МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

ОТЧЁТ

по лабораторной работе 1

по дисциплине

Методология разработки DevOps

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Симанов В.С.

(подпись) (фамилия, и.,о.)

21-ИС\_\_\_\_\_\_\_\_

(шифр группы)

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород

2024

Цель: научиться создавать и управлять репозиториями, использовать тэги для версионирования, реализовать взаимодействие между фронтенд и бэкенд частями на разных версиях.

Задание:

Необходимо создать два отдельных репозитория: frontend и backend.

В каждом репозитории создать ветку main.

Реализация версий 1.0:

Frontend 1.0:

Создайте страницу с полем для ввода текста и кнопкой "Отправить".

Реализуйте логику, при которой нажатие на кнопку отправляет введенные данные на бэкенд.

Backend 1.0:

Реализуйте бэкенд, который принимает данные, отправленные с фронтенда.

Сохраните принятые данные в текстовый файл (data.txt) на сервере.

В каждом репозитории добавьте тэг v1.0 после завершения разработки.

Реализация версии 2.0:

Frontend 2.0:

Модифицируйте версию 1.0, добавив дополнительное поле для ввода и кнопку.

Реализуйте функционал, при котором нажатие на новую кнопку отправляет запрос на бэкенд и выводит результат запроса в отдельное новое поле на странице.

Backend 2.0:

Модифицируйте версию 1.0, добавив функционал чтения данных из data.txt.

Реализуйте логику, при которой бэкенд возвращает содержимое файла (data.txt) в ответ на запрос от фронтенда.

В каждом репозитории добавьте тэг v2.0 после завершения разработки.

Требования:

1. Используйте систему контроля версий Git для управления кодом.

2. Для каждой версии фронтенда и бэкенда должны быть отдельные тэги (v1.0, v2.0).

3. При переходе между версиями (v1.0 и v2.0) все работает корректно.

Хода работы:

Реализуем Frontend 1.0 и Backend 1.0:

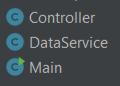
Frontend 1.0:



package com.example.application;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class Application {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(Application.class, args);  
 }  
}

package com.example.application;  
  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.notification.Notification;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.textfield.TextField;  
import com.vaadin.flow.router.Route;  
import org.springframework.http.HttpEntity;  
import org.springframework.http.HttpHeaders;  
import org.springframework.http.MediaType;  
import org.springframework.web.client.RestTemplate;  
  
@Route("")  
public class MainView extends VerticalLayout {  
  
 public MainView() {  
 // Создаем текстовое поле для ввода данных  
 TextField textField = new TextField("Введите данные");  
  
 // Кнопка для отправки данных  
 Button sendButton = new Button("Отправить", event -> {  
 String inputData = textField.getValue();  
 if (!inputData.isEmpty()) {  
 // Отправляем данные на бэкенд  
 String response = sendDataToBackend(inputData);  
 Notification.*show*(response);  
 } else {  
 Notification.*show*("Пожалуйста, введите данные!");  
 }  
 });  
  
 add(textField, sendButton);  
 }  
  
 // Метод для отправки данных на бэкенд  
 private String sendDataToBackend(String data) {  
 try {  
 RestTemplate restTemplate = new RestTemplate();  
 HttpHeaders headers = new HttpHeaders();  
 headers.setContentType(MediaType.*APPLICATION\_JSON*);  
  
 HttpEntity<String> request = new HttpEntity<>(data, headers);  
  
 // Отправка POST-запроса на сервер  
 String response = restTemplate.postForObject("http://localhost:8080/api/save", request, String.class);  
 return response;  
 } catch (Exception e) {  
 e.printStackTrace();  
 return "Ошибка при отправке данных на сервер.";  
 }  
 }  
}

Backend 1.0:

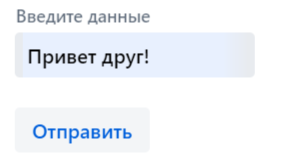


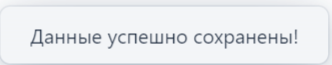
package com.example.backend;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class Main {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(Main.class, args);  
 }  
}

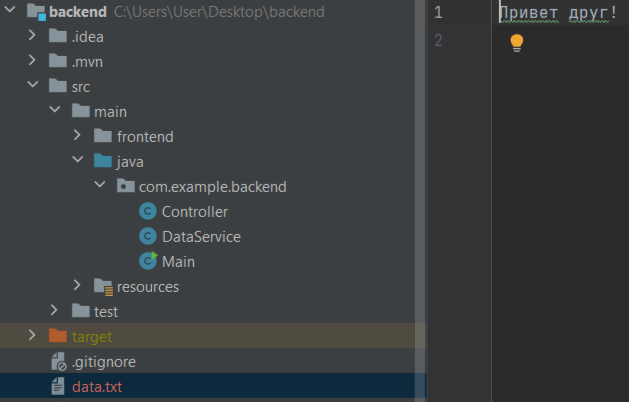
package com.example.backend;  
  
import org.springframework.stereotype.Service;  
  
import java.io.BufferedWriter;  
import java.io.FileWriter;  
import java.io.IOException;  
  
@Service  
public class DataService {  
  
 private static final String FILE\_PATH = "data.txt";  
  
 public boolean saveDataToFile(String data) {  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter(FILE\_PATH, true))) {  
 writer.write(data);  
 writer.newLine();  
 return true;  
 } catch (IOException e) {  
 e.printStackTrace();  
 return false;  
 }  
 }  
}

package com.example.backend;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
@RestController  
@RequestMapping("/api")  
@CrossOrigin(origins = "\*") // Разрешение запросов с фронтенда  
public class Controller {  
  
 @Autowired  
 private DataService dataService;  
  
 @PostMapping("/save")  
 public String saveData(@RequestBody String data) {  
 boolean isSaved = dataService.saveDataToFile(data);  
 if (isSaved) {  
 return "Данные успешно сохранены!";  
 } else {  
 return "Ошибка при сохранении данных.";  
 }  
 }  
}

Как работает:







Переходим в gitbush

В каждом репозитории прописываем get init (инициализирует новый репозиторий Git в текущей папке)

Создаем ветки main - git branch

Создаем пометки (tags)

git tag -a v1.0 -m “your message”

Реализуем Frontend 2.0 и Backend 2.0:

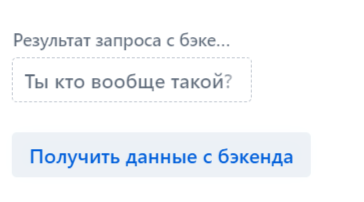
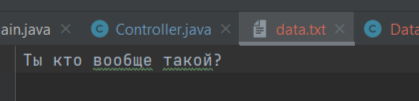
Frontend 1.0:

package com.example.application;  
  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.notification.Notification;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.textfield.TextField;  
import com.vaadin.flow.router.Route;  
import org.springframework.http.HttpEntity;  
import org.springframework.http.HttpHeaders;  
import org.springframework.http.MediaType;  
import org.springframework.web.client.RestTemplate;  
  
@Route("")  
public class MainView extends VerticalLayout {  
  
 // Создаем текстовое полz для ввода данных  
 TextField textField = new TextField("Введите данные");  
  
 TextField textField1 = new TextField("Результат запроса с бэкенда");  
 public MainView() {  
  
 // Кнопка для отправки данных  
 Button sendButton = new Button("Отправить", event -> {  
 String inputData = textField.getValue();  
 if (!inputData.isEmpty()) {  
 // Отправляем данные на бэкенд  
 String response = sendDataToBackend(inputData);  
 Notification.*show*(response);  
 } else {  
 Notification.*show*("Пожалуйста, введите данные!");  
 }  
 });  
  
 textField1 = new TextField("Результат запроса с бэкенда");  
 textField1.setReadOnly(true);  
 Button getButton = new Button("Получить данные с бэкенда", event -> getData());  
  
  
 add(textField, sendButton,textField1,getButton);  
 }  
  
 // Метод для отправки данных на бэкенд  
 private String sendDataToBackend(String data) {  
 try {  
 RestTemplate restTemplate = new RestTemplate();  
 HttpHeaders headers = new HttpHeaders();  
 headers.setContentType(MediaType.*APPLICATION\_JSON*);  
  
 HttpEntity<String> request = new HttpEntity<>(data, headers);  
  
 // Отправка POST-запроса на сервер  
 String response = restTemplate.postForObject("http://localhost:8080/api/save", request, String.class);  
 return response;  
 } catch (Exception e) {  
 e.printStackTrace();  
 return "Ошибка при отправке данных на сервер.";  
 }  
 }  
  
 private void getData() {  
 RestTemplate restTemplate = new RestTemplate();  
 String backendUrl = "http://localhost:8080/api/getBackend"; //  
  
 try {  
 String response = restTemplate.getForObject(backendUrl, String.class);  
 textField1.setValue(response != null ? response : "Нет данных");  
 } catch (Exception e) {  
 Notification.*show*("Ошибка при получении данных с бэкенда");  
 e.printStackTrace();  
 }  
 }  
}

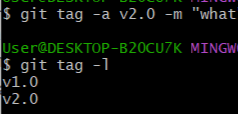
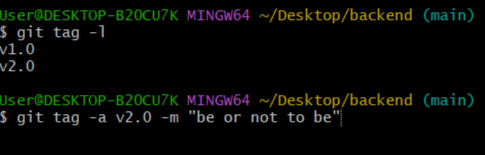
Backend 2.0:

package com.example.backend;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
  
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
  
@RestController  
@RequestMapping("/api")  
@CrossOrigin(origins = "\*") // Разрешение запросов с фронтенда  
public class Controller {  
  
 @Autowired  
 private DataService dataService;  
  
 @PostMapping("/save")  
 public String saveData(@RequestBody String data) {  
 boolean isSaved = dataService.saveDataToFile(data);  
 if (isSaved) {  
 return "Данные успешно сохранены!";  
 } else {  
 return "Ошибка при сохранении данных.";  
 }  
 }  
  
 @GetMapping("/getBackend")  
 public ResponseEntity<String> getBackendDatas() {  
 StringBuilder content = new StringBuilder();  
 try (BufferedReader reader = new BufferedReader(new FileReader("data.txt"))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 content.append(line).append("\n");  
 }  
 return new ResponseEntity<>(content.toString(), HttpStatus.*OK*);  
 } catch (IOException e) {  
 e.printStackTrace();  
 return new ResponseEntity<>("Ошибка при чтении данных", HttpStatus.*INTERNAL\_SERVER\_ERROR*);  
 }  
 }  
}

Как работает:



Переходим в gitbush

Вывод: Я научился создавать и управлять репозиториями, использовать тэги для версионирования, реализовать взаимодействие между фронтенд и бэкенд частями на разных версиях