

## ЛАБОРАТОРНА РОБОТА 3

**Тема:** Реалізація динамічного web-проєкту засобами сервлетів

**Мета:** Створення динамічного web-проєкту з використанням сервлетів

Сервлет — це тип Java-класу, що виконується на сервері та призначений для динамічного формування відповіді на запит, що надходить з мережі в основному по HTTP-протоколу. За суттю, сервлети розширюють функціональні можливості webсерверу.

Обробка запиту статичної web-сторінки відбувається у наступній послідовності: у браузері користувач задає URL (Uniform Resource Locator — уніфікований вказівник інформаційного ресурсу), браузер формує запит за протоколом HTTP та направляє його вказаному в URL web-серверу. Сервер знаходить конкретний файл за запитом та повертає його браузеру у вигляді відповіді через протокол HTTP. HTTP-заголовок відповіді вказує на тип вмісту. Якщо тип MIME (Multipurpose Internet Mail Extensions) позначено як text/plain, то це звичайний текст у форматі ASCII, а якщо — text/html, то це HTML-документ.

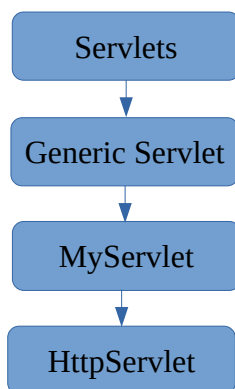
Якщо вміст web-сторінки залежить від часу отримання запиту до неї, то її називають динамічною, а її вміст може змінюватись. Зазвичай, web-сторінка, що створюється як відповідь на запит до бази даних, є динамічною. Більшість сторінок, які отримуються на запити в Інтернеті є динамічними. Використання технології сервлетів для формування динамічних web-сторінок має ряд особливостей: сервлет виконується в адресному просторі web-серверу; обробка декількох запитів клієнта можуть виконуватись в одному процесі; сервлети не залежать від конкретної платформи, тому що розроблюються на мові Java; диспетчер безпеки Java на сервері створює ряд обмежень для захисту його ресурсів; для сервлету є доступними усі функціональні можливості бібліотек класів Java. Сервлет може через механізми сокетів і віддаленого виклику методів (RMI) зв'язуватись з базою даних та іншим програмним забезпеченням.

Сервлети можуть бути вбудовані у різні сервери і виконуються у спеціальному середовищі виконання, який створює контейнер сервлетів або web-контейнер. Web-контейнер може бути компонентом-надбудовою (addon) HTTP-сервера або окремим сервером як Tomcat, GlassFish, JBoss, WildFly та інші. Сервлети інсталиуються до web-контейнерів як частина web-додатку, який також може вміщувати інші web-ресурси: HTML-сторінки, різні зображення та мультимедіа, JSP, XML-файли тощо. Після розгортання web-додатку у web-контейнері створюється та завантажується екземпляр класу сервлету на віртуальну машину Java (JVM) для обслуговування запитів, що надходять до серверу. Сервлет обслуговує кожний запит в окремому потоці. Тому, розробник сервлету вирішує як забезпечується синхронізація доступу до змінних екземпляру та змінних класу, визначає необхідність використання ресурсів баз даних та інше.

Контейнер сервлету (servlet container) забезпечує мережні служби, за допомогою яких отримуються та декодуються запити, формуються та надсилаються відповіді. Усі контейнери сервлетів підтримують HTTP та HTTPS протоколи. Контейнер сервлету може бути розподіленим, тобто дозволяє запускати розподілені web-додатки на декількох віртуальних машинах Java. При цьому віртуальні машини можуть бути запуснені, як на одному, так і на різних комп'ютерах. Контекстом сервлету (servlet context) є об'єкт, що вміщує представлення (вид) web-додатку, в якому запуснено сервлет. Контекст використовується для того, щоб сервлет міг вести журнал подій, отримувати URL-посилання на ресурси, а також встановлювати та зберігати атрибути, які можуть використовувати інші сервлети у додатку. Відображення сервлету (servlet mapping) визначає зв'язок між структурою URL та сервлетом. Як правило використовується для відображення запитів до сервлетів.

Сервлети створюються як дочірні класи пакета javax.servlet. Це дозволяє через класи та інтерфейси пакета javax.servlet описувати та визначати контракти між класом сервлету та середовищем виконання. Середовище виконання надається екземпляру класу сервлета за допомогою відповідного контейнера сервлету. Повний опис пакета за посиланням <http://docs.oracle.com/javaee/7/api/javax/servlet/package-summary.html>

Усі сервлети повинні реалізовувати інтерфейс javax.servlet.Servlet (<http://docs.oracle.com/javaee/7/api/javax/servlet/Servlet.html>) або безпосередньо, або розширюючи клас, який його реалізує, наприклад, HttpServlet:



Інтерфейс Servlet оголошує, але не реалізує методи, які керують сервлетом та його взаємодією з клієнтами. При написанні сервлету розробник повинен реалізовувати деякі або всі методи інтерфейсу сервлету.

Для забезпечення взаємодії з клієнтом використовуються два об'єкта:

- ServletRequest, який встановлює зв'язок від клієнта до серверу;
- ServletResponse, який встановлює зв'язок від сервлета до клієнту.

Обидва створюються на основі відповідних інтерфейсів пакета javax.servlet:

<http://docs.oracle.com/javaee/7/api/javax/servlet/ServletRequest.html>

<http://docs.oracle.com/javaee/7/api/javax/servlet/ServletResponse.html>

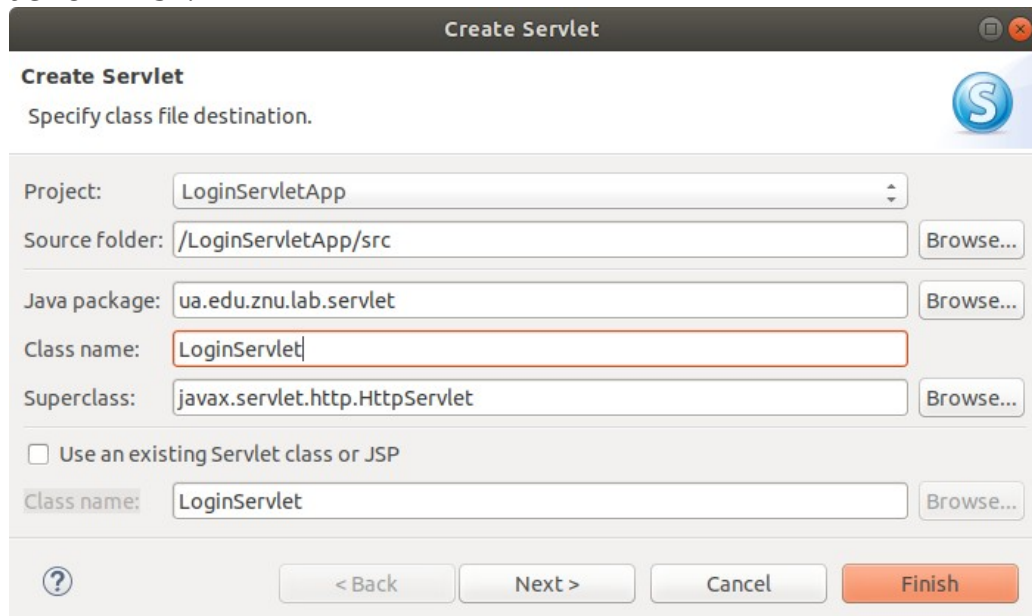
Інтерфейс `ServletRequest` забезпечує сервлету доступ до наданих клієнтом імен параметрів; протоколу, що використовується; імені віддаленого хосту, який виконав запит; імені сервера, який його отримав; вхідному потоку `ServletInputStream`. Сервлети використовують вхідний потік для отримання даних від клієнтів, які використовують протоколи рівня додатків і такі методи як HTTP POST та PUT. Інтерфейси, які розширюють інтерфейс `ServletRequest` дозволяють сервлетам отримувати більш специфічні дані протоколу. Наприклад, інтерфейс `HttpServletRequest` має методи для отримання спеціальної інформації HTTP заголовків.

Інтерфейс `ServletResponse` надає сервлету методи, для відправлення повідомлень клієнту. Він дозволяє сервлету встановлювати довжину вмісту і тип MIME відповіді, а також встановлювати вихідний потік, `ServletOutputStream` та `Writer`, через який сервлет може відправляти дані відповіді. Інтерфейси, що розширюють інтерфейс `ServletResponse` надають додаткові можливості. Наприклад, інтерфейс `HttpServletResponse` має методи, що дозволяють сервлету маніпулювати спеціальною інформацією HTTP заголовків.

## ПРАКТИЧНА ЧАСТИНА

Застосуємо технологію сервлетів до створення динамічного web-проекту призначеного для перевірки логіну та паролю користувача.

1. Створюємо новий динамічний web-проект в IDE Eclipse і задаємо йому ім'я, наприклад, `LoginServletApp`. В навігаторі проектів `Project Explorer` розкриваємо `Java Resources` та на папці `src` клацаємо правою кнопкою миші. В меню обираємо `New -> Servlet`. У віконці `Create Servlet`, що відкрилось, вказуємо ім'я пакету `ua.edu.znu.lab.servlet`, ім'я нового класу `LoginServlet` та натискаємо `Finish`.



2. Перед оголошенням класу є анотація `@WebServlet("/LoginServlet")`, яка вказує контейнеру сервлетів, що сервлет прив'язано до URL: `/LoginServlet`. У ранніх версіях Java EE, замість такої анотації, необхідно було в файлі `web.xml` робити дві записи: `<servlet>` та `<servlet-mapping>`.

Тепер замінюємо URL з `/LoginServlet` на `/login`, для чого змінюємо анотацію на `@WebServlet("/login")`.

У створеному класі `LoginServlet` автоматично утворились конструктор `LoginServlet()` та два методи `doGet` та `doPost`. Ці методи є перевизначеними для базового класу: `HttpServlet`. Метод `doGet` використовується для створення відповіді на `Get`-запит, а метод `doPost` — відповідає на `Post`-запит. Логін-форма буде створюватись у методі `doGet`, а обробка даних форми — в методі `doPost`. Але тому, що `doPost` може знадобитись для показу форми у випадку помилки введення даних, можна створити нову функцію `createForm`, яку можна викликати як з методу `doPost`, так і `doGet`.

3. Додамо функцію `createForm`:

```
protected String createForm(String errMsg) {
    StringBuilder sb = new StringBuilder("<h2>Login</h2>");
    //перевірка відображення повідомлення про помилку
    if (errMsg != null) {
        sb.append("<span style='color: red;'>")
        .append(errMsg)
        .append("</span>");
    }
    //створення форми
    sb.append("<form method='post'>\n")
    .append("User Name: <input type='text'
name='userName'><br>\n")
    .append("Password: <input type='password'
name='password'><br>\n")
    .append("<button type='submit'
name='submit'>Submit</button>\n")
    .append("<button type='reset'>Reset</button>\n")
    .append("</form>");
    return sb.toString();
}
```

4. Тепер змінюємо функцію `doGet` так, щоб з неї викликалась функція `createForm`, що повертає форму у відповідь:

```
protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    response.setContentType("text/html");
    response.getWriter().write(createForm(null));
}
```

Метод `getWriter` викликається на об'єкті `response` та записується вміст форми до нього, який отримується з виклику функції `createForm`. Коли форма

показується вперше, то ніяких повідомлень про помилку ще не може бути, тому в якості аргументу функції `createForm` передається `null`.

Для того, щоб браузер вірно оброблював відповідь необхідно вказувати її тип: `response.setContentType("text/html")` інакше браузер може показувати текст відповіді і не інтерпретувати його, як `html`.

5. Для обробки вмісту форми, яка публікується після натискання кнопки "Submit", змінюємо метод `doPost`:

```
protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    String userName = request.getParameter("userName");
    String password = request.getParameter("password");
    //Створення StringBuilder для рядка відповіді
    StringBuilder responseStr = new StringBuilder();
    if ("admin".equals(userName) && "admin".equals(password)) {
        responseStr.append("<h2>Welcome admin !</h2>")
            .append("You are successfully logged in");
    }
    else {
        //недійсні дані користувача
        responseStr.append(createForm("Invalid user id or password.
Please try again"));
    }
    response.setContentType("text/html");
    response.getWriter().write(responseStr.toString());
}
```

Використання метода `request.getParameter` дозволяє отримати ім'я користувача та пароль від об'єкта запиту. Якщо логін та пароль дійсні, то до рядка відповіді буде додано привітальне повідомлення. Якщо логін та пароль введено не правильно, то відповіддю буде `createForm` з повідомленням про помилку.

На завершенні записується відповідь, яку сформовано в `responseStr`.

6. Для запуску на сервері необхідно клацнути правою кнопкою миші на файлі `LoginServlet.java` в Project Explorer та обрати опцію `Run As | Run on Server`. Попередньо проект не додавався до серверу додатків `WildFly`, тому Eclipse запитає про можливість конфігурації серверу запускати цей сервлет. Зараз слід натиснути кнопку `Finish`, сервлет повинен запуститись за адресою `http://localhost:8080/LoginServletApp/login`. Перевірте роботу сервлету.

Повний текст коду `LoginServlet.java`:

```
package ua.edu.znu.lab.servlet;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```

/**
 * Servlet implementation class LoginServlet
 */
@WebServlet("/login")
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public LoginServlet() {
        super();
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        response.setContentType("text/html");
        response.getWriter().write(createForm(null));
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        String userName = request.getParameter("userName");
        String password = request.getParameter("password");
        //Створення StringBuilder для рядка відповіді
        StringBuilder responseStr = new StringBuilder();
        if ("admin".equals(userName) && "admin".equals(password)) {
            responseStr.append("<h2>Welcome admin !</h2>")
                .append("You are successfully logged in");
        }
        else {
            //недійсні дані користувача
            responseStr.append(createForm("Invalid user id or password! Please
try again"));
        }
        response.setContentType("text/html");
        response.getWriter().write(responseStr.toString());
    }
    protected String createForm(String errMsg) {
        StringBuilder sb = new StringBuilder("<h2>Login</h2>");
        //перевірка відображення повідомлення про помилку
        if (errMsg != null) {
            sb.append("<span style='color: red;'>")
                .append(errMsg)
                .append("</span>");
        }
        //створення форми
        sb.append("<form method='post'>\n")
            .append("User Name: <input type='text' name='userName'><br>\n")
            .append("Password: <input type='password' name='password'><br>\n")
            .append("<button type='submit' name='submit'>Submit</button>\n")
            .append("<button type='reset'>Reset</button>\n")
            .append("</form>");
        return sb.toString();
    }
}

```

Створення HTML форми або іншої сторінки у сервлеті є не дуже зручним, особливо при великих їх кількостях. Для цього краще використовувати JSP або HTML. Однак сервлети достатньо добре використовуються для реалізації контролерів в конфігурації Model-View-Controller (MVC), для обробки запитів, які генерують не текстову відповідь або для створення веб-сервісів або кінцевих точок веб-сокетів.

## **Завдання**

1. Виконайте пункти практичної частини лабораторної роботи. Зробіть скриншоти виконаних завдань.
2. Додайте у проект новий клас, що буде відповідати за утримання логинів та паролів, та створіть в ньому необхідні методи. Внесіть зміни у файл `LoginServlet.java` так, щоб логін та пароль користувача брались для перевірки саме з нового класу.
3. Внесіть зміни у код сторінки так, щоб кількість невдалих спроб пройти верифікацію обмежувалось трьома, а спроба оновити сторінку з формою після цього мало затримку 1 хвилину.
4. Зробіть експорт проекту у war-файл та скопіюйте його до каталогу `[WILDFLY_HOME]/standalone/deployments`, попередньо видаливши з серверу додатків раніше опублікований цей проект. У вікні Console проконтролюйте, що сервер додатків виконав його публікацію та перевірте роботу проекту у звичайному браузері.
5. Підготуйте звіт, в який додайте скриншоти виконаних прикладів та код рішення поставлених задач.

## **Список рекомендованої літератури**

1. Dave Wolf, A.J. Henley *Java EE Web Application Primer: Building Bullhorn: A Messaging App with JSP, Servlets, JavaScript, Bootstrap and Oracle*. - Apress Berkeley, CA. - 2017. - 145 p. <https://doi.org/10.1007/978-1-4842-3195-1>
2. Luciano Manelli, Giulio Zambon. *Beginning Jakarta EE Web Development: Using JSP, JSF, MySQL, and Apache Tomcat for Building Java Web Applications*. - Apress Berkeley, CA. - 2020. - 407 p. <https://doi.org/10.1007/978-1-4842-5866-8>