mesmo que seja culpa do outro motorista."

- Ideia principal na programação:
 - Problemas acontecerão, mas seu programa estará preparado para lidar com eles.
 - "Garbage in, nothing out" (lixo entra, nada sai).
 - "Garbage in, error message out" (lixo entra, mensagem de erro sai).
 - "No garbage allowed in" (nenhum lixo é permitido entrar).



Protegendo seu programa de entradas inválidas

Utilizamos as anotações do pacote jakarta. validation. constraints

Principais anotações

- @NotNull Define que um campo n\u00e3o possa ser nulo.
- @NotEmpty Define que um campo n\u00e3o possar ser vazio.
 Collections, arrays e strings.
- @Max Define o valor máximo para um inteiro.
- @Min Define o valor minímo para um inteiro.
- @Size Define o tamanho para uma String.
- @Email Define que um campo deve conter um email válido.



Integração com o Spring Boot

Basta utilizar a seguinte dependência no pom.xml

E podemos adicionar a anotação @valid na Controller:

```
@PostMapping("/alunos")
public @ResponseBody
Aluno postAluno(@Valid @RequestBody Aluno a) {
    return alunosService.salvaAluno(a);
}
```



Classe Aluno com campos válidos

```
public class Aluno {
        @ld
        @Column
        @NotNull
        private String matricula;
        @Size(min = 3, max = 50)
        @Column
        private String nome:
        @Min(value = 18)
        @Column
        private int idade;
  /** todos os métodos foram ocultados **/
```



Exception: Aluno não encontrado

Caso busque um aluno que não há cadastro de sua matrícula.

```
\textbf{public class} \  \, \textbf{AlunoNaoEncontradoException extends} \  \, \textbf{RuntimeException} \  \, \{
```



Exception: Conflito matrícula

O sistema não pode permitir mais de um aluno com a mesma matrícula.

Programação defensiva no Spring Boot Criando as nossas próprias Exceptions Tratando erros no Spring Boot



Técnicas de tratamento de falhas no Spring Boot

O Spring Boot possui uma abordagem centralizada para tratamento de erros. Sendo assim, um módulo único dedicado somente para tratar de erros. A @ControllerAdvice.



ControllerAdvice: tratando erros

```
@ControllerAdvice
public class Sga2ExceptionHandler {
        @ExceptionHandler(value = AlunoNaoEncontradoException.class)
        public ResponseEntity < String >
            handleAlunoNaoEncontradoException (
                AlunoNaoEncontradoException ex) {
                return ResponseEntity.status(HttpStatus.NOT_FOUND)
                                 .body(ex.getMessage());
        @ExceptionHandler(value = ConflitoMatriculaAlunoException.clas
        public ResponseEntity < String >
            handleAlunoNaoEncontradoException (
                ConflitoMatriculaAlunoException ex) {
                return ResponseEntity.status(HttpStatus.CONFLICT)
                                 .body(ex.getMessage());
```



Tipos opcionais

Os tipos opcionais se expressam pelos objetos envoltos por um optional. O optional em Java tem o objetivo de reduzir a incidência de NullPointerException, então os objetos opcionais abstraem a tarefa de ficar checando se um determinado objeto é null. Além disso, provê uma inferface simplificada para o usuário utilizar. Os tipos opcionais não são exclusivos do Spring Boot!



Exemplo sem e com Optional

```
public char buscaCaracter(String s, int index) {
   if (s == null) return '';
   if (s.length() <= index) return '';
   return s.charAt(index);
}

public char buscaCaracter(String s, int index) {
   return Optional.ofNullable(s)
        .filter(str -> str.length() <= index)
        .map(str -> str.charAt(index))
        .orElse('');
}
```



Principais métodos

Principais métodos

- .isPresent() Devolve se optional está preenchido.
- .get() Pega o valor dentro do optional, caso exista.
- filter (Predicate<T>t) Aplica um filtro ao valor caso n\u00e3o seja null.
- .map(Function<T, U>) Mapeia o optional para outro tipo caso não seja null.
- .orElse(T t) Devolve o valor do opional caso esteja presente, senão devolve um valor padrão.
- .orElseThrow(Supplier<X>s) Lança uma exceção caso o optional esteja vazio.



Exemplo de Optional utilizado com o Java Stream

```
public static void main(String[] args) {
    var i = List.of(1, 2, 3)
        .stream()
        .filter(e -> e % 2 == 1)
        .findFirst() // método que devolve um Optional
        .filter(e -> e > 0)
        .map(e -> e + 1)
        .orElse(-1);
    System.out.println(i);
}
```



Exemplo de Optional lançando nossa Exceção

```
public Aluno buscaAlunoPelaMatricula(String matricula) {
    return alunosRepository.findByld(matricula)
        .orElseThrow(() -> new AlunoNaoEncontradoException(matricula))
```



Obrigado!!

Muito obrigado pela atenção! Alguma dúvida? Bora praticar!!!

"Até a vitória, sempre! "