ЛАБОРАТОРНА РОБОТА 2

Тема: Розробка компонент JSP
Мета: Створення динамічного web-проекту з використанням JSP та JSTL

JSP (Java Server Pages) є технологією, що дозволяє розробникам створювати веб-сторінки, які вміщують статичні та динамічні компоненти. Розробку технології JSP розпочато у 1997 році і згодом JSP було включено до складу Java EE. Сторінка JSP вміщує текст двох типів: статичні данні, які можуть бути в одному з текстових форматів HTML, SVG, WML, або XML, та JSP-елементи, що відповідають за створення динамічного вмісту. Крім того можуть використовуватись бібліотеки JSP-тегів, а також EL (Expression Language), для включення Java-коду у статичний вміст JSP-сторінки.

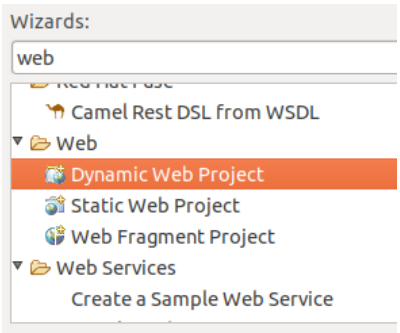
Сторінки компілюються JSP-компілятором в сервлети, які є Java-класами, і виконуються на сервері. Сервлети також можуть бути написані розробником, не використовуючи JSP-сторінки. Ці технології доповнюють одна одну. Контейнери сервлетів, які здатні виконувати JSP-сторінки, написано на платформонезалежній мові Java. JSP-сторінки завантажуються на сервері та керуються зі структури спеціального Java server packet, який має назву Java EE Web Application. Зазвичай сторінки упаковано у файлові архиви .war та .ear.

JSP є однією з високопродуктивних технологій, оскільки весь код сторінки транслюється в java-код сервлету за допомогою компілятора JSP сторінок (наприклад, Jasper), а потім компілюється в байт-код віртуальної машини java (JVM).

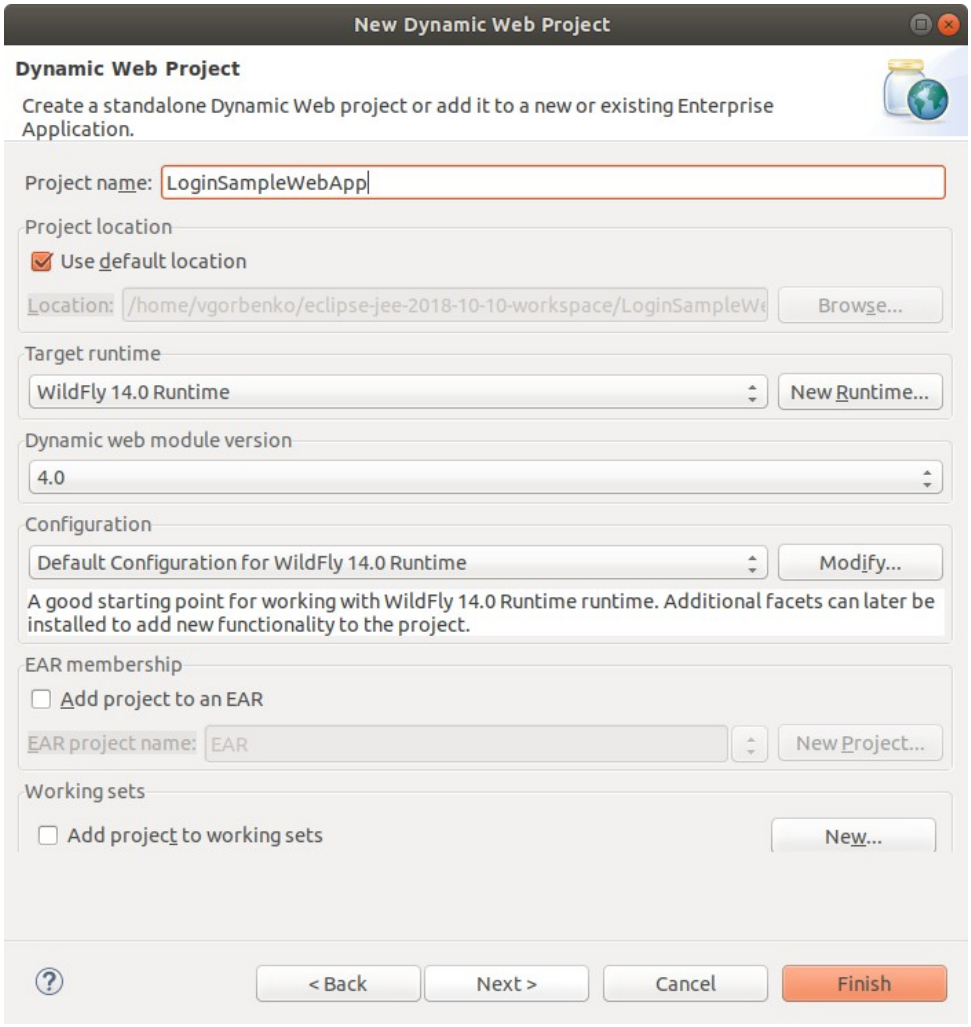
ПРАКТИЧНА ЧАСТИНА

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

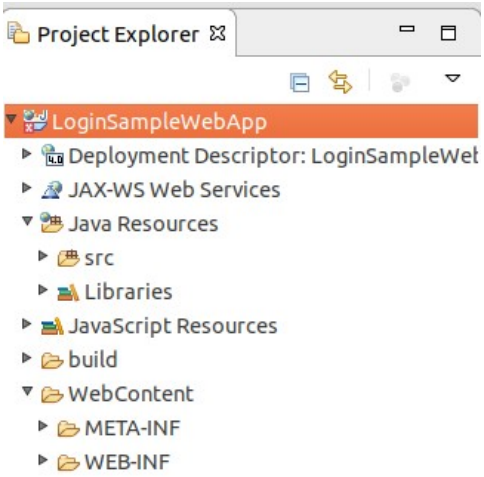
1. Для створення динамічного web-проекту в IDE Eclipse необхідно перейти до меню-візарда проектів: File -> New -> Other. В якості тексту пошуку можна набрати Web, що значно прискорить перехід до необхідного типу проектів. Обираємо Dynamic Web Project:



Наступним буде вікно для створення Dynamic Web Project:



Слід задати ім’я проекту, наприклад, LoginSampleWebApp. Інші налаштування залишаємо за замовчуванням, як показано на рисунку вище. Після натискання Finish утворюється структура нового проекту:



Файли Java будуть знаходитись у папці Java Resources/src. Такі web-ресурси як HTML, JS та CSS файли будуть розміщуватись у папці

WebContent.

2. Для створення jsp-файлу необхідно на папці WebContent клацнути правою кнопкою миші і обрати New -> JSP File. Назвемо новий файл index.jsp. Файл буде відкрито у редакторі, вікно якого буде поділено на дві частини: код та попередній вигляд сторінки. Зробимо заміну назви (затиснута тегами <title> </title>) на Login. Для створення форми Login додамо наступний код

```
<body>
<h2>Login:</h2>
<form method="post">
      User Name: <input type="text" name="userName"><br>
      Password: <input type="password" name="password"><br>
      <button type="submit" name="submit">Submit</button>
      <button type="reset">Reset</button>
</form>
</body>
```

Форма, що позначається тегом form може мати атрибут action, якому можна задати URL за яким данні з форми будуть надіслані, коли користувач натисне кнопку Submit. Якщо цьому атрибуту значення не вказувати, то данні з форми передаються тій самій сторінці, в якій представлено форму. У нашому випадку дані з форми будуть передаватись цьому ж файлу index.jsp.

3. Програмна частина на Java та код описання клієнтської частини (HTML, CSS та JavaScript) можуть знаходитись в одному файлі JSP. Розглянемо можливість поєднувати код Java з кодом HTML, хоча це і не вважається доброю практикою. Код Java записується в JSP між тегами <% %>. Такі блоки Java-коду в JSP називаються скриплетами або сценаріями. Відповідний атрибут може бути встановлено на рівні сторінки JSP. Вони називаються директивами сторінки і включаються між тегами <%@ %>. Створений index.jsp має саме такі директиви і вони знаходяться на початку цього файлу. Тип змісту повідомляє браузеру тип відповіді (html/текст), який повертає сервер і це впливає на відображення відповіді браузером.

У JSP є ряд об'єктів, які допомагають обробляти та генерувати відповіді, автоматично доступні як частина стандарту JSP без спеціального оголошення або імпорту. Ці об’єкти називаються неявними (implicit objects). Список об’єктів надано у таблиці і їх можна використовувати у коді JSP.

Об’єкт	Тип об’єкта	Призначення
request (запит)	javax.servlet.HttpServletRequest	Запит, який потребує обслуговування. Область видимості - запит. Основні методи: *getAttribute*, *getParameter*, *getParameterNames*, *getParameterValues* Запит *request* забезпечує звернення до параметрів запиту через метод *getParameter*, тип запиту (GET, POST, HEAD, тощо) та вхідні HTTP заголовки (cookies, Referer тощо).
response (відповідь)	javax.servlet.HttpServletResponse	Відповідь на запит. Область видимості - сторінка. Тому що потік виводу (out)

		буферизується, можна змінювати коди стану HTTP та заголовки відповідей, навіть якщо це не допускається у звичайному сервлеті, але тільки в тому випадку, якщо ці данні виводу вже були відправлені клієнту.
out (вивід)	javax.servlet.jsp.JspWriter	Об'єкт, що записує у вихідний потік. Область видимості — сторінка. Основні методи: *clear, clearBuffer, flush, getBufferSize, getRemaining.* Розмір буферу можна змінювати і навіть вимикати буферизацію, змінювати значення атрибута *buffer* директиви *page*. Об'єкт *out* використовується практично виключно скриплетами. Вирази JSP автоматично розміщаються у потік виводу, що позбавляє від необхідності явного звернення до *out*.
pageContext (вміст сторінки)	javax.servlet.jsp.pageContext	Вміст JSP-сторінки. Область видимості - сторінка. *pageContext* підтримує доступ до корисних об'єктів та методів, які забезпечують явний доступ реалізації JSP до специфічних об'єктів. Основні методи: *getSession, getPage, findAttribute, getAttribute, getAttributeScope, getAttributeNamesInScope, getException.*
session (сеанс)	javax.servlet.HttpSession	Об'єкт типу *Session*, створюється для клієнта, який присилає запит. Область видимості — сторінка. Основні методи *getId, getValue, getValueNames, putValue.* Сесії створюються автоматично і об'єкт *session* існує навіть якщо немає посилань на вхідні сесії. Єдиним виключенням є випадок, коли розробник вимикає використання сесій через атрибут *session* директиви *page*. У цьому випадку посилання на об'єкт *session* веде до виникнення помилок при трансляції JSP сторінки у сервлет.
application (додаток)	javax.servlet.ServletContext	Контекст сервлету, що отримується із об'єкту конфігурації сервлету при виклику методів: *getServletConfig або getContext*. Область видимості — додаток. Основні методи: *getMimeType, getRealPath.*
config (конфігурація)	javax.servlet.ServletConfig	Об'єкт *ServletConfig* поточної сторінки JSP. Область видимості — сторінка. Основні методи: *getInitParameter, getInitParameterNames*
page (сторінка)	java.lang.Object	Екземпляр класу реалізації поточної сторінки JSP, що обробляє запит. Область видимості - сторінка. Об'єкт є доступним, але, як правило, використовується рідко. За суттю є синонімом для *this*, тому при роботі з Java не потрібен.
exception (виключення)	java.lang.Throwable	Об'єкт *Throwable*, який виводиться у сторінку помилок *error page*. Область видимості - сторінка. Основні методи: *printStackTrace, toString, getMessage, getLocalizedMessage*.

Для створюваної форми будемо використовувати об’єкти *request* та *out*. Спочатку зробимо перевірку щодо використання у формі методу POST, після чого отримаємо значення *username* та *password*. Якщо верифікація за логіном “admin” та паролем “admin” буде вірною, то буде виведено привітальне повідомлення:

```
<%
    String errMsg = null;
    if ("POST".equalsIgnoreCase(request.getMethod()) &&
        request.getParameter("submit") != null)
    {
        String userName = request.getParameter("userName");
        String password = request.getParameter("password");
        if ("admin".equalsIgnoreCase(userName) &&
                "admin".equalsIgnoreCase(password))
        {
                System.out.println("Welcome admin !");
        }
        else
        {
                errMsg = "Invalid user id or password. Please try again";
        }
    }
%>
```

У наведеному коді за допомогою оператору if та "POST".equalsIgnoreCase(request.getMethod()) робиться перевірка, щодо використання у формі методу POST, а request.getParameter("submit") != null перевіряє факт використання кнопки “submit” для відправлення форми.

Для отримання параметрів логіну використовуються request.getParameter(username) та request.getParameter(password). У наведеному прикладі використовуються постійні, наперед задані значення параметрів логіну, але на практиці для порівняння значення можуть братись з бази даних або окремих файлів.

Якщо верифікацію пройдено успішно, то за допомогою об’єкту out (JSPWriter) виводиться повідомлення, в іншому випадку буде задано повідомлення про помилку. Повідомлення про помилку краще виводити перед формою логіну:

```
<h2>Login:</h2>
<%if (errMsg != null) { %>
    <span style="color: red;"><%=;"><%=;"><%=errMsg %></span>
<%} %>
<form method="post">
...
</form>
```

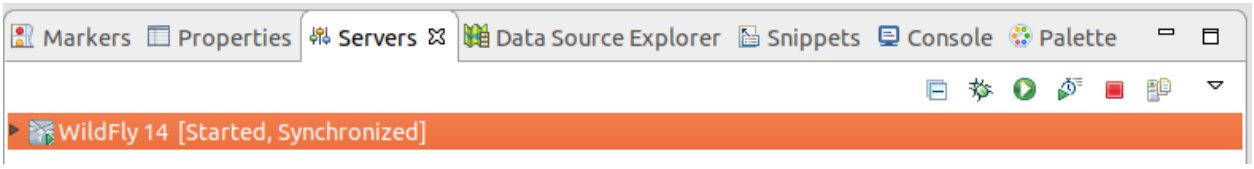
Тут наведено інший блок коду на Java з використанням <% %>. Якщо повідомлення про помилку набуло значення, тобто його значення не пусте, тоді воно буде показано з використанням тега span. Повідомлення про помилку позначено як <%=errMsg %> , що є скороченням від <%out.print(errMsg);%> .

Нижче наведено повний код:

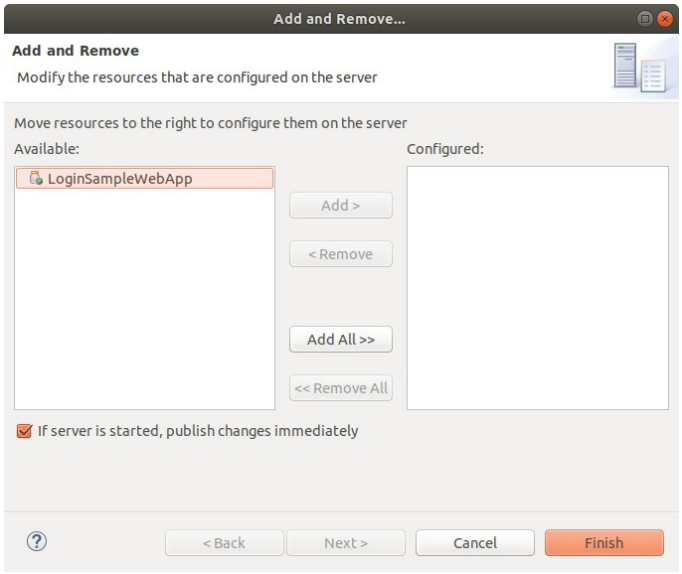
```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Login</title>
</head>
<%
      String errMsg = null;
      if ("POST".equalsIgnoreCase(request.getMethod()) &&
                        request.getParameter("submit") != null)
            { String userName = request.getParameter("userName");
              String password = request.getParameter("password");
      if ("user".equalsIgnoreCase(userName) &&
                  "user".equalsIgnoreCase(password))
      {
            out.println("Welcome admin !");
            return;
      }
      else
      {
            errMsg = "Invalid user id or password. Please try again";
      }
      }
%>

<body>
<h2>Login:</h2>
<%if (errMsg != null) { %>
      <span style="color: red;"><%out.print(errMsg); %></span>
<%} %>
<form method="post">
User Name: <input type="text" name="userName"><br>
Password: <input type="password" name="password"><br>
<button type="submit" name="submit">Submit</button>
<button type="reset">Reset</button>
</form>
</body>
</html>
```

4. Запуск JSP на WildFly

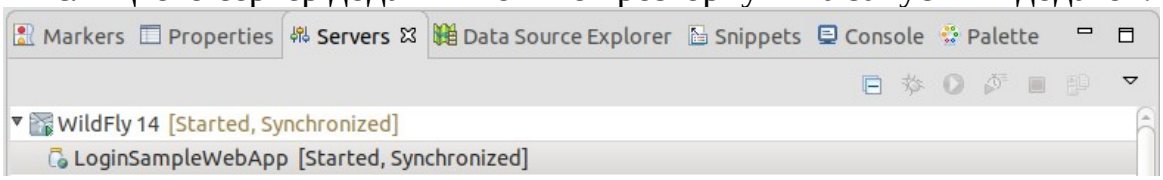
Для того, щоб запустити створену сторінку у web-браузері необхідно розгорнути додаток у контейнері сервлету, який реалізовано на сервері додатків, наприклад, WildFly. Для цього необхідно запустити WildFly:



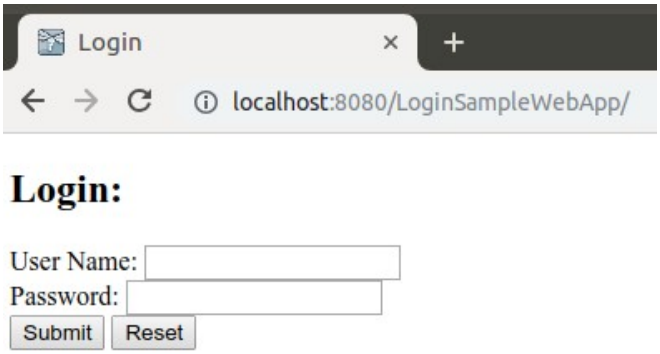
Для розгортання додатку необхідно клацнути правою кнопкою на WildFly у вікні Servers та обрати пункт Add and Remove. У діалоговому вікні, що з’явиться, обираємо проект для розгортання на сервері та натискаємо Finish:



Після цього сервер додатків повинен розгорнути та запустити додаток.



Для перевірки роботи додатку відкриємо його у браузері за відповідною адресою http://localhost:8080/LoginSampleWebApp/



Тепер необхідно ввести “admin” / “admin” в якості логіну та паролю, та натиснути кнопку Submit. У відповідь отримаємо “Welocome admin”, а якщо ввести хибні логін та пароль, то отримаємо повідомлення про помилку введення.

Сторінки JSP створюються динамічно для класів Java, тому для внесених змін у сторінку, для більшості випадків, не доведеться перезавантажувати сервер, а тільки оновити сторінку. Сервер додатків автоматично перекомпілює сторінку, якщо вона змінилася і буде показано змінену сторінку. Переконайтесь в цьому змінивши у коді дійсні логін та пароль, наприклад, на “user / user” або інший і перевірте, що внесені зміни було задіяно автоматично після збереження файлу.

5. Використання JavaBeans в JSP

Як було вказано вище, поєднання в одному файлі JSP Java-коду та тексту або html можливе але не найкраще рішення, тобто не відноситься до кращої практики. Більш того, дизайнер інтерфейсу користувача та програміст, що забезпечує його функціональність можуть бути різними людьми. Таким чином загальною рекомендацією є в JSP файл помістити теги форматування інтерфейсу, а Java-код — в окремі класи. Це також має сенс з точки зору повторного використання Java-коду. Обробка бізнес-логіки може бути перенесена з JSP до JavaBeans, що представляє собою простий Java-об’єкт з атрибутами та get- і set-функціями для них.

JSP має спеціальний тег для використання JavaBeans – jsp:useBean :

```
<jsp:useBean id="name_of_variable" class="name_of_bean_class" scope="scope_of_bean"/>
```

Параметр scope на час життя цього Bean. Його можливими значеннями є application, page, request або session, призначення яких наведено у таблиці нижче.

Значення scope	Описання
page	Bean може використовуватись тільки у поточній сторінці
request	Bean може використовуватись у будь-якій сторінці при обробці того ж самого запиту. Один JSP запит може оброблятись багатьма JSP, якщо одна сторінка переправляє цей запит до іншої сторінки.
session	Bean може використовуватись у тій самій HTTP сесії. Це стає корисним, якщо необхідно зберегти дані користувача під час взаємодії з додатком, наприклад, поточний вміст кошика для покупок в Інтернет-магазині.
application	Bean може використовуватись у будь-якій сторінці в тому ж самому web-додатку. Зазвичай web-додаток розгортається у контейнері web-додатків як war-файл (Web Application Archive). В межах додатку усі JSP у war-файлі можуть використовувати його JavaBeans.

Розглянемо переміщення коду для верифікації користувача в нашому прикладі до JavaBean-класу.

Першим необхідно створити JavaBean клас.

1) У Project Explorer, на папці src треба клацнути правою кнопкою миші та обрати меню New -> Package.

2) Створюємо пакет з ім’ям ua.edu.znu.lab.bean.

3) Клацаємо правою кнопкою на створеному пакеті та обираємо меню New -> Class.

4) Створюємо клас з ім’ям LoginBean.

5) Створюємо два приватних поля типу String:

```
public class LoginBean {
	private String userName;
	private String password;
}
```

6) Клацаємо правою кнопкою миші у будь-якому місці редактору та обираємо меню Source -> Generate Getters and Setter ... Для того, щоб згенерувати функції get- та set- для усіх членів класу, необхідно натиснути кнопку Select All, а також обрати точку вставки функцій (Insertion point) як Last member. Після додавання функцій клас буде виглядати наступним чином:

```
package ua.edu.znu.lab.bean;
public class LoginBean {
	private String userName;
	private String password;
	public String getUserName() {
		return userName;
	}
	public void setUserName(String userName) {
		this.userName = userName;
	}
	public String getPassword() {
		return password;
	}
	public void setPassword(String password) {
		this.password = password;
	}
}
```

7) Тепер додамо до класу ще один метод, який буде виконувати верифікацію username та password:

```
public boolean isValidUser()
{
 return "admin".equals(this.userName) &&
        "admin".equals(this.password);
}
```

8) Передаємо задачу верифікації до створеного bean. В index.jsp замінюємо Java-скриплет, що вище тега <body>, на наступний:

```
<%String errMsg = null; %>
<%if ("POST".equalsIgnoreCase(request.getMethod()) &&
     request.getParameter("submit") != null) {%>
     <jsp:useBean id="loginBean" class="ua.edu.znu.lab.bean.LoginBean">
     <jsp:setProperty name="loginBean" property="*"/>
     </jsp:useBean>
```

```
<%
if (loginBean.isValidUser())
{
   out.println("<h2>Welcome admin !</h2>");
   out.println("You are successfully logged in");
}
else
{
   errMsg = "Invalid user id or password. Please try again";
}
%>
<%} %>
```

В наведеному коді представлено декілька скриплетів — один для декларування змінної errMsg і два інших з блоками if:

- у перший блок if додано тег <jsp:useBean> , що дозволяє створити bean при виконанні умови — представлення форми через натискання кнопки submit;
- тег <jsp:setProperty> встановлює атрибути bean. Його ім’я задається як loginBean, а інші параметри специфікуються як property="*" , тобто значення вказуються неявно, оскільки члени LoginBean іменуються так само, як і поля у формі. Таким чином, під час виконання JSP отримує параметри із об'єкта запиту і призначає значення членам JavaBean з таким самим ім'ям. Якщо імена учасників JavaBean не відповідають параметрам запиту, то необхідно явно встановити значення, наприклад:

```
<jsp:setProperty name="loginBean" property="userName"
                  value="<%=request.getParameter(\"userName\")%>"/>
<jsp:setProperty name="loginBean" property="password"
                  value="<%=request.getParameter(\"password\")%>"/>
```

- останнім викликається метод `isValidUser()`, який у разі вдалої верифікації повертає значення “True” і тоді виводиться привітальне повідомлення, або “False” і тоді виводиться повідомлення про помилку.

Якщо сервер WildFly знаходиться у стані [Started, Synchronized], то збереження проекту автоматично приведе до його розгортання (публікації змін). Тепер перевірте, що проект працює так само, як було без Bean. Визначте, чи можна встановити бізнес-логіку коду Java-скриптів на стороні браузеру?

Зменшення кількості коду Java-скриптів в JSP робить проект більш керованим та зрозумілим. Тому досягнутий результат можна покращити за допомогою JSP Standard Tag Library (JSTL).

6. Використання JSTL

Теги JSTL можуть бути використані для заміни більшої частини Java-коду скриплетів. Теги JSTL класифікуються за п’ятьма групами:

- базові: забезпечують керування потоком та підтримку різних інших речей
- XML: забезпечують обробку XML документів
- i18n: забезпечують інтернаціоналізацію
- SQL: забезпечують доступ до баз даних
- функції: виконують деякі загальні операції з рядками

У своєму дистрибутиві сервер додатків WildFly вже має реалізацію відповідного API для підтримки JSTL. Для його використання добавимо наступну декларацію після першої декларації сторінки (<%@ page language="java" ...>):

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
```

Цей taglib вміщує URL бібліотеки тегів групи Core і його префікс “c”. Зазвичай для групи Core використовується префікс “c”, для XML — X та `uri http://java.sun.com/jsp/jstl/xml`, для i18n — fmt та `uri http://java.sun.com/jsp/jstl/fmt`, для SQL — sql та `uri http://java.sun.com/jsp/jstl/sql`.

Усі теги бібліотеки будуть доступними з використанням префіксу в JSP. Тепер необхідно замінити скриплети на відповідні теги JSTL.

1) Замінюємо `<%String errMsg=null; %>` на наступний набір тегів JSTL:

```
<c:set var="errMsg" value="${null}"/>
<c:set var="displayForm" value="${true}"/>
```

Нова змінна displayForm, яка ініціалізується значенням true, буде використовуватись далі.

2) Тепер необхідно замінити наступний код:

```
<%if ("POST".equalsIgnoreCase(request.getMethod()) &&
                        request.getParameter("submit") != null) {%>
```

на відповідний тег if з бібліотеки JSTL:

```
<c:if test="${\"POST\".equalsIgnoreCase(pageContext.request.method)
&& pageContext.request.getParameter(\"submit\") !=null}">
```

Відповідний об’єкт `request` в JSTL стає доступним через `pageContext`.

3) Теги JavaBean йдуть разом з тегом if і тому ніяких змін не потребують:

```
<jsp:useBean id="loginBean" class="ua.edu.znu.lab.bean.LoginBean">
<jsp:setProperty name="loginBean" property="*"/>
</jsp:useBean>
```

4) Тепер слід додати виклик `loginBean.isValidUser()` та встановлення значення повідомлення відповідно до результату, що повертається від цього методу. Тег if в JSTL не має конструкції типу if-else, тому можна використати тег choose і побудувати конструкцію, що буде працювати як перемикач:

```
<c:choose>
      <c:when test="${!loginBean.isValidUser()}">
         <c:set var="errMsg" value="Invalid user id or password. Please
            try again"/>
      </c:when>
<c:otherwise>
      <h2><c:out value="Welcome admin !"/></h2>
      <c:out value="You are successfully logged in"/>
      <c:set var="displayForm" value="${false}"/>
</c:otherwise>
</c:choose>
```

Якщо введені облікові дані користувача виявились недійсними, то буде встановлено повідомлення про помилку (тег `<c:when </c:when>`). В іншому випадку (тег c:otherwise) буде виведено привітальне повідомлення і значення displayForm отримає значення false, тому що немає сенсу показувати форму логіну, якщо верифікація даних користувача була вдалою.

5) Замінюємо код скриплету:

```
<%if (errMsg != null) { %>
      <span style="color: red;"><%out.print(errMsg); %></span>
<%} %>
```

на відповідну конструкцію тегу if з JSTL:

```
<c:if test="${errMsg != null}">
      <span style="color: red;">
            <c:out value="${errMsg}"></c:out>
      </span>
</c:if>
```

Якщо повідомлення про помилку має не нульове значення, то змінюється відповідний стиль та використовується тег out для його виведення.

6) Останніми змінами буде обгорнення контенту <body> </body> тегом if з JSTL:

```
<c:if test="${displayForm}">
      <body>
            ...
      </body>
</c:if>
```

Таким чином, повний текст нового коду буде виглядати як

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Login</title>
</head>
<c:set var="errMsg" value="${null}"/>
<c:set var="displayForm" value="${true}"/>
<c:if test="${\"POST\".equalsIgnoreCase(pageContext.request.method)
           && pageContext.request.getParameter(\"submit\") != null}">
    <jsp:useBean id="loginBean" class="ua.edu.znu.lab.bean.LoginBean">
    <jsp:setProperty name="loginBean" property="*"/>
    </jsp:useBean>
  <c:choose>
    <c:when test="${!loginBean.isValidUser()}">
       <c:set var="errMsg" value="Invalid user id or password.
            Please try again"/>
    </c:when>
    <c:otherwise>
       <h2><c:out value="Welcome admin !"/></h2>
       <c:out value="You are successfully logged in"/>
       <c:set var="displayForm" value="${false}"/>
    </c:otherwise>
  </c:choose>
</c:if>
<c:if test="${displayForm}">
  <body>
    <h2>Login:</h2>
    <c:if test="${errMsg != null}">
        <span style="color: red;">
           <c:out value="${errMsg}"></c:out>
        </span>
    </c:if>
    <form method="post">
       User Name: <input type="text" name="userName"><br>
       Password: <input type="password" name="password"><br>
       <button type="submit" name="submit">Submit</button>
       <button type="reset">Reset</button>
    </form>
  </body>
</c:if>
</html>
```

Як видно, у наведеному коді відсутні Java скриплети, усі вони були замінені на відповідні теги.

У реальних додатках, після вдалого проходження верифікації користувача додаток, у замін привітального повідомлення, перемкнеться на іншу функціональну сторінку. Для досягнення цього можна використати тег <jsp:forward>.

Завдання

1. Виконайте усі пункти практичної частини лабораторної роботи. Зробіть скриншоти виконаних завдань.

2. Розширте можливості класу *LoginBean* таким чином, щоб значення логинів та паролів брались з відповідного текстового файлу.

3. Використайте тег <jsp:forward> для перемикання на іншу сторінку після вдалої верифікації користувача.

4. Внесіть зміни у код сторінки так, щоб кількість невдалих спроб пройти верифікацію обмежувалось трьома, а спроба оновити сторінку з формою мало затримку 1 хвилину.

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