

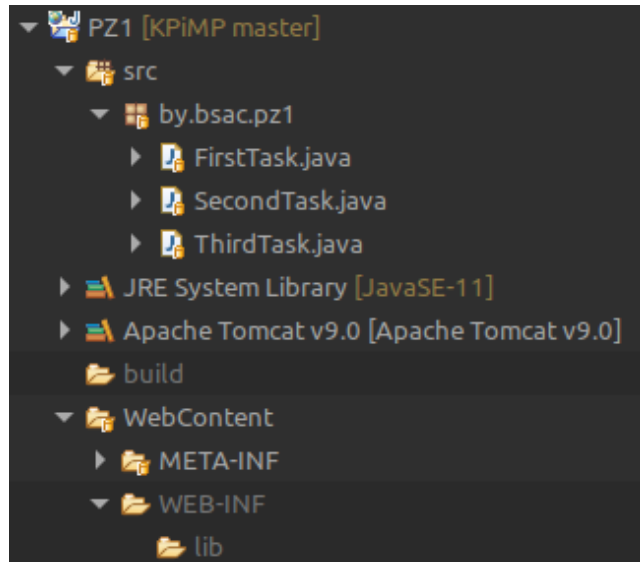
ЗМЕСТ

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

1 ПРАКТИЧНІ ЗАНЯТКИ №1

1.1 Структура проекту

На малюнку 1.1 представлена файлова структура проекту.



Малюнок 1.1 – Файлова структура практичного занятку

1.2 Задання 1

1.2.1 Описання задання.

Написати сервлет, який видає HTML-сторінку з полем для введення з іменем *PI*. Перед полем для введення мусить бути текст *Поля для введення*.

1.2.2 Вихідний код.

Вихідний код з тлумаченнями представлений у лістингу 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 анотація (замість того, щоб писати servlet у web.xml файлі)
12 // Спосіб з написанням web.xml буде розглядатися на наступних практичних занятках
13 // показує на якому URL-адресі буде виконуватися 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 сервлета 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 першадрукавай (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>"
```

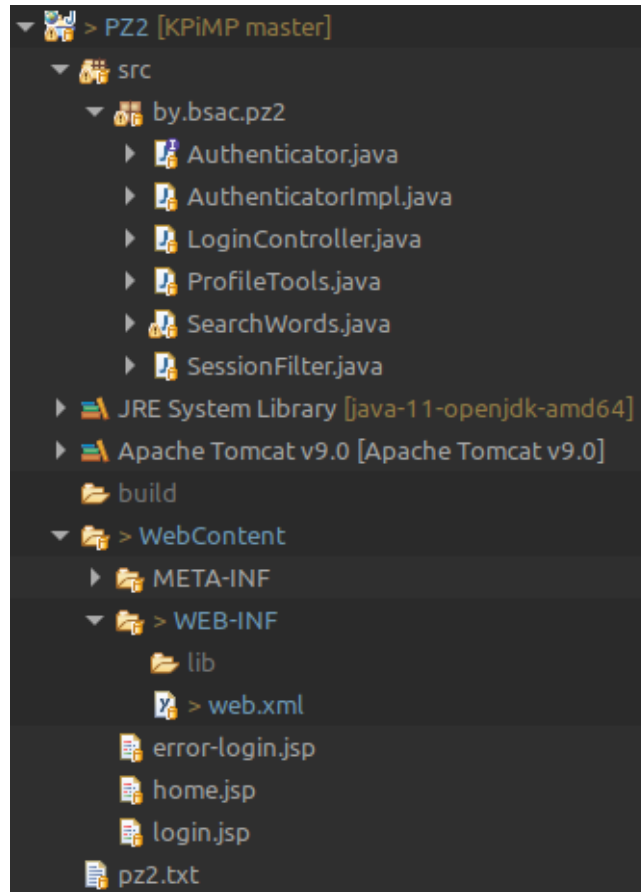
```
55         + "</html>");  
56     }  
57 }
```

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 Апіанне 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 визначиць які метад викликаць 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 метадзе login визначиць метад 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 метад *isLoggedIn* для праверкі статусу карыстальніка.

У лістынг 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      // Падзяляем 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