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

МЕТОДИЧНА РОЗРОБКА

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

Лабораторна робота № 4: «Розроблення вебзастосунку на основі Servlet API та його розгортання в контейнері сервлетів Apache Tomcat» (тема 1.4)

Мета: навчитися створювати сервлети й JSP-представлення, обробляти форми за шаблоном Post/Redirect/Get, керувати сесією користувача, реалізовувати фільтри автентифікації та авторизації, запускати вебзастосунок на вбудованому Tomcat і розгортати WAR-архів на окремому сервері.

Час: 4 години.

Місце заняття: комп'ютерна лабораторія.

Матеріальне забезпечення: ПК з доступом до мережі Internet; JDK 21; IntelliJ IDEA; Maven; PostgreSQL; Apache Tomcat 10.1 (для варіанта з окремим сервером); браузер Chrome; Git; демо-проєкт https://github.com/SidiromUA/shelter-registry (тег `lab-04`).

Після лабораторного заняття здобувачі освіти повинні:

вміти: оголошувати сервлети, фільтри і слухачі анотаціями; читати параметри запиту і передавати дані в JSP; виводити дані засобами EL і JSTL з екрануванням; використовувати forward і redirect; зберігати користувача в сесії; обмежувати доступ за роллю; збирати WAR-архів і розгортати його на Tomcat;

знати: життєвий цикл сервлета, відмінність forward від redirect, механізм сесій на основі cookie `JSESSIONID`, правила безпеки сесій, призначення фільтрів і слухачів, структуру WAR-архіву.

I. Підготовка до заняття

Викладач перевіряє на ПК лабораторії наявність JDK 21, Maven і запущеної БД PostgreSQL. Здобувачі освіти до заняття опрацьовують лекцію 4 і мають власний проєкт з шарами model, dto, repository, service з ЛР 3.

II. Структура заняття

Короткі теоретичні відомості

Контейнер сервлетів (Apache Tomcat) приймає HTTP-запити і передає їх **сервлетам** – класам, що успадковують `HttpServlet` і перевизначають `doGet`/`doPost`. Сервлет існує в одному екземплярі, тому в його полях зберігають лише спільні об'єкти (сервіси). **JSP** з тегами JSTL формує HTML; файли в `WEB-INF` недоступні напряму з браузера. Після успішного POST виконують **redirect** (шаблон PRG), у разі помилок – **forward** на форму з кодом 400. **Сесія** зберігає користувача між запитами; після входу видають новий ідентифікатор сесії. **Фільтр** перевіряє доступ до групи URL, **слухач** `ServletContextListener` створює сервіси під час запуску застосунку.

2.1. Огляд демо-проєкту

```
git checkout lab-04
mvn compile exec:exec      # вбудований Tomcat
# http://localhost:8080/shelters
```

Розгляньте зміни порівняно з `lab-03` (`git diff lab-03 lab-04 --stat`): у `pom.xml` – `<packaging>war</packaging>` і залежність `jakarta.servlet-api` зі `scope=provided`; пакет `web` замість `controller` і `view`; каталог `src/main/webapp` з JSP-сторінками і `web.xml`.

URL	Метод	Обробник	Доступ
`/shelters`	GET	`ShelterServlet` → `shelters.jsp`	усі
`/shelters`	POST	`ShelterServlet` (валідація, PRG)	після входу
`/shelters/delete`	POST	`ShelterDeleteServlet`	лише ADMIN
`/login`	GET, POST	`LoginServlet` → `login.jsp`	усі
`/logout`	POST	`LogoutServlet`	усі

Таблиця 1 – Маршрути демо-проєкту (тег lab-04)

Простежте шлях запиту: `EncodingFilter` → `AuthFilter` → `ShelterServlet.doPost()` → `ShelterService.create()` → redirect на

`/shelters` → `doGet()` → forward на `shelters.jsp`. Сервіси створює `AppContextListener` і зберігає в `ServletContext`:

```
@WebListener
public class AppContextListener
        implements ServletContextListener {
    public void contextInitialized(ServletContextEvent e) {
        ServletContext ctx = e.getServletContext();
        ctx.setAttribute("shelterService", new ShelterService(
            new JdbcShelterRepository(),
            new JdbcDistrictRepository(),
            new JdbcInspectionRepository()));
        ctx.setAttribute("authService",
            new AuthService(new JdbcUserRepository()));
    }
}
```

2.2. Дослідження сесій і доступу

1. Відкрийте http://localhost:8080/shelters без входу. У DevTools → *Application* → *Cookies* перевірте, чи є cookie `JSESSIONID`.
2. Увійдіть як `inspector` / `inspector`. Порівняйте значення `JSESSIONID` до і після входу (має змінитися) і перевірте прапорець *HttpOnly*.
3. Додайте укриття з порожньою назвою і від'ємною місткістю: на вкладці *Network* знайдіть відповідь з кодом 400 і перелік помилок. Додайте коректне укриття: відповідь 302, потім GET /shelters. Натисніть F5 – повторного додавання немає.
4. Спробуйте видалити укриття від імені `inspector` командою curl або через форму, змінену в DevTools, – очікується 403. Увійдіть як `admin` і видаліть тестове укриття.
5. Перевірте, що `http://localhost:8080/WEB-INF/views/shelters.jsp` повертає 404.

```
curl -i -X POST http://localhost:8080/shelters \
     -d "name=Test&capacity=10"
# 302 Location: /login – AuthFilter не пропустив
```

2.3. Індивідуальне завдання

Перетворіть власний проєкт з ЛР 3 на вебзастосунок на Servlet API:

1. У `pom.xml` встановіть `packaging=war`, додайте `jakarta.servlet-api` 6.0 (`provided`), JSTL і вбудований Tomcat для запуску (див. `pom.xml` і `DevServer.java` демо-проєкту).

2. Створіть `AppContextListener`, що створює ваші сервіси і зберігає їх в атрибутах `ServletContext`.
3. Створіть сервлет головної сутності: GET – список у JSP-таблиці (JSTL, `c:out`) і форма додавання; POST – створення через сервіс, у разі успіху flash-повідомлення і redirect, у разі `ValidationException` – код 400, перелік помилок і збережені значення полів.
4. Додайте сервлет видалення (або редагування) об'єкта методом POST; для відсутнього об'єкта – код 404.
5. Реалізуйте вхід і вихід: таблиця користувачів з ролями (щонайменше дві), `LoginServlet` з новою сесією після входу, `LogoutServlet` з `invalidate()`, шапка сторінки з іменем користувача.
6. Створіть `EncodingFilter` (UTF-8) і фільтр авторизації: зміни – лише після входу; щонайменше одна дія – лише для певної ролі (403 для інших).
7. У `web.xml` налаштуйте тайм-аут сесії, `http-only` cookie і сторінки помилок 403, 404, 500. Сторінки мають зберегти доступність з ЛР 3.
8. Запустіть застосунок на вбудованому Tomcat. Додатково: зберіть WAR (`mvn package`) і розгорніть його в окремому Tomcat 10.1.
9. Зафіксуйте зміни, створіть тег `lab-04` і надішліть на GitHub.

Примітка. Паролі в навчальному проєкті поки зберігаються у відкритому вигляді з префіксом `{noop}`; у ЛР 8 їх замінить хешування BCrypt засобами Spring Security.

2.4. Вимоги до звіту

Звіт (PDF або DOC, до 6 сторінок) містить: мету; таблицю маршрутів вашого застосунку (URL, метод, обробник, доступ); фрагменти коду сервлета з обробкою форми, фільтра авторизації і JSP-сторінки зі списком; знімки сторінок (список, форма з помилками, вхід, сторінка помилки 403); знімки DevTools із cookie `JSESSIONID` до і після входу та ланцюжком 302 → 200 після додавання; посилання на репозиторій з тегом `lab-04`; відповіді на 3 контрольні запитання; висновки. У ВУ в завданні «ЛР 4» у поле «Текстова відповідь» вставте посилання на репозиторій, файлом додайте звіт.

2.5. Контрольні запитання

1. Чим контейнер сервлетів відрізняється від вебсервера?

2. Опишіть життєвий цикл сервлета. Чому в його полях не зберігають дані користувача?
3. Чим forward відрізняється від redirect? Навіщо шаблон PRG?
4. Як сервер «пам’ятає» користувача між запитами?
5. Навіщо після входу створювати нову сесію?
6. Як працює ланцюжок фільтрів? Що станеться, якщо не викликати `chain.doFilter()`?
7. Чим автентифікація відрізняється від авторизації? Які коди стану HTTP їм відповідають?
8. Чому JSP розміщують у `WEB-INF`, а Servlet API позначають `scope=provided`?

2.6. Критерії оцінювання (максимально 5 балів)

Бали	Критерій
5	усі сервлети, JSP, фільтри і слухач працюють; PRG і валідація з кодом 400; вхід, вихід і обмеження за роллю (403); налаштовано `web.xml`; застосунок розгорнуто з WAR; тег `lab-04` на GitHub; здобувач вільно пояснює роботу сесій і фільтрів
4	робота виконана повністю, є незначні недоліки (напр. немає сторінок помилок або розгортання WAR)
3	виконано не менше 60 % (напр. немає авторизації за роллю або PRG) або є суттєві помилки
0–2	виконано менше 60 % або здобувач не може пояснити поданий код

Робота, подана після встановленого терміну без поважних причин, оцінюється з пониженням на 1 бал.

III. Підведення підсумків заняття

Викладач перевіряє роботу застосунків, звертає увагу на типові помилки (дані користувача в полях сервлета, відсутність redirect після POST, виведення даних без `c:out`, перевірка ролі лише в JSP без перевірки на сервері), повідомляє, що в ЛР 5 проєкт буде перенесено на Spring Boot.

Методичну розробку розробив:

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

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