

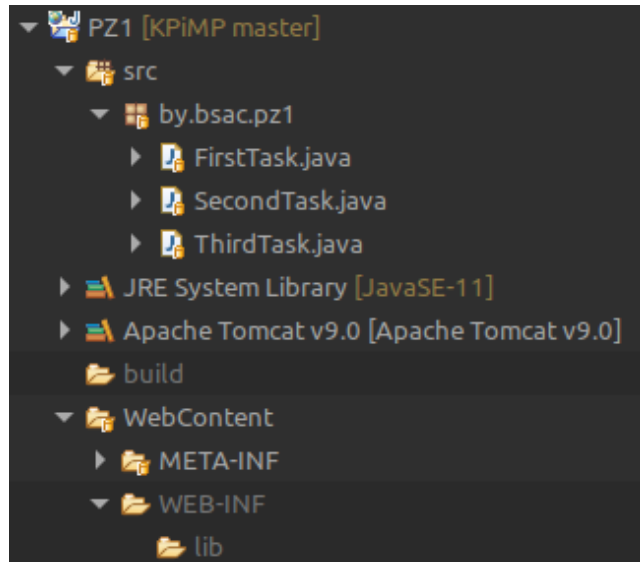
## ЗМЕСТ

1	Практычны занятак №1	2
1.1	Структура праекта	2
1.2	Заданне 1	2
1.3	Заданне 2	3
1.4	Заданне 3	4
2	Практычны занятак №2	7
2.1	Структура праекта	7
2.2	Заданне з тэорыі	7
2.3	Зыходны код. JSP-старонкі	7
2.4	Зыходны код. Java	9
2.5	Зыходны код. web.xml	15
2.6	Індывідуальнае заданне	16
3	Практычны занятак №3	18
3.1	Структура праекта	18
3.2	Заданне з тэорыі	18
3.3	Зыходны код. JSP-старонкі	19
3.4	Старонка allUsers.jsp.	19
3.5	Зыходны код. Java	20
3.6	Індывідуальнае заданне	30
4	Практычны занятак №4	31
4.1	Структура праекта	31
4.2	Заданне з тэорыі	31
4.3	Зыходны код. Java	32
4.4	Індывідуальнае заданне	34
5	Практычны занятак №5	37
5.1	Структура праекта	37
5.2	Заданне 1	37
5.3	Заданне 2	41
5.4	Заданне 3	43
6	Лабараторная работа №2	47
6.1	Структура праекта	47
6.2	Заданне	47
7	Лабараторная работа №4	53
7.1	Заданне	53

# 1 ПРАКТЫЧНЫ ЗАНЯТАК №1

## 1.1 Структура праекта

На малюнку 1.1 прадстаўлена файлавая структура праекта.



Малюнак 1.1 – Файлавая структура практычнага занятку

## 1.2 Заданне 1

### 1.2.1 Апісанне задання.

Напісаць сервлет, які выдае HTML-старонка з полем для ўводу з імем *P1*. Перад полем для ўвода мусіць быць тэкст *Поля для ўвода:*.

### 1.2.2 Зыходны код.

Зыходны код з тлумачэннямі прадстаўлены ў лістынгку 1.1.

```
1 package by.bsac.pz1;
2
3 import java.io.IOException;
4 import javax.servlet.ServletException;
5 import javax.servlet.annotation.WebServlet;
6 import javax.servlet.http.HttpServlet;
7 import javax.servlet.http.HttpServletRequest;
8 import javax.servlet.http.HttpServletResponse;
9
10
11 // WebServlet анатацыя (замест таго, каб анісваць servlet у web.xml файле)
12 // Спосаб з анісаннем web.xml будзе разглядацца на наступных практычных занятках
13 // паказвае на якім URL-адрасе будзе выконвацца клас FirstTask
14 @WebServlet("/FirstTask")
15 public class FirstTask extends HttpServlet {
16     private static final long serialVersionUID = 1L;
```

```

17
18 // Канструктар класа, які выклікае канструктар класа HttpServlet
19 public FirstTask() {
20     super();
21 }
22
23 // Метад doGet выконваецца, калі на старонку сервета адпраўляецца HTTP запыт GET
24 protected void doGet(HttpServletRequest request, HttpServletResponse response)
25     throws ServletException, IOException {
26     // Вяртаем HTML-старонку згодна з заданнем
27     // <label> - тэг, адказвае за надпіс перад тэгам <input>
28     // <input> - тэг, для ўводу даных, дзе
29     // type=text - апісвае, што поле мае тэкставы тып
30     // name=P1 - вызначае імя адпаведна заданню.
31     response.getWriter().append(
32         "<html>"
33         + "<meta charset=UTF-8>"
34         + "<label>Input box:"
35         + "<input type=text name=P1>"
36         + "</label>"
37         + "</html>");
38     }
39 }

```

Лістынг 1.1 – Зыходны код для першага задання

## 1.3 Заданне 2

### 1.3.1 Апісанне задання.

Напісаць сервлет, які выдае HTML-старонку з полем для ўводу з імем *P1* і кнопкай *Submit*. Пасля запаўнення карыстальнікам поля для ўводу і націскання кнопкі *Submit* сервлет мае выдаць такую ж HTML-старонку, у полі *P1* якога мае змяшчацца ўведзенае значэнне, паўторанае 2 разы.

### 1.3.2 Зыходны код.

Зыходны код з тлумачэннямі прадстаўлены ў лістынку 1.2.

```

1 package by.bsac.pz1;
2
3 import java.io.IOException;
4 import javax.servlet.ServletException;
5 import javax.servlet.annotation.WebServlet;
6 import javax.servlet.http.HttpServlet;
7 import javax.servlet.http.HttpServletRequest;
8 import javax.servlet.http.HttpServletResponse;
9
10 //@WebServlet анатацыя (замест таго, каб апісваць servlet у web.xml файле)
11 //Спосаб з апісаннем web.xml будзе разглядацца на наступных практычных занятках
12 //паказвае на якім URL-адрасе будзе выконвацца клас SecondTask
13 @WebServlet("/SecondTask")
14 public class SecondTask extends HttpServlet {
15     private static final long serialVersionUID = 1L;
16
17     // Канструктар класа, які выклікае канструктар класа HttpServlet

```

```

18     public SecondTask() {
19         super();
20     }
21
22     // Метод doGet виконується, коли на сторінку сервілета надіслан HTTP запит GET
23     protected void doGet(HttpServletRequest request, HttpServletResponse response)
24         throws ServletException, IOException {
25         // Вяртаем HTML-сторінку згідно з заданням
26         // <label> - тег, адказвае за надпіс перад тэгам <input>
27         // <input> - тег, для ўводу даных, дзе
28         // type=text (type=submit) - апісвае, што поле мае тэкставы тып (выгляд кнопкі)
29         // name=P1 - вызначае імя адпаведна заданню.
30         // <form> - тэг, якім задаем, што:
31         // method=POST - даныя, атрыманыя ўнутры тэга form, будуць перададзены, як
32         // параметры POST запиту
33         // action - на які URL-адрас, будзе адпраўлены POST запыт
34         response.getWriter().append(
35             "<html>"
36             + "<meta charset=UTF-8>"
37             + "<form action=SecondTask method=POST>"
38             + "<label>Input box:"
39             + "<input type=text name=P1 value=>"
40             + "</label>"
41             + "<input type=submit>"
42             + "</form>"
43             + "</html>");
44     }
45
46     // Метод doPost виконується, коли на сторінку сервілета надіслан HTTP запит POST
47     protected void doPost(HttpServletRequest request, HttpServletResponse response)
48         throws ServletException, IOException {
49         // Атрымліваем з запиту параметр P1 (глядзі метада doGet)
50         String textOfBox = request.getParameter("P1");
51         // Вяртаем HTML-сторінку, згідно заданню
52         // Старонка аналагічна першадрукаванай (глядзі метада doGet),
53         // за выключэннем таго, што для поля ўвода P1 задаецца значэнне (value),
54         // роўнае параметру P1, паўтарае 2 разы.
55         response.getWriter().append(
56             "<html>"
57             + "<meta charset=UTF-8>"
58             + "<form action=SecondTask method=POST>"
59             + "<label>Input box:"
60             + "<input type=text name=P1 value=" + textOfBox + textOfBox + ">"
61             + "</label>"
62             + "<input type=submit>"
63             + "</form>"
64             + "</html>");
65     }
66 }

```

## Лістынг 1.2 – Зыходны код для першага задання

### 1.4 Заданне 3

#### 1.4.1 Апісанне задання.

Напісаць сервілет, які выдае HTML-старонку з лікам *1* і кнопкай *Submit*. Пасля націскання кнопкі *Submit* сервілет павінен выдаць HTML-старонку з лікам *2* і кнопкай *Submit*. Пасля з

лікам 3 і гэтак далей.

#### 1.4.2 Выходны код.

Выходны код з тлумачэннямі прадстаўлены ў лістынку 1.3.

```
1 package by.bsac.pz1;
2
3 import java.io.IOException;
4 import javax.servlet.ServletException;
5 import javax.servlet.annotation.WebServlet;
6 import javax.servlet.http.HttpServlet;
7 import javax.servlet.http.HttpServletRequest;
8 import javax.servlet.http.HttpServletResponse;
9
10 //WebServlet анатацыя (замест таго, каб апісваць servlet у web.xml файле)
11 // Спосаб з апісаннем web.xml будзе разглядацца на наступных практычных занятках
12 // паказвае на якім URL-адрасе будзе выконвацца клас ThirdTask
13 @WebServlet("/ThirdTask")
14 public class ThirdTask extends HttpServlet {
15     private static final long serialVersionUID = 1L;
16
17     // Канструктар класа, які выклікае канструктар класа HttpServlet
18     public ThirdTask() {
19         super();
20     }
21
22     // Метад doGet выконваецца, калі на старонку сервлета адпраўляецца HTTP запыт GET
23     protected void doGet(HttpServletRequest request, HttpServletResponse response)
24         throws ServletException, IOException {
25         // HTML-старонка, якая вяртаецца на запыту GET
26         // <input type=hidden> - схаванае поле ўводу, якое захоўвае лік (пачатковае значэнне - 1)
27         response.getWriter().append(
28             "<html>"
29             + "<meta charset=UTF-8>"
30             + "<form action=ThirdTask method=POST>"
31             + "1"
32             + "<input type=hidden name=P3 value=1>"
33             + "<input type=submit>"
34             + "</form>"
35             + "</html>");
36     }
37
38     // Метад doPost выконваецца, калі на старонку сервлета адпраўляецца HTTP запыт POST
39     protected void doPost(HttpServletRequest request, HttpServletResponse response)
40         throws ServletException, IOException {
41         // Атрымліваем параметр P3 і павялічваем яго на 1
42         int number = Integer.parseInt(request.getParameter("P3")) + 1;
43         // HTML-старонка, якая вяртаецца на запыту POST
44         // Старонка аналагічная першапачатковай, за выключэннем таго, што
45         // пачатковае значэнне 1 было замененае на новае number як
46         // для тэксту, так і для <input type=hidden>
47         response.getWriter().append(
48             "<html>"
49             + "<meta charset=UTF-8>"
50             + "<form action=ThirdTask method=POST>"
51             + number
52             + "<input type=hidden name=P3 value=" + number + ">"
53             + "<input type=submit>"
54             + "</form>"
```

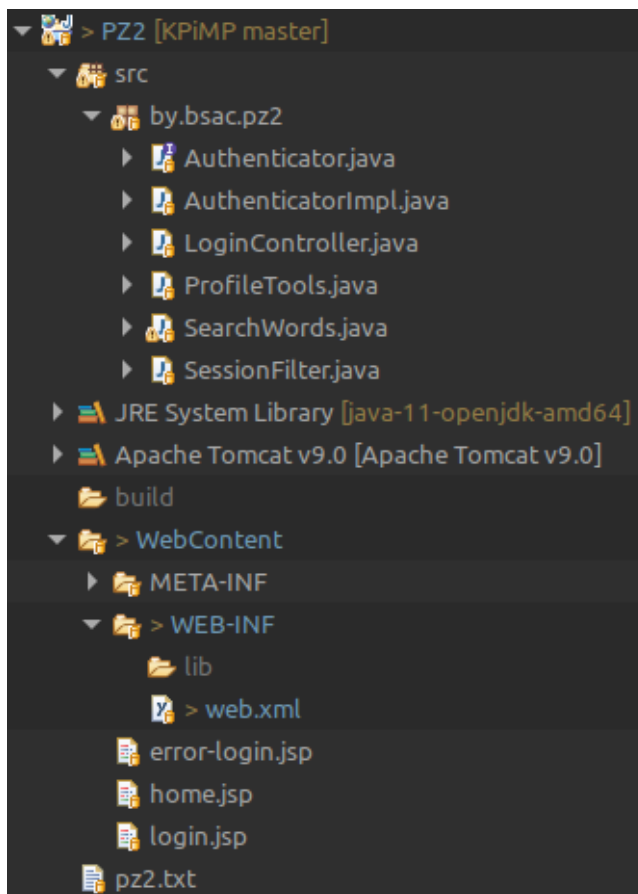
```
55         + "</html>");  
56     }  
57 }
```

Лістынг 1.3 – Выходны код для першага задання

## 2 ПРАКТЫЧНЫ ЗАНЯТАК №2

### 2.1 Структура праекта

На малюнку 2.1 прадстаўлена файлавая структура праекта.



Малюнак 2.1 – Файлавая структура практычнага занятку

### 2.2 Заданне з тэорыі

#### 2.2.1 Апісанне задання.

Стварыць вэб-праграму, якая перанакіроўвае любы URL-адрас на старонку *login*, дзе карыстальніку неабходна прайсці аўтэнтыфікацыю. Пры паспяховай аўтэнтыфікацыі вывесці старонку *home*, пры памылцы ў логіну альбо паролі — вывесці старонку *error-login*.

### 2.3 Зыходны код. JSP-старонкі

У лістынку 2.1 прадстаўлена старонка *error-login.jsp*.

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2     pageEncoding="UTF-8"%>
3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
```

```

4  "http://www.w3.org/TR/html4/loose.dtd">
5  <html>
6  <head>
7  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
8  <title>Login failed</title>
9  </head>
10 <body>
11     Login failed, please
12     <a href="/PZ2/login.jsp">try again</a>.
13 </body>
14 </html>

```

## Лістынг 2.1 – Выходны код для error-login

У лістынку 2.2 прадстаўлена старонка *home.jsp* з дадатковымі тлумачэннямі.

```

1  <%@ page language="java" contentType="text/html; charset=UTF-8"
2      pageEncoding="UTF-8"%>
3  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
4      "http://www.w3.org/TR/html4/loose.dtd">
5  <html>
6  <head>
7  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
8  <title>Home</title>
9  </head>
10 <body>
11     <h1>
12         // Выводзім значэнні, якія захоўваюцца на серверы для дадзенай сесіі
13         Hello,
14         <%=session.getAttribute("user")%>
15         <%=session.getAttribute("password") %>
16         <%=session.getAttribute("loginData") %>
17     </h1>
18
19     // POST форма для індывідуальнага задання
20     <form action="search" method="post">
21         <label>
22             Search
23             <input type="text" name="search" />
24             <input type="submit" value="Search" />
25         </label>
26     </form>
27
28     // POST форма для выхаду з вэб-праграмы
29     <form action="logout" method="post">
30         // Захоўвае значэнне logout для вызначэння функцыі ў метадазе doPost
31         // класа LoginController
32         <input type="hidden" name="authAction" value="logout">
33         <input type="submit" value="Logout" />
34     </form>
35 </body>
36 </html>

```

## Лістынг 2.2 – Выходны код для home

У лістынку 2.3 прадстаўлена старонка *login.jsp* з дадатковымі тлумачэннямі.



```

1  <%@ page language="java" contentType="text/html; charset=UTF-8"
2      pageEncoding="UTF-8"%>
3  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
4      "http://www.w3.org/TR/html4/loose.dtd">
5  <html>
6  <head>
7  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
8  <title>Login</title>
9  </head>
10 <body>
11     // POST форма для аутентифікації
12     <form action="login" method="post">
13         // Захоуває значення login, каб визначыць які метаd выклікаць у doPost класа LoginController
14         <input type="hidden" name="authAction" value="login">
15         <p>
16             Please login by
17             <label for="email">Email</label>
18             // Захоуває значення email, как у метаdze login визначыць метаd аутентифікації
19             // на параметру authType
20             <input type="radio" name="authType" value="email" id="email" checked />
21             or
22             <label for="userName">User name</label>
23             <input type="radio" name="authType" value="userName" id="userName" />
24             <input type="text" name="loginValue">
25         </p>
26         <label for="psw">Password</label>
27         <input type="password" name="psw" id="psw">
28         <p>The characters in a password field are masked (shown as
29             asterisks or circles).</p>
30         <input type="submit" value="Submit" />
31     </form>
32 </body>
33 </html>

```

Лістынг 2.3 – Зыходны код для login

## 2.4 Зыходны код. Java

### 2.4.1 Клас ProfileTools.

Клас *ProfileTools* з'яўляецца дапаможным класам. Ён захоувае імёны атрыбутаў сесіі, якія ўстанаўліваюцца падчас работы вэб-праграмы, а таксама метаd *isLoggedIn* для праверкі статусу карыстальніка.

У лістынгу 2.4 прадстаўлены зыходны код класа *ProfileTools*.

```

1  package by.bsac.pz2;
2
3  import javax.servlet.http.HttpServletRequest;
4  import javax.servlet.http.HttpSession;
5
6
7  public class ProfileTools {
8      public static String SESSION_LOGGEDIN_ATTRIBUTE_NAME = "user";
9      public static String SESSION_LOGGEDIN_ATTRIBUTE_PASSWORD = "password";
10     public static String SESSION_LOGGEDIN_ATTRIBUTE_DATA = "loginData";

```

```

11     public static String SESSION_LOGGEDOUT_ATTRIBUTE_DATA = "logoutData";
12
13     // Правляем ці прайшоў аўтэнтыфікацыю карыстальнік у бягучай сесіі
14     public static boolean isLoggedIn(HttpServletRequest request) {
15         // Атрымліваем бягучую сесію, калі яе няма, то новую не ствараем (false)
16         HttpSession session = request.getSession(false);
17
18         return session != null
19             && session.getAttribute(SESSION_LOGGEDIN_ATTRIBUTE_NAME) != null;
20     }
21 }

```

Лістынг 2.4 – Зыходны код класа ProfileTools

### 2.4.2 Інтэрфейс Authenticator.

Інтэрфейс *Authenticator* вызначае метады, якія маюць быць вызначаны ў класе *AuthenticatorImpl*.

У лістынг 2.5 прадстаўлены зыходны код інтэрфейса *Authenticator*.

```

1 package by.bsac.pz2;
2
3 public interface Authenticator {
4     public boolean authenticateByUserName(String username, String password);
5
6     public boolean authenticateByEmail(String email, String password);
7 }

```

Лістынг 2.5 – Зыходны код інтэрфейса Authenticator

### 2.4.3 Клас AuthenticatorImpl.

Клас *AuthenticatorImpl* захоўвае інфармацыю для ўваходу ў вэб-праграму, рэалізуе праверку ўведзеных даных з данымі доступу.

У лістынг 2.6 прадстаўлены зыходны код класа *AuthenticatorImpl*.

```

1 package by.bsac.pz2;
2
3 public class AuthenticatorImpl implements Authenticator {
4     // Інфармацыя для ўваходу ў вэб-праграму
5     private String username = "Antos";
6     private String password = "password";
7     private String email = "antos@bsac.by";
8
9     @Override
10    public boolean authenticateByUserName(String username, String password) {
11        if ( (getUsername().equalsIgnoreCase(username))
12            && (getPassword().equals(password)) ) {
13            return true;
14        }
15
16        return false;

```

```

17     }
18
19     @Override
20     public boolean authenticateByEmail(String email, String password) {
21         if (getEmail().equalsIgnoreCase(email)
22             && getPassword().equals(password)) {
23             return true;
24         }
25
26         return false;
27     }
28
29     public String getPassword() {
30         return password;
31     }
32
33     public String getUsername() {
34         return username;
35     }
36
37     public String getEmail() {
38         return email;
39     }
40
41     public void setEmail(String email) {
42         this.email = email;
43     }
44 }

```

Лістынг 2.6 – Зыходны код класа *AuthenticatorImpl*

#### 2.4.4 Клас *LoginController*.

Клас *LoginController* з’яўляецца сервлетам вэб-праграмы, які прапановуе старонку для аўтэнтыфікацыі і адказвае за выкананне аўтэнтыфікацыі: праверка ўведзеных даных з правільнымі, перанакіраванне на адпаведную старонку пры аўтэнтыфікацыі.

У лістынг 2.7 прадстаўлены зыходны код класа *LoginController*.

```

1  package by.bsac.pz2;
2
3  import java.io.IOException;
4
5  import java.time.format.DateTimeFormatter;
6  import java.time.LocalDateTime;
7
8  import javax.servlet.ServletException;
9  import javax.servlet.http.HttpServlet;
10 import javax.servlet.http.HttpServletRequest;
11 import javax.servlet.http.HttpServletResponse;
12 import javax.servlet.http.HttpSession;
13
14 public class LoginController extends HttpServlet {
15     private static final long serialVersionUID = 1L;
16
17     public LoginController() {
18         super();

```

```

19     }
20
21     // На запити GET вяртаєм карыстальніку старонку login.jsp
22     protected void doGet(HttpServletRequest request, HttpServletResponse response)
23         throws ServletException, IOException {
24         response.sendRedirect("login.jsp");
25     }
26
27     protected void doPost(HttpServletRequest request, HttpServletResponse response)
28         throws ServletException, IOException {
29         // Атрымліваем параметр authAction з POST запыту
30         String authAction = request.getParameter("authAction");
31
32         // У залежнасці ад параметра authAction вызначаем,
33         // якое дзеянне выконвае карыстальнік
34         if (authAction.contentEquals("login")) {
35             login(request, response);
36         } else if (authAction.equals("logout")) {
37             logout(request, response);
38         }
39     }
40
41     private void login(HttpServletRequest request, HttpServletResponse response)
42         throws ServletException, IOException {
43         // Атрымліваем тып аўтэнтыфікацыі з <input type=radio> элемента
44         String authTypeParam = request.getParameter("authType");
45
46         Authenticator authenticator = new AuthenticatorImpl();
47         boolean isAuthenticated = false;
48
49         // Атрымліваем параметры, перададзеныя ў POST запісе
50         String password = request.getParameter("psw");
51         String authValue = request.getParameter("loginValue");
52
53         if (authTypeParam.contentEquals("email")) {
54             isAuthenticated = authenticator.authenticateByUserEmail(authValue, password);
55         } else {
56             isAuthenticated = authenticator.authenticateByUserName(authValue, password);
57         }
58
59         // Калі аўтэнтыфікацыя прайшла паспяхова
60         if (isAuthenticated) {
61             // Атрымліваем бягучую дату ў фармаце String
62             DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");
63             LocalDateTime now = LocalDateTime.now();
64             String loginData = dtf.format(now);
65
66             // Атрымліваем бягучую сесію з карыстальнікам
67             HttpSession session = request.getSession();
68
69             // Запісваем ў атрыманую сесію атрыбуты
70             // SESSION_LOGGEDIN_ATTRIBUTE_NAME неабходны, каб вызначыць ці прайшоў
71             // карыстальнік аўтэнтыфікацыю
72             session.setAttribute(ProfileTools.SESSION_LOGGEDIN_ATTRIBUTE_NAME, authValue);
73             session.setAttribute(ProfileTools.SESSION_LOGGEDIN_ATTRIBUTE_PASSWORD, password);
74             session.setAttribute(ProfileTools.SESSION_LOGGEDIN_ATTRIBUTE_DATA, loginData);
75             response.sendRedirect("home.jsp");
76             // Калі аўтэнтыфікацыя няверная, вяртаєм старонку з памылкай
77         } else {
78             response.sendRedirect("error-login.jsp");
79         }

```

```

80     }
81
82     private void logout(HttpServletRequest request, HttpServletResponse response)
83         throws ServletException, IOException {
84         HttpSession session = request.getSession(false);
85
86         if (session != null) {
87             // Атримлюємо бягучу дату ў фармаце String
88             DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");
89             LocalDateTime now = LocalDateTime.now();
90             String logoutData = dtf.format(now);
91
92             // Записуємо час вихаду з веб-праграмы
93             session.setAttribute(ProfileTools.SESSION_LOGGEDOUT_ATTRIBUTE_DATA, logoutData);
94
95             // Выдаляем сувязь паміж аб'ектамі і дадзенай сесіяй
96             session.invalidate();
97         }
98
99         response.sendRedirect("login.jsp");
100     }
101 }

```

Лістынг 2.7 – Зыходны код класа LoginController

#### 2.4.5 Фільтр SessionFilter.

Клас *SessionFilter* з’яўляецца фільтрам, які аналізуе запыт карыстальніка (на які URL быў адпраўлены), і перанакіроўвае карыстальніка на старонку *login.jsp*, калі адрас не супадае з тымі, якія былі ўказаны ў *web.xml*.

У лістынг 2.8 прадстаўлены зыходны код класа *SessionFilter*.

```

1  package by.bsac.pz2;
2
3  import java.io.IOException;
4  import java.util.ArrayList;
5  import java.util.StringTokenizer;
6
7  import javax.servlet.Filter;
8  import javax.servlet.FilterChain;
9  import javax.servlet.FilterConfig;
10 import javax.servlet.ServletException;
11 import javax.servlet.ServletRequest;
12 import javax.servlet.ServletResponse;
13 import javax.servlet.http.HttpServletRequest;
14 import javax.servlet.http.HttpServletResponse;
15
16 import by.bsac.pz2.ProfileTools;
17
18 public class SessionFilter implements Filter {
19     private ArrayList<String> ignoredUrlList;
20
21     public SessionFilter() {
22     }
23
24     public void destroy() {

```

```

25     }
26
27     public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
28         throws IOException, ServletException {
29         HttpServletRequest req = (HttpServletRequest) request;
30         HttpServletResponse res = (HttpServletResponse) response;
31
32         // Атримлюємо бягучу URL
33         String requestUri = req.getRequestURI();
34         // Правляємо ці з'являючіся бягучу URL, які необхідно ігнорувати
35         boolean shouldBeIgnored = isIgnoredUrl(requestUri);
36
37         // Калі користувач не пройшов аутентифікацію і URL не ігнорується,
38         // відправляємо користувача на login.jsp
39         if (!shouldBeIgnored && !ProfileTools.isLoggedIn(req)) {
40             res.sendRedirect("login.jsp");
41         } else {
42             chain.doFilter(request, response);
43         }
44     }
45
46     // метод, який викликається перед тим, як буде застосований фільтр
47     public void init(FilterConfig fConfig) throws ServletException {
48         ignoredUrlList = new ArrayList<String>();
49
50         // Читаємо з web.xml параметр (init-param) з іменем ignore-urls
51         String urls = fConfig.getInitParameter("ignore-urls");
52
53         // Додаємо у ignoredUrlList кожну URL-адресу, атриману з web.xml
54         StringTokenizer token = new StringTokenizer(urls, ",");
55         while (token.hasMoreTokens()) {
56             ignoredUrlList.add(token.nextToken());
57         }
58     }
59
60
61     // правляємо ці з'являючіся url тими, які задані фільтром, який ігнорується
62     private boolean isIgnoredUrl(String url) {
63         for (String ignoredUrl : getIgnoredUrlList()) {
64             if (url.startsWith(ignoredUrl)) {
65                 return true;
66             }
67         }
68
69         return false;
70     }
71
72     public ArrayList<String> getIgnoredUrlList() {
73         return ignoredUrlList;
74     }
75
76     public void setIgnoredUrlList(ArrayList<String> urlList) {
77         this.ignoredUrlList = urlList;
78     }
79 }

```

Лістинг 2.8 – Вихідний код класу SessionFilter

## 2.5 Вихідний код. web.xml

Файл *web.xml* апісває, яким сервлетам (фільтрам) адапвлядаюць якія класы, да якіх URL-аў прывязваецца сервлет (фільтр).

У лістынгу 2.9 прадстаўля канфігурацыя файла *web.xml*.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xmlns="http://xmlns.jcp.org/xml/ns/javaee"
4     xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
5 http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
6     version="3.1">
7     <display-name>PZ2</display-name>
8     <welcome-file-list>
9         <welcome-file>index.html</welcome-file>
10        <welcome-file>index.htm</welcome-file>
11        <welcome-file>index.jsp</welcome-file>
12        <welcome-file>default.html</welcome-file>
13        <welcome-file>default.htm</welcome-file>
14        <welcome-file>default.jsp</welcome-file>
15    </welcome-file-list>
16    // Апісанне фільтра
17    <filter>
18        // Назва фільтра
19        <filter-name>SessionFilter</filter-name>
20        // Поўны шлях да класа, які апісвае фільтр
21        <filter-class>by.bsac.pz2.SessionFilter</filter-class>
22        // Пачатковыя параметры фільтра
23        <init-param>
24            <param-name>ignore-urls</param-name>
25            // Спіс URL-аў, якія будуць ігнаравацца фільтрам
26            // Глядзі рэалізацыю ў класе SessionFilter
27            <param-value>/PZ2/login,/PZ2/LoginController,/PZ2/error-login.jsp,
28                /PZ2/login.jsp,/PZ2/search</param-value>
29        </init-param>
30    </filter>
31    // Прывязваем вышэй апісаны фільтр да ўсіх URL-аў (шаблон /*)
32    <filter-mapping>
33        <filter-name>SessionFilter</filter-name>
34        <url-pattern>/*</url-pattern>
35    </filter-mapping>
36    // Апісанне сервлета
37    <servlet>
38        <description></description>
39        // Назва сервлета
40        <display-name>LoginController</display-name>
41        <servlet-name>LoginController</servlet-name>
42        // Поўны шлях да класа, які апісвае сервлет
43        <servlet-class>by.bsac.pz2.LoginController</servlet-class>
44    </servlet>
45    // Звязваем вышэй апісаны сервлет з URL-амі
46    <servlet-mapping>
47        <servlet-name>LoginController</servlet-name>
48        <url-pattern>/login</url-pattern>
49        <url-pattern>/logout</url-pattern>
50    </servlet-mapping>
51    // Аналагічна апісваем сервлет для індывідуальнага задання
52    <servlet>
```

```

53     <description></description>
54     <display-name>SearchWords</display-name>
55     <servlet-name>SearchWords</servlet-name>
56     <servlet-class>by.bsac.pz2.SearchWords</servlet-class>
57 </servlet>
58 <servlet-mapping>
59     <servlet-name>SearchWords</servlet-name>
60     <url-pattern>/search</url-pattern>
61 </servlet-mapping>
62 </web-app>

```

## Лістынг 2.9 – Канфігурацыя web.xml

## 2.6 Індывідуальнае заданне

### 2.6.1 Апісанне задання.

У файле захоўваецца тэкст. Для кожнага слова, якое ўводзіцца ў тэкставае поле HTML-старонкі (праз прабел), вывесці ў *cookie*, колькі разоў яно сустракаецца ў тэксце.

### 2.6.2 Зыходны код

У лістынг 2.10 прадстаўлены зыходны код класа *SearchWords*.

```

1  package by.bsac.pz2;
2
3  import java.io.File;
4  import java.io.IOException;
5  import java.util.ArrayList;
6  import java.util.Arrays;
7  import java.util.HashMap;
8  import java.util.List;
9  import java.util.Scanner;
10
11 import javax.servlet.ServletException;
12 import javax.servlet.http.Cookie;
13 import javax.servlet.http.HttpServlet;
14 import javax.servlet.http.HttpServletRequest;
15 import javax.servlet.http.HttpServletResponse;
16
17 public class SearchWords extends HttpServlet {
18     private static final long serialVersionUID = 1L;
19
20     public SearchWords() {
21         super();
22     }
23
24     protected void doPost(HttpServletRequest request, HttpServletResponse response)
25         throws ServletException, IOException {
26         response.getWriter().append("Served at: ").append(request.getContextPath());
27
28         // Атрымліваем спіс слоў з параметра POST запыту
29         String searchWords = request.getParameter("search");
30         // Падзяляем спіс слоў на асобныя словы і заносім іх у List
31         List<String> inputWords = new ArrayList<String>(Arrays.asList(searchWords.split(" ")));
32

```



```

33      // Чытаем словы з файла pz2.txt і запісваем у str
34      File file = new File("/home/dranser/Documents/BSAC/3 year //KPiMP/PZ2/pz2.txt");
35      Scanner sc = new Scanner(file);
36      String str = "";
37      while (sc.hasNextLine()) {
38          str += sc.nextLine() + " ";
39      }
40
41      // Падазяляем str на асобныя словы і запісваем у List
42      List<String> words = new ArrayList<String>(Arrays.asList(str.split(" ")));
43
44      // Перабіраем усе ўведзеныя словы
45      for (String inputWord : inputWords) {
46          // лічальнік паўтораў бягучага слова
47          int count = 0;
48
49          // Правяраем на супадзенне ўведзенага слова з кожным словам з файла
50          for (String word : words) {
51              // калі словы супадаюць, павялічваем лічальнік
52              if (word.equalsIgnoreCase(inputWord)) {
53                  count++;
54              }
55          }
56
57          // Запісваем у кукі браўзера словы і колькасць яго паўтарэнняў
58          Cookie cookie = new Cookie(inputWord, String.valueOf(count));
59          response.addCookie(cookie);
60      }
61
62      // вяртаем карыстальніка на старонку home
63      response.sendRedirect("home.jsp");
64  }
65 }

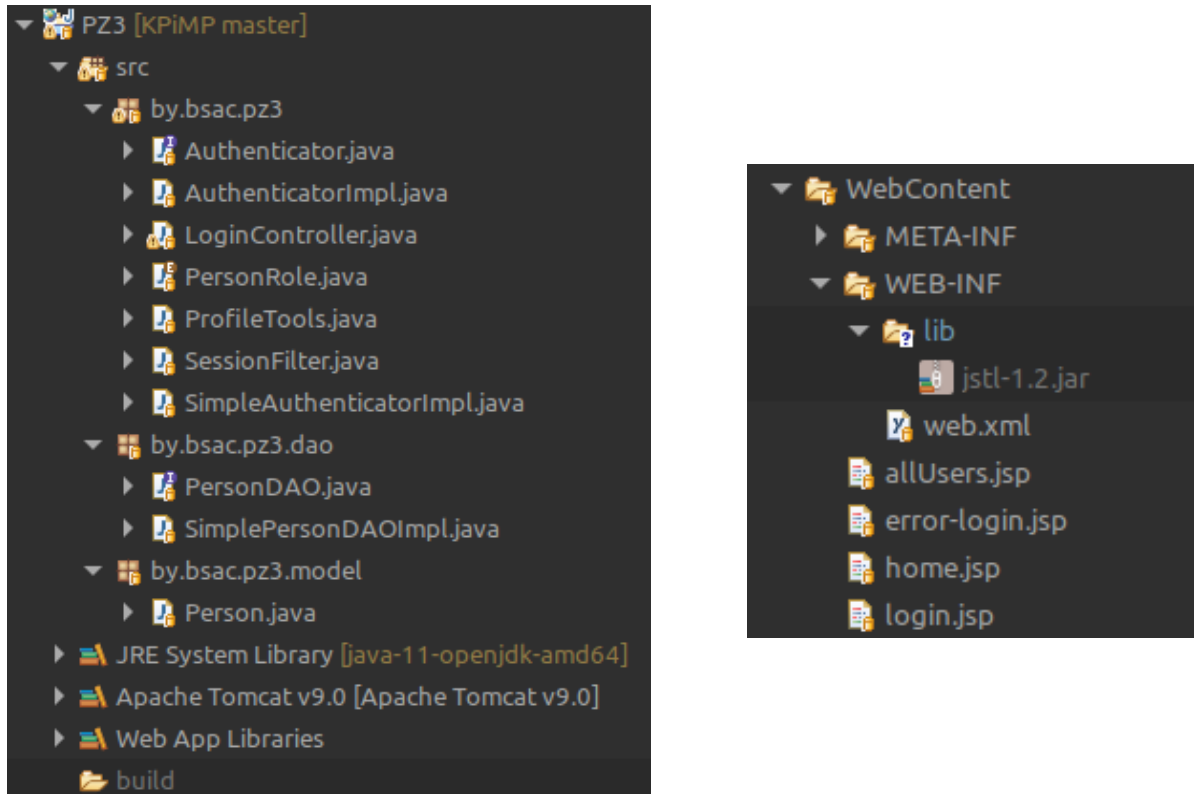
```

Лістынг 2.10 – Зыходны код класа SearchWords

## 3 ПРАКТЫЧНЫ ЗАНЯТАК №3

### 3.1 Структура праекта

На малюнку 3.1 прадстаўлена файлавая структура праекта.



Малюнак 3.1 – Файлавая структура практычнага занятку

Заўважце, што ў *WEB-INF/lib* з’явіўся новы файл *jstl-1.2.jar*. Гэты файл неабходны для магчымасці карыстацца JSTL тэгамі.

Перад выкананнем задання загрузіць дадзеную бібліятэку (*jstl-1.2.jar*) і пакладзіце ў дырэкторыю згодна з малюнкам 3.1.

### 3.2 Заданне з тэорыі

Дадзенае заданне выконваецца на базе практычнага занятку №1 (заданне з тэорыі).

#### 3.2.1 Апісанне задання.

Дабаўце ў вэб-праграму, распрацаваную на першым практычным занятку, клас, які захоўвае інфармацыю пра карыстальнікаў. Змяніце код вэб-праграмы для аўтарызацыі пры дапамозе новага класа. Дабаўце новую старонку, на якой будучы выводзіцца інфармацыя пра ўсіх магчымых карыстальнікаў.

### 3.3 Зыходны код. JSP-старонкі

#### 3.3.1 Абноўленая старонка home.jsp.

Дадавім ўмову *if* для стварэння спасылкі на старонку *allUsers.jsp*, калі бягучы карыстальнік — Адміністратар.

У лістынку 3.1 прадстаўлена абноўленая старонка *home.jsp*.

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
4 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
5   "http://www.w3.org/TR/html4/loose.dtd">
6 <html>
7 <head>
8 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
9 <title>Home</title>
10 </head>
11 <body>
12   <h1>
13     Hello ,
14     <%=session.getAttribute("user")%>
15   </h1>
16   // Праверка пры дапамозе jstl тэга
17   // ці з'яўляецца карыстальнік адміністратарам
18   // isAdmin - пераменная сесіі
19   <c:if test="${isAdmin}">
20     // Калі адміністратар, дадавіць спасылку на старонку
21     <p><a href="allUsers.jsp">View all users</a></p>
22   </c:if>
23
24   <form action="logout" method="post">
25     <input type="hidden" name="authAction" value="logout">
26     <input type="submit" value="Logout" />
27   </form>
28 </body>
29 </html>
```

Лістынг 3.1 – Зыходны код для home.jsp

#### 3.4 Старонка allUsers.jsp.

Дадзеная старонка выводзіць у выглядзе табліцы інфармацыю пра ўсіх магчымых карыстальнікаў (глядзі клас *SimplePersonDAOImpl*).

У лістынку 3.2 прадстаўлена старонка *allUsers.jsp*.

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 // Падключаем стандартныя jstl тэгі, для карыстання c:if i c:for
4 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
5 // Падключаем стандартны jstl тэг, для карыстання fmt:formatDate
6 <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
7
```

```

8  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
9  "http://www.w3.org/TR/html4/loose.dtd">
10 <html>
11 <head>
12 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13 <title>View All Users</title>
14 </head>
15 <body>
16     <h1>View All Users</h1>
17     <a href="home.jsp">Home Page</a>
18     <br />
19     <c:if test="${isAdmin}">
20         <h3>You are Admin</h3>
21         <p>
22             Count of Users
23             <c:out value="${users.size()}" />
24             // Заголоўкі табліцы
25             <table width="100%" border="1">
26                 <tr>
27                     <th>Name</th>
28                     <th>Email</th>
29                     <th>Role</th>
30                     <th>Last login date</th>
31                 </tr>
32                 // Перабіраем кожнага карыстальніка ў
33                 // пераменнай users (бярэцца з сесіі),
34                 // якая ўзяўша сабою спіс аб'ектаў Person
35                 <c:forEach items="${users}" var="user">
36                     <tr>
37                         <td>${user.name}</td>
38                         <td>${user.email}</td>
39                         <td>${user.role}</td>
40                         // Прывараем ці аўтэнтыфікаваўся бягучы карыстальнік
41                         // калі аўтэнтыфікаваўся, выводзім дату аўтэнтыфікацыі,
42                         // інакш - надпіс User did not login
43                         <td><c:if test="${user.loginDate==null}">User did not login</c:if>
44                         <c:if test="${user.loginDate!=null}"><fmt:formatDate pattern="dd-MM-yyyy
HH:mm:ss" value="${user.loginDate}" /></c:if></td>
45                     </tr>
46                 </c:forEach>
47             </table>
48         </p>
49     </c:if>
50 </body>
51 </html>

```

Лістынг 3.2 – Зыходны код для allUsers.jsp

## 3.5 Зыходны код. Java

### 3.5.1 Пералік PersonRole.

Для вызначэння ролі карыстальніка выкарыстоўваецца пералік PersonRole. У ім вызначаюцца дазволеныя назвы для ролі.

У лістынг 3.3 прадстаўлены зыходны код пераліку PersonRole.

```

1 package by.bsac.pz3;
2
3 public enum PersonRole {
4     ADMIN,
5     REGISTERED,
6     GUEST
7 }

```

Лістынг 3.3 – Зыходны код пераліку PersonRole

### 3.5.2 Клас Person

Клас *Person* захоўвае інфармацыю пра карыстальніка.

У лістынг 3.4 прадстаўлены зыходны код класа *Person*.

```

1 package by.bsac.pz3.model;
2
3 import java.util.Date;
4
5 import by.bsac.pz3.PersonRole;
6
7 public class Person {
8     private Long id;
9     private String email;
10    private String name;
11    private String password;
12    private Date loginDate;
13    private PersonRole role;
14
15    public Person(Long id, String email, String name,
16        String password, PersonRole role, Date loginDate) {
17        this.id = id;
18        this.email = email;
19        this.name = name;
20        this.password = password;
21        this.role = role;
22        this.loginDate = loginDate;
23    }
24
25    public Long getId() {
26        return id;
27    }
28
29    public void setId(Long id) {
30        this.id = id;
31    }
32
33    public String getEmail() {
34        return email;
35    }
36
37    public void setEmail(String email) {
38        this.email = email;
39    }
40
41    public String getName() {
42        return name;
43    }
44

```

```

45     public void setName(String name) {
46         this.name = name;
47     }
48
49     public String getPassword() {
50         return password;
51     }
52
53     public void setPassword(String password) {
54         this.password = password;
55     }
56
57     public Date getLoginDate() {
58         return loginDate;
59     }
60
61     public void setLoginDate(Date loginDate) {
62         this.loginDate = loginDate;
63     }
64
65     public PersonRole getRole() {
66         return role;
67     }
68
69     public void setRole(PersonRole role) {
70         this.role = role;
71     }
72
73     @Override
74     public int hashCode() {
75         final int prime = 31;
76         int result = 1;
77         result = prime * result + ((email == null) ? 0 : email.hashCode());
78         result = prime * result + ((id == null) ? 0 : id.hashCode());
79         result = prime * result + ((loginDate == null) ? 0 : loginDate.hashCode());
80         result = prime * result + ((name == null) ? 0 : name.hashCode());
81         result = prime * result + ((password == null) ? 0 : password.hashCode());
82         result = prime * result + ((role == null) ? 0 : role.hashCode());
83         return result;
84     }
85
86     @Override
87     public boolean equals(Object obj) {
88         if (this == obj)
89             return true;
90         if (obj == null)
91             return false;
92         if (getClass() != obj.getClass())
93             return false;
94         Person other = (Person) obj;
95         if (email == null) {
96             if (other.email != null)
97                 return false;
98         } else if (!email.equals(other.email))
99             return false;
100         if (id == null) {
101             if (other.id != null)
102                 return false;
103         } else if (!id.equals(other.id))
104             return false;
105         if (loginDate == null) {

```

```

106         if (other.loginDate != null)
107             return false;
108     } else if (!loginDate.equals(other.loginDate))
109         return false;
110     if (name == null) {
111         if (other.name != null)
112             return false;
113     } else if (!name.equals(other.name))
114         return false;
115     if (password == null) {
116         if (other.password != null)
117             return false;
118     } else if (!password.equals(other.password))
119         return false;
120     if (role != other.role)
121         return false;
122     return true;
123 }
124 }

```

Лістынг 3.4 – Зыходны код класа Person

### 3.5.3 Інтэрфейс PersonDAO.

Інтэрфейс *PersonDAO* вызначае метады, якія неабходна будзе рэалізаваць у класе *SimplePersonDAOImpl*.

У лістынг 3.5 прадстаўлены зыходны код інтэрфейса *PersonDAO*.

```

1  package by.bsac.pz3.dao;
2
3  import java.util.Set;
4
5  import by.bsac.pz3.model.Person;
6
7  public interface PersonDAO {
8      public Long add(Person person);
9      public Long save(Person person);
10     public boolean delete(Person person);
11     public Person findByName(String name);
12     public boolean update(Person person);
13     public Person findByEmail(String email);
14     public Set<Person> getAll();
15 }

```

Лістынг 3.5 – Зыходны код інтэрфейса PersonDAO

### 3.5.4 Клас SimplePersonDAOImpl.

Дадзены клас захоўвае інфармацыю пра карыстальнікаў, якія могуць аўтэнтыфікавацца ў вэб-праграме, і магчымасці пошуку карыстальнікаў па іх даным.

У лістынг 3.6 прадстаўлены зыходны код класа *SimplePersonDAOImpl*.

```

1  package by.bsac.pz3.dao;
2
3  import java.util.Set;

```

```

4 import java.util.HashSet;
5
6 import by.bsac.pz3.model.Person;
7 import by.bsac.pz3.PersonRole;
8
9 public class SimplePersonDAOImpl implements PersonDAO {
10     private Set<Person> persons = new HashSet<Person>();
11     private static SimplePersonDAOImpl simplePersonDAOImpl = new SimplePersonDAOImpl();
12
13     private SimplePersonDAOImpl() {
14         // Карыстальнікі, якія могуць аўтэнтыфікавацца ў вэб-праграме
15         persons.add(new Person(1L, "john@john.com", "john", "john123", PersonRole.ADMIN, null));
16         persons.add(new Person(2L, "peter@peter.com", "peter", "peter123", PersonRole.REGISTERED,
17             null));
18         persons.add(new Person(3L, "alex@alex.com", "alex", "alex123", PersonRole.REGISTERED,
19             null));
20     }
21
22     // Вяртае статычную пераменную, для магчымасці дабаўляць новых карыстальнікаў (рэгістрацыя)
23     public static SimplePersonDAOImpl getInstance() {
24         return simplePersonDAOImpl;
25     }
26
27     @Override
28     // Дабаўленне новага карыстальніка
29     public Long add(Person person) {
30         Long personId = -1L;
31
32         try {
33             persons.add(person);
34             personId = person.getId();
35         } catch (Exception e) {
36             e.printStackTrace();
37         }
38
39         return personId;
40     }
41
42     @Override
43     // Захаванне карыстальніка
44     public Long save(Person person) {
45         if (persons.contains(person)) {
46             persons.remove(person);
47         }
48
49         persons.add(person);
50
51         return person.getId();
52     }
53
54     @Override
55     // Выдаленне карыстальніка
56     public boolean delete(Person person) {
57         return persons.remove(person);
58     }
59
60     @Override
61     // Пошук карыстальніка па імені
62     public Person findByName(String name) {
63         System.out.println("findByName -> All users =" + getAll());
64
65         // Перабіраем усіх карыстальнікаў і параўноўваем іх

```



```

63         // електронную пошту з дадзенай
64         for (Person currentPerson : persons) {
65             if (name.equals(currentPerson.getName())) {
66                 System.out.println("findByName -> " + currentPerson);
67                 // Калі знайшлі супадзенне, вяртаем карыстальніка
68                 return currentPerson;
69             }
70         }
71         // калі не знайшлі супадзенне, вяртаем пустату
72         return null;
73     }
74
75     @Override
76     // Абнаўленне карыстальніка
77     public boolean update(Person person) {
78         if (persons.contains(person)) {
79             persons.remove(person);
80         }
81
82         return persons.add(person);
83     }
84
85     @Override
86     // Пошук карыстальніка па электроннай пошце
87     // пошук адбываецца аналагічна findByName
88     public Person findByEmail(String email) {
89         for (Person currentPerson : persons) {
90             if (email.contentEquals(currentPerson.getEmail())) {
91                 return currentPerson;
92             }
93         }
94
95         return null;
96     }
97
98     @Override
99     // Атрымаць мноства ўсіх карыстальнікаў
100    public Set<Person> getAll() {
101        return getPersons();
102    }
103
104    // Атрымаць мноства ўсіх карыстальнікаў
105    public Set<Person> getPersons() {
106        return persons;
107    }
108
109    public void setPersons(Set<Person> persons) {
110        this.persons = persons;
111    }
112 }

```

Лістынг 3.6 – Зыходны код класа SimplePersonDAOImpl

### 3.5.5 Абноўлены інтэрфейс Authenticator.

Абнавім інтэрфейс *Authenticator* для таго, каб ён мог працаваць з класам Person.

У лістынг 3.7 прадстаўлены зыходны код абноўленага інтэрфейса *Authenticator*.

```

1 package by.bsac.pz3;
2
3 import by.bsac.pz3.model.Person;
4
5 public interface Authenticator {
6     public Person authenticateByUsername(String username, String password);
7
8     public Person authenticateByEmail(String email, String password);
9 }

```

Лістынг 3.7 – Зыходны код інтэрфейса Authenticator

### 3.5.6 Абноўлены клас AuthenticatorImpl.

Абнавім клас *AuthenticatorImpl*, каб ён ствараў аб'ект класа *Person* з інфармацыяй пра карыстальніка па ўмаўчанню. Зменім логіку метадаў аўтэнтыфікацыі, каб яны вярталі аб'ект класа *Person*.

У лістынг 3.8 прадстаўлены зыходны код абноўленага класа *AuthenticatorImpl*.

```

1 package by.bsac.pz3;
2
3 import java.util.Date;
4
5 import by.bsac.pz3.model.Person;
6
7 public class AuthenticatorImpl implements Authenticator {
8     // Даныя на ўмаўчанню для аўтэнтыфікацыі
9     private String username = "Antos";
10    private String password = "password";
11    private String email = "antos@bsac.by";
12
13    private Long id;
14    private PersonRole role;
15    private Date loginDate;
16
17    private Person person;
18
19    // Пры стварэнні аб'екта класа AuthenticatorImpl
20    // ствараецца аб'ект класа Person з данымі на ўмаўчанню
21    public AuthenticatorImpl() {
22        person = new Person(999L, email, username, password, PersonRole.ADMIN, null);
23    }
24
25    @Override
26    public Person authenticateByUsername(String username, String password) {
27        if ( (getUsername().equalsIgnoreCase(username))
28            && (getPassword().equals(password)) ) {
29            return getPerson();
30        }
31
32        return null;
33    }
34
35    @Override
36    public Person authenticateByEmail(String email, String password) {
37        if (getEmail().equalsIgnoreCase(email)

```

```

38         && getPassword().equals(password)) {
39             return getPerson();
40         }
41
42         return null;
43     }
44
45     public Long getId() {
46         return id;
47     }
48
49     public void setId(Long id) {
50         this.id = id;
51     }
52
53     public PersonRole getRole() {
54         return role;
55     }
56
57     public void setRole(PersonRole role) {
58         this.role = role;
59     }
60
61     public Date getLoginDate() {
62         return loginDate;
63     }
64
65     public void setLoginDate(Date loginDate) {
66         this.loginDate = loginDate;
67     }
68
69     public Person getPerson() {
70         return person;
71     }
72
73     public void setPerson(Person person) {
74         this.person = person;
75     }
76
77     public void setUsername(String username) {
78         this.username = username;
79     }
80
81     public void setPassword(String password) {
82         this.password = password;
83     }
84
85     public String getPassword() {
86         return password;
87     }
88
89     public String getUsername() {
90         return username;
91     }
92
93     public String getEmail() {
94         return email;
95     }
96
97     public void setEmail(String email) {
98         this.email = email;

```

```
99     }
100 }
```

### Лістынг 3.8 – Зыходны код класа AuthenticatorImpl

#### 3.5.7 Клас SimpleAuthenticatorImpl.

Дадзены клас прадастаўляе метады аўтэнтыфікацыі для новага спосабу захоўвання карыстальнікаў (класы Person і SimplePersonDAOImpl).

У лістынг 3.9 прадстаўлены зыходны код класа *SimpleAuthenticatorImpl*.

```
1  package by.bsac.pz3;
2
3  import by.bsac.pz3.dao.PersonDAO;
4  import by.bsac.pz3.dao.SimplePersonDAOImpl;
5  import by.bsac.pz3.model.Person;
6
7  public class SimpleAuthenticatorImpl implements Authenticator {
8      private PersonDAO personDAO = SimplePersonDAOImpl.getInstance();
9
10     @Override
11     public Person authenticateByUsername(String username, String password) {
12         Person person = getPersonDAO().findByName(username);
13
14         if (isValidPassword(person, password)) {
15             return person;
16         }
17
18         return null;
19     }
20
21     @Override
22     public Person authenticateByEmail(String email, String password) {
23         Person person = getPersonDAO().findByEmail(email);
24
25         if (isValidPassword(person, password)) {
26             return person;
27         }
28
29         return null;
30     }
31
32     public PersonDAO getPersonDAO() {
33         return personDAO;
34     }
35
36     public void setPersonDAO(PersonDAO personDAO) {
37         this.personDAO = personDAO;
38     }
39
40     private boolean isValidPassword(Person person, String password) {
41         return person != null && person.getPassword().equals(password);
42     }
43 }
```

### Лістынг 3.9 – Зыходны код класа SimpleAuthenticatorImpl

### 3.5.8 Абнoўлены клас LoginController.

Абнaвiм клас *LoginController*, каб ён працаваў з класам *Person* i ў залежнасцi ад ролi карыстальнiка дабаўляў неабходную iнфармацыю ў параметры сесii з уласцiвасцяў аб'екта.

У лiстынгy 3.10 прадстаўлены зыходны код абнoўленага класа *LoginController*.

```
1 package by.bsac.pz3;
2
3 import java.io.IOException;
4
5 import java.time.format.DateTimeFormatter;
6 import java.time.LocalDateTime;
7
8 import javax.servlet.RequestDispatcher;
9 import javax.servlet.ServletException;
10 import javax.servlet.http.HttpServlet;
11 import javax.servlet.http.HttpServletRequest;
12 import javax.servlet.http.HttpServletResponse;
13 import javax.servlet.http.HttpSession;
14
15 import by.bsac.pz3.dao.PersonDAO;
16 import by.bsac.pz3.dao.SimplePersonDAOImpl;
17 import by.bsac.pz3.model.Person;
18
19 public class LoginController extends HttpServlet {
20     private static final long serialVersionUID = 1L;
21     private PersonDAO personDAO = SimplePersonDAOImpl.getInstance();
22
23     public LoginController() {
24         super();
25     }
26
27     protected void doGet(HttpServletRequest request, HttpServletResponse response)
28         throws ServletException, IOException {
29         response.sendRedirect("login.jsp");
30     }
31
32     protected void doPost(HttpServletRequest request, HttpServletResponse response)
33         throws ServletException, IOException {
34         System.out.println("Login -> DoPost");
35         String authAction = request.getParameter("authAction");
36
37         if (authAction.contentEquals("login")) {
38             login(request, response);
39         } else if (authAction.equals("logout")) {
40             logout(request, response);
41         }
42     }
43
44     private void login(HttpServletRequest request, HttpServletResponse response)
45         throws ServletException, IOException {
46         String authTypeParam = request.getParameter("authType");
47         Authenticator authenticator = new SimpleAuthenticatorImpl();
48         Person person;
49         String password = request.getParameter("psw");
50         String authValue = request.getParameter("loginValue");
51
52         if (authTypeParam.contentEquals("email")) {
53             person = authenticator.authenticateByEmail(authValue, password);
54         } else {
```

```

55         person = authenticator.authenticateByUserName(authValue, password);
56     }
57
58     // Калі аўтэнтыфікацыя прайшла паспяхова (вярнуўся аб'ект класа Person)
59     if (person != null) {
60         HttpSession session = request.getSession();
61
62         // устанаўліваем дату аўтэнтыфікацыі для карыстальніка
63         person.setLoginDate(ProfileTools.generateLoginDate());
64         getPersonDAO().save(person);
65
66         session.setAttribute(ProfileTools.SESSION_LOGGEDIN_ATTRIBUTE_NAME, person.getName());
67
68         // калі карыстальнік адміністратар, то запісваем у пераменную сесіі
69         // іфармацыю пра ўсіх карыстальнікаў
70         if (ProfileTools.isAdmin(person)) {
71             session.setAttribute(ProfileTools.PERSON_IS_ADMIN, true);
72             session.setAttribute(ProfileTools.ALL_PERSONS_ATTRIBUTE_NAME,
getPersonDAO().getAll());
73         }
74         response.sendRedirect("home.jsp");
75     } else {
76         response.sendRedirect("error-login.jsp");
77     }
78 }
79
80 public void setPersonDAO(PersonDAO personDAO) {
81     this.personDAO = personDAO;
82 }
83
84 public PersonDAO getPersonDAO() {
85     return personDAO;
86 }
87
88 private void logout(HttpServletRequest request, HttpServletResponse response)
89     throws ServletException, IOException {
90     HttpSession session = request.getSession(false);
91
92     if (session != null) {
93         session.invalidate();
94     }
95
96     response.sendRedirect("login.jsp");
97 }
98 }

```

Лістынг 3.10 – Зыходны код класа LoginController

## 3.6 Індывідуальнае заданне

### 3.6.1 Апісанне задання.

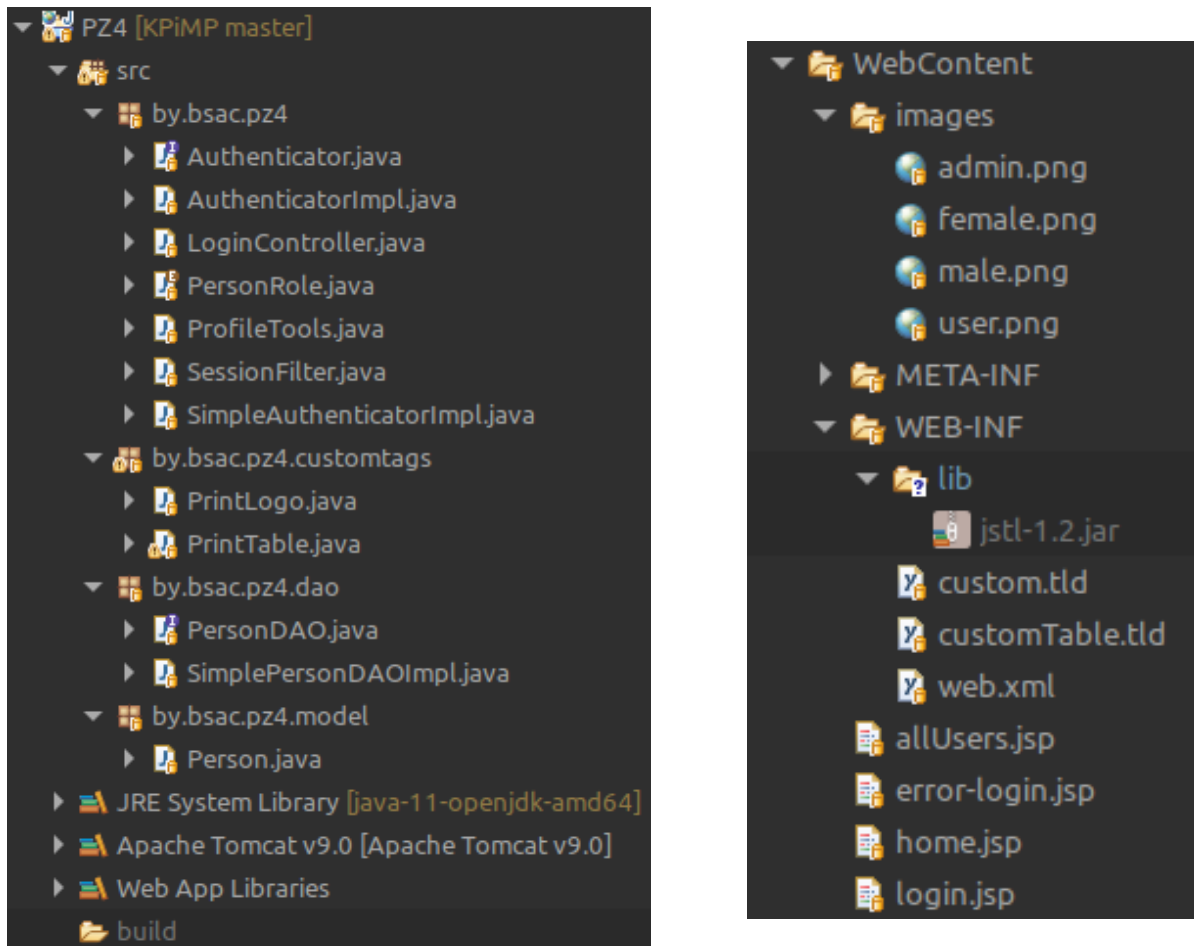
У табліцы (глядзі *allUsers.jsp* вывесці паведамленне, што карыстальнік не ўваходзіў у вэб-праграму, калі дата ўваходу не вызначана, інакш вывесці дату (для фармавання даты скарыстацца тэгам *fmt*).

Рашэнне дадзенага задання глядзі ў лістынг 3.2.

## 4 ПРАКТЫЧНЫ ЗАНЯТАК №4

### 4.1 Структура праекта

На малюнку 4.1 прадстаўлена файлавая структура праекта.



Малюнак 4.1 – Файлавая структура практычнага занятку

### 4.2 Заданне з тэорыі

Дадзенае заданне выконваецца на базе прыкладнага занятка №3 (заданне з тэорыі).

#### 4.2.1 Апісанне задання.

Арганізаваць вывад лагатыпа карыстальніка пры дапамозе карыстальніцкіх тэгаў.

Для гэтага перад тым, як пачаць выконваць заданне, неабходна пакласці карцінкі лагатыпаў у *WebContent/images*.

## 4.3 Зыходны код. Java

### 4.3.1 Клас PrintLogo.

Дадзены клас адказвае за выкананне дзеянняў (пабудова поўнага шляху да карцінкі лагатыпа) пры выкліку тэга ў JSP старонцы.

У лістынгу 4.1 прадстаўлены зыходны код класа *PrintLogo*.

```
1 package by.bsac.pz4.customtags;
2
3 import javax.servlet.jsp.JspException;
4 import javax.servlet.jsp.JspWriter;
5 import javax.servlet.jsp.tagext.TagSupport;
6
7 // Дадзены клас будзе выконвацца ў месцы, дзе будзе выклікацца
8 // карыстальніцкі тэг
9 @SuppressWarnings("serial")
10 public class PrintLogo extends TagSupport {
11     public static final String ADMIN_IMAGE = "admin";
12     public static final String USER_IMAGE = "user";
13     public static final String IMAGE_PATH = "images/";
14     public static final String IMAGE_EXTENSION = ".png";
15
16     private boolean isAdmin;
17
18     // Дадзены метаад стварае поўны шлях да карцінкі лагатыпа і
19     // выводзіць атрыманы шлях замест выкліку тэга ў JSP старонцы
20     public int doStartTag() throws JspException {
21         JspWriter out = pageContext.getOut();
22
23         try {
24             StringBuilder fullImagePath = new StringBuilder().append(IMAGE_PATH);
25
26             // У залежнасці ад ролі карыстальніка дабаўляем адпаведную карцінку
27             if (isAdmin) {
28                 fullImagePath.append(ADMIN_IMAGE);
29             } else {
30                 fullImagePath.append(USER_IMAGE);
31             }
32
33             fullImagePath.append(IMAGE_EXTENSION);
34
35             // вяртаем шлях на месца выкліку тэга
36             out.print(fullImagePath.toString());
37         } catch (Exception e) {
38             e.printStackTrace();
39         }
40
41         return SKIP_BODY;
42     }
43
44     public boolean isAdmin() {
45         return isAdmin;
46     }
47
48     public void setIsAdmin(boolean isAdmin) {
49         this.isAdmin = isAdmin;
50     }
```



## Лістынг 4.1 – Зыходны код класа PrintLogo

## 4.3.2 Канфігурацыя тэга custom.tld.

Для таго, каб вэб-праграма ведала пра створаны тэг, неабходна апісаць яго. Для гэтага апішам тэг у файле *custom.tld* і пакладзем яго ў *WEB-INF*.

У лістынг 4.2 прадстаўлена апісанне *custom.tld*.

```

1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE taglib PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.2//EN"
3   "http://java.sun.com/j2ee/dtd/web-jsptaglibrary_1_2.dtd">
4 <taglib>
5   <tlib-version>1.2</tlib-version>
6   <jsp-version>2.0</jsp-version>
7   <short-name>m</short-name>
8   // Які адрас задаваць для апісання тэга
9   <uri>PZ4</uri>
10  <tag>
11    // Назва тэга
12    <name>printUserLogo</name>
13    // Шлях да класа, які апісвае тэг
14    <tag-class>by.bsac.pz4.customtags.PrintLogo</tag-class>
15    // Дадатковыя атрыбуты, якія можна перадаваць унутры тэга
16    <attribute>
17      <name>isAdmin</name>
18      <required>true</required>
19      <rtexprvalue>true</rtexprvalue>
20    </attribute>
21  </tag>
22 </taglib>

```

## Лістынг 4.2 – Апісанне custom.tld

## 4.3.3 Абноўленая старонка home.jsp

Абнавім старонку *home.jsp*, каб на ёй выводзіўся лагатып карыстальніка; шлях лагатыпа ствараўся пры дапамозе карыстальніцкага тэга.

У лістынг 4.3 прадстаўлена абноўленая старонка *home.jsp*.

```

1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
4   // Падключаем намі створаны тэг
5   // uri павінен супадаць з тым, які анікалі ў custom.tld
6 <%@ taglib uri="PZ4" prefix="m" %>
7 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
8   "http://www.w3.org/TR/html4/loose.dtd">
9 <html>
10 <head>
11 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12 <title>Home</title>
13 </head>
14 <body>

```

```

15     <h1>
16         Hello ,
17         <%=session.getAttribute("user")%>
18     </h1>
19     // Дабаўляем html-тэг <img src=> для ўстаўкі лагатыпу,
20     // у месцы для крыніцы лагатыпу выклікаем наш тэг з дадатковым
21     // атрыбутам isAdmin, і даем яму значэнне роўнае параметру
22     // з сесіі isAdmin
23     <img src='<m:printUserLogo isAdmin="${isAdmin}" />' width="64" height="64" />
24     <c:if test="${isAdmin}">
25     <p><a href="allUsers.jsp">View all users</a></p>
26     </c:if>
27     <form action="logout" method="post">
28         <input type="hidden" name="authAction" value="logout">
29         <input type="submit" value="Logout" />
30     </form>
31 </body>
32 </html>

```

### Лістынг 4.3 – Старонка home.jsp

## 4.4 Індывідуальнае заданне

### 4.4.1 Апісанне задання.

На старонцы *allUsers.jsp* табліцу *users* вывесці пры дапамозе карыстальніцкага тэга.

### 4.4.2 Клас PrintTable

Як было апісана вышэй, для рэалізацыі карыстальніцкага тэга неабходна стварыць клас, які будзе вызначаць логіку тэга.

У лістынг 4.4 прадстаўлены зыходны код абноўленага класа *PrintTable*.

```

1  package by.bsac.pz4.customtags;
2
3  import java.text.DateFormat;
4  import java.text.SimpleDateFormat;
5  import java.util.Date;
6  import java.util.Set;
7
8  import javax.servlet.http.HttpServletRequest;
9  import javax.servlet.http.HttpSession;
10 import javax.servlet.jsp.JspException;
11 import javax.servlet.jsp.JspWriter;
12 import javax.servlet.jsp.tagext.TagSupport;
13
14 import by.bsac.pz4.ProfileTools;
15 import by.bsac.pz4.model.Person;
16
17 @SuppressWarnings("serial")
18 public class PrintTable extends TagSupport {
19     public int doStartTag() throws JspException {
20         JspWriter out = pageContext.getOut();
21
22         HttpServletRequest request = (HttpServletRequest)pageContext.getRequest();

```

```

23     HttpSession session = request.getSession();
24
25     // Атрымлівае з параметраў сесіі інфармацыю пра карыстальнікаў
26     Set<Person> users = (Set<Person>)
session.getAttribute(ProfileTools.ALL_PERSONS_ATTRIBUTE_NAME);
27
28     try {
29         // Пачынаем html-табліцу
30         StringBuilder table = new StringBuilder().append("<table width=\"100%\" border=\"1\">");
31         // Дабаўляем загалоўкі для табліцы
32         table.append(
33             "<tr>\n"
34                 + "<th>Name</th>\n"
35                 + "<th>Email</th>\n"
36                 + "<th>Role</th>\n"
37                 + "<th>Last login date</th>\n"
38             + "</tr>");
39
40
41         DateFormat formatter = new SimpleDateFormat("dd-M-yyyy hh:mm:ss");
42
43         // Перабіраем усіх карыстальнікаў, спіс каторых атрымалі з параметраў сесіі
44         for (Person user : users) {
45             Date loginDate = user.getLoginDate();
46
47             // Калі карыстальнік не ўваходзіў у вэб-праграму
48             // выводзіць наведамленне User have not logged у поле для даты ўваходу
49             String strDate = "User have not logged";
50             // Калі карыстальнік уваходзіў,
51             // запісваем бягучую дату
52             if (loginDate != null) {
53                 strDate = formatter.format(loginDate);
54             }
55
56             // Ствараем радок у табліцы для бягучага карыстальніка
57             table.append(
58                 "<tr>" +
59                 "<td>" + user.getName() + "</td>" +
60                 "<td>" + user.getEmail() + "</td>" +
61                 "<td>" + user.getRole() + "</td>" +
62                 "<td>" + strDate + "</td>" +
63                 "</tr>"
64             );
65         }
66
67         // Завершаем табліцу
68         table.append("</table>");
69
70         // Выводзім атрыманую табліцу замест выкліку тэга ў allUsers.jsp
71         out.print(table.toString());
72     } catch (Exception e) {
73         e.printStackTrace();
74     }
75
76     return SKIP_BODY;
77 }
78 }

```

Лістынг 4.4 – Зыходны код класа PrintTable

#### 4.4.3 Апісанне тэга customTable.tld.

У лістынг 4.5 прадстаўлена апісанне тэга *customTable.tld*.

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE taglib PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.2//EN"
3   "http://java.sun.com/j2ee/dtd/web-jsptaglibrary_1_2.dtd">
4 <taglib>
5   <tlib-version>1.2</tlib-version>
6   <jsp-version>2.0</jsp-version>
7   <short-name>t</short-name>
8   <uri>table</uri>
9   <tag>
10     <name>customTable</name>
11     <tag-class>by.bsac.pz4.customtags.PrintTable</tag-class>
12   </tag>
13 </taglib>
```

Лістынг 4.5 – Апісанне тэга customTable.tld

#### 4.4.4 Абноўленая старонка allUsers.jsp

Заменім стварэнне табліцы карыстальнікаў у самой старонцы на выклік уласнага тэга.

У лістынг 4.6 прадстаўлена абноўленая старонка *allUsers.jsp*.

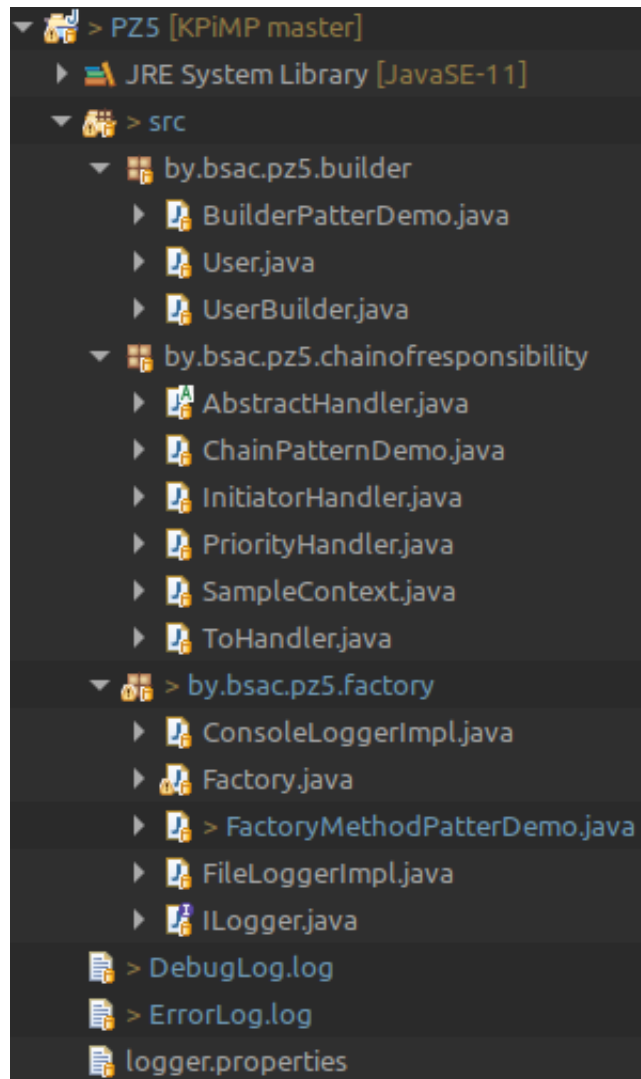
```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
4 <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
5   // Падключаем тэгі, створаны ў індывідуальным заданні
6 <%@ taglib uri="table" prefix="t" %>
7
8 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
9   "http://www.w3.org/TR/html4/loose.dtd">
10 <html>
11 <head>
12 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13 <title>View All Users</title>
14 </head>
15 <body>
16   <h1>View All Users</h1>
17   <a href="home.jsp">Home Page</a>
18   <br />
19   <c:if test="${isAdmin}">
20     <h3>You are Admin</h3>
21     <p>
22       Count of Users
23       <c:out value="${users.size()}" />
24       // Выклікаем створаны тэгі,
25       // які створыць у гэтым месцы html-табліцу карыстальнікаў
26       <t:customTable />
27     </c:if>
28 </body>
29 </html>
```

Лістынг 4.6 – Старонка allUsers.jsp

## 5 ПРАКТЫЧНЫ ЗАНЯТАК №5

### 5.1 Структура праекта

На малюнку 5.1 прадстаўлена файлавая структура праекта.



Малюнак 5.1 – Файлавая структура практычнага занятку

### 5.2 Заданне 1

#### 5.2.1 Апісанне задання.

Пры дапамозе шаблона *Factory* рэалізаваць логіку, якая выводзіць паведамленні ў *debug* альбо ў *error* на экран альбо запісвае ў файл.

Інтэрфейс *ILogger* апісвае два асноўных метады *debug* з параметрам *msg* і *error* з параметрам *msg*, дзе *msg* - паведамленне, якое трэба вывесці на кансолі або запісаць у файл. Створаны два класы, такія як *ConsoleLoggerImpl* – для вываду паведамлення на кансоль і *FileLoggerImpl* –

для запису паведамлення ў файл. Так жа створаны клас `Factory`, які з дапамогай метады `getLogger` вяртае `FileLoggerImpl` альбо `ConsoleLoggerImpl` на падставе ўласцівасці `FileLogging = true` (або `false`), што знаходзіцца у файле `logger.properties`. Калі значэнне `true`, то `getLogger` вяртае `FileLoggerImpl` інакш `ConsoleLoggerImpl`. У класе `FactoryMethodPatternDemo` прыведзены прыклад выкарыстання шаблону `Factory` на практыцы.

У лістынг 5.1 прадстаўлены зыходны код інтэрфейса *`ILogger`*.

```
1 package by.bsac.pz5.factory;
2
3 public interface ILogger {
4     public void debug(String msg);
5     public void error(String msg);
6 }
```

Лістынг 5.1 – Зыходны код інтэрфейса `ILogger`

У лістынг 5.2 прадстаўлены зыходны код класа *`ConsoleLoggerImpl`*.

```
1 package by.bsac.pz5.factory;
2
3 public class ConsoleLoggerImpl implements ILogger {
4
5     ConsoleLoggerImpl() {}
6
7     @Override
8     public void debug(String msg) {
9         System.out.println("DEBUG:" + msg);
10    }
11
12    @Override
13    public void error(String msg) {
14        System.err.println("ERROR: " + msg);
15    }
16
17 }
```

Лістынг 5.2 – Зыходны код класа `ConsoleLoggerImpl`

У лістынг 5.3 прадстаўлены зыходны код класа *`FileLoggerImpl`*.

```
1 package by.bsac.pz5.factory;
2
3 import java.io.BufferedWriter;
4 import java.io.File;
5 import java.io.FileWriter;
6 import java.io.IOException;
7 import java.text.SimpleDateFormat;
8 import java.util.Date;
9
10 public class FileLoggerImpl implements ILogger {
11     private static int count = 0;
12     private static StringBuilder bufferString = new StringBuilder("");
13 }
```

```

14     @Override
15     public void debug(String msg) {
16         try {
17             if (count < 5) {
18                 Date date = new Date();
19                 SimpleDateFormat formatter = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss");
20                 String strDate = formatter.format(date);
21
22                 bufferString.append "[" + strDate + "] DEBUG: " + String.valueOf(count) + msg);
23
24                 count++;
25             } else {
26                 File file = new File("DebugLog.log");
27
28                 if (!file.exists()) {
29                     file.createNewFile();
30                 }
31
32                 FileWriter fileWriter = new FileWriter(file.getName(), true);
33                 BufferedWriter bufferWriter = new BufferedWriter(fileWriter);
34                 bufferWriter.write(bufferString.toString());
35                 bufferWriter.close();
36                 count = 0;
37                 bufferString = new StringBuilder("");
38             }
39         } catch (IOException e) {
40             e.printStackTrace();
41         }
42     }
43
44     @Override
45     public void error(String msg) {
46         try {
47             File file = new File("ErrorLog.log");
48
49             if (!file.exists()) {
50                 file.createNewFile();
51             }
52
53             FileWriter fileWriter = new FileWriter(file.getName(), true);
54             BufferedWriter bufferWriter = new BufferedWriter(fileWriter);
55             bufferWriter.write("ERROR: " + msg);
56             bufferWriter.close();
57         } catch (IOException e) {
58             e.printStackTrace();
59         }
60     }
61 }

```

### Лістынг 5.3 – Зыходны код класа FileLoggerImpl

У лістынг 5.4 прадстаўлены зыходны код класа *Factory*.

```

1 package by.bsac.pz5.factory;
2
3 import java.io.File;
4 import java.io.FileInputStream;
5 import java.io.IOException;
6 import java.io.InputStream;

```

```

7  import java.util.Properties;
8
9  public class Factory {
10     public ILogger getLogger() {
11         if (isFileLoggingEnabled()) {
12             return new FileLoggerImpl();
13         } else {
14             return new ConsoleLoggerImpl();
15         }
16     }
17
18     private boolean isFileLoggingEnabled() {
19         Properties p = new Properties();
20
21         try {
22             InputStream input = new FileInputStream("logger.properties");
23             p.load(input);
24             String fileLoggingValue = p.getProperty("FileLogging");
25
26             if ("true".equalsIgnoreCase(fileLoggingValue)) {
27                 return true;
28             } else {
29                 return false;
30             }
31         } catch (IOException e) {
32             return false;
33         }
34     }
35 }

```

Лістынг 5.4 – Вихідний код класа *Factory*

У лістингу 5.5 представлено вихідний код класа *Factory*.

```

1  package by.bsac.pz5.factory;
2
3  public class FactoryMethodPatternDemo {
4
5     public static void main(String args[]) throws InterruptedException {
6         Factory f = new Factory();
7         ILogger msgLogger = f.getLogger();
8
9         for (int i = 0; i < 6; i++) {
10             msgLogger.debug("Sample debug message\n");
11             Thread.sleep(1000);
12         }
13
14         msgLogger.error("Sample error message\n");
15     }
16 }

```

Лістынг 5.5 – Вихідний код класа *FactoryMethodPatternDemo*



## 5.3 Заданне 2

Пры дапамозе шаблона Builder стварыць клас, які можа ствараць карыстальнікаў з уласцівасцямі, у залежнасці ад выкліканага канструктара.

У лістынг 5.6 прадстаўлены зыходны код класа *User*.

```
1 package by.bsac.pz5.builder;
2
3 public class User {
4     private String firstName;
5     private String lastName;
6     private int age;
7     private final String phone;
8     private final String address;
9
10    User(UserBuilder builder) {
11        this.firstName = builder.firstName;
12        this.lastName = builder.lastName;
13        this.age = builder.age;
14        this.phone = builder.phone;
15        this.address = builder.address;
16    }
17
18    public String getFirstName() {
19        return firstName;
20    }
21
22    public String getLastName() {
23        return lastName;
24    }
25
26    public int getAge() {
27        return age;
28    }
29
30    public String getPhone() {
31        return phone;
32    }
33
34    public String getAddress() {
35        return address;
36    }
37
38    @Override
39    public String toString() {
40        return "User: " + this.firstName +
41            ", " + this.lastName +
42            ", " + this.phone +
43            ", " + this.address;
44    }
45 }
```

Лістынг 5.6 – Зыходны код класа User

У лістынг 5.7 прадстаўлены зыходны код класа *UserBuilder*.

```

1 package by.bsac.pz5.builder;
2
3 public class UserBuilder {
4     String firstName;
5     String lastName;
6     int age;
7     String phone;
8     String address;
9
10    public UserBuilder(String firstName, String lastName) {
11        this.firstName = firstName;
12        this.lastName = lastName;
13    }
14
15    public UserBuilder age(int age) {
16        this.age = age;
17        return this;
18    }
19
20    public UserBuilder phone(String phone) {
21        this.phone = phone;
22        return this;
23    }
24
25    public UserBuilder address(String address) {
26        this.address = address;
27        return this;
28    }
29
30    public User build() {
31        User user = new User(this);
32        return user;
33    }
34 }

```

Лістынг 5.7 – Зыходны код класа UserBuilder

У лістынг 5.8 прадстаўлены зыходны код класа *BuilderPatternDemo*.

```

1 package by.bsac.pz5.builder;
2
3 public class BuilderPatterDemo {
4     public static void main(String[] args) {
5         User user1 = new UserBuilder("John", "Doe")
6             .age(30)
7             .phone("1234567")
8             .address("Fake address 1234")
9             .build();
10
11        System.out.println(user1);
12
13        User user2 = new UserBuilder("Ivan", "Petrov")
14            .age(40)
15            .phone("5655")
16            .build();
17
18        System.out.println(user2);
19
20        User user3 = new UserBuilder("Super", "Man")

```

```

21         .build();
22
23         System.out.println(user3);
24     }
25 }

```

Лістынг 5.8 – Зыходны код класа BuilderPatternDemo

### 5.4 Заданне 3

У лістынг 5.9 прадстаўлены зыходны код класа *AbstractHandler*.

```

1 package by.bsac.pz5.chainofresponsibility;
2
3 public abstract class AbstractHandler {
4     protected AbstractHandler nextHandler;
5
6     public void setNextHandler(AbstractHandler nextHandler) {
7         this.nextHandler = nextHandler;
8     }
9
10    public boolean handle(SampleContext context) {
11        boolean processedChain = process(context);
12
13        if (processedChain && nextHandler != null) {
14            processedChain = nextHandler.handle(context);
15        }
16
17        return processedChain;
18    }
19
20    abstract protected boolean process(SampleContext context);
21 }

```

Лістынг 5.9 – Зыходны код класа AbstractHandler

У лістынг 5.10 прадстаўлены зыходны код класа *ChainPatternDemo*.

```

1 package by.bsac.pz5.chainofresponsibility;
2
3 public class ChainPatternDemo {
4     private static AbstractHandler getChainOfLoggers() {
5         AbstractHandler initiatorHandler = new InitiatorHandler();
6         AbstractHandler toHandler = new ToHandler();
7         AbstractHandler priorityHandler = new PriorityHandler();
8
9         priorityHandler.setNextHandler(initiatorHandler);
10        initiatorHandler.setNextHandler(toHandler);
11
12        return priorityHandler;
13    }
14
15    public static void main(String[] args) {
16        SampleContext context = new SampleContext("boss", "support", 11, "Fix");

```

```

17
18     AbstractHandler handlerChain = getChainOfLoggers();
19     boolean validContext = handlerChain.handle(context);
20
21     if (validContext) {
22         System.out.println(context.msg);
23     }
24 }
25 }

```

Лістынг 5.10 – Зыходны код класа ChainPatternDemo

У лістынг 5.11 прадстаўлены зыходны код класа *InitiatorHandler*.

```

1 package by.bsac.pz5.chainofresponsibility;
2
3 public class InitiatorHandler extends AbstractHandler {
4     private String initiator = "boss";
5
6     @Override
7     protected boolean process(SampleContext context) {
8         boolean result = context.getFrom().equalsIgnoreCase(initiator);
9         System.out.println(this.getClass() + ": result = " + result);
10
11         return result;
12     }
13 }

```

Лістынг 5.11 – Зыходны код класа InitiatorHandler

У лістынг 5.12 прадстаўлены зыходны код класа *PriorityHandler*.

```

1 package by.bsac.pz5.chainofresponsibility;
2
3 public class PriorityHandler extends AbstractHandler {
4     private int priorityLevel = 10;
5
6     @Override
7     protected boolean process(SampleContext context) {
8         boolean result = context.getPriority() < priorityLevel;
9         System.out.println(this.getClass() + ": result = " + result);
10
11         return result;
12     }
13 }

```

Лістынг 5.12 – Зыходны код класа PriorityHandler

У лістынг 5.13 прадстаўлены зыходны код класа *SampleContext*.

```

1 package by.bsac.pz5.chainofresponsibility;
2
3 public class SampleContext {
4     public String from;

```

```

5     public String to;
6     public int priority;
7     public String msg;
8
9     public SampleContext(String from, String to, int priority, String msg) {
10         this.from = from;
11         this.to = to;
12         this.priority = priority;
13         this.msg = msg;
14     }
15
16     public String getFrom() {
17         return from;
18     }
19
20     public void setFrom(String from) {
21         this.from = from;
22     }
23
24     public String getTo() {
25         return to;
26     }
27
28     public void setTo(String to) {
29         this.to = to;
30     }
31
32     public int getPriority() {
33         return priority;
34     }
35
36     public void setPriority(int priority) {
37         this.priority = priority;
38     }
39
40     public String getMsg() {
41         return msg;
42     }
43
44     public void setMsg(String msg) {
45         this.msg = msg;
46     }
47
48 }

```

Лістынг 5.13 – Зыходны код класа SampleContext

У лістынг 5.14 прадстаўлены зыходны код класа *ToHandler*.

```

1 package by.bsac.pz5.chainofresponsibility;
2
3 public class ToHandler extends AbstractHandler {
4     private String validTo = "support";
5
6     @Override
7     protected boolean process(SampleContext context) {
8         boolean result = validTo.equalsIgnoreCase(context.getTo());
9         System.out.println(this.getClass() + ": result = " + result);
10    }

```

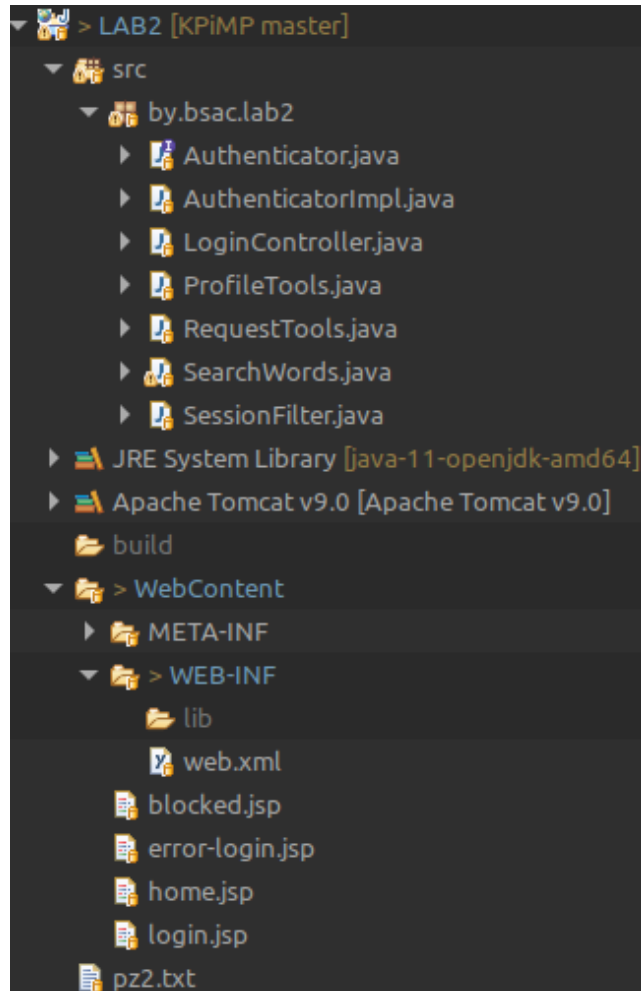
```
11         return result;
12     }
13 }
```

Лістынг 5.14 – Зыходны код класа ToHandler

## 6 ЛАБАРАТОРНАЯ РАБОТА №2

### 6.1 Структура праекта

На малюнку 6.1 прадстаўлена файлавая структура праекта.



Малюнак 6.1 – Файлавая структура практычнага занятку

### 6.2 Заданне

Неабходна рэалізаваць механізм блакіроўкі карыстальніка, які няверна ўвёў пароль 3 разы, а таксама выводзіць дату і час блакіроўкі.

Дадзенае заданне выконваецца на базе практычнага занятку №2.

#### 6.2.1 Старонка блакіроўкі blocked.jsp.

У лістынгу 6.1 прадстаўлена старонка блакіроўкі *blocked.jsp*.

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

```

2     pageEncoding="UTF-8"%>
3 <!DOCTYPE html PUBLIC "-//W3C/DTD HTML 4.01 Transitional//EN"
4     "http://www.w3.org/TR/html4/loose.dtd">
5 <html>
6 <head>
7 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
8 <title>Login failed</title>
9 </head>
10 <body>
11     <p> You are blocked since you tried to type incorrect login or password 3 times.
12     <p> Blocked date: ${blockedDate}
13     <p> Blocked time: ${blockedTime}
14 </body>
15 </html>

```

Лістынг 6.1 – Старонка блакіроўкі blocked.jsp

### 6.2.2 Клас RequestTools.

Дадзены клас забяспечвае механізмы праверкі спроб аўтэнтыфікацыі ў вэб-праграму і блакіроўкі карыстальніка, калі перавышана колькасць няверных логінаў ці пароляў.

У лістынг 6.2 прадстаўлены клас *RequestTools*.

```

1 package by.bsac.lab2;
2
3 import java.io.IOException;
4 import java.time.LocalDate;
5 import java.time.LocalTime;
6
7 import javax.servlet.ServletException;
8 import javax.servlet.http.HttpServletRequest;
9 import javax.servlet.http.HttpServletResponse;
10
11 public class RequestTools {
12     // Павялічвае лічыльнік няверных уводаў даных
13     public static void addLoginAttemptToSession(HttpServletRequest req,
14         HttpServletResponse resp) throws IOException, ServletException {
15         // Атрымлівае колькасць няверных уводаў з параметра сесіі
16         int attempts = Integer.parseInt(req.getSession().getAttribute("failedLoginAttemptsCount")
17             .toString());
18         // Павялічваем лічыльнік і запісваем назад у параметр сесіі
19         req.getSession().setAttribute("failedLoginAttemptsCount", attempts + 1);
20     }
21
22     // Праверка на блакіроўку карыстальніка
23     public static boolean isBlocked(HttpServletRequest req, HttpServletResponse resp)
24         throws IOException, ServletException {
25         // Калі карыстальнік не быў заблакаваны, то запісвае ў параметры сесіі
26         // час блакіроўкі
27         if (req.getSession().getAttribute("blockedDate") == null
28             || req.getSession().getAttribute("blockedTime") == null) {
29             req.getSession().setAttribute("blockedDate", LocalDate.now());
30             req.getSession().setAttribute("blockedTime", LocalTime.now());
31         }
32         System.out.println("isBlocked");
33         // Калі значэнне лічыльніка >= 3, то вяртае True (карыстальнік заблакаваны)
34         return

```



```

Integer.parseInt(req.getSession().getAttribute("failedLoginAttemptsCount").toString()) >= 3 ?
true : false;
35 }
36
37 public static void redirectToBlockedPage(HttpServletRequest req,
38     HttpServletResponse resp) throws IOException, ServletException {
39     req.getRequestDispatcher("/blocked.jsp").forward(req, resp);
40 }
41 }

```

## Лістынг 6.2 – Клас RequestTools

### 6.2.3 Абноўлены клас фільтра SessionFilter.

Абноўлены фільтр мае правяраць ці быў заблакаваны карыстальнік, калі карыстальнік заблакаваны перанакіроўваць на старонку блакіроўкі, інакш выставіць лічыльнік роўны 0 і перанакіраваць на старонку для ўвахода.

У лістынг 6.3 прадстаўлены абноўлены клас *SessionFilter*.

```

1 package by.bsac.lab2;
2
3 import java.io.IOException;
4 import java.util.ArrayList;
5 import java.util.StringTokenizer;
6
7 import javax.servlet.Filter;
8 import javax.servlet.FilterChain;
9 import javax.servlet.FilterConfig;
10 import javax.servlet.ServletException;
11 import javax.servlet.HttpServletRequest;
12 import javax.servlet.HttpServletResponse;
13 import javax.servlet.http.HttpServletRequest;
14 import javax.servlet.http.HttpServletResponse;
15
16 import by.bsac.lab2.ProfileTools;
17
18 public class SessionFilter implements Filter {
19     private ArrayList<String> ignoredUrlList;
20
21     public SessionFilter() {
22     }
23
24     public void destroy() {
25     }
26
27     public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
28         throws IOException, ServletException {
29         HttpServletRequest req = (HttpServletRequest) request;
30         HttpServletResponse res = (HttpServletResponse) response;
31
32         // Калі ў сесіі няма параметра failedLoginAttemptsCount,
33         // стварыць яго і прызначыць значэнне 0
34         if (req.getSession().getAttribute("failedLoginAttemptsCount") == null) {
35             req.getSession().setAttribute("failedLoginAttemptsCount", 0);
36         }
37

```

```

38     String requestUri = req.getRequestURI();
39     boolean shouldBeIgnored = isIgnoredUrl(requestUri);
40
41     // Проверьць ці заблакаваны карыстальнік у дадзенай сесіі
42     // калі заблакаваны, то перанакіраваць на старонку блакіроўкі,
43     // інакш на старонку для ўваходу
44     if (RequestTools.isBlocked(req, res)) {
45         RequestTools.redirectToBlockedPage(req, res);
46     } else if (!shouldBeIgnored && !ProfileTools.isLoggedIn(req)) {
47         req.getRequestDispatcher("/login").forward(req, res);
48     } else {
49         chain.doFilter(request, response);
50     }
51 }
52
53 public void init(FilterConfig fConfig) throws ServletException {
54     ignoredUrlList = new ArrayList<String>();
55     String urls = fConfig.getInitParameter("ignore-urls");
56     StringTokenizer token = new StringTokenizer(urls, ",");
57
58     while (token.hasMoreTokens()) {
59         ignoredUrlList.add(token.nextToken());
60     }
61 }
62
63 private boolean isIgnoredUrl(String url) {
64     for (String ignoredUrl : getIgnoredUrlList()) {
65         if (url.startsWith(ignoredUrl)) {
66             return true;
67         }
68     }
69
70     return false;
71 }
72
73 public ArrayList<String> getIgnoredUrlList() {
74     return ignoredUrlList;
75 }
76
77 public void setIgnoredUrlList(ArrayList<String> urlList) {
78     this.ignoredUrlList = urlList;
79 }
80 }

```

Лістынг 6.3 – Клас SessionFilter

## 6.2.4 Абноўлены клас LoginController.

Дадавіў у клас *LoginController* логіку блакіроўкі карыстальніка.

У лістынг 6.4 прадстаўлены абноўлены клас *LoginController*.

```

1 package by.bsac.lab2;
2
3 import java.io.IOException;
4
5 import java.time.format.DateTimeFormatter;
6 import java.time.LocalDateTime;

```

```

7
8 import javax.servlet.ServletException;
9 import javax.servlet.http.HttpServlet;
10 import javax.servlet.http.HttpServletRequest;
11 import javax.servlet.http.HttpServletResponse;
12 import javax.servlet.http.HttpSession;
13
14 public class LoginController extends HttpServlet {
15     private static final long serialVersionUID = 1L;
16
17     public LoginController() {
18         super();
19     }
20
21     protected void doGet(HttpServletRequest request, HttpServletResponse response)
22         throws ServletException, IOException {
23         response.sendRedirect("login.jsp");
24     }
25
26     protected void doPost(HttpServletRequest request, HttpServletResponse response)
27         throws ServletException, IOException {
28         String authAction = request.getParameter("authAction");
29
30         if (authAction.contentEquals("login")) {
31             login(request, response);
32         } else if (authAction.equals("logout")) {
33             logout(request, response);
34         }
35     }
36
37     private void login(HttpServletRequest request, HttpServletResponse response)
38         throws ServletException, IOException {
39         String authTypeParam = request.getParameter("authType");
40         Authenticator authenticator = new AuthenticatorImpl();
41         boolean isAuthenticated = false;
42         String password = request.getParameter("psw");
43         String authValue = request.getParameter("loginValue");
44
45         if (authTypeParam.contentEquals("email")) {
46             isAuthenticated = authenticator.authenticateByEmail(authValue, password);
47         } else {
48             isAuthenticated = authenticator.authenticateByUsername(authValue, password);
49         }
50
51         if (isAuthenticated) {
52             DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");
53             LocalDateTime now = LocalDateTime.now();
54             String loginData = dtf.format(now);
55
56             HttpSession session = request.getSession();
57
58             session.setAttribute(ProfileTools.SESSION_LOGGEDIN_ATTRIBUTE_NAME, authValue);
59             session.setAttribute(ProfileTools.SESSION_LOGGEDIN_ATTRIBUTE_PASSWORD, password);
60             session.setAttribute(ProfileTools.SESSION_LOGGEDIN_ATTRIBUTE_DATA, loginData);
61
62             // Калі аўтэнтыфікацыя прайшла паспяхова, абнуліць лічылнік
63             // і выдаць дату блакіроўкі
64             request.getSession().setAttribute("failedLoginAttemptsCount", 0);
65             request.getSession().removeAttribute("blockedDate");
66             request.getSession().removeAttribute("blockedTime");
67             response.sendRedirect("home.jsp");

```

```

68     } else {
69         // калі аўтэнтыфікацыя правалілася,
70         // па'явічыць лічылінік
71         RequestTools.addLoginAttemptToSession(request, response);
72
73         // калі карыстальнік заблакаваны (увёў няправільна 3 раз),
74         // перанакіраваць на старонку блакіроўкі
75         if (RequestTools.isBlocked(request, response)) {
76             RequestTools.redirectToBlockedPage(request, response);
77         } else {
78             response.sendRedirect("error-login.jsp");
79         }
80     }
81 }
82
83 private void logout(HttpServletRequest request, HttpServletResponse response)
84     throws ServletException, IOException {
85     HttpSession session = request.getSession(false);
86
87     if (session != null) {
88         DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");
89         LocalDateTime now = LocalDateTime.now();
90         String logoutData = dtf.format(now);
91
92         session.setAttribute(ProfileTools.SESSION_LOGGEDOUT_ATTRIBUTE_DATA, logoutData);
93
94         session.invalidate();
95     }
96
97     response.sendRedirect("login.jsp");
98 }
99 }

```

Лістынг 6.4 – Клас LoginController

## 7 ЛАБАТОРНАЯ РАБОТА №4

### 7.1 Заданне

Патэрн *Builder*. Распрацаваць мадэль сістэмы *Музыкальны калектыў*. Напісаць код праграмы, які дазваляе ствараць музыкальныя, танцавальныя, змешаныя клубы.

У лістынг 7.1 прадстаўлены зыходны код праграмы з тлумачэннямі.

```
1 package by.bsac.lab4;
2
3 // Дадзены клас змяшчае інфармацыю пра клуб
4 class Club {
5     private String dance;
6     private String music;
7     private int people;
8
9     public void setDance(String dance) {
10         this.dance = dance;
11     }
12
13     public void setMusic(String music) {
14         this.music = music;
15     }
16
17     public void setPeople(int people) {
18         this.people = people;
19     }
20
21     @Override
22     public String toString() {
23         return "Club [dance=" + dance + ", music=" + music + ", people=" + people + "];"
24     }
25 }
26
27 // Абстрактны клас, які будзе аб'ект класа Club
28 abstract class ClubBuilder {
29     protected Club club;
30
31     public Club getClub() {
32         return club;
33     }
34
35     public void createClub() {
36         club = new Club();
37     }
38
39     public abstract void buildDance();
40
41     public abstract void buildMusic();
42
43     public abstract void buildPeople();
44 }
45
46 // Музыкальны клуб са значэннямі па ўмаўчэнню
47 class MusicClubBuilder extends ClubBuilder {
48     public void buildMusic() {
49         club.setMusic("vocal");
```

```

50     }
51
52     public void buildPeople() {
53         club.setPeople(5);
54     }
55
56     public void buildDance() {
57         club.setDance("");
58     }
59 }
60
61 // Клуб, де єсть і танцюри, і співаки
62 class CommonClubBuilder extends ClubBuilder {
63     public void buildMusic() {
64         club.setMusic("vocal");
65     }
66
67     public void buildPeople() {
68         club.setPeople(5);
69     }
70
71     public void buildDance() {
72         club.setDance("DANCE");
73     }
74 }
75
76 // Танцювальні клуб
77 class DanceClubBuilder extends ClubBuilder {
78     public void buildDance() {
79         club.setDance("DANCE");
80     }
81
82     public void buildPeople() {
83         club.setPeople(5);
84     }
85
86     public void buildMusic() {
87         club.setMusic("");
88     }
89 }
90
91 // Клас, які визначає, які клуб буде побудовані,
92 // у залежності від типу параметра
93 class Director {
94     private ClubBuilder clubBuilder;
95
96     public void setClubBuilder(ClubBuilder cb) {
97         clubBuilder = cb;
98     }
99
100     public Club getClub() {
101         return clubBuilder.getClub();
102     }
103
104     public void constructClub() {
105         clubBuilder.createClub();
106         clubBuilder.buildMusic();
107         clubBuilder.buildDance();
108         clubBuilder.buildPeople();
109     }
110 }

```

```

111
112 public class BuilderDemo {
113     public static void main(String[] args) {
114         Director director = new Director();
115
116         MusicClubBuilder musicClub = new MusicClubBuilder();
117         DanceClubBuilder danceClub = new DanceClubBuilder();
118         CommonClubBuilder commonClub = new CommonClubBuilder();
119
120         director.setClubBuilder(musicClub);
121         director.constructClub();
122
123         Club club1 = director.getClub();
124
125         director.setClubBuilder(danceClub);
126         director.constructClub();
127
128         Club club2 = director.getClub();
129
130         director.setClubBuilder(commonClub);
131         director.constructClub();
132
133         Club club3 = director.getClub();
134
135         System.out.println(club1);
136         System.out.println(club2);
137         System.out.println(club3);
138     }
139 }

```

Лістынг 7.1 – Вихідний код програми