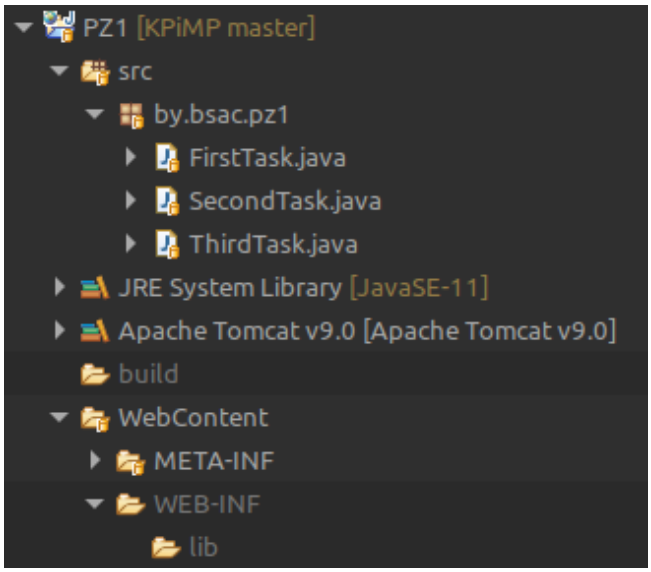
ЗМЕСТ

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 апісваць servlet у web.xml файле)
12 // Спосаб з апісаннем 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 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 - вызначаем 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      // Метад doGet выконваецца, калі на старонку сервлета адпраўляецца HTTP запыт GET
23      protected void doGet(HttpServletRequest request, HttpServletResponse response)
24              throws ServletException, IOException {
25          // Вяртаем HTML-старонку згодна з заданнем
26          // <label> - тэг, адказвае за надпіс перад тэгам <input>
27          // <input> - тэг, для ўводу даных, дзе
28          // type=text (type=submit) - апісвае, што поле мае тэкставы тып (выгляд кнопкі)
29          // name=P1 - вызначаем імя адпаведна заданню.
30          // <form> - тэг, якім задаем, што:
31          // method=POST - даныя, атрыманыя ўнутры тэга form, будуць перададзены, як
32          // параметры POST запыту
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      // Метад doPost выконваецца, калі на старонку сервлета адпраўляецца HTTP запыт POST
47      protected void doPost(HttpServletRequest request, HttpServletResponse response)
48              throws ServletException, IOException {
49          // Атрымліваем з запыту параметр P1 (глядзі метад doGet)
50          String textOfBox = request.getParameter("P1");
51          // Вяртаем HTML-старонку, згодна заданню
52          // Старонка аналагічна першапачатковай (глядзі метад doGet),
53          // за выключэннем таго, што для поля ўвода P1 задаецца значэнне (value),
54          // роўнае параметру P1, паўторанае 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 сервлет, які выдае HTML-старонку з лікам *1* і кнопкай *Submit*. Пасля націскання кнопкі *Submit* сервлет павінен выдаць HTML-старонку з лікам *2* і кнопкай *Submit*. Пасля з

лікам *3* і гэтак далей.

1.4.2 Зыходны код.

Зыходны 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 тага, 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 па 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 па запыту POST
44         // Старонка аналагічная першапачатковай, за выключэннем тага, што
45         // пачатковае значэнне 1 было заменена 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>"
```

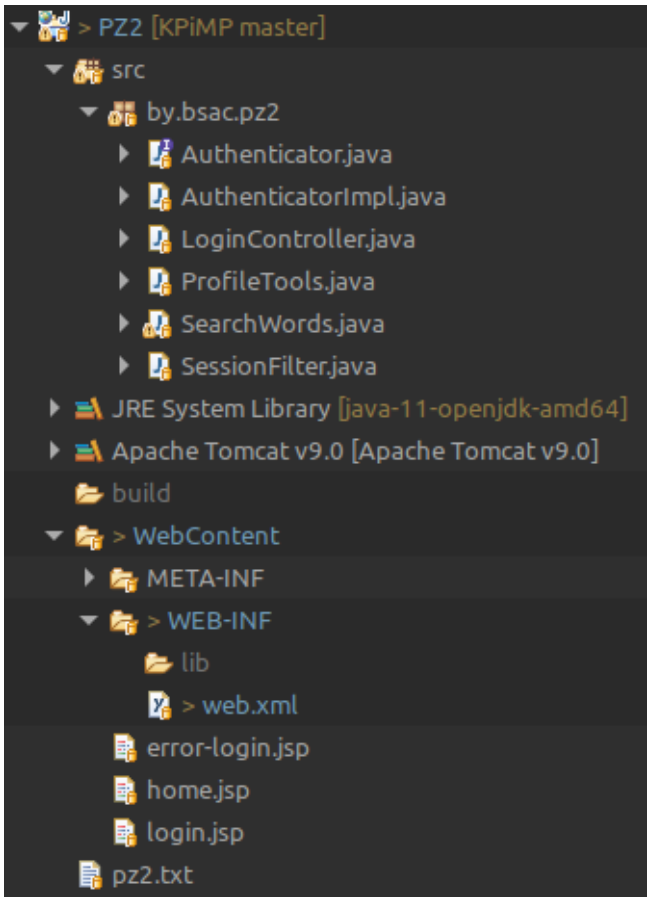
```
55                 + "</html>");
56         }
57   }
```

Лістынг 1.3 – Зыходны 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 паспяховаў 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 – Зыходны 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 метадзе 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 – Зыходны код для home

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

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Login</title>
</head>
<body>
    // POST форма для аўтэнтыфікацыі
    <form action="login" method="post">
        // Захоўвае значэнне login, каб вызначыць які метад выклікаць у doPost класа LoginController
        <input type="hidden" name="authAction" value="login">
        <p>
            Please login by
            <label for="email">Email</label>
            // Захоўвае значэнне email, как у метадзе login вызначыць метад аўтэнтыфікацыі
            // па параметру authType
            <input type="radio" name="authType" value="email" id="email" checked />
            or
            <label for="userName">User name</label>
            <input type="radio" name="authType" value="userName" id="userName" />
            <input type="text" name="loginValue">
        </p>
        <label for="psw">Password</label>
        <input type="password" name="psw" id="psw">
        <p>The characters in a password field are masked (shown as
            asterisks or circles).</p>
        <input type="submit" value="Submit" />
    </form>
</body>
</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 захоўвае імёны атрыбутаў сесіі, якія ўстанаўліваюцца падчас работы вэб-праграмы, а таксама метад *isLoggedIn* для праверкі статусу карыстальніка.

У лістынгу 2.4 прадстаўлены зыходны код класа *ProfileTools*.

```java
package by.bsac.pz2;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;


public class ProfileTools {
    public static String SESSION_LOGGEDIN_ATTRIBUTE_NAME = "user";
    public static String SESSION_LOGGEDIN_ATTRIBUTE_PASSWORD = "password";
    public static String SESSION_LOGGEDIN_ATTRIBUTE_DATA = "loginData";
```

```
11      public static String SESSION_LOGGEDOUT_ATTRIBUTE_DATA = "logoutData";
12
13      // Правяраем 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 лістынгу 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 authenticateByUserEmail(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 лістынгу 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 authenticateByUserEmail(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 прапаноўвае старонку для аўтэнтыфікацыі і адказвае за выкананне аўтэнтыфікацыі: праверка ўведзеных даных з правільнымі, перанапраўленне на адпаведную старонку пры аўтэнтыфікацыі.

У лістынгу 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      // Па запыту GET вяртаем 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 isAuthentificated = false;
48
49          // Атрымліваем параметры, перададзеныя ў POST запыце
50          String password = request.getParameter("psw");
51          String authValue = request.getParameter("loginValue");
52
53          if (authTypeParam.contentEquals("email")) {
54              isAuthentificated = authenticator.authenticateByUserEmail(authValue, password);
55          } else {
56              isAuthentificated = authenticator.authenticateByUserName(authValue, password);
57          }
58
59          // Калі аўтэнтыфікацыя прайшла паспяхова
60          if (isAuthentificated) {
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 вызначаць 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, вяртаем 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              // Атрымліваем бягучую 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              // Запісваем час выхаду з вэб-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

2.5 Зыходны код. web.xml