ЗАТВЕРДЖУЮ
Начальник кафедри інформаційних технологій та систем електронних комунікацій, к.т.н., доцент
Назарій БУРАК
«____» _____ 20___ р.

МЕТОДИЧНА РОЗРОБКА

для проведення лекційного заняття із здобувачами освіти першого (бакалаврського) рівня вищої освіти спеціальності F3 «Комп'ютерні науки»
з дисципліни «WEB-технології та WEB-програмування»

Лекція № 3. Тема 1.3: «Побудова об'єктно-орієнтованих вебзастосунків на Java
(у т.ч. з урахуванням інклюзивності)»

Мета: сформувати у здобувачів освіти розуміння багаторівневої архітектури вебзастосунку та шаблону MVC, навички моделювання предметної області об'єктами Java, обробки винятків, а також знання принципів доступності (інклюзивності) вебзастосунків відповідно до рекомендацій WCAG.

Час: 2 години.

Місце заняття: лекційна аудиторія (або онлайн у MS Teams).

Матеріальне забезпечення: мультимедійний проєктор, ПК з доступом до мережі Internet, IntelliJ IDEA, браузер із засобами розробника, презентація до лекції, електронний курс у «Віртуальному університеті».

Навчальні питання:

1. Принципи ООП у веброзробці.
2. Шаблон MVC і багаторівнева архітектура: контролер, сервіс, репозиторій.
3. Моделювання предметної області. Сутності та DTO.
4. Обробка винятків і валідація даних.
5. Доступність (інклюзивність) вебзастосунків. Рекомендації WCAG 2.2.
6. Засоби перевірки доступності.

Після лекційного заняття здобувачі освіти повинні:

знати: призначення шарів багаторівневої архітектури, компоненти шаблону MVC, відмінність сутності від DTO, ієрархію винятків Java, чотири

принципи WCAG (POUR) і основні критерії доступності;

вміти: розкладати код вебзастосунку на шари, проєктувати класи предметної області, передавати залежності через конструктор, обробляти помилки валідації, створювати доступні вебсторінки та перевіряти їх засобами Lighthouse і axe.

Література:

1. Віртуальне навчальне середовище ЛДУ БЖД «Віртуальний університет» [Електронний ресурс]. – Доступний з: https://virt.ldubgd.edu.ua/course/view.php?id=2111
2. Sierra K., Bates B., Gee T. Head First Java. 3rd ed. – O'Reilly Media, 2022. – 752 p.
3. Fowler M. Patterns of Enterprise Application Architecture. – Addison-Wesley, 2002. – 560 p.
4. Web Content Accessibility Guidelines (WCAG) 2.2 [Електронний ресурс]. – W3C, 2023. – Доступний з: https://www.w3.org/TR/WCAG22/
5. ДСТУ EN 301 549:2022. Вимоги доступності до продуктів та послуг ІКТ.

І. Підготовка до заняття

Перед початком заняття викладач перевіряє роботу проєктора, доступ до демо-проєкту (тег `lab-03`) і браузера з розширенням axe DevTools для демонстрації аудиту доступності.

ІІ. Зміст лекції

Вступ

У лабораторній роботі № 2 увесь код роботи з базою даних містився в кількох класах DAO та методі `main`. Для навчального прикладу цього достатньо, але реальний вебзастосунок – це сотні класів, над якими працює команда. Щоб код можна було розвивати, тестувати і змінювати по частинах, його поділяють на **шари** з чіткими обов'язками. Водночас вебзастосунок має бути зручним для **всіх** користувачів, зокрема людей з порушеннями зору, слуху чи моторики, – це вимога доступності (інклюзивності).

1. Принципи ООП у веброзробці

Принцип	Суть	Приклад у демо-проєкті
Інкапсуляція	дані об'єкта змінюються лише через його методи	стан укриття змінює лише метод `applyInspection()`
Абстракція	робота з інтерфейсом, а не з реалізацією	сервіс залежить від інтерфейсу `ShelterRepository`
Поліморфізм	різні реалізації одного інтерфейсу взаємозамінні	`JdbcShelterRepository` (ЛР 3) → Spring Data JPA (ЛР 8)
Наслідування	спільна поведінка в базовому класі	власні винятки успадковують `RuntimeException`

Таблиця 1 – Принципи ООП у вебзастосунку

Також застосовують принципи **SOLID**: зокрема, принцип єдиної відповідальності (кожен клас має одну причину для змін) і принцип інверсії залежностей (верхні шари залежать від абстракцій, а не від конкретних класів).

2. Шаблон MVC і багаторівнева архітектура

> **MVC** (Model–View–Controller) – шаблон, що розділяє застосунок на **модель** (дані і бізнес-логіка), **представлення** (відображення даних користувачеві) та **контролер** (приймає запит, звертається до моделі, обирає представлення).

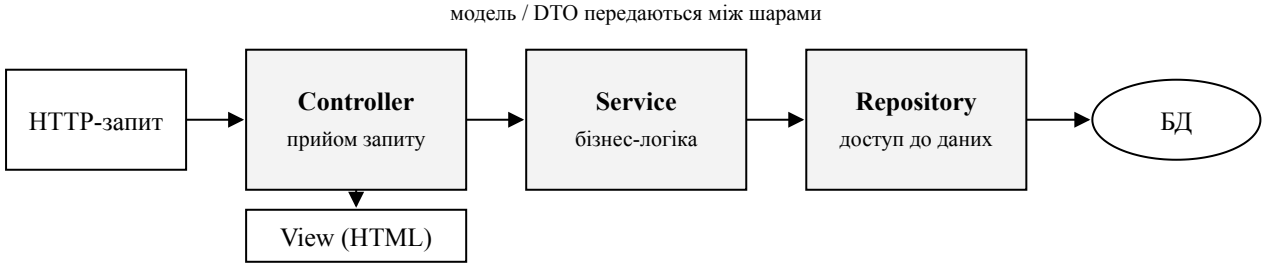

Рисунок 1 – Багаторівнева архітектура вебзастосунку

Контролер перетворює HTTP-запит на виклик сервісу і вибирає представлення; він не містить бізнес-правил і SQL. **Сервіс** реалізує бізнес-логіку: перевіряє дані, застосовує правила предметної області, керує транзакціями. **Репозиторій** (DAO) інкапсулює доступ до БД. Кожен шар звертається лише до нижчого. Залежності передаються через **конструктор** – так їх легко замінити, наприклад на тестову заглушку:

```
public class ShelterService {
    // інтерфейси, а не JDBC-класи
    private final ShelterRepository shelters;
    private final DistrictRepository districts;

    public ShelterService(ShelterRepository shelters,
                          DistrictRepository districts) {
        this.shelters = shelters;
        this.districts = districts;
    }
}
// «збирання» застосунку (у Spring — контейнер, тема 2.1)
ShelterService service = new ShelterService(
        new JdbcShelterRepository(), new JdbcDistrictRepository());
```

Лістинг 1 – Впровадження залежностей через конструктор

3. Моделювання предметної області. Сутності та DTO

Сутність (entity) – клас, що відповідає об’єкту предметної області і, як правило, таблиці БД: `Shelter`, `District`, `Inspection`. Сутність містить не лише поля, а й поведінку – бізнес-правила, що стосуються саме цього об’єкта. Перелічувані значення (тип укриття, стан, результат перевірки) задають типами `enum`, а не рядками – компілятор не дасть записати неіснуюче значення.

```
public enum ShelterStatus {
    READY("Готове"), LIMITED("Обмежено готове"),
    NOT_READY("Не готове");
    // ...
}
public class Shelter {
    private ShelterStatus status = ShelterStatus.NOT_READY;

    // бізнес-правило — всередині сутності
    public void applyInspection(InspectionResult result,
                                LocalDate date) {
        this.status = ShelterStatus.from(result);
        this.lastInspectionDate = date;
    }
}
```

Лістинг 2 – Сутність з бізнес-правилом

DTO (Data Transfer Object) – простий об'єкт для передавання даних між шарами чи назовні: лише ті поля, які потрібні конкретній сторінці або запиту API. Наприклад, `ShelterView` містить назву району і «людські» підписи стану, а `CreateShelterRequest` – дані з форми. DTO відокремлюють зовнішній вигляд від внутрішньої моделі: зміна таблиці БД не ламає сторінки і API. У Java 16+ DTO зручно оголошувати як `record`.

4. Обробка винятків і валідація даних

Винятки в Java поділяють на **перевірювані** (checked, напр. `SQLException` – компілятор вимагає їх обробити) і **неперевірювані** (unchecked, нащадки `RuntimeException`). Шар репозиторію перетворює технічні `SQLException` на власний неперевірюваний `DataAccessException`, щоб верхні шари не залежали від JDBC. Для ситуацій предметної області створюють власні винятки: `NotFoundException` (об'єкт не знайдено → HTTP 404), `ValidationException` (некоректні дані → HTTP 400).

Сервіс перевіряє **всі** поля і повертає повний перелік помилок, а не лише першу, – користувачеві не доведеться надсилати форму багато разів. Контролер перехоплює виняток і показує сторінку з повідомленнями:

```
public String create(CreateShelterRequest request) {
    try {
        service.create(request);
        return listPage.render(service.findAll(), List.of(),
                               "Укриття додано");
    } catch (ValidationException e) {
        return listPage.render(service.findAll(),
                               e.getErrors(), null);
    }
}
```

Лістинг 3 – Обробка помилок валідації в контролері

Дані, отримані від користувача, перед виведенням на сторінку обов'язково **екранують** (< → `<`), інакше можлива атака XSS – впровадження шкідливого скрипта.

5. Доступність (інклюзивність) вебзастосунків. WCAG 2.2

Доступність (accessibility, a11y) – можливість користуватися вебзастосунком для всіх людей, зокрема з порушеннями зору (незрячі користуються **екранними читачами**), слуху, моторики (керують лише

клавіатурою) чи когнітивних функцій. Це вимога законодавства (в Україні – ДСТУ EN 301 549, у ЄС – European Accessibility Act) і водночас показник якості: доступний сайт зручніший для всіх. Міжнародний стандарт – **WCAG 2.2** (W3C), рівні відповідності A, AA (рекомендований мінімум), AAA. Чотири принципи (POUR):

Принцип	**Приклади вимог**
Сприйманість (Perceivable)	текстова альтернатива зображень (`alt`); контраст тексту не менше 4,5:1; інформація не передається лише кольором
Керованість (Operable)	усі функції доступні з клавіатури; видимий фокус; посилання «Перейти до основного вмісту»
Зрозумілість (Understandable)	мова сторінки (`lang="uk"`); підписи полів форми; зрозумілі повідомлення про помилки
Надійність (Robust)	коректна семантична розмітка, сумісна з допоміжними технологіями; атрибути ARIA, де потрібно

Таблиця 2 – Принципи WCAG

Практичні правила: використовуйте **семантичні елементи** (`<header>`, `<nav>`, `<main>`, `<button>`) замість `<div onclick>`; дотримуйтеся ієрархії заголовків `h1`–`h3`; кожне поле форми пов'язуйте з `<label for>`, групи перемикачів – з `<fieldset>` і `<legend>`; для таблиць задавайте `<caption>` і `<th scope>`; повідомлення про помилки оголошуйте атрибутом `role="alert"`, а про успіх – `role="status"`.

```
<label for="capacity">Місткість, осіб</label>
<input id="capacity" name="capacity" type="number"
       min="1" required aria-describedby="capacity-hint">
<p id="capacity-hint">Скільки людей вміщує укриття</p>

<div class="errors" role="alert">
  <h2>Виправте помилки</h2> …
</div>
```

Лістинг 4 – Доступне поле форми та оголошення помилок

Інклюзивність стосується не лише інтерфейсу, а й даних: у демо-проєкті для кожного укриття зберігається ознака доступності для маломобільних груп населення (пандус, ліфт), яку відображає вебзастосунок.

6. Засоби перевірки доступності

Lighthouse (DevTools → Lighthouse → Accessibility) – автоматичний аудит з оцінкою 0–100 і переліком порушень. **axe DevTools** – розширення браузера з детальним описом кожного порушення і посиланням на критерій WCAG. **WAVE**

– онлайн-перевірка опублікованих сторінок. Автоматичні засоби знаходять лише частину проблем, тому обов'язкова **ручна перевірка**: пройти сторінку тільки клавіатурою (Tab, Shift+Tab, Enter, Space) і з екранним читачем (VoiceOver – Cmd+F5 у macOS, NVDA – у Windows). У демо-проєкті файл `web/inaccessible-example.html` містить навмисні порушення, а `docs/accessibility.md` – їх розбір і виправлення.

Висновки

Багаторівнева архітектура розділяє вебзастосунок на контролери, сервіси і репозиторії; шаблон MVC відокремлює дані від їх відображення. Сутності містять дані та бізнес-правила, DTO – лише дані для обміну. Власні винятки й повна валідація роблять поведінку застосунку передбачуваною. Доступність за WCAG – обов'язкова вимога якості: семантична розмітка, підписи, контраст, керування з клавіатури. Цю архітектуру ми перенесемо на сервлети (тема 1.4) і Spring Boot (модуль 2).

Контрольні запитання

1. Які обов'язки має кожен шар: контролер, сервіс, репозиторій?
2. Що означає кожна літера в абревіатурі MVC?
3. Чому сервіс має залежати від інтерфейсу репозиторію, а не від JDBC-класу?
4. Чим сутність відрізняється від DTO?
5. Чим перевірювані винятки відрізняються від неперевірюваних?
6. Назвіть чотири принципи WCAG і по одному прикладу вимог.
7. Чому не можна передавати інформацію лише кольором?
8. Як зробити поле форми доступним для екранного читача?
9. Які засоби перевірки доступності ви знаєте? Чому потрібна ручна перевірка?

Завдання для самостійної роботи

Опрацювати матеріал лекції. Ознайомитися з рекомендаціями WCAG 2.2 та засобами автоматизованої перевірки доступності. Обрати тему та погодити індивідуальне завдання на курсову роботу. Пройти тест № 3 у «Віртуальному університеті».

III. Підведення підсумків заняття

Узагальнити основні положення лекції, відповісти на запитання здобувачів освіти. Наголосити на зв’язку матеріалу з лабораторною роботою № 3 та курсовою роботою. Видати завдання для самостійної роботи.

Методичну розробку розробив:

Доцент кафедри ІТтаСЕК, к.т.н. Роман ГОЛОВАТИЙ

Методична розробка обговорена на засіданні кафедри ІТтаСЕК
Протокол № ____ від «____» _____ 20___ р.