ЗВІТ

до лабораторної роботи № 1
на тему:
"Встановлення IntelliJ IDEA Community Edition та Apache Tomcat. Ознайомлення з базовою структурою Java Web-застосунків."

Виконала:
студентка групи **МІТ-21**
Йовхимищ Діана

Мета роботи: Ознайомитися з інтегрованим середовищем розробки IntelliJ IDEA Community Edition.Встановити та налаштувати Apache Tomcat для роботи з Java веб-застосунками.Розгорнути базовий веб-застосунок із використанням сервлету та перевірити його функціонування.

Необхідні інструменти та середовища:

1. JDK (Java Development Kit): Завантажити та встановити останню версію JDK. https://www.oracle.com/cis/java/technologies/downloads/#jdk23-windows
2. IntelliJ IDEA Community Edition: Інтегроване середовище розробки для Java. https://www.jetbrains.com/idea/download/?section=windows
3. Програмний інструмент управління проектами Apache Maven. https://maven.apache.org
4. Apache Tomcat: Сервер застосунків для розгортання веб-додатків на Java. https://tomcat.apache.org/index.html
5. Встановлений браузер для тестування веб-застосунку.

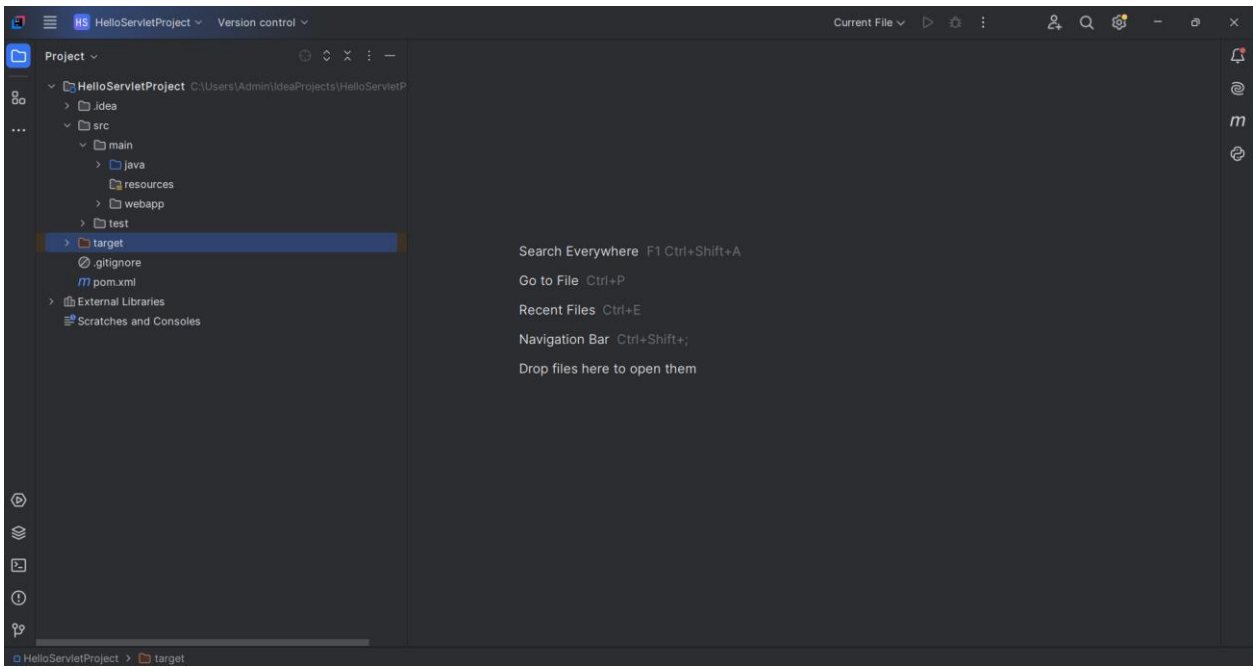
Завдання:

Завдання 1: Встановлення IntelliJ IDEA та налаштування проєкту

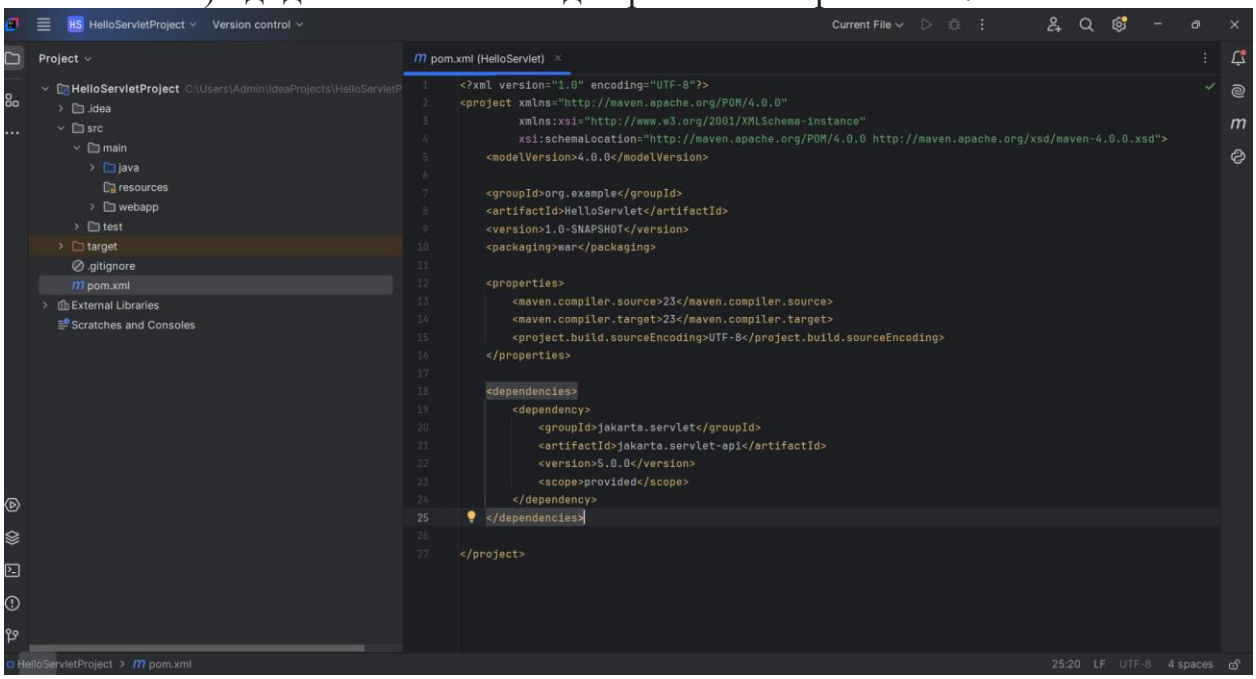
1. Перейдіть на офіційний сайт JetBrains: https://www.jetbrains.com/idea/ та завантажте Community Edition.
2. Виконайте встановлення IntelliJ IDEA, дотримуючись інструкцій майстра встановлення.
3. Після встановлення запустіть IntelliJ IDEA.
4. Створіть новий проєкт:

    1. Виберіть New Project.
    2. Оберіть тип проєкту: Maven.
    3. Укажіть JDK, установлену на вашому комп'ютері.
    4. Дайте назву проєкту: HelloServletProject.
    5. Натисніть Finish.

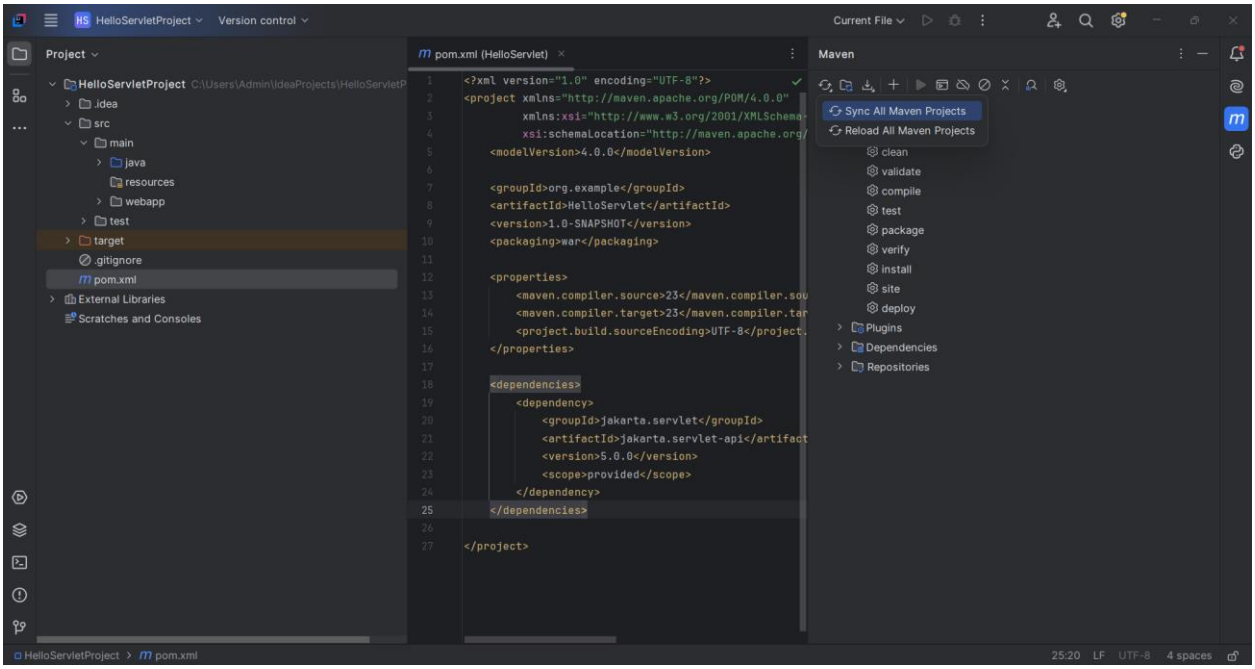
5. Перевірте структуру проєкту. Зверніть увагу, що Maven автоматично створює папки src/main/java, src/main/resources тощо.



6. Відкрийте файл pom.xml (основний файл для керування залежностями Maven) і додайте залежність для роботи із сервлетами.

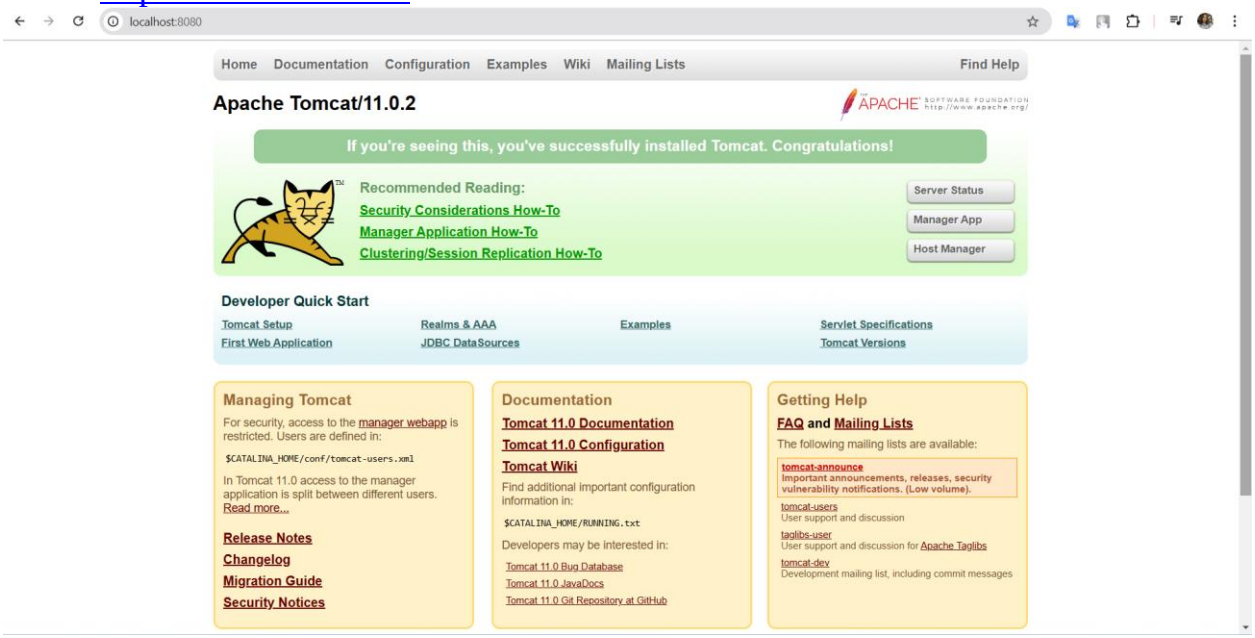


7. Виконайте синхронізацію Maven (натисніть Reload Maven Project у панелі).



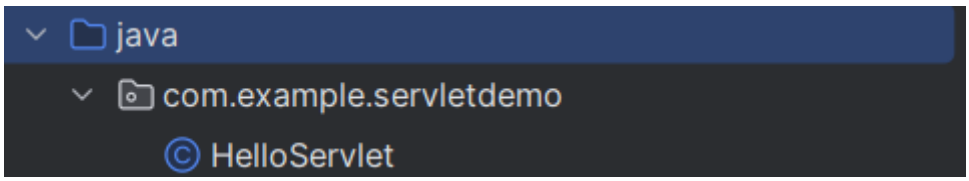
Завдання 2: Встановлення та налаштування Apache Tomcat

1. Завантажте останню версію Apache Tomcat з офіційного сайту: https://tomcat.apache.org/.
2. Розпакуйте завантажений архів у зручне місце (наприклад, C:/Tomcat).
3. Перейдіть у папку bin і запустіть файл startup.bat (Windows) або startup.sh (Linux/MacOS).
4. Відкрийте браузер і перевірте, чи Tomcat успішно працює, перейшовши за адресою: http://localhost:8080

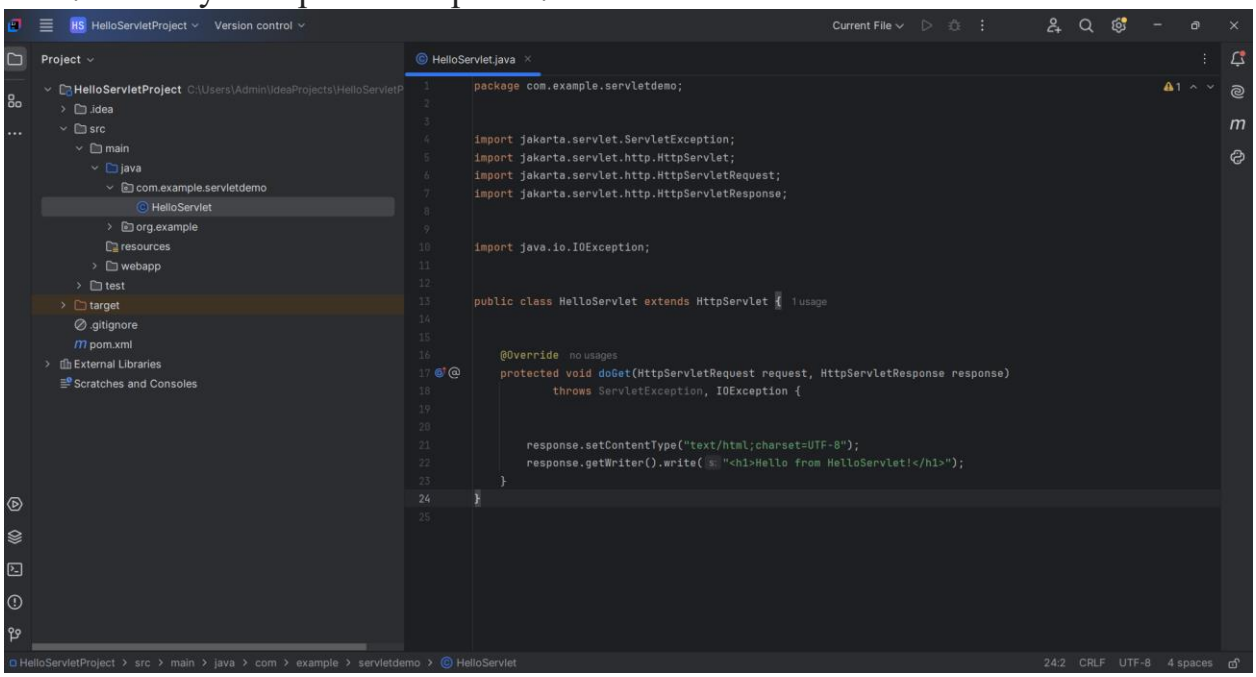


Завдання 3: Створення сервлету

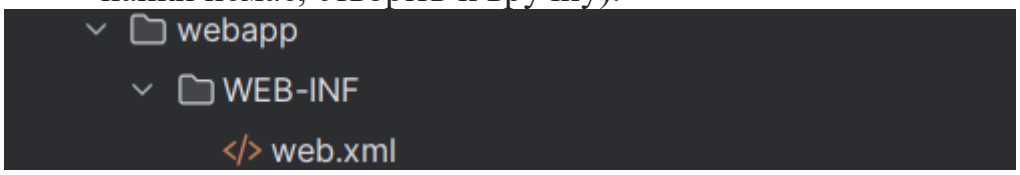
1. У вашому проєкті створіть новий Java-клас у папці src/main/java. Наприклад, створіть пакет com.example.servletdemo і в ньому клас HelloServlet.



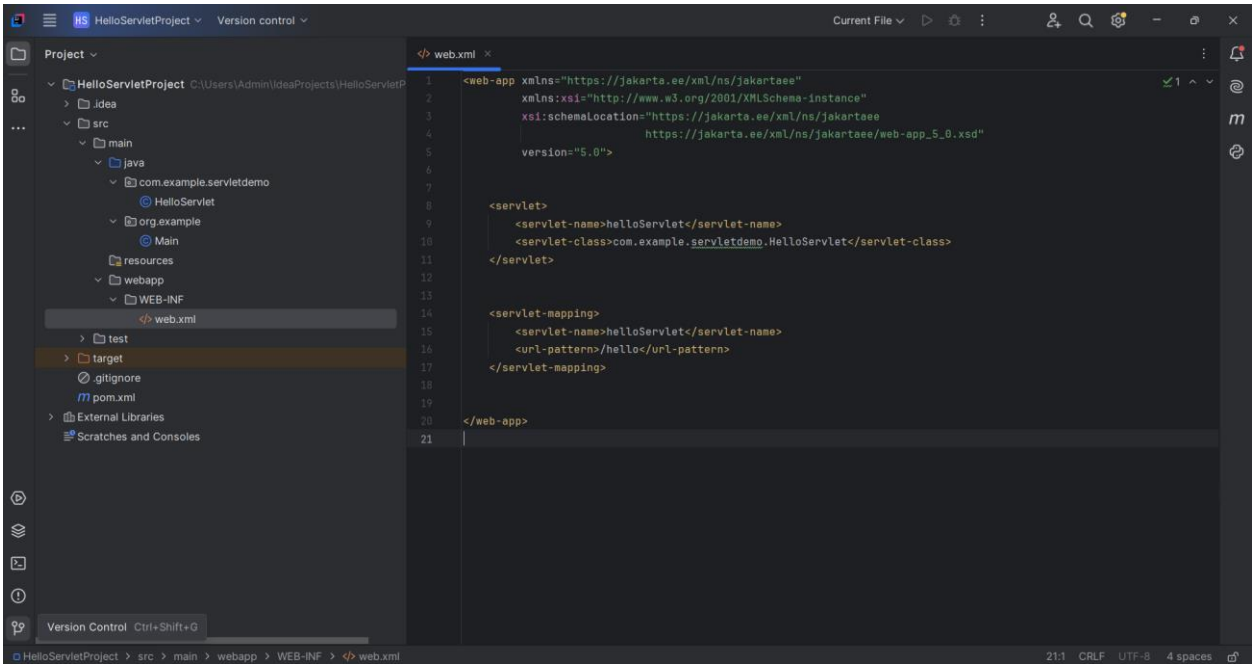
2. Реалізуйте простий сервлет.



3. Створіть файл web.xml у папці src/main/webapp/WEB-INF/ (якщо цієї папки немає, створіть її вручну).

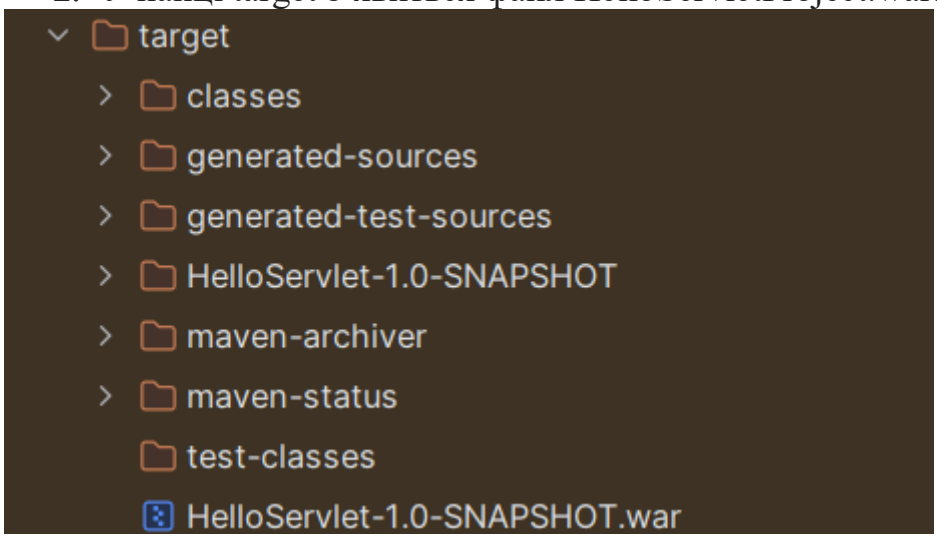


4. Додайте конфігурацію сервлету.

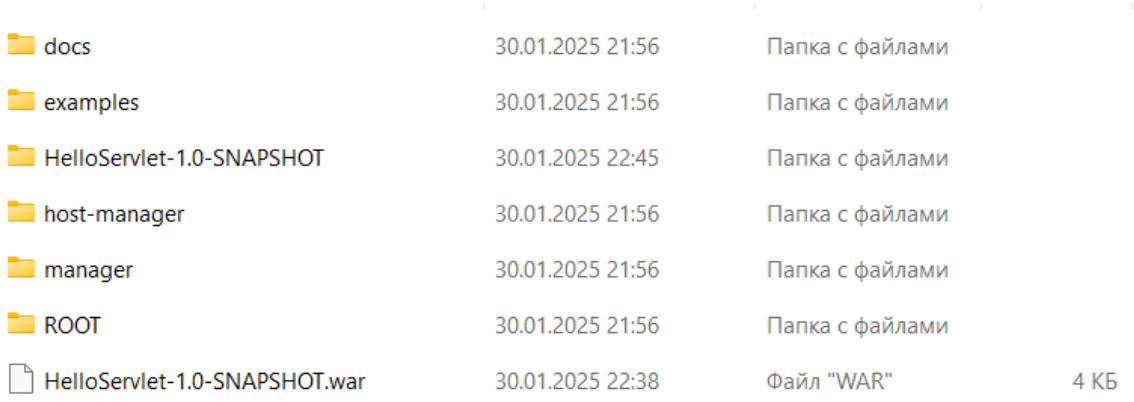


Завдання 4: Розгортання застосунку на Tomcat

1. Зберіть ваш проєкт, виконавши команду Maven:
   mvn clean package.
2. У папці target з’явиться файл HelloServletProject.war.



3. Скопіюйте цей .war файл у папку webapps вашого Tomcat.



4. Перезавантажте Tomcat (запустіть shutdown.bat і startup.bat).
5. Відкрийте браузер і перейдіть за адресою:
   http://localhost:8080/Lab1HelloServlet-1.0-SNAPSHOT/hello
6. Ви побачите повідомлення:
   Hello from HelloServlet!

Результат:

Контрольні питання:

1. Як Tomcat забезпечує виконання сервлетів?

Apache Tomcat є сервером застосунків (application server), який підтримує виконання Java Servlet та JavaServer Pages (JSP). Він працює як контейнер для сервлетів, забезпечуючи їх життєвий цикл: ініціалізацію, обробку запитів і відповідей, а також завершення роботи. Коли веб-застосунок розгортається на Tomcat, сервер автоматично завантажує сервлети, викликає їх методи

(наприклад, doGet або doPost) для обробки HTTP-запитів і повертає результат клієнту.

2. Що таке файл web.xml і яку роль він виконує у проєкті?

Файл web.xml (дескриптор розгортання) — це конфігураційний файл, який використовується для налаштування веб-застосунків у Java EE. Він містить інформацію про сервлети, їх URL-мапінги, параметри ініціалізації, фільтри, слухачі подій та інші налаштування. У контексті сервлетів, web.xml визначає, який клас сервлету відповідає за обробку певного URL.

3. Як працює анотація @WebServlet і чим вона відрізняється від налаштування в web.xml?

Анотація @WebServlet дозволяє налаштувати сервлет без використання файлу web.xml. Вона додається безпосередньо до класу сервлету і визначає URL-мапінг та інші параметри.

Відмінності:

web.xml: Вимагає окремого файлу для конфігурації, що може бути зручно для централізованого управління налаштуваннями.

@WebServlet: Зменшує кількість коду та спрощує конфігурацію, але менш гнучкий для складних налаштувань.

4. Що означає <scope>provided</scope> у залежності Maven?

Тег <scope>provided</scope> у Maven вказує, що залежність (наприклад, jakarta.servlet-api) буде надана середовищем виконання (наприклад, Apache Tomcat) і не повинна бути включена до фінального архіву (WAR-файлу). Це допомагає уникнути конфліктів версій і зменшує розмір архіву.

З лабораторної роботи:

```
<dependency>
    <groupId>jakarta.servlet</groupId>
    <artifactId>jakarta.servlet-api</artifactId>
    <version>5.0.0</version>
    <scope>provided</scope>
</dependency>
```

Це означає, що сервлет-API буде використовуватися під час компіляції, але не буде включено до WAR-файлу, оскільки Tomcat вже містить цю бібліотеку.

Висновок

У ході лабораторної роботи я виконала встановлення та налаштування середовища розробки IntelliJ IDEA Community Edition, сервера Apache Tomcat та засобу керування залежностями Maven. Ознайомившись із базовою структурою Java Web-застосунку, я створила перший веб-додаток із використанням сервлету, налаштувала його конфігурацію та успішно розгорнула на сервері Tomcat.

У процесі роботи я розглянула різні підходи до конфігурації сервлетів, зокрема використання файлу **web.xml** та анотації **@WebServlet**. Також вивчила принципи роботи Apache Tomcat як контейнера сервлетів, його механізм обробки HTTP-запитів та управління життєвим циклом сервлетів.

Окрім цього, опрацювала процес керування залежностями у **Maven**, зокрема використання області дії **<scope>provided</scope>**, що дозволяє уникати конфліктів версій бібліотек у середовищі виконання.

В результаті я отримала практичний досвід створення та запуску Java Web-застосунку, що є важливим кроком у вивченні серверної розробки на Java.