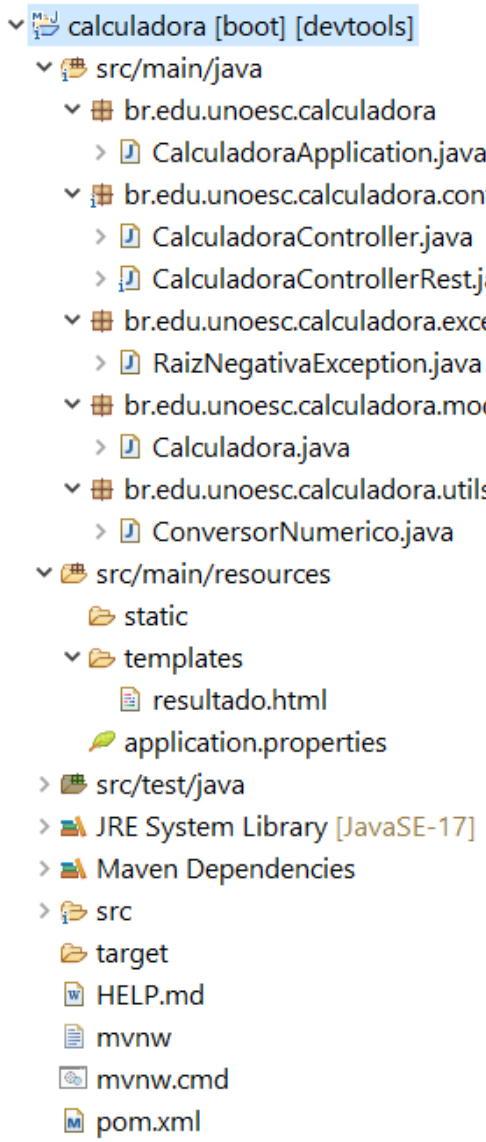
Calculadora com Redirecionamentos

Estrutura do projeto:



Programa principal:

```java
package br.edu.unoesc.calculadora;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class CalculadoraApplication {

    public static void main(String[] args) {
        SpringApplication.run(CalculadoraApplication.class, args);
    }

}
```

Classe de conversão numérica:

```java
package br.edu.unoesc.calculadora.utils;

public class ConversorNumerico {
    public static boolean ehNumerico(String strNumero) {
        if (strNumero == null) {
            return false;
        }

        // 10,65 => 10.65
        String numero = strNumero.replaceAll(",", ".");

        // Utilizando tratamento de exceções
        try {
            Double.parseDouble(numero);
            return true; // Significa que a string contém um Double válido
        } catch (NumberFormatException e) {
            return false;
        }

        // Utilizando expressões regulares
//      return numero.matches("[-+]?\\d*\\.?\\d*");
    }

    public static Double converterParaDouble(String strNumero) {
        if (strNumero == null) {
            return 0d;
        }

        String numero = strNumero.replaceAll(",", ".");
        if (ehNumerico(numero)) {
            return Double.parseDouble(numero);
        }

        return 0d;
    }
}
```

Classe personalizada de tratamento de exceção:

```java
package br.edu.unoesc.calculadora.exceptions;

@SuppressWarnings("serial")
public class RaizNegativaException extends RuntimeException {
    public RaizNegativaException(String msg) {
        super(msg);
    }
}
```

Classe `Calculadora`:

```java
package br.edu.unoesc.calculadora.model;

import java.util.logging.Logger;

import br.edu.unoesc.calculadora.exceptions.RaizNegativaException;
import br.edu.unoesc.calculadora.utils.ConversorNumerico;

public class Calculadora {
    private static Logger logger = Logger.getLogger(Calculadora.class.getName());

    public static Double somar(String n1, String n2) {
        if (!ConversorNumerico.ehNumerico(n1) || !ConversorNumerico.ehNumerico(n2)) {
            return 0d;
        }

        Double numero1 = ConversorNumerico.converterParaDouble(n1);
        Double numero2 = ConversorNumerico.converterParaDouble(n2);
        Double resultado = numero1 + numero2;

        logger.info("Calculando... " + numero1 + " + " + numero2 + " = " + resultado);

        return resultado;
    }

    // Método sobrecarregado
    public static Double subtrairQuery(Double n1, Double n2) {
        return subtrair(n1.toString(), n2.toString());
    }

    public static Double subtrair(String n1, String n2) {
        Double numero1 = ConversorNumerico.converterParaDouble(n1);
        Double numero2 = ConversorNumerico.converterParaDouble(n2);
        Double resultado = numero1 - numero2;

        logger.info("Calculando... " + numero1 + " - " + numero2 + " = " + resultado);

        return resultado;
    }

    public static Double multiplicar(String n1, String n2) {
        Double numero1 = ConversorNumerico.converterParaDouble(n1);
        Double numero2 = ConversorNumerico.converterParaDouble(n2);
        Double resultado = numero1 * numero2;

        logger.info("Calculando... " + numero1 + " * " + numero2 + " = " + resultado);

        return resultado;
    }

    public static Double dividir(String n1, String n2) {
        String erro = "Erro de divisão por zero!";

        Double numero1 = ConversorNumerico.converterParaDouble(n1);
        Double numero2 = ConversorNumerico.converterParaDouble(n2);

        if (numero2 == 0) {
            logger.info(erro);
            throw new ArithmeticException(erro);
        }

        Double resultado = numero1 / numero2;

        logger.info("Calculando... " + numero1 + " / " + numero2 + " = " + resultado);

        return resultado;
    }

    public static Double calcularMedia(String n1, String n2) {
        Double numero1 = ConversorNumerico.converterParaDouble(n1);
        Double numero2 = ConversorNumerico.converterParaDouble(n2);
        Double resultado = (numero1 + numero2) / 2.0;

        logger.info("Calculando... (" + numero1 + " + " + numero2 + ")/2 = " + resultado);

        return resultado;
    }

    public static Double calcularPotencia(String n1, String n2) {
        Double base = ConversorNumerico.converterParaDouble(n1);
        Double expoente = ConversorNumerico.converterParaDouble(n2);
        Double resultado = Math.pow(base, expoente);

        logger.info("Calculando... " + base + " ^ " + expoente + " = " + resultado);

        return resultado;
    }

    public static Double calcularRaizQuadradra(String num) {
        String erro = "Erro! Não existe raiz quadrada de um número negativo!";

        Double numero = ConversorNumerico.converterParaDouble(num);

        if (numero < 0) {
            logger.info(erro);
            throw new RaizNegativaException(erro);
        }

        Double resultado = Math.sqrt(numero);

        logger.info("Calculando... raiz quadrada de " + numero + " = " + resultado);

        return resultado;
    }
}
```

Classe `CalculadoraController`:

```java
package br.edu.unoesc.calculadora.controller;

import javax.servlet.http.HttpServletRequest;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import br.edu.unoesc.calculadora.model.Calculadora;

@Controller
public class CalculadoraController {
    @GetMapping("/soma-query")
    public String somaQuery(@RequestParam(value = "n1", defaultValue = "0") String n1,
                            @RequestParam(value = "n2", defaultValue = "0") String n2,
                            RedirectAttributes atributos) {
        atributos.addAttribute("v1", "40");
        atributos.addFlashAttribute("v2", "42");

        return "redirect:/soma-path/" + n1 + "/" + n2;
    }

    @GetMapping("/soma-path/{numero1}/{numero2}")
    public String somaPath(@PathVariable String numero1,
                           @PathVariable String numero2,
                           @ModelAttribute("v1") Double v1,
                           @ModelAttribute("v2") Double v2,
                           ModelMap model) {
        System.out.println(v1 + " " + v2);
        System.out.println(model.getAttribute("v1") + " " + model.getAttribute("v2"));

        model.addAttribute("resultado", Calculadora.somar(numero1, numero2));

        return "resultado";
    }
    //----------------
    @GetMapping("/soma-forward-query")
    public String somaForwardQuery(@RequestParam(value = "n1", defaultValue = "0") String n1,
                                   @RequestParam(value = "n2", defaultValue = "0") String n2,
                                   ModelMap model,
                                   HttpServletRequest requisicao) {
        model.addAttribute("v1", "40");
        requisicao.setAttribute("v2", "2");

        return "forward:/soma-forward-path/" + n1 + "/" + n2;
    }

    @GetMapping("/soma-forward-path/{numero1}/{numero2}")
    public String somaForwardPath(@PathVariable String numero1,
                                  @PathVariable String numero2,
                                  Model model,
                                  HttpServletRequest requisicao) {
        System.out.println(requisicao.getAttribute("v1"));
        System.out.println(requisicao.getAttribute("v2"));

        model.addAttribute("resultado", Calculadora.somar(numero1, numero2));

        return "resultado";
    }
}
```

Classe `CalculadoraRestController`:

```java
package br.edu.unoesc.calculadora.controller;

import java.net.URI;

import javax.servlet.http.HttpServletRequest;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.ui.Model;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;
import org.springframework.web.servlet.view.RedirectView;

import br.edu.unoesc.calculadora.exceptions.RaizNegativaException;
import br.edu.unoesc.calculadora.model.Calculadora;

@RestController
public class CalculadoraRestController {
    @GetMapping("/somar-query")
    public Double somarQuery(@RequestParam(value = "n1", defaultValue = "0") Double n1,
                             @RequestParam(value = "n2", defaultValue = "0") Double n2) {
        return this.somarPath(n1.toString(), n2.toString()); // Chama método somarPath() logo abaixo
    }

    @RequestMapping(value = "/somar-path/{numero1}/{numero2}", method = RequestMethod.GET)
    public Double somarPath(@PathVariable("numero1") String numero1,
                            @PathVariable("numero2") String numero2) {
        return Calculadora.somar(numero1, numero2);
    }
    //----------------------
    // Redirect: Lado cliente
    @GetMapping("/subtrair-query")
    public ResponseEntity<Void> subtrairQuery(@RequestParam(value = "n1", defaultValue = "0") Double n1,
                                              @RequestParam(value = "n2", defaultValue = "0") Double n2) {
        return ResponseEntity.status(HttpStatus.FOUND).location(URI.create("/subtrair-path/" + n1 + "/" + n2)).build();
    }

    @GetMapping("/subtrair-path/{numero1}/{numero2}")
    public Double subtrairPath(@PathVariable("numero1") String numero1,
                               @PathVariable("numero2") String numero2) {
        return Calculadora.subtrair(numero1, numero2);
    }
    //----------------------
    // Redirect: Lado cliente
    @GetMapping("/multiplicar-query")
    public RedirectView multiplicarQuery(@RequestParam(value = "n1", defaultValue = "0") String n1,
                                         @RequestParam(value = "n2", defaultValue = "0") String n2,
                                         RedirectAttributes atributos) {
        atributos.addAttribute("v1", 40);
        atributos.addFlashAttribute("v2", 2);

        return new RedirectView("/multiplicar-path/" + n1 + "/" + n2);
    }

    @GetMapping("/multiplicar-path/{numero1}/{numero2}")
    public Double multiplicarPath(@PathVariable String numero1,
                                  @PathVariable String numero2,
                                  @ModelAttribute("v1") Double v1,
                                  @ModelAttribute("v2") Double v2,
                                  Model model) {
        System.out.println(model.getAttribute("v1"));
        System.out.println(model.getAttribute("v2"));
        System.out.println(v1 + " " + v2);

        return Calculadora.multiplicar(numero1, numero2);
    }
    //----------------------
    // Redirect: Lado cliente
    @GetMapping("/dividir-query")
    public ModelAndView dividirQuery(@RequestParam(value = "n1", defaultValue = "0") String n1,
                                     @RequestParam(value = "n2", defaultValue = "0") String n2,
                                     ModelMap modeloMap) {
        modeloMap.addAttribute("valor", 42);

        return new ModelAndView("redirect:/dividir-path/" + n1 + "/" + n2, modeloMap);
    }

    @GetMapping("/dividir-path/{n1}/{n2}")
    public String dividirPath(@PathVariable String n1,
                              @PathVariable String n2,
                              @ModelAttribute("valor") String valor,
                              Model modelo) {
        System.out.println(valor);
        System.out.println(modelo.getAttribute("valor"));

        try {
            return Calculadora.dividir(n1, n2).toString();
        } catch (ArithmeticException e) {
            return e.getMessage();
        }
    }
    //----------------------
    // Forward: Lado servidor (transparente para o cliente/navegador)
    @GetMapping("/calcular-media-query")
    public ModelAndView calcularMediaQuery(@RequestParam(value = "n1", defaultValue = "0") String n1,
                                           @RequestParam(value = "n2", defaultValue = "0") String n2,
                                           HttpServletRequest requisicao) {
        requisicao.setAttribute("valor", "42");

        return new ModelAndView("forward:/calcular-media-path/" + n1 + "/" + n2);
    }

    @GetMapping("/calcular-media-path/{numero1}/{numero2}")
    public Double calcularMediaPath(@PathVariable String numero1,
                                    @PathVariable String numero2,
                                    HttpServletRequest requisicao) {
        System.out.println(requisicao.getAttribute("valor"));

        return Calculadora.calcularMedia(numero1, numero2);
    }
    //----------------------
    @GetMapping("/calcular-potencia-query")
    public Double calcularPotenciaQuery(@RequestParam(value = "base", defaultValue = "0") String base,
                                        @RequestParam(value = "expoente", defaultValue = "0") String expoente) {
        return this.calcularPotenciaPath(base, expoente);
    }

    @GetMapping("/calcular-potencia-path/{base}/{expoente}")
    public Double calcularPotenciaPath(@PathVariable String base,
                                       @PathVariable String expoente) {
        return Calculadora.calcularPotencia(base, expoente);
    }
    //----------------------
    @GetMapping("/calcular-raiz-query")
    public String calcularRaizQuery(@RequestParam(value = "numero", defaultValue = "0") String numero) {
        return this.calcularRaizPath(numero);
    }

    @GetMapping("/calcular-raiz-path/{numero}")
    public String calcularRaizPath(@PathVariable String numero) {
        try {
            return Calculadora.calcularRaizQuadradra(numero).toString();
        } catch (RaizNegativaException e) {
            return e.getMessage();
        }
    }
}
```