

Calculadora com Redirecionamentos

Estrutura do projeto:

- ▼ calculadora [boot] [devtools]
 - ▼ src/main/java
 - ▼ br.edu.unoesc.calculadora
 - > CalculadoraApplication.java
 - ▼ br.edu.unoesc.calculadora.controller
 - > CalculadoraController.java
 - > CalculadoraControllerRest.java
 - ▼ br.edu.unoesc.calculadora.exceptions
 - > RaizNegativaException.java
 - ▼ br.edu.unoesc.calculadora.model
 - > Calculadora.java
 - ▼ br.edu.unoesc.calculadora.utils
 - > ConversorNumerico.java
 - ▼ src/main/resources
 - static
 - ▼ templates
 - resultado.html
 - application.properties
 - > src/test/java
 - > JRE System Library [JavaSE-17]
 - > Maven Dependencies
 - > src
 - target
 - HELP.md
 - mvnw
 - mvnw.cmd
 - pom.xml

Programa principal:

```
1 package br.edu.unoesc.calculadora;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 public class CalculadoraApplication {
8
9     public static void main(String[] args) {
10         SpringApplication.run(CalculadoraApplication.class, args);
11     }
12 }
13 }
```

Classe de conversão numérica:

```
1 package br.edu.unoesc.calculadora.utils;
2
3 public class ConversorNumerico {
4     public static boolean ehNumerico(String strNumero) {
5         if (strNumero == null) {
6             return false;
7         }
8
9         // 10,65 => 10.65
10        String numero = strNumero.replaceAll(",", ".");
11
12        // Utilizando tratamento de exceções
13        try {
14            Double.parseDouble(numero);
15            return true; // Significa que a string contém um Double válido
16        } catch (NumberFormatException e) {
17            return false;
18        }
19
20        // Utilizando expressões regulares
21        // return numero.matches("[+]?\\d*\\.?\\d*");
22    }
23
24    public static Double converterParaDouble(String strNumero) {
25        if (strNumero == null) {
26            return 0d;
27        }
28
29        String numero = strNumero.replaceAll(",", ".");
30        if (ehNumerico(numero)) {
31            return Double.parseDouble(numero);
32        }
33
34        return 0d;
35    }
36 }
```

Classe personalizada de tratamento de exceção:

```
1 package br.edu.unoesc.calculadora.exceptions;
2
3 @SuppressWarnings("serial")
4 public class RaizNegativaException extends RuntimeException {
5     public RaizNegativaException(String msg) {
6         super(msg);
7     }
8 }
```

Classe Calculadora:

```
1 package br.edu.unoesc.calculadora.model;
2
3 import java.util.logging.Logger;
4
5 import br.edu.unoesc.calculadora.exceptions.RaizNegativaException;
6 import br.edu.unoesc.calculadora.utils.ConversorNumerico;
7
8 public class Calculadora {
9     private static Logger logger = Logger.getLogger(Calculadora.class.getName());
10
11     public static Double somar(String n1, String n2) {
12         if (!ConversorNumerico.ehNumerico(n1) || !ConversorNumerico.ehNumerico(n2)) {
13             return 0d;
14         }
15
16         Double numero1 = ConversorNumerico.converterParaDouble(n1);
17         Double numero2 = ConversorNumerico.converterParaDouble(n2);
18         Double resultado = numero1 + numero2;
19
20         logger.info("Calculando... " + numero1 + " + " + numero2 + " = " + resultado);
21
22         return resultado;
23     }
24
25     // Método sobrecarregado
26     public static Double subtrairQuery(Double n1, Double n2) {
27         return subtrair(n1.toString(), n2.toString());
28     }
29
30     public static Double subtrair(String n1, String n2) {
31         Double numero1 = ConversorNumerico.converterParaDouble(n1);
32         Double numero2 = ConversorNumerico.converterParaDouble(n2);
33         Double resultado = numero1 - numero2;
34
35         logger.info("Calculando... " + numero1 + " - " + numero2 + " = " + resultado);
36
37         return resultado;
38     }
39
40     public static Double multiplicar(String n1, String n2) {
41         Double numero1 = ConversorNumerico.converterParaDouble(n1);
42         Double numero2 = ConversorNumerico.converterParaDouble(n2);
43         Double resultado = numero1 * numero2;
44
45         logger.info("Calculando... " + numero1 + " * " + numero2 + " = " + resultado);
46
47         return resultado;
48     }
49
50     public static Double dividir(String n1, String n2) {
51         String erro = "Erro de divisão por zero!";
52
53         Double numero1 = ConversorNumerico.converterParaDouble(n1);
54         Double numero2 = ConversorNumerico.converterParaDouble(n2);
55
56         if (numero2 == 0) {
57             logger.info(erro);
58             throw new ArithmeticException(erro);
59         }
60
61         Double resultado = numero1 / numero2;
62
63         logger.info("Calculando... " + numero1 + " / " + numero2 + " = " + resultado);
64
65         return resultado;
66     }
67
68     public static Double calcularMedia(String n1, String n2) {
69         Double numero1 = ConversorNumerico.converterParaDouble(n1);
70         Double numero2 = ConversorNumerico.converterParaDouble(n2);
71         Double resultado = (numero1 + numero2) / 2.0;
72
73         logger.info("Calculando... (" + numero1 + " + " + numero2 + ")/2 = " + resultado);
74
75         return resultado;
76     }
77
78     public static Double calcularPotencia(String n1, String n2) {
79         Double base = ConversorNumerico.converterParaDouble(n1);
80         Double expoente = ConversorNumerico.converterParaDouble(n2);
81         Double resultado = Math.pow(base, expoente);
82
83         logger.info("Calculando... " + base + " ^ " + expoente + " = " + resultado);
84
85         return resultado;
86     }
87
88     public static Double calcularRaizQuadrada(String num) {
89         String erro = "Erro! Não existe raiz quadrada de um número negativo!";
90
91         Double numero = ConversorNumerico.converterParaDouble(num);
92
93         if (numero < 0) {
94             logger.info(erro);
95             throw new RaizNegativaException(erro);
96         }
97
98         Double resultado = Math.sqrt(numero);
99
100         logger.info("Calculando... raiz quadrada de " + numero + " = " + resultado);
101
102         return resultado;
103     }
104 }
```

Classe CalculadoraController:

```
1 package br.edu.unoesc.calculadora.controller;
2
3 import javax.servlet.http.HttpServletRequest;
4
5 import org.springframework.stereotype.Controller;
6 import org.springframework.ui.Model;
7 import org.springframework.ui.ModelMap;
8 import org.springframework.web.bind.annotation.GetMapping;
9 import org.springframework.web.bind.annotation.ModelAttribute;
10 import org.springframework.web.bind.annotation.PathVariable;
11 import org.springframework.web.bind.annotation.RequestParam;
12 import org.springframework.web.servlet.mvc.support.RedirectAttributes;
13
14 import br.edu.unoesc.calculadora.model.Calculadora;
15
16 @Controller
17 public class CalculadoraController {
18     @GetMapping("/soma-query")
19     public String somaQuery(@RequestParam(value = "n1", defaultValue = "0") String n1,
20                             @RequestParam(value = "n2", defaultValue = "0") String n2,
21                             RedirectAttributes atributos) {
22         atributos.addAttribute("v1", "40");
23         atributos.addFlashAttribute("v2", "42");
24
25         return "redirect:/soma-path/" + n1 + "/" + n2;
26     }
27
28     @GetMapping("/soma-path/{numero1}/{numero2}")
29     public String somaPath(@PathVariable String numero1,
30                             @PathVariable String numero2,
31                             @ModelAttribute("v1") Double v1,
32                             @ModelAttribute("v2") Double v2,
33                             ModelMap model) {
34         System.out.println(v1 + " " + v2);
35         System.out.println(model.getAttribute("v1") + " " + model.getAttribute("v2"));
36
37         model.addAttribute("resultado", Calculadora.somar(numero1, numero2));
38
39         return "resultado";
40     }
41     //-----
42     @GetMapping("/soma-forward-query")
43     public String somaForwardQuery(@RequestParam(value = "n1", defaultValue = "0") String n1,
44                                     @RequestParam(value = "n2", defaultValue = "0") String n2,
45                                     ModelMap model,
46                                     HttpServletRequest requisicao) {
47         model.addAttribute("v1", "40");
48         requisicao.setAttribute("v2", "2");
49
50         return "forward:/soma-forward-path/" + n1 + "/" + n2;
51     }
52
53     @GetMapping("/soma-forward-path/{numero1}/{numero2}")
54     public String somaForwardPath(@PathVariable String numero1,
55                                    @PathVariable String numero2,
56                                    Model model,
57                                    HttpServletRequest requisicao) {
58         System.out.println(requisicao.getAttribute("v1"));
59         System.out.println(requisicao.getAttribute("v2"));
60
61         model.addAttribute("resultado", Calculadora.somar(numero1, numero2));
62
63         return "resultado";
64     }
65 }
```

Classe CalculadoraRestController:

```
1 package br.edu.unoesc.calculadora.controller;
2
3 import java.net.URI;
4
5 import javax.servlet.http.HttpServletRequest;
6
7 import org.springframework.http.HttpStatus;
8 import org.springframework.http.ResponseEntity;
9 import org.springframework.ui.Model;
10 import org.springframework.ui.ModelMap;
11 import org.springframework.web.bind.annotation.GetMapping;
12 import org.springframework.web.bind.annotation.ModelAttribute;
13 import org.springframework.web.bind.annotation.PathVariable;
14 import org.springframework.web.bind.annotation.RequestMapping;
15 import org.springframework.web.bind.annotation.RequestMethod;
16 import org.springframework.web.bind.annotation.RequestParam;
17 import org.springframework.web.bind.annotation.RestController;
18 import org.springframework.web.servlet.ModelAndView;
19 import org.springframework.web.servlet.mvc.support.RedirectAttributes;
20 import org.springframework.web.servlet.view.RedirectView;
21
22 import br.edu.unoesc.calculadora.exceptions.RaizNegativaException;
23 import br.edu.unoesc.calculadora.model.Calculadora;
24
25 @RestController
26 public class CalculadoraRestController {
27     @GetMapping("/somar-query")
28     public Double somarQuery(@RequestParam(value = "n1", defaultValue = "0") Double n1,
29                             @RequestParam(value = "n2", defaultValue = "0") Double n2) {
30         return this.somarPath(n1.toString(), n2.toString()); // Chama método somarPath() logo abaixo
31     }
32
33     @RequestMapping(value = "/somar-path/{numero1}/{numero2}", method = RequestMethod.GET)
34     public Double somarPath(@PathVariable("numero1") String numero1,
35                             @PathVariable("numero2") String numero2) {
36         return Calculadora.somar(numero1, numero2);
37     }
38     //-----
39     // Redirect: Lado cliente
40     @GetMapping("/subtrair-query")
41     public ResponseEntity<Void> subtrairQuery(@RequestParam(value = "n1", defaultValue = "0") Double n1,
42                                                @RequestParam(value = "n2", defaultValue = "0") Double n2) {
43         return ResponseEntity.status(HttpStatus.FOUND).location(URI.create("/subtrair-path/" + n1 + "/" + n2)).build();
44     }
45
46     @GetMapping("/subtrair-path/{numero1}/{numero2}")
47     public Double subtrairPath(@PathVariable("numero1") String numero1,
48                               @PathVariable("numero2") String numero2) {
49         return Calculadora.subtrair(numero1, numero2);
50     }
51     //-----
52     // Redirect: Lado cliente
53     @GetMapping("/multiplicar-query")
54     public RedirectView multiplicarQuery(@RequestParam(value = "n1", defaultValue = "0") String n1,
55                                          @RequestParam(value = "n2", defaultValue = "0") String n2,
56                                          RedirectAttributes atributos) {
57         atributos.addAttribute("v1", 40);
58         atributos.addFlashAttribute("v2", 2);
59
60         return new RedirectView("/multiplicar-path/" + n1 + "/" + n2);
61     }
62
63     @GetMapping("/multiplicar-path/{numero1}/{numero2}")
64     public Double multiplicarPath(@PathVariable String numero1,
65                                  @PathVariable String numero2,
66                                  @ModelAttribute("v1") Double v1,
67                                  @ModelAttribute("v2") Double v2,
68                                  Model model) {
69         System.out.println(model.getAttribute("v1"));
70         System.out.println(model.getAttribute("v2"));
71         System.out.println(v1 + " " + v2);
72
73         return Calculadora.multiplicar(numero1, numero2);
74     }
75     //-----
76     // Redirect: Lado cliente
77     @GetMapping("/dividir-query")
78     public ModelAndView dividirQuery(@RequestParam(value = "n1", defaultValue = "0") String n1,
79                                      @RequestParam(value = "n2", defaultValue = "0") String n2,
80                                      ModelMap modeloMap) {
81         modeloMap.addAttribute("valor", 42);
82
83         return new ModelAndView("redirect:/dividir-path/" + n1 + "/" + n2, modeloMap);
84     }
85
86     @GetMapping("/dividir-path/{n1}/{n2}")
87     public String dividirPath(@PathVariable String n1,
88                              @PathVariable String n2,
89                              @ModelAttribute("valor") String valor,
90                              Model modelo) {
91         System.out.println(valor);
92         System.out.println(modelo.getAttribute("valor"));
93
94         try {
95             return Calculadora.dividir(n1, n2).toString();
96         } catch (ArithmeticException e) {
97             return e.getMessage();
98         }
99     }
100     //-----
101     // Forward: Lado servidor (transparente para o cliente/navegador)
102     @GetMapping("/calcular-media-query")
103     public ModelAndView calcularMediaQuery(@RequestParam(value = "n1", defaultValue = "0") String n1,
104                                             @RequestParam(value = "n2", defaultValue = "0") String n2,
105                                             HttpServletRequest requisicao) {
106         requisicao.setAttribute("valor", "42");
107
108         return new ModelAndView("forward:/calcular-media-path/" + n1 + "/" + n2);
109     }
110
111     @GetMapping("/calcular-media-path/{numero1}/{numero2}")
112     public Double calcularMediaPath(@PathVariable String numero1,
113                                     @PathVariable String numero2,
114                                     HttpServletRequest requisicao) {
115         System.out.println(requisicao.getAttribute("valor"));
116
117         return Calculadora.calcularMedia(numero1, numero2);
118     }
119     //-----
120     @GetMapping("/calcular-potencia-query")
121     public Double calcularPotenciaQuery(@RequestParam(value = "base", defaultValue = "0") String base,
122                                          @RequestParam(value = "expoente", defaultValue = "0") String expoente) {
123         return this.calcularPotenciaPath(base, expoente);
124     }
125
126     @GetMapping("/calcular-potencia-path/{base}/{expoente}")
127     public Double calcularPotenciaPath(@PathVariable String base,
128                                        @PathVariable String expoente) {
129         return Calculadora.calcularPotencia(base, expoente);
130     }
131     //-----
132     @GetMapping("/calcular-raiz-query")
133     public String calcularRaizQuery(@RequestParam(value = "numero", defaultValue = "0") String numero) {
134         return this.calcularRaizPath(numero);
135     }
136
137     @GetMapping("/calcular-raiz-path/{numero}")
138     public String calcularRaizPath(@PathVariable String numero) {
139         try {
140             return Calculadora.calcularRaizQuadrada(numero).toString();
141         } catch (RaizNegativaException e) {
142             return e.getMessage();
143         }
144     }
145 }
```