

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Український державний університет науки і технологій**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №5**

**з дисципліни «Компонентні технології програмного забезпечення»**

**на тему:**

**«**Розробка web-компонентів на основі сервлетів.**»**

Виконав:

Студент гр. ПЗ2011

Кулик С.В.

Прийняв:

Андрющенко В. О.

Дніпро, 2023

**Тема.** Розробка web-компонентів на основі сервлетів.

**Теоретичні відомості**

***Загальне уявлення про веб-додаток Java***

Веб-додаток Java створює інтерактивні веб-сторінки, що містять різні типи мов розмітки (HTML, XML і т.д.), а також динамічний вміст. Цей вміст зазвичай формується веб-компонентами, наприклад сторінками JavaServer(JSP), Сервлетами і компонентами JavaBean, які дозволяють змінювати дані і здійснювати їх тимчасове зберігання, взаємодіяти з базами даних і вебслужбами, а також відображати вміст у відповідь на запити клієнтів.

Так як багато завдань розробки веб-додатків можуть повторюватися або вимагати наявності надлишкового шаблонного коду, то для зменшення кількості спільних дій слід застосовувати веб-платформи. Багато платформ (наприклад, JavaServer Faces) надають бібліотеки для створення шаблонів сторінок і управління сеансами, а також часто забезпечують повторне використання коду.

***Загальні відомості про Java EE***

Java EE (Enterprise Edition) являє собою широко використовувану платформу, яка містить набір взаємозв'язаних технологій, які істотно скорочують вартість і складність розробки, розгортання багаторівневих серверних додатків, а також управління ними. Платформа Java EE заснована на платформі Java SE і надає набір інтерфейсів API (інтерфейсів розробки додатків) для розробки і запуску портуємих, надійних, масштабованих і безпечних серверних додатків.

Java EE в числі інших містить наступні компоненти:

* Enterprise JavaBeans (EJB): керована серверна архітектура компонентів, що використовується для інкапсуляції бізнес-логіки додатка. Технологія EJB дозволяє здійснювати швидку і спрощену розробку розподілених, транзакційних, безпечних і переносимих додатків, заснованих на технології Java.
* Інтерфейс API збереження стану Java (Java Persistence API, JPA): інфраструктура, що дозволяє розробникам керувати даними за допомогою об'єктно-реляційного зіставлення (ORM) в додатках, створених на платформі Java.

**Текст програми**

**Файл UserController.java**

package com.example.lab5;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet(

name = "UserController",

urlPatterns = {"/processuser"}

)

public class UserController extends HttpServlet {

public UserController() {

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

UserController.RequestUser customer = UserController.RequestUser.*fromRequestParameters*(request);

customer.setAsRequestAttributes(request);

List violations = customer.validate();

Iterator var5 = violations.iterator();

while(var5.hasNext()) {

Object obj = var5.next();

System.*out*.println(obj.toString());

}

if (!violations.isEmpty()) {

request.setAttribute("violations", violations);

}

String url = this.determineUrl(violations);

request.getRequestDispatcher(url).forward(request, response);

}

private String determineUrl(List violations) {

return !violations.isEmpty() ? "/" : "/WEB-INF/file.jsp";

}

private static class RequestUser {

private final String login;

private final String password;

private final String email;

private RequestUser(String login, String password, String email) {

this.login = login;

this.password = password;

this.email = email;

}

public static UserController.RequestUser fromRequestParameters(HttpServletRequest request) {

return new UserController.RequestUser(request.getParameter("login"), request.getParameter("password"), request.getParameter("email"));

}

public void setAsRequestAttributes(HttpServletRequest request) {

request.setAttribute("login", this.login);

request.setAttribute("password", this.password);

request.setAttribute("email", this.email);

}

public List validate() {

List violations = new ArrayList();

if (!UserController.LoginValidator.*validate*(this.login)) {

violations.add("error!");

}

if (!UserController.PasswordValidator.*validate*(this.password)) {

violations.add("error");

}

if (!UserController.EmailValidator.*validate*(this.email)) {

violations.add("error Email!");

}

return violations;

}

}

private static class PasswordValidator {

private static final String *passwordRegex* = "^(?=.\*[0-9])(?=.\*[a-z])(?=.\*[A-Z])(?=\\S+$).{8,}$";

private PasswordValidator() {

}

public static boolean validate(String password) {

return password.matches("^(?=.\*[0-9])(?=.\*[a-z])(?=.\*[A-Z])(?=\\S+$).{8,}$");

}

}

private static class LoginValidator {

private static final String *loginRegex* = "^[a-zA-Z0-9\_-]{3,16}$";

private LoginValidator() {

}

public static boolean validate(String login) {

return login.matches("^[a-zA-Z0-9\_-]{3,16}$");

}

}

private static class EmailValidator {

private static final String *emailRegex* = "^[\\w\\.-]+@[\\w\\.-]+\\.[\\w]{2,}$";

private EmailValidator() {

}

public static boolean validate(String email) {

return email.matches("^[\\w\\.-]+@[\\w\\.-]+\\.[\\w]{2,}$");

}

}

}

**Файл index.jsp**

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Реєстрація</title>

</head>

<body>

<h1>Зареєструватися</h1>

<**c:if** test="${violations != null}">

<**c:forEach** items="${violations}" var="violation">

<p>${violation}.</p>

</**c:forEach**>

</**c:if**>

<form action="${pageContext.request.contextPath}/processuser" method="post">

<label for="login">Логін : </label>

<input type="text" name="login" id="login" value="${login}">

<label for="password">Пароль: </label>

<input type="text" name="password" id="password" value="${password}">

<label for="email">Email: </label>

<input type="text" name="email" id="email" value="${email}">

<br><br><input type="submit" name="signup" value="Зареєструватися">

</form>

</body>

</html>

**Файл file.jsp**

<%@ **page** contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Дані реєстрації</title>

</head>

<body>

<h1>Дякуємо за реєстрацію</h1>

<h2>Ви створили аккаунт з наступними даними:</h2>

<p><strong>Логін: </strong> ${login}</p>

<p><strong>Пароль: </strong> ${password}</p>

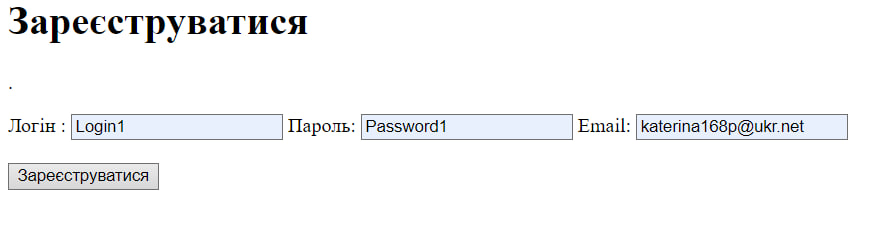
<p><strong>Email: </strong>${email}</p>

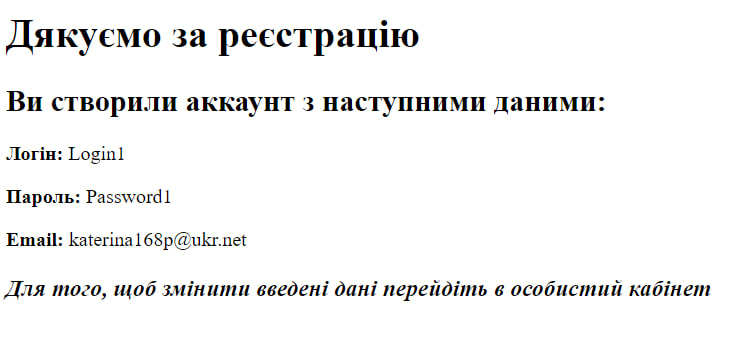
<h3><i>Для того, щоб змінити введені дані перейдіть в особистий кабінет</i></h3>

</body>

</html>

**Результати роботи програми**



**Висновок**

Розробка веб-компонентів на базі сервлетів - це процес створення елементів веб-додатків, які використовують сервлети як основу для обробки запитів від клієнтів. Цей підхід може бути реалізований з використанням різних мов програмування, таких як Java або Kotlin, і може бути використаний для створення різноманітних веб-додатків, таких як онлайн-магазини, соціальні мережі або системи управління контентом. Розробка веб-компонентів на базі сервлетів може включати в себе розробку HTML-шаблонів, CSS-стилів, JavaScript-скриптів та інших елементів веб-сторінок. Крім того, вона може вимагати налаштування веб-сервера, баз даних та інших засобів розробки. Основна перевага використання сервлетів полягає в їх здатності забезпечувати динамічну генерацію вмісту, що може змінюватися в залежності від запитів користувачів. Крім того, використання сервлетів дозволяє створювати безпечні веб-додатки, які захищені від атак з боку зловмисників.