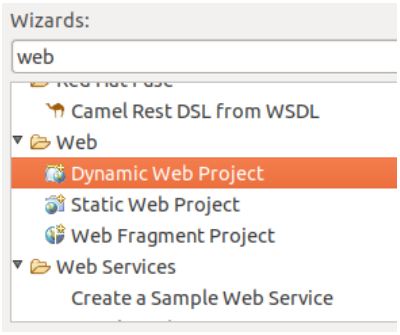
ЛАБОРАТОРНА РОБОТА

Тема: Моніторинг корпоративної мережі
Мета: Створення web-проекту для моніторингу визначених ресурсів

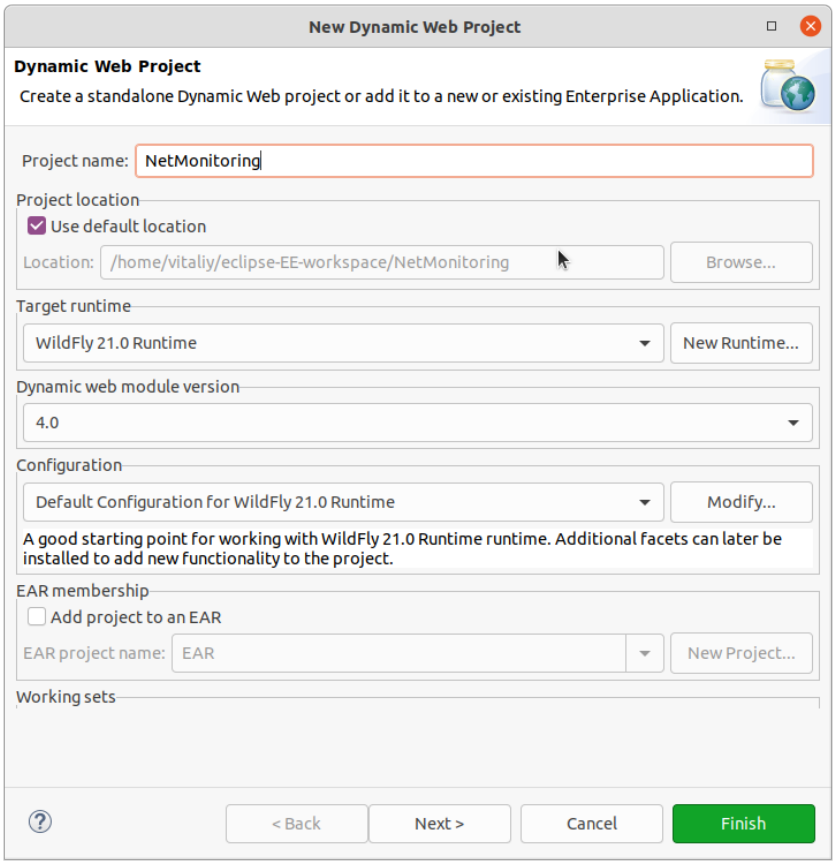
ПРАКТИЧНА ЧАСТИНА

Для розробки проекту припустимо застосування корпоративних технології Java.

1. Створюємо динамічний web-проекту в IDE Eclipse. Для чого 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, NetMonitoring. 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. Новий файл — index.jsp. Замінимо автоматично створений код файлу на наступний:

```
      <%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Monitoring Net</title>
</head>
<%String errMsg = null; String testMsg = null;%>
<%if ("POST".equalsIgnoreCase(request.getMethod()) &&
     request.getParameter("submit") != null) {%>
           <jsp:useBean id="testNet" class="ua.edu.znu.netmonitor.TestNet">
           <jsp:setProperty name="testNet" property="*"/>
           </jsp:useBean>
      <%
if("URL".equalsIgnoreCase(testNet.getTest())) testMsg = testNet.doTestURL();
if("IP".equalsIgnoreCase(testNet.getTest())) testMsg = testNet.doTestIP();
            errMsg = testNet.getTest();
      %>
      <%} %>
<body>
<h2>Test</h2>
<%if (errMsg != null) { %>
            <span style="color: red;"><%out.print(errMsg); %></span><br>
      <%} %>
<%if (testMsg != null) { %>
            <span style="color: blue;"><%out.print(testMsg); %></span><br>
      <%} %>
      <form method="post">
            Type of test:<br>
            <div>
               <input type="radio" id="testURL" name="test" value="URL"
checked>URL
            </div>
            <div>
              <input type="radio" id="testIP" name="test" value="IP">IP
            </div>
            <button type="submit" name="submit">Submit</button>
            <button type="reset">Reset</button>
      </form>
</body>
      </html>
```

Файл забезпечує простий інтерфейс для створення запиту на тестування ресурсів мережі та відображення результатів.

3. Додамо до проекту клас JavaBeans, що буде виконувати бізнес-логіку проекту:

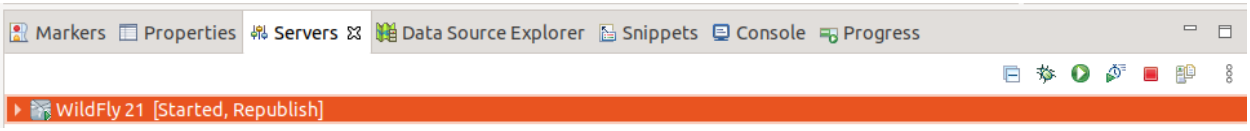
```
package ua.edu.znu.netmonitor;

import java.net.HttpURLConnection;
import java.net.SocketTimeoutException;
import java.net.URL;
import java.net.UnknownHostException;

public class TestNet {
	private boolean testURL;
	private boolean testIP;
	private String test;

	public String getTest() {
		return test;
	}

	public void setTest(String test) {
		this.test = test;
	}

	public boolean getTestURL() {
		return testURL;
	}

	public boolean getTestIP() {
		return testIP;
	}

	public void setTestURL(boolean testURL) {
		this.testURL = testURL;
	}

	public void setTestIP(boolean testIP) {
		this.testIP = testIP;
	}

	public String doTestURL() {
		String resultTest = "<br>";
	 for (int ipCount = 0; ipCount < ipAddresses.length; ipCount++)
	 {
	     final int index = ipCount;
	     resultTest += index + "<br>";
	     resultTest += ipAddresses[index][0]+"  "+ipAddresses[index][1]+"  "+"<br>";
	     resultTest += testURL(
	           ipAddresses[index][0],
	           ipAddresses[index][1],
	           ipAddresses[index][2]);
	 }
	 return resultTest;
	}

	public String doTestIP() {
		String resultTest = "<br>";
	 for (int ipCount = 0; ipCount < ipAddresses.length; ipCount++)
	 {
	     final int index = ipCount;
	     resultTest += index + "<br>";
	     resultTest += ipAddresses2[index][0]+"  "+ipAddresses2[index][1]+"  "+"<br>";
	     resultTest += testURL(
	           ipAddresses2[index][0],
	           ipAddresses2[index][1],
	           ipAddresses2[index][2]);
	 }
	 return resultTest;
	}

    private String testURL(String ipType, String url, String timeout)
    {
        String resultTest = "";
        String result = "";
        int processReturnValue = 0;
        boolean wasLastStatusError = false;
        HttpURLConnection connection = null;
```

```
    int code = -1;

    String ip = url.startsWith("http://") ? url : ("http://" + url);

        try
        {
            URL siteURL = new URL(ip);
            connection = (HttpURLConnection) siteURL.openConnection();
            connection.setRequestMethod("GET");
            connection.connect();
            code = connection.getResponseCode();

            if (code >= 400)
                processReturnValue = 1;

            result = String.valueOf(code) + ": "
                    + connection.getResponseMessage();
        }
        catch (UnknownHostException | SocketTimeoutException e)
        {
            result = e.toString();
            processReturnValue = 1;
        }
        catch (Exception e)
        {
            result = e.toString();
            processReturnValue = -1;
        }
        finally
        {
            if (connection != null)
            {
                connection.disconnect();
                connection = null;
            }

        }

        switch (processReturnValue)
        {
            case 0: // No error
                resultTest += "Network Connection Established to " + ipType
    + " Network (" + url + ")"+result+"<br>";
                break;
            case 1: // Connection error
                if (!wasLastStatusError)
                {
                   resultTest += "Network Connection Error to " + ipType
                             + " Network (" + url + ")" + result+"<br>";
                    wasLastStatusError = true;
                }
                break;
            case -1: // Configuration or programming error
                if (!wasLastStatusError)
                {
                   resultTest += "Configuration or Programming Error to "
                             + ipType + " Network (" + url + ")" + result+"<br>";
                    wasLastStatusError = true;
                }
                break;
        } // end case

    return resultTest;
       }

       private static String[][] ipAddresses =
{
        {
                "Open", "www.google.com", "3000"
        },
        {
                "VPN", "znu.edu.ua", "5000"
        }
};
private static String[][] ipAddresses2 =
{
        {
                "Open", "216.58.214.228", "3000"
```

```
            },
            {
                    "Open", "192.168.0.1", "5000"
            }
    };
}
```

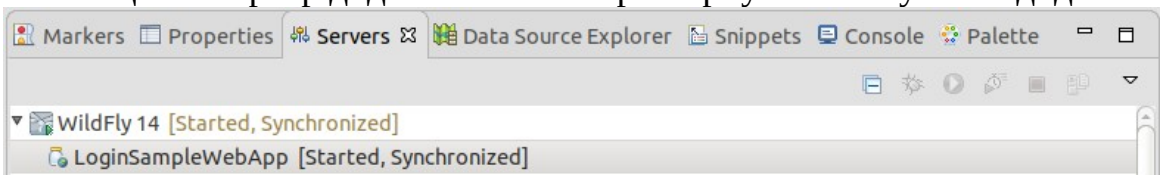
4. Запуск на WildFly

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:

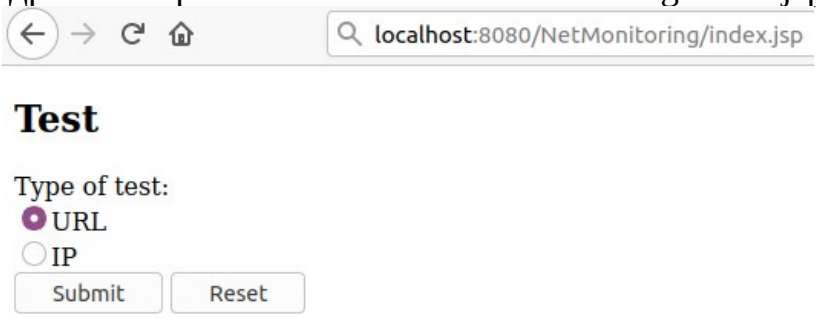


Для розгортання додатку необхідно клацнути правою кнопкою на WildFly у вікні Servers та обрати пункт Add and Remove. У діалоговому вікні, що з’явиться, обираємо проект для розгортання на сервері та натискаємо Finish.

Після цього сервер додатків повинен розгорнути та запустити додаток.



Для перевірки роботи додатку відкриємо його у браузері за відповідною адресою http://localhost:8080/NetMonitoring/index.jsp



Якщо обрати тип тестування ресурсів мережі та натиснути кнопку Submit, то у відповідь отримаємо, наприклад:

Завдання

1. Реалізуйте представлений проект та виконайте тестування мережі за типом “URL” та “IP”. Зробіть скриншоти виконаних завдань.

2. Поясніть принцип роботи проекту.

3. Визначте межу розширення *ipAddresses* та *ipAddresses2*. Чим вона обумовлена?

4. Внесіть зміни у код сторінки так, щоб оновлення відбувалось у автоматичному режимі з періодом 30 сек.

5. Підготуйте звіт, в який додайте скриншоти виконаних прикладів та код рішення поставлених задач.

Список рекомендованої літератури

1. Перри Б. Java сервлеты и JSP: сборник рецептов / Б.Перри - М.:КУДИЦ-ПРЕСС, 2006.-768 с.

2. Шпильман С. JSTL: Практическое руководство для JSP-программистов / С.Шпильман - М.: КУДИЦ-ОБРАЗ, 2004. - 272 с.

3. Шилдт Г. Java 8. Полное руководство / Г.Шилдт. - М.:ООО"И.Д. Вильямс", 2015. - 1376 с.