

PROGRAMOWANIE W JĘZYKU JAVA

Prowadzący: dr hab. inż. **Jan Prokop**, prof. PRz, e-mail: *jprokop@prz.edu.pl*,
Politechnika Rzeszowska, Wydział Elektrotechniki i Informatyki

LABORATORIUM 5

Temat: Java EE Platform (Servlets, JSP, JSF, EJB)

1. Java Web Applications - Serwlety

1.1. Przykład kompilacji, instalacji i uruchomienia serwletu

- Kod źródłowy serwletu `FirstServlet.java`

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class FirstServlet extends HttpServlet {
    protected void processRequest(HttpServletRequest request, HttpServletResponse
                                response) throws ServletException, IOException {
        response.setContentType("text/html; charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            out.println("<html>");
            out.println("<head>");
            out.println("<title>FirstServlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet FirstServlet at: " +
                        request.getContextPath() + "</h1>");

            out.println("</body>");
            out.println("</html>");
        } finally {
            out.close();
        }
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
                        throws ServletException, IOException {
        processRequest(request, response);
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
                        throws ServletException, IOException {
        processRequest(request, response);
    }
}
```

- Plik konfiguracyjny `web.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
version="2.5">
    <servlet>
        <servlet-name>FirstServlet</servlet-name>
        <servlet-class>FirstServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>FirstServlet</servlet-name>
        <url-pattern>/</url-pattern>
```

```
</servlet-mapping>
</web-app>
```

• Kompilacja

```
C:\>javac FirstServlet.java -classpath "C:\Program Files\Apache Software
Foundation\Tomcat 6.0\lib\servlet-api.jar"
```

• Struktura katalogów i plików	• Uruchomienie
+apache-tomcat-6.0 _+webapps _+_FirstServlet _+_WEB-INF _+_classes _+_FirstServlet.class _web.xml	http://localhost:8080/FirstServlet

1.2. Przykłady obsługi żądań protokołu HTTP

• Kod serwletu obsługi żądań klienta RequestInfo.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class RequestInfo extends HttpServlet {
    public void doGet(HttpServletRequest request,
        HttpServletResponse response) throws IOException, ServletException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println("<head>");
        out.println("<title>Request Information Example</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h3>Request Information Example</h3>");
        out.println("Method: " + request.getMethod() + "<br/>");
        out.println("Request URI: " + request.getRequestURI() + "<br/>");
        out.println("Protocol: " + request.getProtocol() + "<br/>");
        out.println("PathInfo: " + request.getPathInfo() + "<br/>");
        out.println("Remote Address: " + request.getRemoteAddr());
        out.println("</body>");
        out.println("</html>");
    }
    public void doPost(HttpServletRequest request,
        HttpServletResponse response) throws IOException, ServletException {
        doGet(request, response);
    }
}
```

• Kod licznika odwiedzin strony SimpleCounterServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class SimpleCounterServlet extends HttpServlet {
    int count;
    public void init() throws ServletException {
        count = 0;
        log("Method init, count = " + count);
    }
    public void destroy(){
```

```

        log("Method destroy, count = " + count);
    }
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        response.setContentType("text/html; charset=UTF-8");
        PrintWriter out = response.getWriter();
        StringBuffer sb = new StringBuffer();
        sb.append("<html>");
        sb.append("<head>");
        sb.append("<title>SimpleCounterServlet</title>");
        sb.append("</head>");
        sb.append("<body>");
        count++;
        sb.append("<p>Since loading, this servlet has been accessed<b> ");
        sb.append(count);
        sb.append(" </b>times</p>");
        sb.append("</body>");
        sb.append("</html>");
        out.println(new String(sb));
        out.close();
    }
}

```

1.3. Przykład przekazywania parametrów z formularza strony HTML do serwletu i generowania odpowiedzi w formacie HTML

- **Formularz** ShowParametersServlet.html

```

<html>
<head><title>ShowParameterstServlet</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
</head>
<body>
<h3>Formularz</h3>
<form method="GET" action="http://localhost:8080/ShowParametersServlet">
  <table>
    <tr><td>Nazwisko:</td><td><input type="text" size="20" name="name"></td></tr>
    <tr><td>E-mail:</td><td><input type="text" size="20" name="mail"></td></tr>
    <tr><td>Płeć:</td><td><input type="radio" name="sex" value="K"> Kobieta</td></tr>
    <tr><td></td><td><input type="radio" name="sex" value="M"> Mężczyzna</td></tr>
    <tr><td><input type="reset" value="Wyczyść"></td><td><input type="submit"
value="Wyślij"></td></tr>
  </table>
</form>
</body>
</html>

```

- **Plik serwletu** ShowParametersServlet.java

```

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ShowParametersServlet extends HttpServlet {
    String name, mail, sex;
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        name = request.getParameter("name");
        mail = request.getParameter("mail");
        sex = request.getParameter("sex");
        response.setContentType("text/html; charset=UTF-8");
        PrintWriter out = response.getWriter();
        out.println(createHTML());
        out.close();
    }
}

```

```

    }
    protected String createHTML() {
        StringBuffer sb = new StringBuffer();
        sb.append("<html>");
        sb.append("<head>");
        sb.append("<title>Title</title>");
        sb.append("</head>");
        sb.append("<body>");
        sb.append("<h3>Twój wybór</h3>");
        sb.append("<table border=\"1\" bgcolor=\"#ffff00\">");
        sb.append("<tr><td>Nazwisko: </td><td>" + name + "</td></tr>");
        sb.append("<tr><td>E-mail: </td><td>" + mail + "</td></tr>");
        sb.append("<tr><td>Płeć: </td><td>" + sex + "</td></tr>");
        sb.append("</table>");
        sb.append("</body>");
        sb.append("</html>");
        return (new String(sb));
    }
}

```

1.4. Przykład aplikacji Web o nazwie Quiz - HTML + serwlet

- Strona index.html

```

<!DOCTYPE html>
<html>
    <head>
        <title>Quiz</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <form method="get" action="/Quiz">
            <b>1. Która technologia platformy Java EE związana jest z logika
biznesową aplikacji ?</b><br/>
            <input type="radio" name="q1" value="Servlet">Servlet<br/>
            <input type="radio" name="q1" value="JSP">JSP<br/>
            <input type="radio" name="q1" value="JSF">JSF<br/>
            <input type="radio" name="q1" value="EJB">EJB<br/>
            <br/>
            <b>2. W jakim kontenerze są przetwarzane serwlety na platformie Java EE ?
</b><br/>
            <input type="radio" name="q2" value="EJB">EJB<br/>
            <input type="radio" name="q2" value="CDI">CDI<br/>
            <input type="radio" name="q2" value="WEB">WEB<br/>
            <input type="radio" name="q2" value="EEE">EEE<br/>
            <br/>
            <input type="submit" value="Sprawdź">
        </form>
    </body>
</html>

```

- Serwlet Quiz.java

```

package jp;
import java.io.IOException;
import java.io.PrintWriter;
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 = "Quiz", urlPatterns = {"/Quiz"})
public class Quiz extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        String q1 = request.getParameter("q1");
        String q2 = request.getParameter("q2");
        String q3 = request.getParameter("q3");
        String q4 = request.getParameter("q4");

        if (q1.equals("EJB")) {
            out.println("Pytanie 1 - " + "OK");
        } else {
            out.println("Pytanie 1 - " + "NO");
        }

        if (q2.equals("WEB")) {
            out.println("Pytanie 2 - " + "OK");
        } else {
            out.println("Pytanie 2 - " + "NO");
        }
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }
}

```

1.5. Aplikacja Web o nazwie Calculator - HTML + serwlet

- Strona index.html

```

<!DOCTYPE html>
<html>
    <head>
        <title>Servlet calculator</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <form method="get" action="http://Calculator" >
            a: <input name="a" type="text" />
            <select name="operator">
                <option value="addition"> + </option>
                <option value="subtraction"> - </option>
                <option value="multiplication"> * </option>
                <option value="division"> / </option>
            </select>
            b: <input name="b" type="text" />
            <input type="submit" value="=" />
        </form>
    </body>
</html>

```

- **Servlet Calculator.java**

```
package jp;
import java.io.IOException;
import java.io.PrintWriter;
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 = "Calculator", urlPatterns = {"/Calculator"})
public class Calculator extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            String a = request.getParameter("a");
            String b = request.getParameter("b");
            String operator = request.getParameter("operator");
            switch (operator) {
                case "addition":
                    out.println((Double.parseDouble(a) + Double.parseDouble(b)));
                    break;
                case "subtraction":
                    out.println(Double.parseDouble(a) - Double.parseDouble(b));
                    break;
                case "multiplication":
                    out.println(Double.parseDouble(a) * Double.parseDouble(b));
                    break;
                case "division":
                    out.println(Double.parseDouble(a) / Double.parseDouble(b));
                    break;
                default:
                    out.println(Double.parseDouble(a) + Double.parseDouble(b));
                    break;
            }
        }
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }
}
```

1.6. Przykład przekazywania parametrów do serwletu z aplikacji

- **Przekazywanie parametrów z aplikacji ShowParametersServletApplication.java**

```
import java.net.*;
import java.io.*;
public class ShowParametersServletApplication {
    public static void main(String[] args) {
        try {
```

```
URL url = new URL("http://localhost:8080/MyGetPostServlet?name=Jan
                  +Prokop&mail=jprokop@prz.edu.pl&sex=M");
BufferedReader in = new BufferedReader(
    new InputStreamReader(url.openStream()));
String line;
while ((line = in.readLine()) != null) {
    System.out.println(line);
}
in.close();
} catch (IOException e) {
    e.printStackTrace();
}
}
```

1.7. Inne przykłady serwletów

- **Generowanie danych XML z tablicy języka Java**

```
protected void doGet(HttpServletRequest request, HttpServletResponse
                    response) throws ServletException, IOException {
    String[][] data = {{ "1234", "Author 1", "Title 1"},
                      { "3456", "Author 2", "Title 2"} };
    response.setContentType("text/xml; charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        out.println("<?xml version=\"1.0\"?>");
        out.println("<bookstore>");
        out.println("<book isbn=\"" + data[0][0] + "\">");
        out.println("<author>" + data[0][1] + "</author>");
        out.println("<title>" + data[0][2] + "</title>");
        out.println("</book>");
        // ...
        out.println("</bookstore>");
    } finally {
        out.close();
    }
}
```

- **Generowanie grafiki**

```
import javax.servlet.annotation.WebServlet;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.awt.*;
import java.awt.image.*;
import com.sun.image.codec.jpeg.*;

public class ImageServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse
                        response) throws ServletException, IOException {
        BufferedImage image = new BufferedImage(640, 480,
                                                BufferedImage.TYPE_INT_RGB);

        Graphics g = image.getGraphics();
        g.setColor(Color.yellow);
        g.fillOval(100, 100, 250, 250);
        response.setContentType("image/jpeg");
        JPEGImageEncoder encoder =
            JPEGCodec.createJPEGEncoder(response.getOutputStream());
        encoder.encode(image);
    }
}
```

- **Przekierowanie do strony**

```
import java.io.*;
import java.sql.Date;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class PageRedirect extends HttpServlet {

    public void doGet(HttpServletRequest request,
                      HttpServletResponse response) throws
                      ServletException, IOException {

        response.setContentType("text/html");
        String webpage = new String("http://java.prz.edu.pl");
        response.setStatus(response.SC_MOVED_TEMPORARILY);
        response.setHeader("Location", webpage);

    }
}
```

- **Serwlet realizujący transformację XSLT - SimpleXSLTServlet.java**

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.xml.transform.*;
import javax.xml.transform.stream.*;
public class SimpleXSLTServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html; charset=UTF-8");
        PrintWriter out = response.getWriter();
        StreamSource source = new StreamSource("http://localhost:8080/
                                                MyGetPostServlet/sample.xml");
        StreamSource style = new StreamSource("http://localhost:8080/
                                                MyGetPostServlet/sample.xsl");
        StreamResult result = new StreamResult(out);
        try {
            TransformerFactory transFactory = TransformerFactory.newInstance();
            Transformer transformer = transFactory.newTransformer(style);
            transformer.transform(source, result);
        } catch (Exception e){
            resp.getWriter().print(e.getMessage());
        }
    }
}
```

sample.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="lista.xsl"?>
<lista>
    <pracownik foto="jk.jpg" www="http://www.onet.pl"> CV Pracownik1
        <nazwisko>Jan Kowalski</nazwisko>
        <uczelnia>Politechnika Rzeszowska</uczelnia>
        <wydzial>Wydział Elektrotechniki i Informatyki</wydzial>
        <adres>ul. W. Pola 2, 35-959 Rzeszów</adres>
        <telefon>(0-prefix-17) 8651384</telefon>
        <mail>jk@prz.edu.pl</mail>
        <wzrost>175</wzrost>
    </pracownik>
```



```

    <pracownik foto="jn.jpg" www="http://www.wp.pl"> CV Pracownik2
      <nazwisko>Jan Nowak</nazwisko>
      <uczelnia>Politechnika Rzeszowska</uczelnia>
      <wydzial>Wydział Elektrotechniki i Informatyki</wydzial>
      <adres>ul. W. Pola 2, 35-959 Rzeszów</adres>
      <telefon>(0-prefix-17) 8651384</telefon>
      <mail>jn@prz.edu.pl</mail>
      <wzrost>175</wzrost>
    </pracownik>
  </lista>

```

sample.xsl

```

<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
    <h2>Wzrost większy niż 180 cm</h2>
    <table border="1">
      <tr bgcolor="#00ff00">
        <th>Imię i Nazwisko</th>
        <th>Wzrost</th>
      </tr>
      <xsl:for-each select="lista/pracownik">
        <tr>
          <td><xsl:value-of select="nazwisko"/></td>
          <xsl:choose>
            <xsl:when test="wzrost>'180'">
              <td bgcolor="#ff00ff">
                <xsl:value-of select="wzrost"/></td>
            </xsl:when>
            <xsl:otherwise>
              <td><xsl:value-of select="wzrost"/></td>
            </xsl:otherwise>
          </xsl:choose>
        </tr>
      </xsl:for-each>
    </table>
  </body>
</html>
</xsl:template>
</xsl:stylesheet>

```

- **Serwlet logowania** - LoginForm.java

```

import java.io.*;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;

@WebServlet(name = "LoginForm", urlPatterns = {"/LoginForm"})
public class LoginForm extends HttpServlet {

    public static String USER_KEY = "ServletLogin.user";
    public static String FIELD_USER = "username";
    public static String FIELD_PASSWORD = "password";

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        java.io.PrintWriter out = response.getWriter();
        response.setHeader("Expires", "01 Jan 2015 00:00:00 GMT");
    }
}

```

```

        String uri = request.getRequestURI();
        HttpSession session = request.getSession();
        String user = (String) session.getAttribute(USER_KEY);
        if (user == null) {
            login(out, uri);
            return;
        }
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Welcome</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<center><h2>Login OK !</h2>");
        out.println("</center><br><br>");
        out.println("</body>");
        out.println("</html>");
        out.flush();
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        java.io.PrintWriter out = response.getWriter();
        HttpSession session = request.getSession(true);
        String user = (String) session.getAttribute(USER_KEY);
        if (user == null) {
            String username = request.getParameter(FIELD_USER);
            String password = request.getParameter(FIELD_PASSWORD);
            if (!validUser(username, password)) {
                out.println("<html>");
                out.println("<title>Invalid User</title>");
                out.println("<body><center><h2>" + "Invalid User!</h2><br>");
                out.println("Press the \"Back\" button to try again");
                out.println("</center></body></html>");
                out.flush();
                return;
            }
            session.setAttribute(USER_KEY, username);
        }
        response.sendRedirect(request.getRequestURI());
    }

    protected void login(java.io.PrintWriter out, String uri) throws
java.io.IOException {
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Login</title>");
        out.println("<center><h2>Welcome! Please login</h2>");
        out.println("<br><form method=POST action=\"" + uri + "\">");
        out.println("<table>");
        out.println("<tr><td>User ID:</td>");
        out.println("<td><input type=text name=\"" + FIELD_USER + " "
size=30></td></tr>");
        out.println("<tr><td>Password:</td>");
        out.println("<td><input type=password name=\"" + FIELD_PASSWORD + " "
size=10></td></tr>");
        out.println("</table><br>");
        out.println("<input type=submit value=\"Login\">");
        out.println("</form></center></body></html>");
    }

    protected boolean validUser(String username, String password) {
        boolean valid = false;
    }

```

```
        if ((username != null) && (username.length() > 0)) {  
            valid = username.equals("admin") && password.equals("admin");  
        }  
        return valid;  
    }  
}
```

Zadania

1. Napisać serwlet, który dane z formularza strony HTML zapisuje w pliku XML i zwraca aktualną zawartość tego pliku w postaci tabeli HTML
2. Napisać aplikację JavaFX, która dane wprowadzone z pól tekstowych interfejsu graficznego wysyła do serwletu, a serwlet je odsyła do aplikacji w formacie kodu HTML interpretowanego w oknie aplikacji.

2. Java Web Applications - technologia JSP

2.1. Przykład strony JSP

- Plik SimpleJSPExample.jsp

```
<html>
<body>
<% java.util.Date date = new java.util.Date(); %>
<p>Time: <%= date %></p>
<%
    out.println("RemoteHost: " + request.getRemoteHost() + "<br/>");
    out.println("Method: " + request.getMethod() + "<br/>");
    out.println("RequestURI: " + request.getRequestURI() + "<br/>");
    out.println("Protocol: " + request.getProtocol() + "<br/>");
%>
</body>
</html>
```

2.2. Przykład zastosowania komponentu JavaBean w dokumencie JSP

- JSP akcje - kod formularza JSPuseBean.html

```
<html>
<body>
<form method="post" action="PrintUserData.jsp">
<h3>Wprowadź dane</h3>
<table border="0">
  <tr>
    <td>Name:</td>
    <td><input type="text" name="username" size="25"></td>
  </tr>
  <tr>
    <td>Email:</td>
    <td><input type="text" name="email" size="25"></td>
  </tr>
  <tr>
    <td>Age:</td>
    <td><input type="text" name="age" size="5"></td>
  </tr>
</table>
<p><input type="submit" value="Wyślij"></p>
</form>
</body>
</html>
```

- JSP akcje - kod ziarna SimpleBean.java

```
package jp;
import java.io.Serializable;
public class DataBean implements Serializable {
    String username, email;
    int age;
    public DataBean() {
    }
    public void setUsername(String value) {
        username = value;
    }
    public void setEmail(String value) {
        email = value;
    }
    public void setAge(int value) {
        age = value;
    }
}
```

```
}  
public String getUsername() {  
    return username;  
}  
public String getEmail() {  
    return email;  
}  
public int getAge() {  
    return age;  
}  
}
```

- **JSP akcje - kod odpowiedzi** PrintUserData.jsp

```
<jsp:useBean id="user" class="jp.DataBean" scope="session"/>  
<jsp:setProperty name="user" property="*" />  
<html>  
<body>  
<h3>Twoje dane</h3>  
<table border="1">  
    <tr bgcolor="red">  
        <th>Parametr</th><th>Wartość</th>  
    </tr>  
    <tr>  
        <td bgcolor="blue">Name: </td>  
        <td><%=user.getUsername()%></td>  
    </tr>  
    <tr>  
        <td bgcolor="blue">Email:</td>  
        <td><%=user.getEmail()%></td>  
    </tr>  
    <tr>  
        <td bgcolor="blue">Age:</td>  
        <td><%=user.getAge()%></td>  
    </tr>  
</table>  
</body>  
</html>
```

3. Java Web Applications - Technologia JSF

3.1. Przykład strony JSF

- JSF - plik index.xhtml

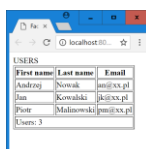
```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">
  <h:head>
    <title>
      Facelet Title
    </title>
  </h:head>
  <h:body>
    <h2>Ajax Example</h2>
    <h:form>
      <h:inputText id="inputName" value="#{userData.name}"> </h:inputText>
      <h:commandButton value="Show Message">
        <f:ajax execute="inputName" render="outputMessage" />
      </h:commandButton>
      <h2><h:outputText id="outputMessage"
        value="#{userData.welcomeMessage !=null ?
          userData.welcomeMessage : ''}" />
    </h2>
    </h:form>
  </h:body>
</html>
```

- JSF - plik HelloWorld.java

```
package jp;

import java.io.Serializable;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
@ManagedBean(name = "userData", eager = true) // application-scoped
@SessionScoped
public class HelloWorld implements Serializable {
    private static final long serialVersionUID = 1L;
    private String name;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getWelcomeMessage() {
        return "Hello " + name;
    }
}
```

3.2. Przykład strony JSF – dane ArrayList do tabeli HTML



The screenshot shows a web browser window with a table titled 'USERS'. The table has three columns: 'First name', 'Last name', and 'Email'. It contains three rows of data: (Andrzej, Nowak, andr@xx.pl), (Jan, Kowalski, jk@xx.pl), and (Piotr, Malinowski, pm@xx.pl). Below the table, it says 'Rows: 3'.

First name	Last name	Email
Andrzej	Nowak	andr@xx.pl
Jan	Kowalski	jk@xx.pl
Piotr	Malinowski	pm@xx.pl

- JSF - plik index.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
```

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core">
  <h:head>
    <title>Facelet Title</title>
  </h:head>
  <h:body>
    <h:outputText value="USERS" />
    <h:dataTable value="#{userBean.userList}" var="user" border="1">
      <h:column>
        <f:facet name="header">
          <h:outputText value="First name" />
        </f:facet>
        <h:outputText value="#{user.firstName}" />
      </h:column>
      <h:column>
        <f:facet name="header">
          <h:outputText value="Last name" />
        </f:facet>
        <h:outputText value="#{user.lastName}" />
      </h:column>
      <h:column>
        <f:facet name="header">
          <h:outputText value="Email" />
        </f:facet>
        <h:outputText value="#{user.email}" />
      </h:column>
      <f:facet name="footer">
        <h:outputText value="Users: " />
        <h:outputText value="#{userBean.userCount}" />
      </f:facet>
    </h:dataTable>
  </h:body>
</html>

```

- JSF - plik UserBean.java

```

package jp;

import javax.inject.Named;
import javax.enterprise.context.Dependent;
import java.util.ArrayList;
import java.util.List;
import javax.annotation.PostConstruct;

@Named(value = "userBean")
@Dependent
public class UserBean {

    private List<User> userList;

    @PostConstruct
    private void init() {
        userList = new ArrayList<>();
        userList.add(new User("Andrzej", "Nowak", "an@xx.pl"));
        userList.add(new User("Jan", "Kowalski", "jk@xx.pl"));
        userList.add(new User("Piotr", "Malinowski", "pm@xx.pl"));
    }

    public List<User> getUserList() {
        return userList;
    }
}

```

```

    public int getUserCount() {
        return userList.size();
    }
}

```

- JSF - plik User.java

```

package jp;
public class User {

    private final String firstName;
    private final String lastName;
    private final String email;

    public User(String firstName, String lastName, String email) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
    }

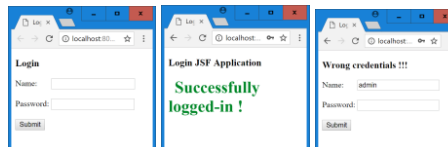
    public String getFirstName() {
        return firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public String getEmail() {
        return email;
    }
}

```

3.3. Przykład strony JSF – Logowanie



- JSF - plik index.xhtml

```

<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html">
    <h:head>
        <title>Login JSF Application</title>
    </h:head>
    <h:body>
        <h3>#{userLogin.message}</h3>
        <h:form id="loginForm">
            <h:outputLabel value="Name: &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; " />
            <h:inputText value="#{userLogin.username}" /> <br/><br/>
            <h:outputLabel value="Password: " />
            <h:inputSecret value="#{userLogin.password}"></h:inputSecret><br/><br/>
            <h:commandButton value="Submit"
                action="#{userLogin.login}"></h:commandButton>
        </h:form>
    </h:body>

```


</html>

- JSF - plik UserLogin.java

```
package jp;

import javax.inject.Named;
import javax.enterprise.context.RequestScoped;

@RequestScoped
@Named("userLogin")
public class UserLogin {

    public UserLogin() {
    }
    private String message = "Login";
    private String username;
    private String password;

    public String login() {
        if ("admin".equalsIgnoreCase(username) && "admin".equalsIgnoreCase(password))
        {
            message = "Successfully logged-in !";
            return "success";
        } else {
            message = "Wrong credentials !!!";
            return "index";
        }
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    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;
    }
}
```

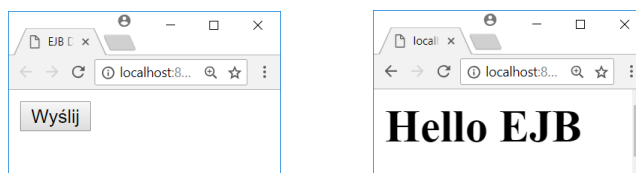
- JSF - plik successs.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:head>
    <title>Login JSF Application</title>
```

```
<style type="text/css">
    .successText {
        color: green;
        margin: 12px;
        font-weight: bold;
        font-size: 32px;
    }
</style>
</h:head>
<h:body>
    <h3>Login JSF Application</h3>
    <h:outputLabel class="successText" value="#{userLogin.message}" />
</h:body>
</html>
```

4. Java Enterprise Application - Technologia EJB

4.1. EJB z klientem w postaci serwletu



- Plik strony `index.html`

```
<!DOCTYPE html>
<html>
  <head>
    <title>EJB Demo</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  </head>
  <body>
    <form method="get" action="http://localhost:8080/EnterpriseApplication-
war/NewServlet">
      <input type="submit" value="Wyślij">
    </form>
  </body>
</html>
```

- Plik interfejsu `HelloInterface.java`

```
package jp;
import javax.ejb.Remote;
@Remote
public interface HelloInterface {
    String sayHello(String text);
}
```

- Plik bean-a EJB `HelloBean.java`

```
package jp;
import javax.ejb.Stateless;
import javax.ejb.LocalBean;
@Stateless
@LocalBean
public class HelloBean implements HelloInterface {
    @Override
    public String sayHello(String text) {
        return "Hello " + text;
    }
}
```

- Plik klienta, tj. serwletu `HelloServletClient.java`

```
package jp;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.ejb.EJB;
```

```

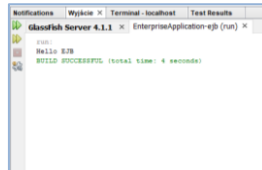
@WebServlet(name = "NewServlet", urlPatterns = {"/NewServlet"})
public class HelloServletClient extends HttpServlet {

    @EJB
    private HelloInterface helloBean;

    protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            out.println("<h1>" + helloBean.sayHello("EJB") + "</h1>");
            out.close();
        }
    }
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }
}

```

4.2. Klient aplikacyjny EJB do przykładu 4.1



- Plik klienta aplikacyjnego HelloClientApplication.java

```

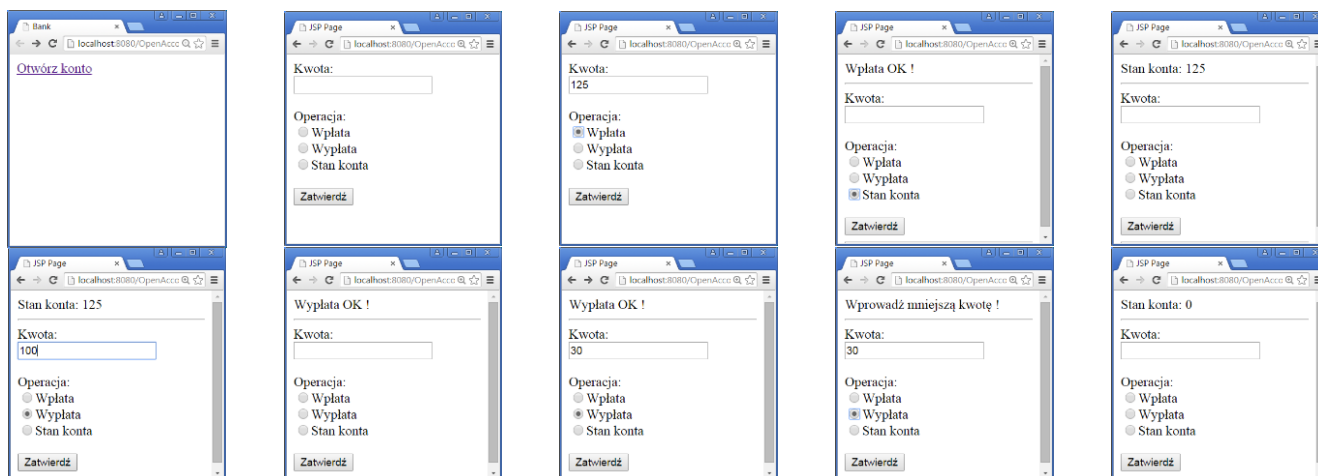
package jp;

import javax.naming.InitialContext;
import javax.naming.NamingException;
public class HelloClientApplication {
    public HelloClientApplication() {
    }
    public static void main(String[] args) {
        try {
            InitialContext ic = new InitialContext();
            HelloInterface hb = (HelloInterface)ic.lookup("jp.HelloInterface");
            System.out.println(hb.sayHello("EJB"));
        } catch (NamingException ex) {
        }
    }
}

```

5. Java Enterprise Application – aplikacja w architekturze MVC „BANK”

Uruchomić aplikację Java EE obsługi konta „banku” o wyglądzie i kodach źródłowych jak na rysunkach i listingu poniżej.



- **Kod źródłowy interfejsu BankRemote.java**

```
package jp;
import javax.ejb.Remote;
@Remote
public interface BankRemote {
    /**
     *
     * @param amount
     * @return
     */
    boolean credit(int amount);
    void deposit(int amount);
    int getBalance();
}
```

- **Kod źródłowy klasy BankAccountBean.java**

```
package jp;
import javax.ejb.Stateful;
@Stateful(mappedName = "myBank")
public class BankAccountBean implements BankRemote {
    private int amount = 0;
    @Override
    public boolean credit(int amount) {
        if (amount <= this.amount) {
            this.amount -= amount;
            return true;
        } else {
            return false;
        }
    }
    @Override
    public void deposit(int amount) {
        this.amount += amount;
    }
    @Override
    public int getBalance() {
        return amount;
    }
}
```

- **Kod źródłowy pliku index.jsp**

```
<!DOCTYPE html>
<html>
  <head>
    <title>Bank</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  </head>
  <body>
    <div>
      <a href="http://localhost:8080/OpenAccount">
        Otwórz konto
      </a>
    </div>
  </body>
</html>
```

- **Kod źródłowy pliku operation.jsp**

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <form action="operationprocess.jsp">
      Kwota: <br/>
      <input type="text" name="amount"/><br/><br/>
      Operacja: <br/>
      <input type="radio" name="operation" value="deposit"/>Wpłata<br/>
      <input type="radio" name="operation" value="withdraw"/>Wypłata<br/>
      <input type="radio" name="operation"
        value="checkbalance"/>Stan konta<br/>
      <br/>
      <input type="submit" value="Zatwierdź">
    </form>
  </body>
</html>
```

- **Kod źródłowy pliku operationprocess.jsp**

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <title>JSP Page</title>
  </head>
  <body>
    <%@ page import="jp.*" %>
    <%
      BankRemote remote =
        (BankRemote)session.getAttribute("remote");
      String operation = request.getParameter("operation");
      String amount = request.getParameter("amount");
      if (operation != null) {
        if (operation.equals("deposit")) {
          remote.deposit(Integer.parseInt(amount));
          out.print("Wpłata OK !");
        }
      }
    %>
```

```

        else if (operation.equals("withdraw")) {
            boolean status = remote.credit(Integer.parseInt(amount));
            if (status) {
                out.print("Wypłata OK !");
            }
            else {
                out.println("Wprowadź mniejszą kwotę !");
            }
        }
        else {
            out.println("Stan konta: " +
                        remote.getBalance());
        }
    }
    %>
<hr/>
<jsp:include page="operation.jsp"></jsp:include>
<hr/>
</body>
</html>

```

- **Kod źródłowy pliku OpenAccount.java**

```

package jp;
import java.io.IOException;
import javax.naming.InitialContext;
import javax.naming.NamingException;
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 = "OpenAccount", urlPatterns = {"/OpenAccount"})
public class OpenAccount extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        try {
            InitialContext context = new InitialContext();
            BankRemote b;
            b = (BankRemote) context.lookup("myBank");
            request.getSession().setAttribute("remote", b);
            request.getRequestDispatcher("/operation.jsp").
                forward(request, response);
        }
        catch (NamingException | ServletException | IOException e) {
            System.out.println(e);
        }
    }
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        doGet(request, response);
    }
}

```

Zadania

5.1. Rozbudować aplikację z punktu 5.2 o stronę logowania do “banku” jak na rysunku korzystając z kodu poniżej oraz możliwość wylogowania się.



- **Kod źródłowy pliku login.jsp**

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <title>Login</title>
  </head>
  <body>
    <h2>Login</h2>
    <form method="post" action="Receive.jsp">
      User Name <input type="text" name="user"><br/>
      Password <input type="password" name="pass"><br/>
      <input type="submit" value="Zaloguj">
    </form>
  </body>
</html>
```

- **Kod źródłowy pliku Receive.jsp**

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head><title>JSP Page</title></head>
  <body>
    <jsp:useBean id="myid" class="jp.ValidateBean" />
    <jsp:setProperty name="myid" property="user" />
    <jsp:setProperty name="myid" property="pass" />
    You name:
    <jsp:getProperty name="myid" property="user" /><br/>
    You password:
    <jsp:getProperty name="myid" property="pass" /><br/>
    <br/>You are a<%=myid.validate("jp","java")%> user !
  </body>
</html>
```

- **Kod źródłowy pliku ValidateBean.java**

```
package jp;
public class ValidateBean {
  String user; String pass;
  public ValidateBean( ) {
  }
  public void setUser(String user) {
    this.user = user;
  }
  public String getUser( ) {
    return user;
  }
  public void setPass(String pass) {
    this.pass = pass;
  }
  public String getPass( ) {
    return pass;
  }
  public String validate(String s1,String s2) {
    if(s1.equals(user) && s2.equals(pass))
      return "VALID";
    else
```



```
        return "INVALID";
    }
}
```

5.2. Rozbudować aplikację z punktu 5.2 wprowadzając transakcje korzystając z kodu poniżej.

• Kod źródłowy pliku AccountBean.java

```
import javax.annotation.Resource;
import javax.ejb.Stateless;
import javax.ejb.TransactionManagement;
import javax.ejb.TransactionManagementType;
import javax.transaction.UserTransaction;
@Stateless
@TransactionManagement(value=TransactionManagementType.BEAN)
public class AccountBean implements AccountBeanLocal {
    @Resource
    private UserTransaction userTransaction;
    public void transferFund(Account fromAccount, double fund , Account toAccount)
        throws Exception {
        try {
            userTransaction.begin();
            confirmAccountDetail(fromAccount);
            withdrawAmount(fromAccount, fund);
            confirmAccountDetail(toAccount);
            depositAmount(toAccount, fund);
            userTransaction.commit();
        }
        catch (InvalidAccountException exception){
            userTransaction.rollback();
        }
        catch (InsufficientFundException exception){
            userTransaction.rollback();
        }
        catch (PaymentException exception){
            userTransaction.rollback();
        }
    }
    private void confirmAccountDetail(Account account)
        throws InvalidAccountException {
    }
    private void withdrawAmount() throws InsufficientFundException {
    }
    private void depositAmount() throws PaymentException{
    }
}
```

5.3. Zmienić strony widoku na JSF

5.4. Opracować klienta „banku” w postaci aplikacji JavaFX.

Inne zadania podaje prowadzący w trakcie zajęć laboratoryjnych