**Міністерство освіти і науки України**

**Національний технічний університет**

**«Дніпровська політехніка»**

Изображение выглядит как текст, Шрифт, снимок экрана, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

**ЗВІТ**

**про виконання лабораторної роботи**

**з дисципліни**

**«Поглиблене програмування на Java»**

Виконав: ст. гр. 122-22ск-1

Куліш В.М.

Прийняв: Мінєєв О.С.

**Дніпро**

**2025**

Тема: Створення парсинг проекту.

Мета: Отримати навички роботи з Spring, H2 та API.

Хід роботи:

1. Було створено Spring проект за допомогою start.spring.io.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 1. Каталог проекту.

1. Було створено контролер, сервіс, конфіг для під’єднання до БД, парсер, модель запису та парсер.

Лістинг (AppController)

package com.project.parsing.controller;

import com.fasterxml.jackson.core.JsonProcessingException;

import com.fasterxml.jackson.databind.ObjectMapper;

import com.project.parsing.Parser;

import com.project.parsing.db.WriteToDB;

import com.project.parsing.excel.WriteToExcel;

import com.project.parsing.model.Unit;

import com.project.parsing.service.ApplicationService;

import javafx.application.Platform;

import javafx.fxml.FXML;

import javafx.scene.control.Button;

import javafx.scene.control.TextArea;

import javafx.scene.control.TextField;

import javafx.scene.text.Text;

import org.springframework.stereotype.Controller;

import org.springframework.web.client.RestTemplate;

import java.net.URL;

import java.util.ArrayList;

import java.util.List;

import java.util.ResourceBundle;

@Controller

public class AppController {

public static final String BaseURL = "https://www.work.ua/jobs-it-";

public static String ProductQuery = "";

public static String FinalURL=BaseURL + ProductQuery.replace(" ","");

public static int JobCounter = 0;

public static List<Unit> Result;

public static Parser parser = new Parser();

public static WriteToDB DBWriter = new WriteToDB();

public static WriteToExcel ExcelWriter = new WriteToExcel();

public static ApplicationService service = new ApplicationService(new RestTemplate(), new ObjectMapper());

public Text dollarToGrn;

public Text EuroToGrn;

@FXML

private ResourceBundle resources;

@FXML

private URL location;

@FXML

private Button PrevBtn;

@FXML

private Button SrchBtn;

@FXML

private Button WriteToExcelBtn;

@FXML

private Button WriteToDBBtn;

@FXML

private TextArea aboutJobTextField;

@FXML

private TextArea jobNameTextField;

@FXML

private Button nextBtn;

@FXML

private TextArea salaryTextField;

@FXML

private TextField srchField;

@FXML

void initialize() {

SrchBtn.setOnAction(actionEvent -> { // Search for jobs

if(!srchField.getText().isEmpty()){

ProductQuery = srchField.getText().replace(" ","+");

FinalURL = BaseURL + ProductQuery.replace(" ","");

Result = parser.ParseURLs(FinalURL);

Platform.runLater(()->{

jobNameTextField.appendText(Result.get(JobCounter).getJobName());

aboutJobTextField.appendText(Result.get(JobCounter).getAboutJob());

salaryTextField.appendText(Result.get(JobCounter).getJobSalaryRange());

});

}

});

nextBtn.setOnAction(actionEvent -> { // Next job

if(JobCounter+1 <= Result.size()-1){

JobCounter++;

jobNameTextField.clear();

aboutJobTextField.clear();

salaryTextField.clear();

jobNameTextField.appendText(Result.get(JobCounter).getJobName());

aboutJobTextField.appendText(Result.get(JobCounter).getAboutJob());

salaryTextField.appendText(Result.get(JobCounter).getJobSalaryRange());

}else{

System.out.println("Out of range");

}

});

PrevBtn.setOnAction(actionEvent -> { // Previous job

if(JobCounter-1 >= 0){

JobCounter--;

jobNameTextField.clear();

aboutJobTextField.clear();

salaryTextField.clear();

jobNameTextField.appendText(Result.get(JobCounter).getJobName());

aboutJobTextField.appendText(Result.get(JobCounter).getAboutJob());

salaryTextField.appendText(Result.get(JobCounter).getJobSalaryRange());

}else{

System.out.println("Out of range");

}

});

WriteToExcelBtn.setOnAction(actionEvent -> { // Create or replace excel file with data

if(!Result.isEmpty()){

ExcelWriter.toExcel(Result);

}else{

System.out.println("Empty array.");

}

});

WriteToDBBtn.setOnAction(actionEvent -> { // Add jobs to database

if(!Result.isEmpty()){

DBWriter.saveUnitsToDatabase(Result);

}else{

System.out.println("Empty array.");

}

});

loadExchangeRate();

}

public void loadExchangeRate() { // Get exchange rate from PrivatBank api

new Thread(()->{

List<String> exchRate = new ArrayList<>();

String ApiURL = "https://api.privatbank.ua/p24api/pubinfo?exchange&coursid=5";

try {

exchRate=service.fetchDataFromURL(ApiURL);

} catch (JsonProcessingException e) {

throw new RuntimeException(e);

}

List<String> finalExchRate = exchRate;

Platform.runLater(()->{

EuroToGrn.setText(finalExchRate.get(0));

dollarToGrn.setText(finalExchRate.get(1));

});

}).start();

}

}

Лістинг (WriteToDB)

package com.project.parsing.db;  
  
import com.project.parsing.model.Unit;  
  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.PreparedStatement;  
import java.sql.SQLException;  
import java.util.List;  
  
public class WriteToDB {  
 private static final String *URL* = "jdbc:postgresql://localhost:5432/labResult";  
 private static final String *USER* = "postgres";  
 private static final String *PASSWORD* = "212030";  
 public void saveUnitsToDatabase(List<Unit> units){  
 String sql = "INSERT INTO jobs (jobname,aboutjob,salary) VALUES(?, ?, ?)";  
 try(Connection conn = DriverManager.*getConnection*(*URL*, *USER*, *PASSWORD*);  
 PreparedStatement preparedStatement = conn.prepareStatement(sql)){  
 for(Unit job:units){  
 preparedStatement.setString(1,job.getJobName());  
 preparedStatement.setString(2,job.getAboutJob());  
 preparedStatement.setString(3,job.getJobSalaryRange());  
 preparedStatement.addBatch();  
 }  
 preparedStatement.executeBatch();  
 System.*out*.println("Запис виконано успішно!");  
 }catch (SQLException e){  
 e.printStackTrace();  
 }  
  
 }  
}

Лістинг (WriteToExcel)

package com.project.parsing.excel;  
  
import com.project.parsing.model.Unit;  
import org.apache.poi.ss.usermodel.Row;  
import org.apache.poi.ss.usermodel.Sheet;  
import org.apache.poi.ss.usermodel.Workbook;  
import org.apache.poi.xssf.usermodel.XSSFWorkbook;  
  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.util.List;  
  
public class WriteToExcel {  
 public static final String *filePath* = "./jobs.xlsx";  
 public void toExcel(List<Unit> units){  
 Workbook workbook = new XSSFWorkbook();  
 Sheet sheet = workbook.createSheet("Result");  
 Row headerRow = sheet.createRow(0);  
 headerRow.createCell(0).setCellValue("Название");  
 headerRow.createCell(1).setCellValue("Описание");  
 headerRow.createCell(2).setCellValue("Зарплата");  
 for(int i = 0; i<units.size();i++){  
 Unit job = units.get(i);  
 Row row = sheet.createRow(i+1);  
  
 row.createCell(0).setCellValue(job.getJobName());  
 row.createCell(1).setCellValue(job.getAboutJob());  
 row.createCell(2).setCellValue(job.getJobSalaryRange());  
 }  
 try(FileOutputStream fileOut = new FileOutputStream(*filePath*)){  
 workbook.write(fileOut);  
 System.*out*.println("Запис завершено.");  
 System.*out*.println("Файл створено за шляхом: " + *filePath*);  
  
 }catch (IOException e){  
 e.printStackTrace();  
 }  
 try{  
 workbook.close();  
 }catch (IOException e){  
 e.printStackTrace();  
 }  
 }  
}

Лістинг (Unit)

package com.project.parsing.model;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
@Data  
@NoArgsConstructor  
@AllArgsConstructor  
public class Unit {  
 String jobName;  
 String aboutJob;  
 String jobSalaryRange;  
}

Лістинг (Parser)

package com.project.parsing;  
  
import com.project.parsing.model.Unit;  
import lombok.SneakyThrows;  
import org.jsoup.Jsoup;  
import org.jsoup.nodes.Document;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class Parser {  
 @SneakyThrows  
 public Unit ParseUnit(String URL){ // Get job name, description and salary  
 Document global = Jsoup.*connect*(URL).get();  
 String jobName = global.getElementById("h1-name").text();  
 String aboutJob = global.getElementById("job-description").text();  
 String jobSalaryRange = global.select("#center > div.design-section.design-wrapper.block-relative > div > div > div.col-md-8 > div > div.card.rounded-0-top.rounded-20-top-xs.pb-lg > div.wordwrap.mt-lg.sm\\:mt-0 > ul > li:nth-child(1) > span.strong-500").text();  
  
 Unit unit = new Unit(jobName, aboutJob, jobSalaryRange);  
 return unit;  
 }  
 @SneakyThrows  
 public List<Unit> ParseURLs(String baseURL){// Get jobs name, description and salary from first page  
 String mainPageURL = "https://www.work.ua";  
 String finalLink;  
 Document global = Jsoup.*connect*(baseURL).get();  
 List<String> urls = global.getElementById("pjax-jobs-list").stream()  
 .map(element -> element.select("a").attr("href"))  
 .filter(href->href!=null && !href.isBlank() && href.contains("/jobs/")&& !href.contains("by-company"))  
 .distinct()  
 .toList();  
 List<Unit> result = new ArrayList<>();  
 if(urls.isEmpty()){  
 System.*out*.println("Empty list after parsing given URL");  
 result.add(new Unit("0","0","0"));  
 return result;  
 }else{  
 for(String links: urls){  
 finalLink = mainPageURL+links;  
 result.add(ParseUnit(finalLink));  
 }  
 }  
 return result;  
 }  
}

Лістинг (JavaFXApplication)

package com.project.parsing;  
  
import javafx.application.Application;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Parent;  
import javafx.scene.Scene;  
import javafx.stage.Stage;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.context.ConfigurableApplicationContext;  
  
@SpringBootApplication()  
public class JavaFXApplication extends Application {  
 private ConfigurableApplicationContext springContext;  
 private Parent rootNode;  
  
 public static void main(String[] args) {  
 Application.*launch*(args);  
 }  
 @Override  
 public void init() throws Exception {  
 springContext = SpringApplication.*run*(JavaFXApplication.class);  
 FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("/fxml/main-view.fxml"));  
 fxmlLoader.setControllerFactory(springContext::getBean);  
 rootNode = fxmlLoader.load();  
 }  
 @Override  
 public void start(Stage stage) throws Exception {  
 stage.setScene(new Scene(rootNode));  
 stage.show();  
 }  
  
 @Override  
 public void stop() {  
 springContext.close();  
 }  
}