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 № 6: «Конфігурація застосунку Spring Boot, профілі середовищ, керування залежностями засобами Maven/Gradle. Збирання та запуск застосунку»
(тема 2.2)

Мета: навчитися виносити конфігурацію застосунку Spring Boot у файли YAML і змінні оточення, створювати типізовані налаштування з перевіркою значень, налаштовувати профілі dev і prod, керувати залежностями засобами Maven, налаштовувати журналювання і Actuator, збирати виконуваний JAR і Docker-образ та запускати застосунок разом з БД через Docker Compose.

Час: 4 години.

Місце заняття: комп'ютерна лабораторія.

Матеріальне забезпечення: ПК з доступом до мережі Internet; JDK 21; IntelliJ IDEA; Maven; PostgreSQL; Docker Desktop (за наявності); Postman або curl; браузер Chrome; Git; демо-проєкт https://github.com/SidiromUA/shelter-registry (тег `lab-06`).

Після лабораторного заняття здобувачі освіти повинні:

вміти: описувати конфігурацію у форматі YAML; використовувати заповнювачі і змінні оточення; створювати класи `@ConfigurationProperties` з `@Validated`; створювати і активувати профілі; аналізувати дерево залежностей Maven; журналювати події через SLF4J з правильними рівнями; вмикати ендпоінти Actuator; збирати JAR і Docker-образ; запускати багатоконтейнерний застосунок через Docker Compose;

знати: пріоритет джерел властивостей, правила роботи із секретами, призначення профілів, області видимості залежностей і фази життєвого циклу Maven, рівні журналювання, поняття образу, контейнера, тому, багатоетапного збирання.

I. Підготовка до заняття

Викладач перевіряє на ПК лабораторії наявність JDK 21, Maven, PostgreSQL і (за можливості) Docker. Здобувачі освіти до заняття опрацьовують лекцію 6 і мають власний проєкт Spring Boot з ЛР 5 (біни, `JdbcClient`, REST-контролер).

II. Структура заняття

Короткі теоретичні відомості

Spring Boot читає налаштування з багатьох **джерел**; за однакових ключів перемагає джерело з вищим пріоритетом: аргументи командного рядка → змінні оточення → файл профілю `application-<профіль>.yml` → основний `application.yml` → значення за замовчуванням. Запис `${DB_PASSWORD:shelter}` бере значення змінної оточення або значення за замовчуванням. Групу власних налаштувань описують записом з анотацією `@ConfigurationProperties(prefix=…)`; `@Validated` перевіряє значення під час запуску. **Профіль** активують властивістю `spring.profiles.active` або змінною `SPRING_PROFILES_ACTIVE`. Maven керує прямими і транзитивними залежностями, їх області видимості задає `<scope>`. Журнали пишуть через SLF4J з рівнями ERROR, WARN, INFO, DEBUG, TRACE. **Docker-образ** описує Dockerfile, а **Docker Compose** запускає застосунок разом з БД.

2.1. Огляд демо-проєкту

```
git checkout lab-06
git diff lab-05 lab-06 --stat
mvn spring-boot:run                # профіль dev — БД H2, PostgreSQL не потрібен
# http://localhost:8080/api/shelters/overdue
# http://localhost:8080/h2-console   (JDBC URL: jdbc:h2:mem:shelters, користувач sa)
```

Файл	Призначення
`application.yml`	спільні налаштування, власний блок `registry:`, Actuator, `server.port: ${PORT:8080}`
`application-dev.yml`	H2 у пам'яті, схема і дані під час кожного запуску, H2-консоль, DEBUG і журнал SQL
`application-prod.yml`	PostgreSQL зі змінних `DB_URL`, `DB_USER`, `DB_PASSWORD`; журнал у файл
`RegistryProperties`	типізовані налаштування з перевіркою `@NotBlank`, `@Min`, `@Email`
`Dockerfile`, `.dockerignore`	багатоетапне збирання образу; запуск від непривілейованого користувача
`docker-compose.yml`	застосунок + PostgreSQL 17, том для даних, перевірка стану БД

Таблиця 1 – Нові файли демо-проєкту (тег lab-06)

Знайдіть у журналі запуску рядок з активним профілем і SQL-запити рівня DEBUG. Порівняйте `pom.xml` з тегом `lab-05`: додано H2, H2-консоль, стартери validation і actuator.

2.2. Дослідження конфігурації

1. Зберіть JAR (`mvn clean package`) і запустіть його з різними параметрами (табл. 2). Для кожного запуску перевірте результат запитом у Postman або curl.
2. Запустіть застосунок з `--registry.inspection-interval-days=0` і прочитайте повідомлення «APPLICATION FAILED TO START». Поясніть, яка анотація його спричинила.
3. Задайте одночасно змінну `SERVER_PORT=9000` і аргумент `--server.port=9090`. Визначте, на якому порту запущено застосунок, і поясніть чому.
4. Виконайте `mvn dependency:tree` і знайдіть, через які стартери в проєкт потрапили Tomcat, Jackson і HikariCP.
5. Перевірте `GET /actuator/health` і `GET /actuator/info` (версія має збігатися з `pom.xml`). Переконайтеся, що `/actuator/env` недоступний (404).
6. За наявності Docker виконайте `docker compose up --build`, дочекайтеся стану healthy для БД і перевірте `GET /api/shelters`. Виконайте `docker compose ps`, `docker compose logs app`, `docker`

`compose down`; запустіть ще раз і переконайтеся, що дані збереглися в томі.

Запуск	**Очікуваний результат**
`java -jar target/shelter-registry-0.6.0.jar`	профіль dev, H2, працює без PostgreSQL
`SPRING_PROFILES_ACTIVE=prod java -jar …`	профіль prod, PostgreSQL з ЛР 2, журнал у `logs/`
`java -jar … --registry.inspection-interval-days=30`	інший перелік `/api/shelters/overdue`
`PORT=9091 java -jar …`	застосунок на порту 9091

Таблиця 2 – Запуски для перевірки

2.3. Індивідуальне завдання

Доповніть власний проєкт з ЛР 5:

1. Замініть `application.properties` на `application.yml`; задайте назву застосунку, порт через `${PORT:8080}` і профіль за замовчуванням dev.
2. Створіть профіль **dev** (H2 у пам’яті в режимі сумісності з вашою СУБД або локальна БД; автоматичне створення схеми і тестових даних; DEBUG для вашого пакета) і профіль **prod** (PostgreSQL; адреса, користувач і пароль – лише зі змінних оточення; INFO; журнал у файл). Перенесіть SQL-скрипти в `src/main/resources/db`.
3. Додайте блок власних налаштувань вашої предметної області (щонайменше три параметри, наприклад ліміти, інтервали, контактні дані) і клас `@ConfigurationProperties` з `@Validated` і обмеженнями. Використайте налаштування в бізнес-логіці і створіть REST-маршрут, результат якого залежить від них.
4. Додайте журналювання в сервіс: INFO для створення і видалення, WARN для відхилених даних, DEBUG для параметрів пошуку; використовуйте параметри `{}`; не журналюйте паролі і персональні дані.
5. Підключіть Actuator і відкрийте назовні лише `health` та `info`; у `info` виведіть назву і версію застосунку з POM.
6. Створіть `Dockerfile` з багатоетапним збиранням і запуском не від root, `.dockerignore` і `docker-compose.yml` із сервісами застосунку і БД (том, перевірка стану, змінні оточення для профілю prod).

7. Переконайтеся, що застосунок запускається в обох профілях, а за наявності Docker – через `docker compose up --build`.
8. Перевірте, що в репозиторії немає реальних паролів; додайте до `.gitignore` каталоги `target/`, `logs/` і локальні файли з секретами.
9. Оновіть `README.md` (запуск у профілях dev і prod, змінні оточення, запуск у Docker), зафіксуйте зміни, створіть тег `lab-06` і надішліть на GitHub.

Додаткові завдання (за бажанням): створіть аналогічний опис проєкту для Gradle (`build.gradle.kts`) і зберіть його командою `./gradlew bootJar`; додайте профіль `test` з `@ActiveProfiles("test")` у тесті; зберіть образ командою `mvn spring-boot:build-image` і порівняйте його розмір з образом з Dockerfile; реалізуйте бін з різними реалізаціями для профілів dev і prod (`@Profile`).

2.4. Вимоги до звіту

Звіт (PDF або DOC, до 6 сторінок) містить: мету; вміст `application.yml` і файлів профілів; код класу налаштувань і його використання; фрагмент сервісу з журналюванням; знімки журналу запуску в профілях dev і prod (рядок активного профілю), помилки запуску з некоректним налаштуванням, відповіді `/actuator/health` і маршруту, що залежить від налаштувань; фрагмент `mvn dependency:tree` з поясненням; Dockerfile, docker-compose.yml і знімок `docker compose ps` (за наявності Docker); посилання на репозиторій з тегом `lab-06`; відповіді на 3 контрольні запитання; висновки. У ВУ в завданні «ЛР 6» у поле «Текстова відповідь» вставте посилання на репозиторій, файлом додайте звіт.

2.5. Контрольні запитання

1. Навіщо відокремлювати конфігурацію від коду? Де зберігати секрети?
2. Назвіть джерела властивостей у порядку пріоритету.
3. Як задати властивість `spring.datasource.password` змінною оточення?
4. Чим `@ConfigurationProperties` кращий за `@Value` для групи налаштувань?
5. Як створити й активувати профіль? Що робить `@Profile`?
6. Що таке транзитивні залежності? Для чого області `runtime` і `test`?
7. Які фази виконує `mvn package`?

8. Які рівні журналювання ви використали і чому?
9. Чим образ відрізняється від контейнера? Навіщо багатоетапне збирання і том у Compose?

2.6. Критерії оцінювання (максимально 5 балів)

Бали	Критерій
5	конфігурація в YAML; профілі dev і prod працюють; секрети лише зі змінних оточення; типізовані налаштування з перевіркою використано в логіці; журналювання з правильними рівнями; Actuator; Dockerfile і Compose працюють; тег `lab-06` на GitHub; здобувач вільно пояснює пріоритет джерел і роботу контейнерів
4	робота виконана повністю, є незначні недоліки (напр. Compose не перевірено, неточні рівні журналів)
3	виконано не менше 60 % (напр. немає профілю prod або Dockerfile) або є суттєві помилки (пароль у репозиторії)
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 (табуляція в YAML, паролі в репозиторії, `localhost` замість імені сервісу в Compose, `System.out.println` замість логера, відкриті всі ендпоінти Actuator, профіль dev у промисловому запуску), повідомляє, що в ЛР 7 буде розроблено вебсторінки на Thymeleaf і повноцінний REST API з валідацією і документацією.

Методичну розробку розробив:

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

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