Міністерство освіти і науки України

Національний технічний університет

«Дніпровська політехніка»

ЗВІТ

про виконання лабораторної роботи №1

з дисципліни

«Поглиблене програмування в середовищі Java»

Виконав:

студент групи 122-22ск-1

Куліков Ярослав Андрійович

Перевірив:

Доцент кафедри ФІТ

Мінєєв Олександр Сергійович

Дніпро
2025

Лабораторна робота №1

Тема: Розробка додатка мовою Java на базі фреймворку Spring Boot.

Мета: Навчитися створювати Spring Boot додатки, реалізовувати спілкування клієнта/сервера через REST, також навчитися адаптувати базу даних в додаток та створювати Frontend для додатка.

Хід роботи

1. У пакеті com.example.dniproapartmentfinder.config визначаємо клас RestTemplateConfig, який є конфігураційним класом Spring, позначеним анотацією @Configuration. У цьому класі метод restTemplate(), анотований як @Bean, створює та повертає екземпляр класу RestTemplate. Цей об'єкт реєструється в контексті Spring як бін і може бути використовується для виконання HTTP-запитів до зовнішніх ресурсів, таких як веб-API. Це забезпечує зручне налаштування та повторне використання RestTemplate у додатку для отримання даних із сервісів пошуку квартир:

```
package com.example.dniproapartmentfinder.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.client.RestTemplate;

@Configuration
public class RestTemplateConfig {
    @Bean
    public RestTemplate restTemplate() {
        return new RestTemplate();
    }
}
```

2. У пакеті com.example.dniproapartmentfinder.model визначаємо клас Apartment, який є JPA-сутністю, позначеною анотацією @Entity. Цей клас представляє модель квартири в системі пошуку квартир. Він містить поля: id (первинний ключ із автоматичною генерацією), title (назва квартири), url (посилання на оголошення), priceUah і priceUsd (ціни в гривнях і доларах, тип Double для підтримки null-значень), а також datePosted (дата розміщення). Для кожного поля реалізовані геттери і сеттери для доступу до даних. Клас використовується для відображення об'єктів квартир на таблицю в базі даних, що дозволяє зберігати, отримувати та обробляти інформацію про квартири в додатку:

```java
package com.example.dniproapartmentfinder.model;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class Apartment {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String title;
    private String url;
    private Double priceUah; // Має бути Double, а не double
    private Double priceUsd; // Має бути Double, а не double
    private String datePosted;

    // Геттери і сеттери
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }
    public String getTitle() { return title; }
    public void setTitle(String title) { this.title = title; }
    public String getUrl() { return url; }
    public void setUrl(String url) { this.url = url; }
    public Double getPriceUah() { return priceUah; }
    public void setPriceUah(Double priceUah) { this.priceUah = priceUah; }
    public Double getPriceUsd() { return priceUsd; }
    public void setPriceUsd(Double priceUsd) { this.priceUsd = priceUsd; }
    public String getDatePosted() { return datePosted; }
    public void setDatePosted(String datePosted) { this.datePosted =
datePosted; }
}
```

3. У пакеті com.example.dniproapartmentfinder.parser визначаємо клас OlxParser, позначаємо анотацією @Service, що робить його сервісом Spring. Клас відповідає за парсинг оголошень про квартири з сайту OLX.ua (URL: https://www.olx.ua/uk/nedvizhimost/kvartiry/dnepr/?currency=UAH). Метод parseApartments() використовує бібліотеку Jsoup для завантаження HTML-сторінки, витягує дані (заголовок, URL, ціну в гривнях, дату розміщення) за допомогою CSS-селекторів і зберігає їх у список об’єктів Apartment. Логування через SLF4J допомагає відстежувати процес і виявляти помилки, такі як відсутність даних чи проблеми з підключенням. Парсер обробляє винятки (IOException) і повертає список квартир, який може бути використаний для подальшої обробки чи збереження в базі даних:

```java
package com.example.dniproapartmentfinder.parser;

import com.example.dniproapartmentfinder.model.Apartment;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Service;
```

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

@Service
public class OlxParser {
    private static final Logger logger =
LoggerFactory.getLogger(OlxParser.class);

    public List<Apartment> parseApartments() {
        List<Apartment> apartments = new ArrayList<>();
        try {
            // Додаємо userAgent і таймаут для уникнення блокування
            Document doc =
Jsoup.connect("https://www.olx.ua/uk/nedvizhimost/kvartiry/dnepr/?currency=UAH"
)
                    .userAgent("Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36")
                    .timeout(10000) // Таймаут 10 секунд
                    .get();

            // Логування HTML для діагностики (закоментуйте після перевірки)
            // logger.debug("HTML content: {}", doc.html());

            // Оновлений селектор для контейнерів оголошень
            Elements listings = doc.select("div.css-1g5933j");
            logger.info("Found {} listings", listings.size());

            if (listings.isEmpty()) {
                logger.warn("No listings found. Check if the selector 'div.css-
1g5933j' is correct or if the page structure has changed.");
            }

            for (Element listing : listings) {
                Apartment apt = new Apartment();

                // Заголовок
                Element titleElement = listing.selectFirst("h4.css-1g61gc2");
                if (titleElement != null) {
                    apt.setTitle(titleElement.text());
                } else {
                    apt.setTitle("No title");
                    logger.warn("Title not found for listing: {}",
listing.html());
                }

                // URL
                Element linkElement = listing.selectFirst("a.css-1tqlkj0");
                if (linkElement != null) {
                    apt.setUrl("https://www.olx.ua" +
linkElement.attr("href"));
                } else {
                    apt.setUrl("");
                    logger.warn("URL not found for listing: {}",
listing.html());
                }

                // Ціна
                Element priceElement = listing.selectFirst("p[data-testid=ad-
price].css-uj7mm0");
                if (priceElement != null) {
                    String priceText = priceElement.text().replaceAll("[^0-9]",
"");
                    try {
                        apt.setPriceUah(Double.parseDouble(priceText));
                    } catch (NumberFormatException e) {
                        apt.setPriceUah(0.0);
                        logger.error("Failed to parse price: {}", priceText,
```

```
e);
                    }
                } else {
                    apt.setPriceUah(0.0);
                    logger.warn("Price not found for listing: {}",
listing.html());
                }

                // Дата
                Element dateElement = listing.selectFirst("p[data-
testid=location-date].css-vbz67q");
                if (dateElement != null) {
                    apt.setDatePosted(dateElement.text());
                } else {
                    apt.setDatePosted("No date");
                    logger.warn("Date not found for listing: {}",
listing.html());
                }

                apartments.add(apt);
                logger.debug("Parsed apartment: {}", apt.getTitle());
            }
        } catch (IOException e) {
            logger.error("Error while parsing OLX: {}", e.getMessage(), e);
        }
        return apartments;
    }
}
```

4. У пакеті com.example.dniproapartmentfinder.controller визначаємо клас ApartmentController та позначаємо анотаціями @RestController і @RequestMapping("/api/apartments"). Це REST-контролер, який обробляє HTTP-запити, пов’язані з квартирами. Контролер використовує сервіс ApartmentService, введений через @Autowired. Він містить три методи: getAllApartments() повертає список квартир у форматі JSON через GET-запит до /api/apartments; refreshData() оновлює дані через /api/apartments/refresh і перенаправляє на головну сторінку зі статусом 302; exportToExcel() експортує дані у файл Excel через /api/apartments/export, повертаючи бінарний масив із відповідними заголовками та статусом 200. Контролер забезпечує API для взаємодії з даними про квартири у проєкті:

```
package com.example.dniproapartmentfinder.controller;

import com.example.dniproapartmentfinder.model.Apartment;
import com.example.dniproapartmentfinder.service.ApartmentService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.io.IOException;
import java.util.List;
```

```java
@RestController
@RequestMapping("/api/apartments")
public class ApartmentController {
    @Autowired
    private ApartmentService apartmentService;

    @GetMapping
    public List<Apartment> getAllApartments() {
        return apartmentService.getAllApartments();
    }

    @GetMapping("/refresh")
    public ResponseEntity<String> refreshData() {
        apartmentService.refreshApartmentData();
        // Перенаправлення на головну сторінку після оновлення
        return ResponseEntity.status(HttpStatus.FOUND)
                .header("Location", "/")
                .body("Redirecting to home page...");
    }

    @GetMapping("/export")
    public ResponseEntity<byte[]> exportToExcel() throws IOException {
        byte[] excelBytes = apartmentService.exportToExcel();
        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_OCTET_STREAM);
        headers.setContentDispositionFormData("attachment", "apartments.xlsx");
        return new ResponseEntity<>(excelBytes, headers, HttpStatus.OK);
    }
}
```

5. У пакеті com.example.dniproapartmentfinder.repository визначаємо інтерфейс ApartmentRepository та позначаємо анотацією @Repository. Він успадковує JpaRepository<Apartment, Long> із Spring Data JPA, що забезпечує автоматичну реалізацію методів для роботи з сутністю Apartment у базі даних. Тип первинного ключа — Long, що відповідає полю id у моделі Apartment. Репозиторій надає стандартні операції, такі як збереження, пошук, отримання всіх записів і видалення квартир, без необхідності писати їх вручну. У проєкті він слугує шаром доступу до даних, дозволяючи іншим компонентам (наприклад, сервісам) взаємодіяти з базою даних для зберігання та обробки інформації про квартири:

```java
package com.example.dniproapartmentfinder.repository;

import com.example.dniproapartmentfinder.model.Apartment;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface ApartmentRepository extends JpaRepository<Apartment, Long> {
}
```

6. У файлу db.sql за допомогою SQL-запиту створюємо таблиці apartment у базі даних проєкту dniproapartmentfinder. Таблиця має шість колонок: id (тип BIGINT, первинний ключ з автоінкрементом), title (тип VARCHAR(255), назва

квартири), url (тип VARCHAR(255), посилання на оголошення), price_uah (тип DOUBLE, ціна в гривнях), price_usd (тип DOUBLE, ціна в доларах) і date_posted (тип VARCHAR(100), дата розміщення). Структура таблиці відповідає моделі Apartment у Java-коді, яка використовується як JPA-сутність. Таблиця слугує для зберігання даних про квартири, отриманих із парсингу OLX, і забезпечує їх обробку через репозиторій ApartmentRepository у проєкті:

```
-- db.sql
CREATE TABLE apartment (
                           id BIGINT AUTO_INCREMENT PRIMARY KEY,
                           title VARCHAR(255),
                           url VARCHAR(255),
                           price_uah DOUBLE,
                           price_usd DOUBLE,
                           date_posted VARCHAR(100)
);
```

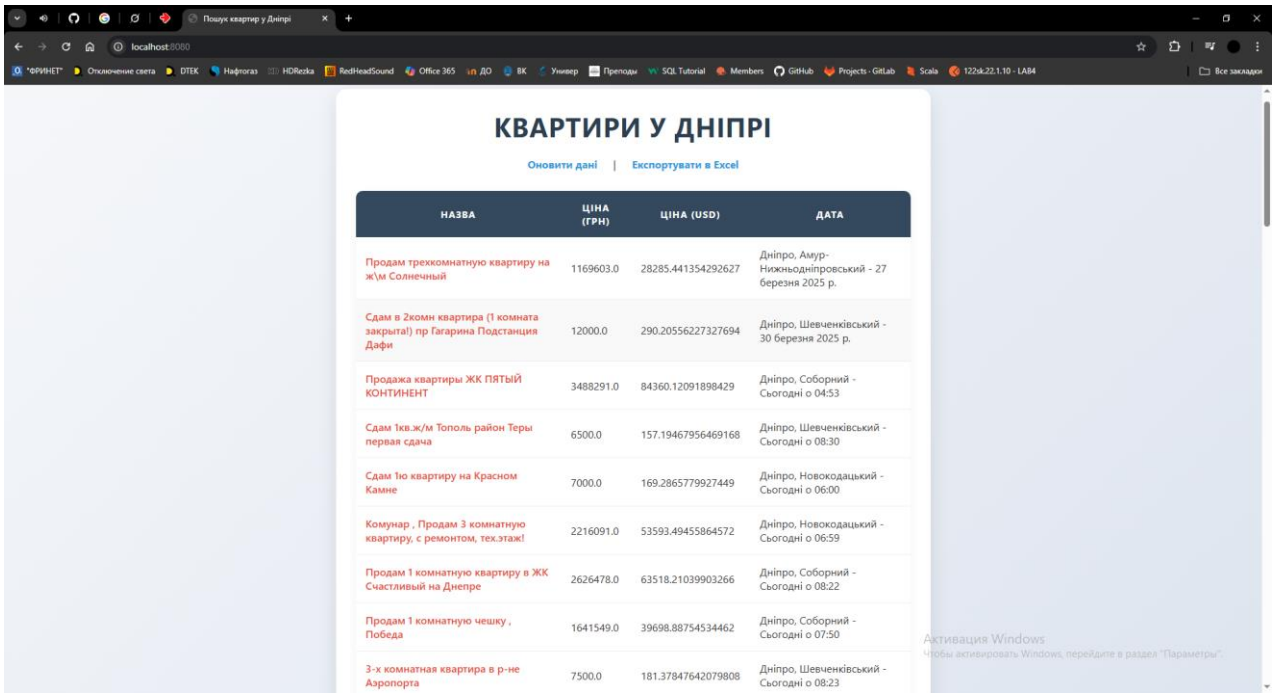
7. Створюємо HTML-шаблон із Thymeleaf для проєкту dniproapartmentfinder для того, щоб визначити вебінтерфейс для відображення квартир. Він включає стилі CSS для оформлення (градієнтний фон, таблиця з ефектами наведення, адаптивний дизайн до 600px) і динамічний вміст через Thymeleaf. Заголовок сторінки — 'Пошук квартир у Дніпрі', основний вміст — таблиця з колонками 'Назва', 'Ціна (грн)', 'Ціна (USD)', 'Дата'. Дані вставляються через цикл th:each по списку apartments, отриманому від ApartmentController. Назва квартири є посиланням на її URL. Навігація включає посилання на оновлення даних (/api/apartments/refresh) і експорт у Excel (/api/apartments/export). Шаблон забезпечує зручний інтерфейс для перегляду та управління даними про квартири:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Пошук квартир у Дніпрі</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: 'Segoe UI', Arial, sans-serif;
            background: linear-gradient(135deg, #f5f7fa 0%, #c3cfe2 100%);
            min-height: 100vh;
            padding: 40px 20px;
            color: #333;
        }

        .container {
            max-width: 900px;
            margin: 0 auto;
```

```
        background: #fff;
        border-radius: 15px;
        box-shadow: 0 10px 30px rgba(0, 0, 0, 0.1);
        padding: 30px;
    }

    h1 {
        text-align: center;
        font-size: 2.5em;
        color: #2c3e50;
        margin-bottom: 20px;
        text-transform: uppercase;
        letter-spacing: 1px;
    }

    .nav-links {
        text-align: center;
        margin-bottom: 30px;
    }

    .nav-links a {
        text-decoration: none;
        color: #3498db;
        font-weight: bold;
        padding: 10px 20px;
        transition: all 0.3s ease;
    }

    .nav-links a:hover {
        color: #2980b9;
        background: #ecf0f1;
        border-radius: 5px;
    }

    table {
        width: 100%;
        border-collapse: separate;
        border-spacing: 0;
        overflow: hidden;
        border-radius: 10px;
        box-shadow: 0 5px 15px rgba(0, 0, 0, 0.05);
    }

    th {
        background: #34495e;
        color: #fff;
        padding: 15px;
        text-transform: uppercase;
        font-size: 0.9em;
        letter-spacing: 1px;
    }

    td {
        padding: 15px;
        border-bottom: 1px solid #eee;
        transition: background 0.3s ease;
    }

    tr:hover td {
        background: #f9f9f9;
    }

    td a {
        color: #e74c3c;
        text-decoration: none;
        font-weight: 500;
        transition: color 0.3s ease;
    }
```

```
        td a:hover {
            color: #c0392b;
            text-decoration: underline;
        }

        /* Адаптивний дизайн */
        @media (max-width: 600px) {
            .container {
                padding: 20px;
            }

            h1 {
                font-size: 1.8em;
            }

            th, td {
                padding: 10px;
                font-size: 0.9em;
            }

            .nav-links a {
                display: inline-block;
                margin: 5px 0;
            }
        }
    </style>
</head>
<body>
<div class="container">
    <h1>Квартири у Дніпрі</h1>
    <div class="nav-links">
        <a href="/api/apartments/refresh">Оновити дані</a> |
        <a href="/api/apartments/export">Експортувати в Excel</a>
    </div>

    <table>
        <tr>
            <th>Назва</th>
            <th>Ціна (грн)</th>
            <th>Ціна (USD)</th>
            <th>Дата</th>
        </tr>
        <tr th:each="apt : ${apartments}">
            <td><a th:href="${apt.url}" th:text="${apt.title}"></a></td>
            <td th:text="${apt.priceUah}"></td>
            <td th:text="${apt.priceUsd}"></td>
            <td th:text="${apt.datePosted}"></td>
        </tr>
    </table>
</div>
</body>
</html>
```

Результат запуску проєкту:



Висновок: Навчився створювати Spring Boot додатки, реалізовувати спілкування клієнта/сервера через REST, також навчився адаптувати базу даних в додаток та створювати Frontend для додатка.