|  |
| --- |
| Министерство науки и высшего образования Российской Федерации |
| федеральное государственное бюджетное образовательное учреждение высшего образования |

ОТЧЕТ

по учебной практике

|  |
| --- |
| УП.04.01  Внедрение и поддержка программного обеспечения |

|  |
| --- |
| Профессионального модуля ПМ.04  Сопровождение и обслуживание программного обеспечения компьютерных систем |

|  |
| --- |
| Специальность 09.02.07  Информационные системы и программирование |

Студент:       Полжайковский Алексей Дмитриевич

*фамилия, имя, отчество*

Группа: П50-3-22

Руководитель по практической подготовке от техникума:

Белова Дарья Романовна

*фамилия, имя, отчество*  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2025 года

Содержание

[Практическая работа №1 3](#_Toc211247849)

# Практическая работа №1

Цель работы: создать приложение, включающее страницы «Калькулятор» и «Конвертер валют». Страница «Калькулятор» должна содержать функционал базового калькулятора. «Конвертер валют» – страница, на которой находится 2 выпадающих списка: в первом находится валюта, из которой надо перевести деньги, а во втором в какую валюту надо перевести.

Ход работы:

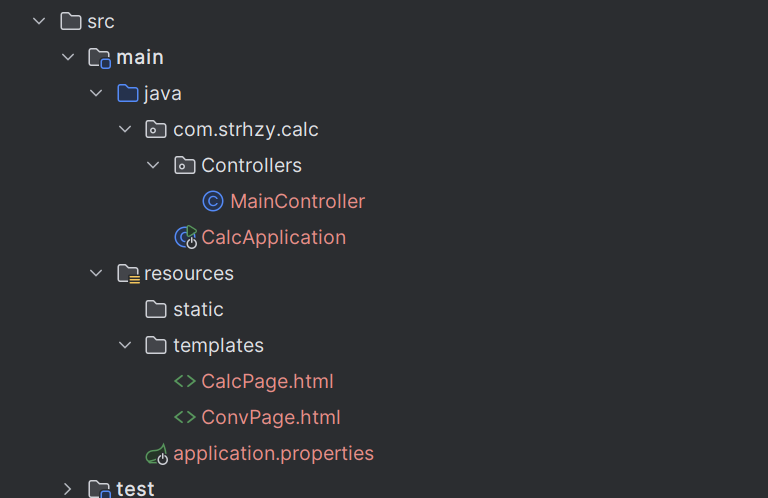


Рисунок 1 - Структура проекта

Главным классом приложения, содержащим метод main, который запускает Spring Boot приложение является файл **CalcApplication**.

Для обработки HTTP запросов пользователей был создан **MainController**, здесь в зависимости от логики контроллер возвращает соответствующие шаблоны.

**Static** – папка для хранения статических ресурсов, в представленной структуре там хранятся стили для некоторых шаблонов.

**Templates** – папка с HTML-шаблонами, которые использует Thymeleaf для рендеринга страниц.

Аннотации, используемые в работе:

**@Controller** – указывает на класс контроллер.

**@RequestMapping** – говорит о том, что все методы в этом контроллере будут обрабатывать запросы, начинающиеся с определенного пути, в коде, представленном ниже, это /home.

**@GetMapping** – используется для указания, что метод контроллера будет обрабатывать GET-запросы.

**@PostMapping** – используется для указания, что метод контроллера будет обрабатывать POST-запросы.

**@RequestParam** – используется для привязки параметров запроса к методам контроллера. @RequestParam(name = «paramName», required = false, defaultValue = «defaultValue»), где name – имя параметра, required – указатель, является ли параметр обязательным, defaultValue – значение по умолчанию.

**Model** – это специальный объект, который используется для передачи данных из контроллера в HTML-шаблон.

Код программы:

1. MainController.java:

package com.strhzy.calc.Controllers;  
  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestParam;  
  
@Controller  
public class MainController {  
 @GetMapping("/")  
 public String showCalculator() {  
 return "CalcPage";  
 }  
  
 @PostMapping("/calculate")  
 public String calculate(@RequestParam double num1, @RequestParam double num2, @RequestParam String operation, Model model) {  
 double result = 0;  
 switch (operation) {  
 case "+":  
 result = num1 + num2;  
 break;  
 case "-":  
 result = num1 - num2;  
 break;  
 case "\*":  
 result = num1 \* num2;  
 break;  
 case "/":  
 if (num2 == 0) {  
 model.addAttribute("error", "Деление на ноль невозможно");  
 return "CalcPage";  
 }  
 result = num1 / num2;  
 break;  
 }  
  
 model.addAttribute("result", result);  
  
 return "CalcPage";  
 }  
  
 @GetMapping("/conv")  
 public String showConverter() {  
 return "ConvPage";  
 }  
  
 @PostMapping("/convert")  
 public String convert(@RequestParam double num, @RequestParam String currency, Model model) {  
 double result = switch (currency) {  
 case "usd" -> num / 90;  
 case "eur" -> num / 100;  
 default -> 0;  
 };  
 model.addAttribute("result", result);  
 return "ConvPage";  
 }  
}

**Метод showCalculator()** обрабатывает GET-запрос на путь /home и отображает страницу с калькулятором main.html.

**Метод calculate()** обрабатывает POST-запрос с данными для выполнения арифметической операции. Принимает три параметра через @RequestParam: два числа (num1 и num2) и операцию (operation). В зависимости от операции вычисляет результат и передает его в модель, чтобы отобразить на странице result.html.

**Метод converter()** обрабатывает GET-запрос на путь /home/converter для конвертации валют. Принимает три параметра: валюты для конвертации (val1 и val2) и сумму для конвертации (amount). Вычисляет результат конвертации на основе предустановленных курсов валют и отображает его на странице converter.html, через метод **getConversionRate()**, который возвращает курс обмена между валютами, которые указаны в запросе.

**Метод showCalculator()** обрабатывает GET-запрос на путь / и отображает страницу с калькулятором CalcPage.html.

**Метод calculate()** обрабатывает POST-запрос с данными для выполнения арифметической операции. Принимает три параметра через @RequestParam: два числа (num1 и num2) и операцию (operation) и выводит на этой странице CalcPage.html..

**Метод showConverter()** обрабатывает GET-запрос на путь /conv и отображает страницу с калькулятором ConvPage.html.

**Метод convert()** обрабатывает GET-запрос на путь /home/converter для конвертации валют. Принимает два параметра: валюта для конвертации (currency) и сумму для конвертации (num). Вычисляет результат конвертации на основе предустановленных курсов валют и отображает его на странице ConvPage.html.

Изображение выглядит как текст, снимок экрана, линия, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 2 - CalcPage.html

Изображение выглядит как текст, снимок экрана, линия, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 3 - CalcPage.html с результатом

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 4 - ConvPage.html

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 5 - ConvPage.html с результатом

Вывод: в ходе выполнения работы было создано приложение, содержащее 2 страницы «Калькулятор» и «Конвертер валют», реализован функционал базового калькулятора, а также конвертора валют. Использованы аннотации @GetMapping, @PostMapping и @RequestParam.