

Guideline setup

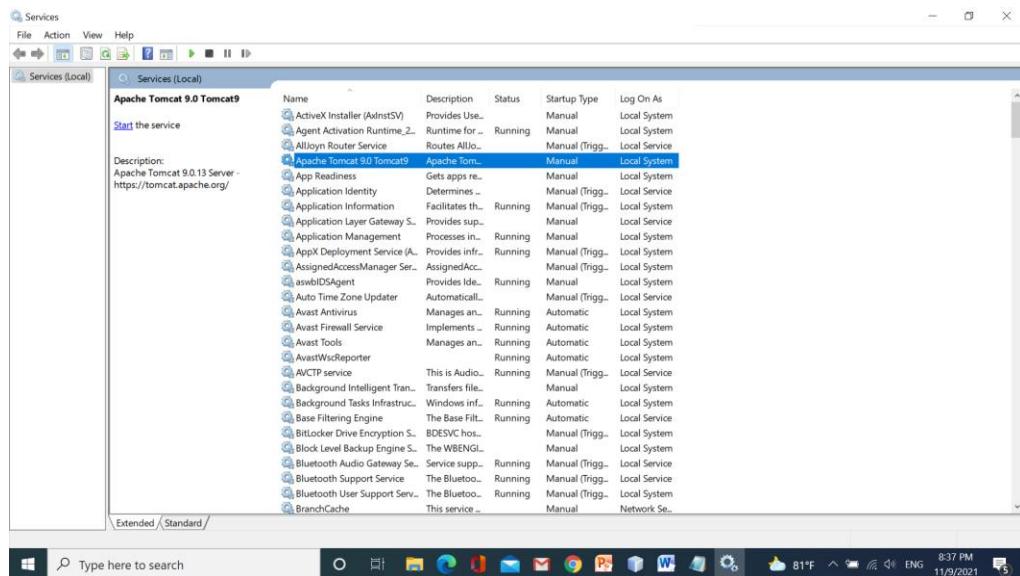
- 1) Download and install Webserver (Tomcat)

[How to download and install Tomcat 9 on windows - YouTube](#)

Or you can use any webserver(for example: GlassFish server).

- 2) Start or Stop webserver by the service window

Open the service window



Start or stop by cmd

[How to Start and Stop Apache Tomcat from the Command Line \(Windows\) | Webucator](#)

- 3) Install JavaEE in the Netbean (use this to create a new web app)

[I don't see "Java Web" in the NetBeans IDE under available projects - Tips \(sentientmindz.com\)](#)

- 4) Download **sqljdbc4.jar** (Use this to connect to the MS SQL Server)

[Download Microsoft JDBC Driver 4.2 for SQL Server from Official Microsoft Download Center](#)

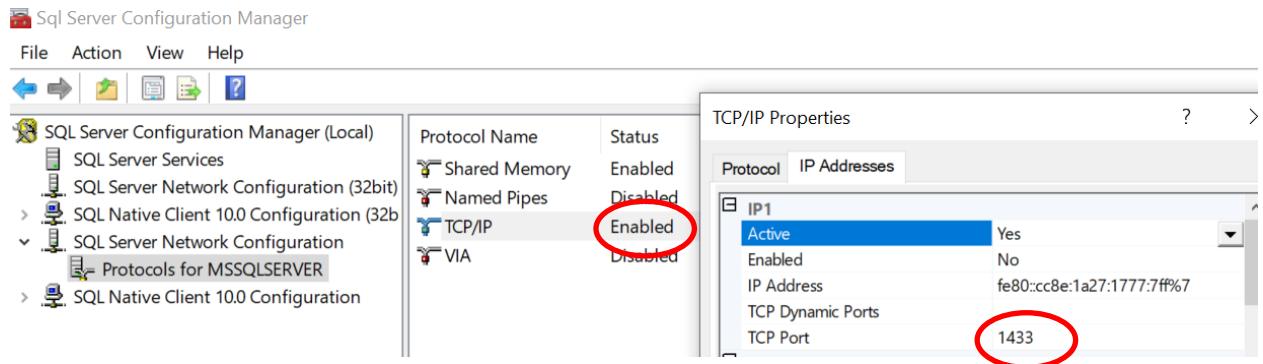
- 5) Add a new user/account to manage your app on the Tomcat server.

Open file “tomcat-users.xml” in path C:\Program Files\Apache Software Foundation\Tomcat 9.0\conf. type like below

```
-->
<user username="admin" password="admin" roles="roles1,admin,manager-script"/>
</tomcat-users>
```

And save (**you must log in to your computer by administrator account**)

- 6) To your web app can connect to the SQL server. Open “SQL Server Configuration Manager” and set up all ports in the SQLServer like the figure



7) Create a database for doing all workshops and assignments. Run the code below

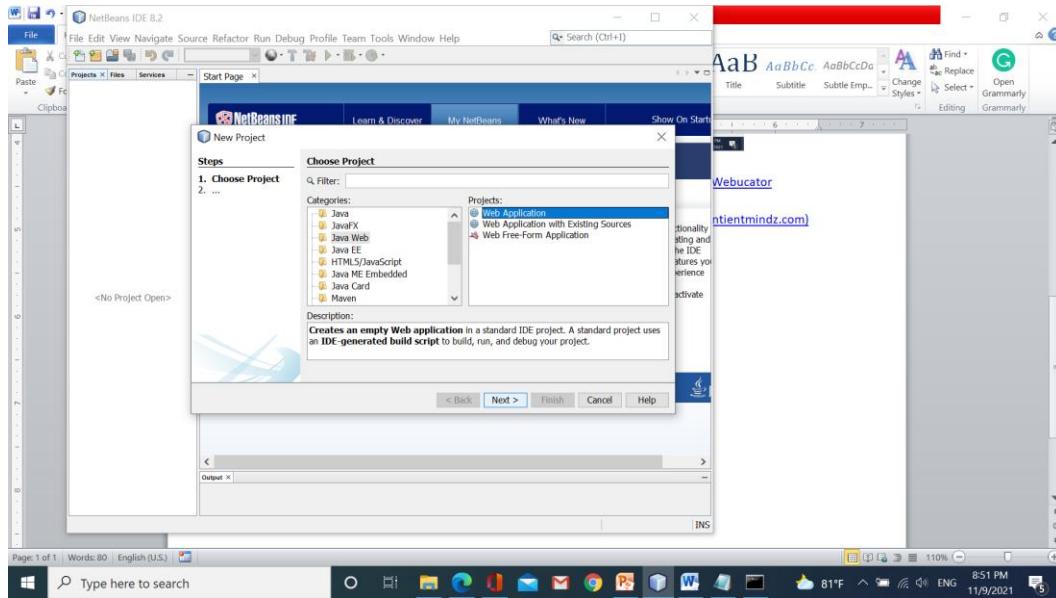
```

create database PlantShop
GO
use PlantShop
GO
create table Accounts(
    accID int identity(1,1)primary key,
    email varchar(30) unique,
    password varchar(30),
    fullname varchar(30),
    phone varchar(12),
    status int check(status =1 or status=0)-- 1:active; 0:inactive
    role int check(role=1 or role=0) --:admin, 0:user
)
GO
create table Categories(
    CateID int identity(1,1) primary key,
    CateName varchar(30)
)
GO
create table Plants
(
    PID int identity(1,1) primary key,
    PName varchar(30),
    price int check(price>=0),
    imgPath varchar(50),
    description text,
    status int, --1:active, 0:inactive
    CateID int foreign key references Categories(CateID)
)
GO
create table Orders(
    OrderID int identity(1,1) primary key,
    OrdDate date,
    shipdate date,
    status int check(status =1 or status=2 or status=3)--1:processing,
    2: completed, 3: cancel
    AccID int foreign key references Accounts(AccID)
)
GO
create table OrderDetails(
    DetailID int identity(1,1) primary key,
    OrderID int foreign key references Orders(OrderID),
    FID int foreign key references Plants(PID),
    quantity int check(quantity>=1)
)

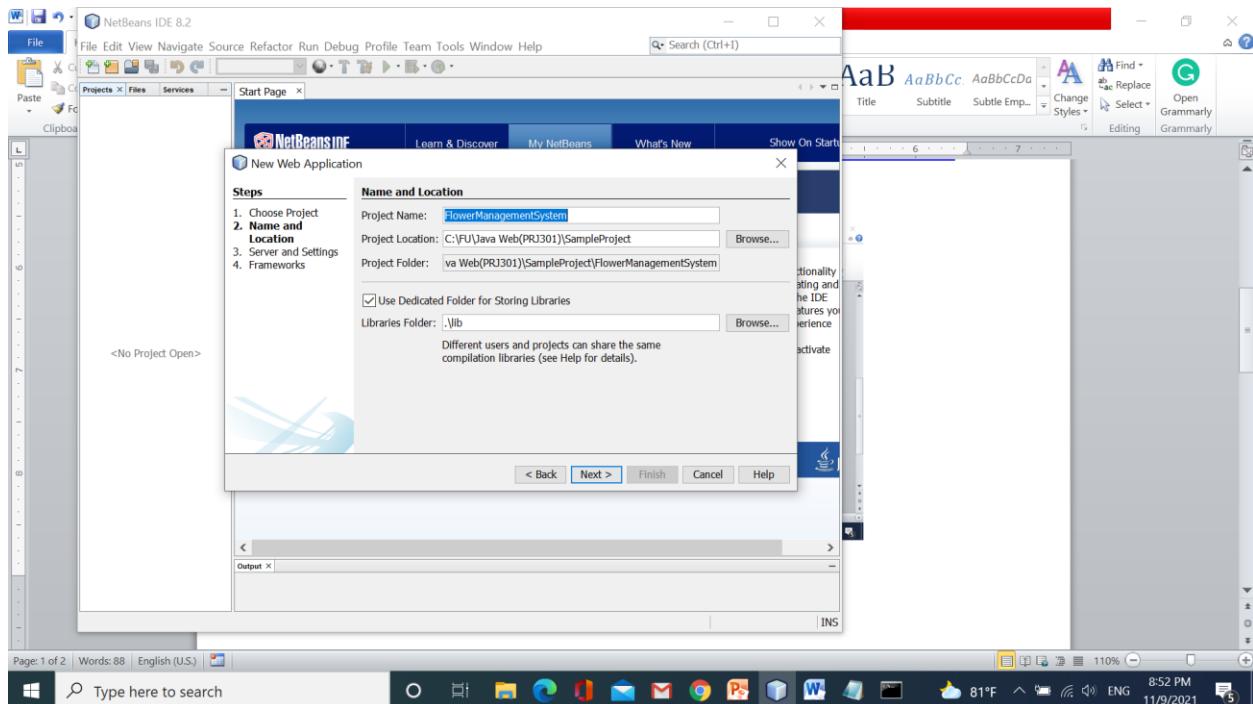
```

Create a new Java Web Project

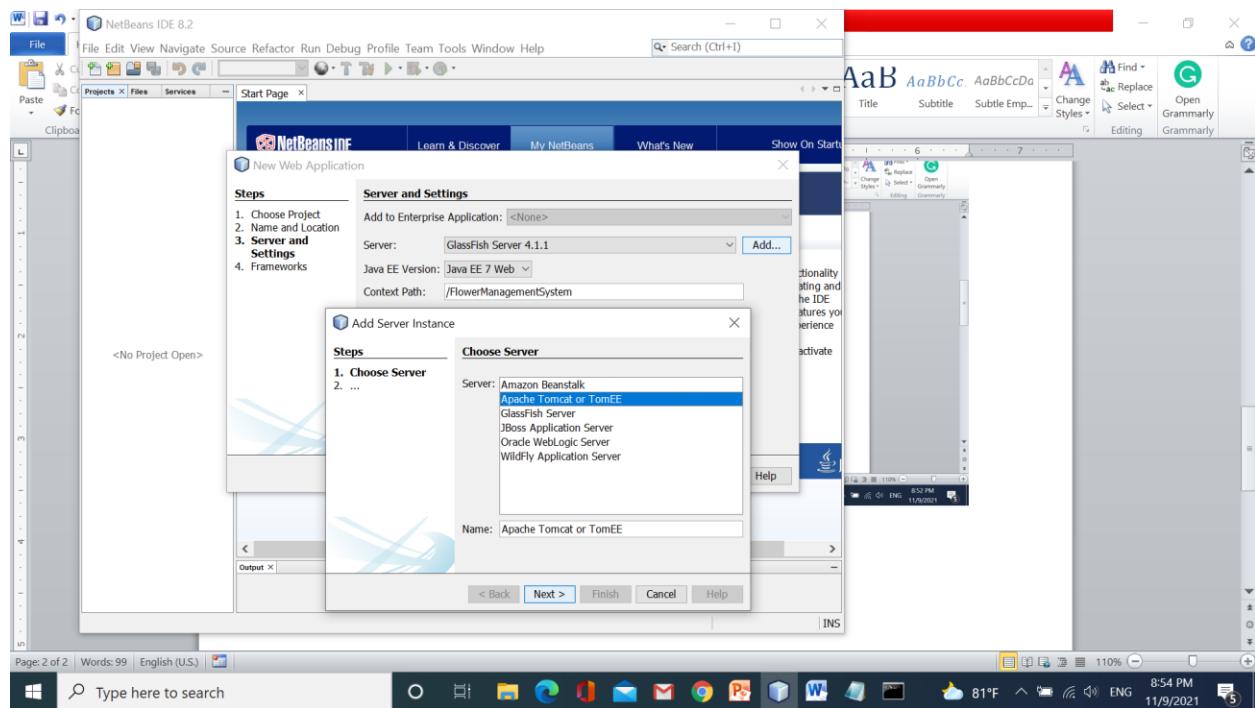
step 1: open Netbean/new Project/Java web/Web application



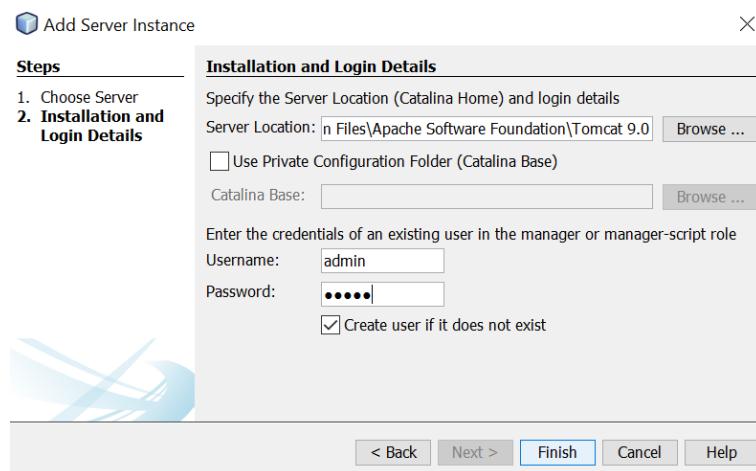
step 2: give a project name, project location and click Next



Step 3: Click Add and choose web server Tomcat, or any webserver, click Next



Step 4: choose the server location and input username="admin", password="admin" (account of the user that is stored in the file tomcat_user.xml)/click finish



Problem: Develop a web app to introduce plants that are orchids, roses,..etc. Users can buy plants from this app, track their orders. Besides, this app supports a dashboard for the admin. The website has three roles: guest, admin, and members.

The common functions for all roles (guest/member) :

- Search Plants by their name or category
- View information of plants
- Buy plants

The guest functions:

- Registrate a new account

The member functions after login:

- log out to the system
- View himself/herself orders (all days, or filter from date to date).
- Cancel the order (only cancel orders whose status is processing)
- Order again (only perform on orders whose status was canceled)
- Change his/her profile

The admin functions after login:

- Manage accounts: view all accounts, block/unblock an account
- View all orders, filter orders between from date and to date, filter orders of any customer.
- Manage plants (view,create,update plants)
- Manage categories: view, create, update
- Log out

Requirement:

- The problem is broken down into some parts that include basic parts and advanced parts
- The basic parts are used to get the workshop's mark. These parts will be guided step by step
- The advanced parts are used to get the assignment's mark. You must perform these parts by yourself.
- Workshops will be evaluated each week. The assignment is evaluated in the 10th week.
- All basic tasks are highlighted, others are advanced tasks
- The practical exam will be performed in the 9th week.

Workshop 1: Learn JDBC, Servlet

#HTML #Form # JDBC #Request Object #Response Object # Servlet

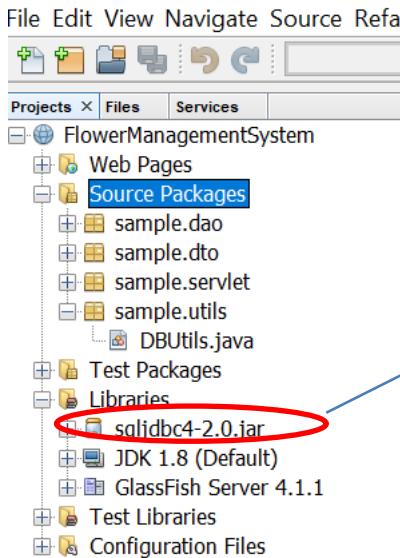
Learning outcome:

- Learn JDBC
- Learn servlet
- Perform the login function

Part 1: Learning JDBC

Step 1: The student creates some packages, files like the figure:

FlowerManagementSystem - NetB



Add this file from downloaded driver

DBUtils.java

FlowerManagementSystem - NetBeans IDE 8.2

The screenshot shows the NetBeans IDE interface with the title "FlowerManagementSystem - NetBeans IDE 8.2". The menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help. The toolbar has standard icons for file operations. The Projects tab is selected, showing a tree view of the project structure. Under "Web Pages", there is a "Source Packages" node which is expanded, showing "sample.dao", "sample.dto", "sample.servlet", and "sample.utils" (containing "DBUtils.java"). Below "Source Packages" is a "Test Packages" node. Under "Libraries", the "sqljdbc4-2.0.jar" file is highlighted with a red circle and has a blue arrow pointing from it to a callout box. Other items in the "Libraries" node include "JDK 1.8 (Default)" and "GlassFish Server 4.1.1". There are also "Test Libraries" and "Configuration Files" nodes.

```
4  * and open the template in the editor.
5  *
6  package sample.utils;
7
8  import java.sql.Connection;
9  import java.sql.DriverManager;
10
11 /**
12  *
13  * @author user
14  */
15 public class DBUtils {
16
17     /**
18      * makeConnection()
19      * Connection cn=null;
20      * String IP="localhost";
21      * String instanceName="DESKTOP-1RCVITI";
22      * String port="1433";
23      * String uid="sa";
24      * String pwd="123456";
25      * String db="PlantShop";
26      * String driver="com.microsoft.sqlserver.jdbc.SQLServerDriver";
27      * String url="jdbc:sqlserver://" +IP+"\\\"+ instanceName+":"+port
28      *           +"\\dataasename"+db+":user"+uid+":password"+pwd;
29      * Class.forName(driver);
30      * cn=DriverManager.getConnection(url);
31      * return cn;
32  }
33 }
```

Change instanceName,username,password based on yours

Account.java

FlowerManagementSystem - NetBeans IDE 8.2

```
4 * and open the template in the editor.
5 */
6 package sample.dto;
7
8 /**
9  * ...
10 */
11 public class Account {
12     private int accID;
13     private String email;
14     private String password;
15     private String fullname;
16     private int status;
17     private String phone;
18     private int role;
19
20     public Account() {...2 lines}
21     public Account(int accID, String email, String password, String fullname, int status, String phone, int role) {...9 lines}
22     public int getAccID() {...3 lines}
23     public void setAccID(int accID) {...3 lines}
24     public String getEmail() {...3 lines}
25     public void setEmail(String email) {...3 lines}
26     public String getPassword() {...3 lines}
27     public void setPassword(String password) {...3 lines}
28     public String getFullscreen() {...3 lines}
29     public void setFullscreen(String fullname) {...3 lines}
30     public int getStatus() {...3 lines}
31     public void setStatus(int status) {...3 lines}
32     public String getPhone() {...3 lines}
33     public void setPhone(String phone) {...3 lines}
34     public int getRole() {...3 lines}
35     public void setRole(int role) {...3 lines}
36 }
37 }
```

Output Notifications

11:11 INS

AccountDAO.java

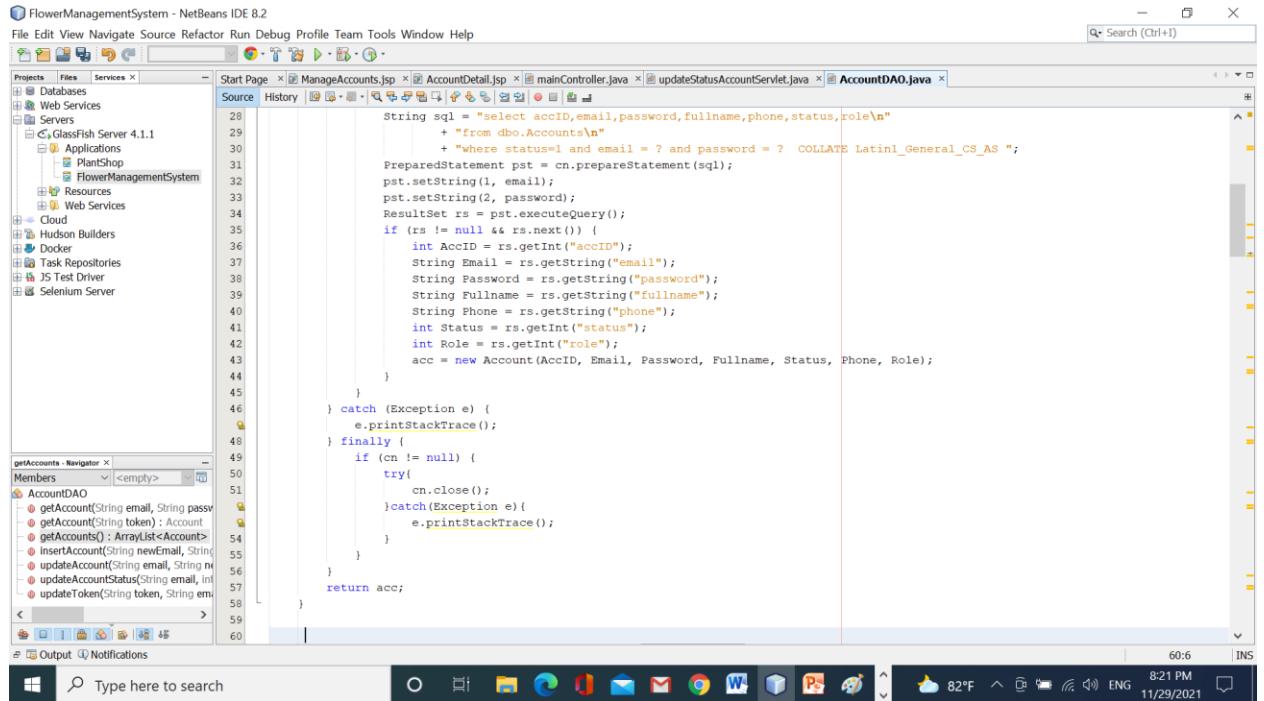
You write the method “getAccount(String email, String password)” to return the account that equals the given parameters.

NetBeans IDE 8.2

```
public static Account getAccount(String email, String password) {
    Connection cn = null;
    Account acc = null;
    try {
        cn = DBUtils.makeConnection();
        if (cn != null) {
            String sql = "select accID,email,password,fullname,phone,status,role\n"
                    + "from dbo.Accounts\n"
                    + "where status=1 and email = ? and password = ? COLLATE Latin1_General_CS_AS ";
            PreparedStatement pst = cn.prepareStatement(sql);
            pst.setString(1, email);
            pst.setString(2, password);
            ResultSet rs = pst.executeQuery();
            if (rs != null && rs.next()) {
                int AccID = rs.getInt("accID");
                String Email = rs.getString("email");
                String Password = rs.getString("password");
                String Fullname = rs.getString("fullname");
                String Phone = rs.getString("phone");
                int Status = rs.getInt("status");
                int Role = rs.getInt("role");
                acc = new Account(AccID, Email, Password, Fullname, Status, Phone, Role);
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        if (cn != null) {
            try {
                cn.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
```

Output Notifications

30:75 INS



Step 2: to test the code, we create a file named “testConnection.java” in the package “sample.dto”. In the table Accounts, you must input sample tuples like

DESKTOP-IRCVITT....p - dbo.Accounts DB_FlowerShop.sql...lantShop (sa (53))							
	accID	email	password	fullname	phone	status	role
▶	1	test@gmail.com	test	test	123456	1	0
*	2	admin@gmail.com	admin	Administrator	123456	1	1
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

and run this file with input data email= “test@gmail.com”, password=“test”.

```

package sample.dto;

import java.util.ArrayList;
import sample.dao.AccountDAO;
/*...4 lines */
public class TestConnection {
    public static void main(String[] args) {
        //test login
        Account acc=AccountDAO.getAccount("test@gmail.com", "test");
        if(acc!=null){
            if(acc.getRole()==1)
                System.out.println("i am an admin");
            else
                System.out.println("i am a user");
        }
        else System.out.println("login fail");
    }
}

```

So, you have learned about JDBC (just a little, ☺).

The advanced challenge:

- 1) In the file AccountDAO, write the function

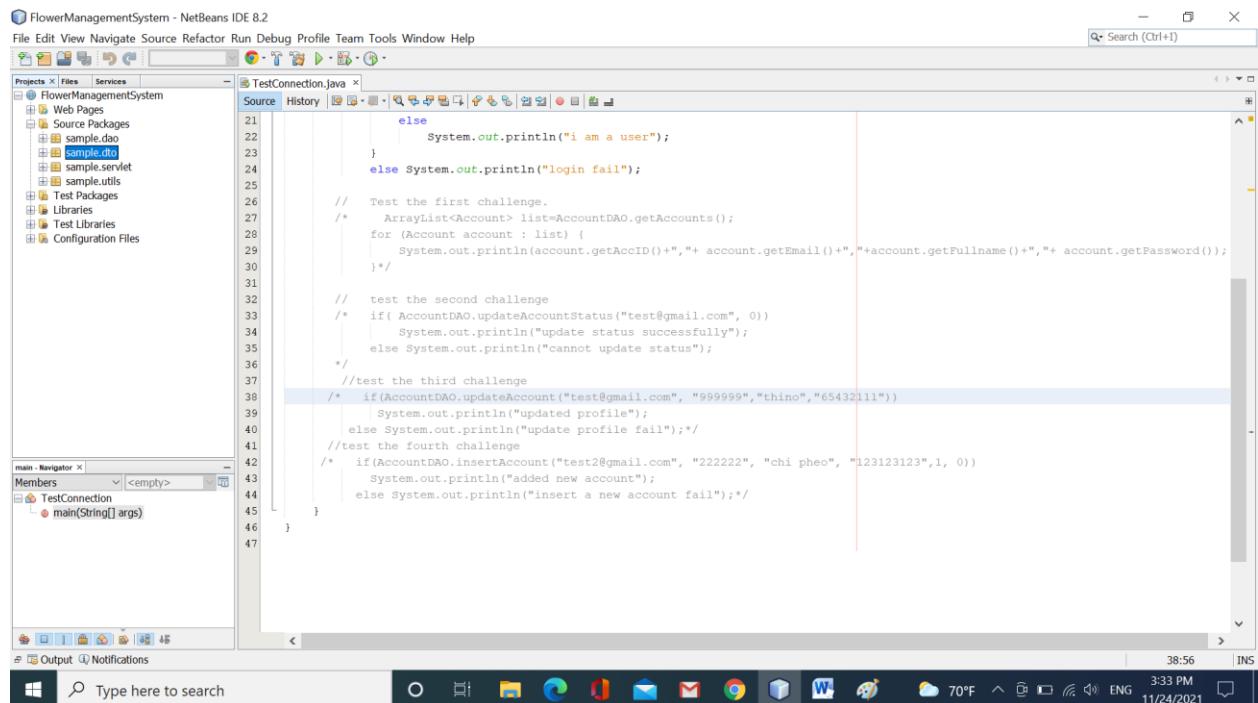
`public static ArrayList<Account> getAccounts(){...}`. This function returns the list of accounts that are got from the table Account. You test this function in the main method.

- 2) In the file AccountDAO, write a function named `public static boolean updateAccountStatus(String email, int status)`. This function will update the account's status based on the given email and return true/false

- 3) In the file AccountDAO, write a function named `public static boolean updateAccount(String email, String newPassword, String newFullscreen, String newPhone)`. This function is used to update the full name, phone, password of the account based on the given email and returns true/false.

- 4) In the file AccountDAO, write a function named `public static boolean insertAccount(String newEmail, String newPassword, String newFullscreen, String newPhone, int newStatus,int newRole)`. This function is used to insert a new account into the table Accounts and return true/false.

You can use the sample to test the methods above

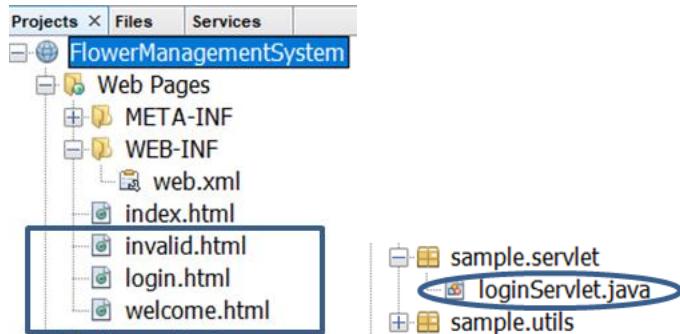


The screenshot shows the NetBeans IDE interface with the following details:

- Title Bar:** FlowerManagementSystem - NetBeans IDE 8.2
- Menu Bar:** File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
- Toolbar:** Standard NetBeans toolbar with icons for file operations.
- Project Explorer:** Shows the project structure under "FlowerManagementSystem".
- Code Editor:** The main window displays the `TestConnection.java` file. The code contains several sections of Java code, including:
 - A user login verification section.
 - A section for testing the first challenge, which prints account details if successful.
 - A section for testing the second challenge, which updates account status.
 - A section for testing the third challenge, which updates account profile.
 - A section for testing the fourth challenge, which inserts a new account.
- Navigator:** Shows the `main(String[] args)` method in the `TestConnection` class.
- Status Bar:** Shows system information like time (38:56), battery level (70%), and date (11/24/2021).

Part 2: perform the login function.

Step 1: You create three files HTML in the folder “Web Pages” and one file `loginServlet` in the package “sample.servlet” like the figure:



`login.html`

The image shows the 'login.html' file in a code editor. The code is as follows:

```
7 <html>
8   <head>
9     <title>TODO supply a title</title>
10    <meta charset="UTF-8">
11    <meta name="viewport" content="width=device-width, initial-scale=1.0">
12  </head>
13  <body>
14    <form action="loginServlet" method="post">
15      <table>
16        <tr>
17          <td>email</td>
18          <td><input type="text" name="txtemail"></td></tr>
19        <tr><td>password</td>
20          <td><input type="password" name="txtpassword"></td></tr>
21        <tr><td colspan="2"><input type="submit" value="login"></td></tr>
22      </table>
23    </form>
24  </body>
25 </html>
```

Annotations in the code editor highlight several parts of the code with blue ovals:

- A blue oval surrounds the `action="loginServlet"` attribute in the `<form>` tag.
- A blue oval surrounds the `name="txtemail"` attribute in the first `<input>` tag inside the `<tr>`.
- A blue oval surrounds the `name="txtpassword"` attribute in the second `<input>` tag inside the `<tr>`.
- A blue oval surrounds the `value="login"` attribute in the `<input type="submit" value="login">` tag.

`invalid.html`

The screenshot shows a code editor interface with the following details:

- Title Bar:** The title bar displays "Start Page" and "invalid.html".
- Toolbar:** The toolbar includes icons for Source, History, and various file operations like Open, Save, Print, and Find.
- Code Editor Area:** The main area shows the following HTML code:

```
<!DOCTYPE html>
<!-- To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
--&gt;
&lt;html&gt;
    &lt;head&gt;
        &lt;title&gt;TODO supply a title&lt;/title&gt;
        &lt;meta charset="UTF-8"&gt;
        &lt;meta name="viewport" content="width=device-width, initial-scale=1.0" /&gt;
    &lt;/head&gt;
    &lt;body&gt;
        &lt;p&gt;invalid email or password&lt;/p&gt;
        &lt;p&gt;&lt;a href="login.html"&gt;please, login again&lt;/a&gt;&lt;/p&gt;
    &lt;/body&gt;
&lt;/html&gt;</pre>
```
- Status Bar:** A status bar at the bottom indicates "File is saved" and shows the file path "C:\Users\Public\Documents\NetBeansProjects\MyProject\src\invalid.html".

Welcome.html

loginServlet.java

right-click in the package “sample.servlet”/new servlet/finish.

Steps

1. Choose File Type
2. Name and Location
3. Configure Servlet Deployment

Configure Servlet Deployment

Register the Servlet with the application by giving the Servlet an internal name (Servlet Name). Then specify patterns that identify the URLs that invoke the Servlet. Separate multiple patterns with commas.

Add information to deployment descriptor (web.xml)

Class Name:	sample.servlet.loginServlet
Servlet Name:	loginServlet
URL Pattern(s):	/loginServlet
Initialization Parameters:	
Name	Value
New	
Edit	

```
Start Page × loginServlet.java ×
Source History | Back Forward Stop Refresh Reload Stop Refresh Reload
32     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
33         throws ServletException, IOException {
34     response.setContentType("text/html;charset=UTF-8");
35     try (PrintWriter out = response.getWriter()) {
36         /* TODO output your page here. You may use following sample code. */
37         String email=request.getParameter("txtemail");
38         String password=request.getParameter("txtpassword");
39         Account acc=null;
40         try {
41             acc=AccountDAO.getAccount(email, password);
42             if(acc!=null){
43                 //admin
44                 if(acc.getRole()==1){
45                     //chuyen qua admin home page
46
47                 }
48                 //user
49                 else{
50                     //chuyen qua welcome page
51                     response.sendRedirect("welcome.html");
52
53                 }
54             }
55             else response.sendRedirect("invalid.html");
56         } catch (Exception e) {
57             e.printStackTrace();
58         }
59     }
60 }
61 }
```

Open the file web.xml to set up the first page to run.

Start Page |  web.xml | 

Source General Servlets Filters Pages References Security History  Welcome Files 

Welcome Files

Welcome Files: 

Use comma(,) to separate multiple welcome files.

[Go To Source\(s\)](#)

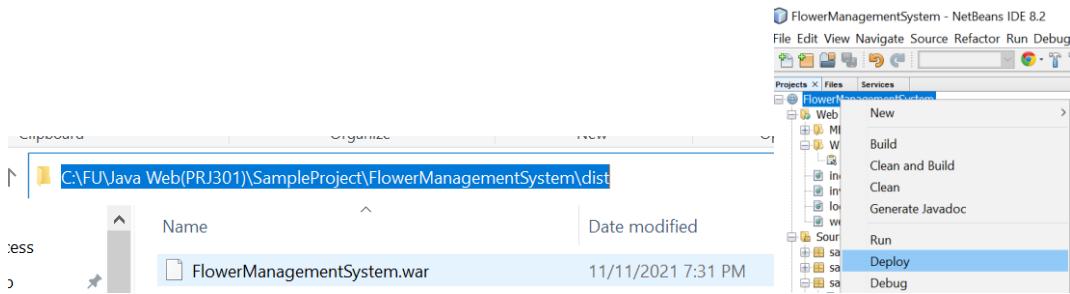
The diagram shows the `web.xml` configuration file with several annotations:

- A blue bracket on the right side spans from the end of the `<session-timeout>` element to the end of the `</web-app>` element.
- A blue arrow points from the text "Annotations for this page" to the `<welcome-file>` element at line 17.

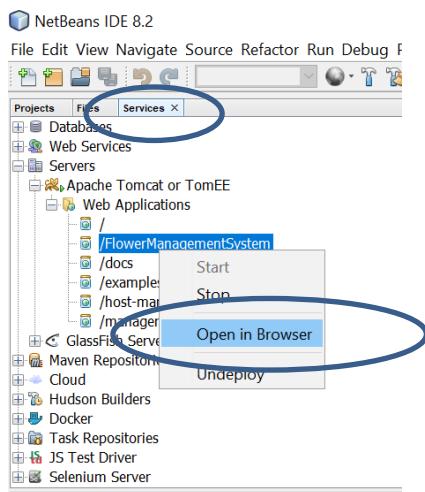
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
    <servlet>
        <servlet-name>loginServlet</servlet-name>
        <servlet-class>sample.servlet.loginServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>loginServlet</servlet-name>
        <url-pattern>/loginServlet</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>30</session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>login.html</welcome-file>
    </welcome-file-list>
</web-app>
```

Now, you click “clean and build”/right-click on the project/deploy.

Deploy your app to copy *.war to the webserver. From now, you can run the web without needing Netbean.



And then, click open in the browser



The result:

The left screenshot shows a browser window with the URL <http://localhost:8080/FlowerManagementSystem>. The page contains a login form with fields for 'email' (value: test@gmail.com) and 'password' (value: ****). Below the form is a 'login' button. The right screenshot shows the same browser window after logging in, displaying the text 'welcome'.



invalid email or password

[please, login again](#)

Congratulations!!!! you have finished workshop 1 ☺ ☺

Workshop 2: Learn JDBC , Servlet

#HTML #Form # JDBC # Servlet

Learning outcome:

- Students can create the register form
- Students use technologies of JDBC
- Students can create servlets to insert a new account into the database.

Part 1: how to create a registration form

Step 1: Students edit the file index.html like the code below:

```
<html>
  <head>
    <title>TODO supply a title</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="mycss.css" type="text/css" />
  </head>
  <body>
    <header>
      <nav>
        <ul>
          <li><a href=""></a> </li>
          <li><a href="index.html">Home</a></li>
          <li><a href="registration.html">Register</a></li>
          <li><a href="login.html" >Login</a></li>
          <li><form action="searchServlet" method="post" class="formsearch">
            <input type="text" name="txtsearch">
            <select name="searchby">
              <option value="byname">by name</option><option value="bycate">by category</option>
            </select>
            <input type="submit" value="search" name="action" >
          </form></li>
        </ul>
      </nav>
    </header>
    <section>

    </section>
    <footer>
      <p></p>
    </footer>
  </body>
```

➤ All pages contain header tag the same, they are differences in the section tag.

Step 2: Student create a new file named “mycss.css”

```
*{  
    box-sizing:border-box;  
}  
html{ font-family: arial;  
    font-size: 12px;  
}  
header,section,footer,nav,ul { float: left; width: 100%;}  
header, nav, footer{ height: 100px;  
    background-color: #e1f5eb;  
}  
section { height: auto;}  
header li{width:10%;  
    list-style-type: none;  
    text-align: center;  
    float:left;  
}  
a { color: #006666;  
    font-size: 1.5em;  
    text-decoration: none;  
}  
  
header li:last-child { width:30%; float:right;}  
#logo { width: 100px; height: 100px; }  
  
.form table{width:50%;}  
  
.product {width: 20%; height: auto; float:left; margin: 5%;}  
  
.order { border: 1px solid #cccccc; width:100%; margin: 1%;}  
.order tr:first-child{ width:100%; background-color:#e1eee7 ; color: #006666;}  
.order td{ color:black; width: 15%;}  
  
.plantimg { width:100px; height: 100px;}  
  
.shopping { width:70%;}
```

Step 3: Create a new file named “registration.html”

```

16     <nav>
17         <ul>
18             <li><a href=""></a> </li>
19             <li><a href="index.html">Home</a></li>
20             <li><a href="registration.html">Register</a></li>
21             <li><a href="login.html" >Login</a></li>
22             <li><form action="searchServlet" method="post" class="form" ')>
23                 <input type="text" name="txtsearch">
24                 <select name="searchby"><option>by name</option>
25                     <option>by category</option>
26                 </select>
27                 <input type="submit" value="search" name="action" >
28             </form></li>
29         </ul>
30     </nav>
31 </header>
32 <section>
33     <form action="registerServlet" method="post" class="form" ')>
34         <h1>Register</h1>
35         <table>
36             <tr><td>email</td><td><input type="text" name="txtemail" required=""></td></tr>
37             <tr><td>full name</td><td><input type="text" name="txtfullname" required=""></td></tr>
38             <tr><td>password</td><td><input type="password" name="txtpassword" required=""></td></tr>
39             <tr><td>phone</td><td><input type="text" name="txtphone"></td></tr>
40             <tr><td colspan="2"><input type="submit" value="create"></td></tr>
41         </table>
42     </form>
43 </section>
44 <footer>
45     <p></p>
46 </footer>
47 </body>
48 </html>

```

Step 4: create a new file named “registerServlet.java” in the package “sample.servlet”

```

24     * methods.
25
26     * @param request servlet request
27     * @param response servlet response
28     * @throws ServletException if a servlet-specific error occurs
29     * @throws IOException if an I/O error occurs
30     */
31    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
32        throws ServletException, IOException {
33        response.setContentType("text/html;charset=UTF-8");
34        try (PrintWriter out = response.getWriter()) {
35            /* TODO output your page here. You may use following sample code. */
36            String email=request.getParameter("txtemail");
37            String fullname=request.getParameter("txtfullname");
38            String password=request.getParameter("txtpassword");
39            String phone=request.getParameter("txtphone");
40            int status=1; //default a new account's status is 1
41            int role=0;//default a new account's role is 0
42            if(AccountDAO.insertAccount(email, password, fullname, phone, status, role)){
43                response.sendRedirect("index.html");
44            }
45            else response.sendRedirect("errorpage.html");
46        }
47    }
48
49    HttpServlet methods. Click on the + sign on the left to edit the code.
50
51
52
53
54
55
56
57
58
59

```

Step 5: Create a file named “errorpage.html”

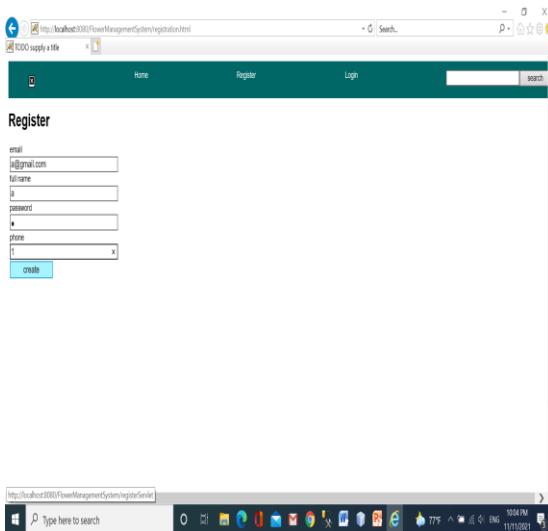
```

1 <!DOCTYPE html>
2 <!--
3 To change this license header, choose License Headers in Project Properties.
4 To change this template file, choose Tools | Templates
5 and open the template in the editor.
6 -->
7 <html>
8   <head>
9     <title>TODO supply a title</title>
10    <meta charset="UTF-8">
11    <meta name="viewport" content="width=device-width, initial-scale=1.0">
12  </head>
13  <body>
14    <p>Something wrong, please click <a href="index.html">here</a> to return index page</p>
15  </body>
16 </html>
17

```

Step 6: run your app, click the link “register”

Sample



→ After running, the web returns the index page.

Part 2: do the search function.

Step 1: input the table Categories and Plants some tuples.

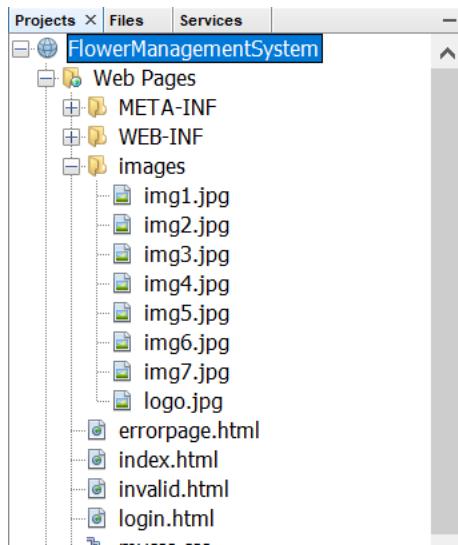
DESKTOP-IRCVITT....- dbo.Categories

	CateID	CateName
▶	1	orchid
	2	roses
	3	others
*	NULL	NULL

DESKTOP-IRCVITT....hop - dbo.Plants DB_FlowerShop.sql...ntShop (sa (51))*

	PID	PName	price	imgPath	description	status	CateID
▶	1	vanda	100	images/img1.jpg	this is a vanda o...	1	1
	2	white rose	90	images/img4.jpg	this is a rose	1	2
	5	lan h? di?p	70	images/img2.jpg	hoa lan	1	1
	6	lan h?ai	140	images/img3.jpg	hoa lan	1	1
	7	c?y hoa h?ng cam	200	images/img5.jpg	hoa hong	1	2
	8	monstera	80	images/img6.jpg	cay la kieng	1	3
	9	var monstera	400	images/img7.jpg	cay la kieng	1	3
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Step 2: Prepare your images



Step 3: In the package sample.dto, create a new file named “Plant.java”

FlowerManagementSystem - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects Services

Plant.java

```

4  * and open the template in the editor.
5  */
6 package sample.dao;
7 /**
8  * ...
9  */
10 public class Plant {
11     private int id;
12     private String name;
13     private int price;
14     private String imgpath;
15     private String description;
16     private int status;
17     private int cateid;
18     private String catename;
19     public Plant() {...2 lines}
20     public Plant(int id, String name, int price, String imgpath, String description, int status, int cateid, String catename) {...10 lines}
21     public String getName() {...3 lines}
22     public void setName(String name) {...3 lines}
23     public int getPrice() {...3 lines}
24     public void setPrice(int price) {...3 lines}
25     public String getImgpath() {...3 lines}
26     public void setImgpath(String imgpath) {...3 lines}
27     public String getDescription() {...3 lines}
28     public void setDescription(String description) {...3 lines}
29     public int getStatus() {...3 lines}
30     public void setStatus(int status) {...3 lines}
31     public int getCateid() {...3 lines}
32     public void setCateid(int cateid) {...3 lines}
33     public String getCatename() {...3 lines}
34     public void setCatename(String catename) {...3 lines}
35     public int getId() {...3 lines}
36     public void setId(int id) {...3 lines}
37 }

```

Navigator Members

- setDescription(String description)
- setId(int id)
- setImgpath(String imgpath)
- setName(String name)
- setPrice(int price)
- setStatus(int status)
- cateid : int
- catename : String
- description : String

Output Notifications

Type here to search

6:20 3:57 PM 68°F ENG 11/24/2021

Step 4: In the package “sample.dao”, create a new file named “PlantDAO.java”.

FlowerManagementSystem - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects Files Services

PlantDAO.java

```

13 import sample.utils.DBUtils;
14 /**
15  * ...
16 */
17 public class PlantDAO {
18     public static ArrayList<Plant> getPlants(String keyword, String searchby) {
19         ArrayList<Plant> list = new ArrayList<>();
20         Connection cn=null;
21         try {
22             cn=DBUtils.makeConnection();
23             if(cn!=null && searchby!=null) {
24                 String sql = "select PID,FName,price,imgPath,description,status,Plants.CateID as 'CateID',Catename\n"
25                     + "from Plants join Categories on Plants.CateID=Categories.CateID\n";
26                 if(searchby.equalsIgnoreCase("byname"))
27                     sql+= "where Plants.FName like ?";
28                 else sql+= "where Catename like ?";
29                 PreparedStatement pst=cn.prepareStatement(sql);
30                 pst.setString(1, "%" + keyword + "%");
31                 ResultSet rs=pst.executeQuery();
32                 if(rs!=null) {
33                     while(rs.next()) {
34                         int id=rs.getInt("PID");
35                         String name=rs.getString("FName");
36                         int price=rs.getInt("price");
37                         String imgpath=rs.getString("imgPath");
38                         String description=rs.getString("description");
39                         int status=rs.getInt("status");
40                         int cateid=rs.getInt("CateID");
41                         String catename=rs.getString("Catename");
42                         Plant plant=new Plant(id, name, price, imgpath, description, status, cateid, catename);
43                         list.add(plant);
44                     }
45                 }
46             } catch (Exception e) {
47             e.printStackTrace();
48         }
49     }
50 }

```

Navigator Members

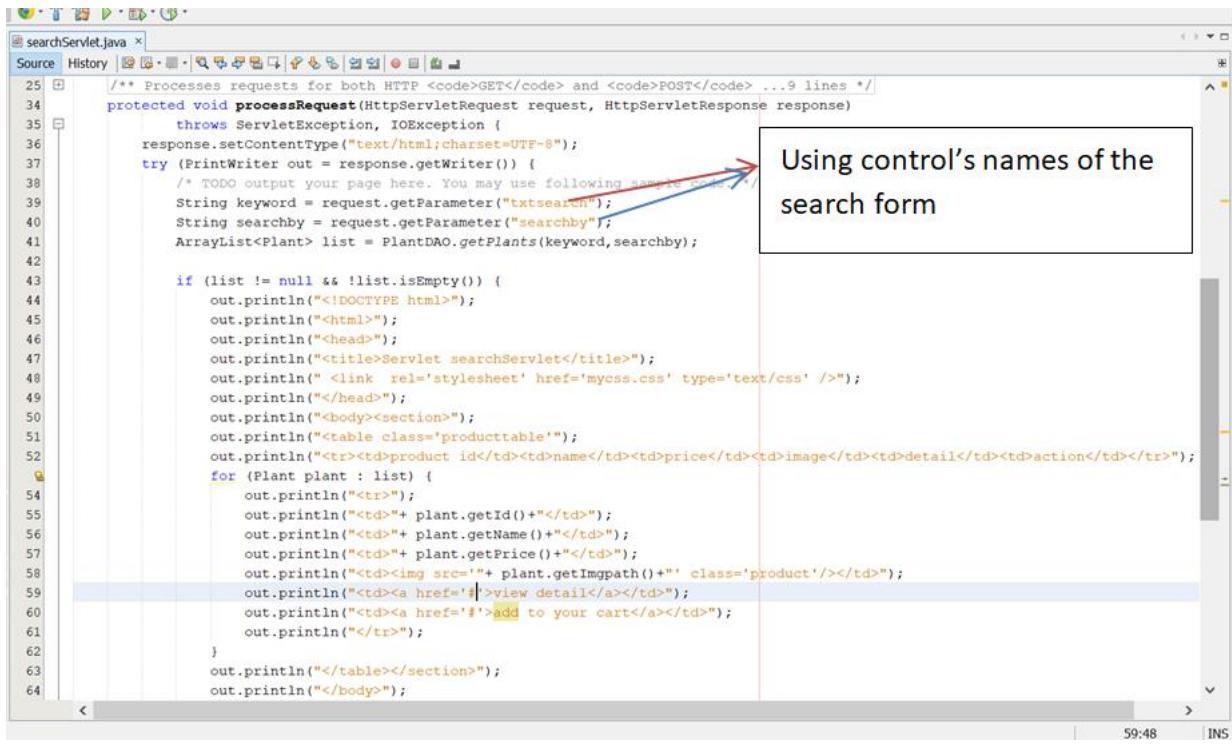
- PlantDAO
- getPlants(String keyword, String searchby)

Output Notifications

Type here to search

13:29 3:58 PM 68°F ENG 11/24/2021

Step 5: In the package “sample.servlet”, create a new file named “searchServlet.java”



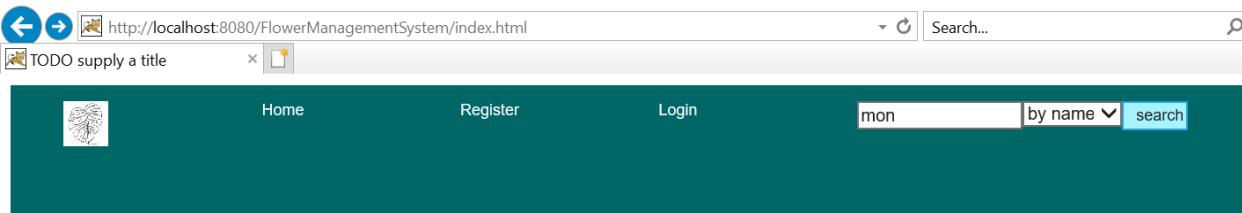
```

25 	/** Processes requests for both HTTP <code>GET</code> and <code>POST</code> ...9 lines */
26 	protected void processRequest(HttpServletRequest request, HttpServletResponse response)
27 		throws ServletException, IOException {
28 		response.setContentType("text/html;charset=UTF-8");
29 		try (PrintWriter out = response.getWriter()) {
30 			/* TODO output your page here. You may use following sample code */
31 			String keyword = request.getParameter("txtsearch");
32 			String searchby = request.getParameter("searchby");
33 			ArrayList<Plant> list = PlantDAO.getPlants(keyword,searchby);
34
35 			if (list != null && !list.isEmpty()) {
36 				out.println("<!DOCTYPE html>");
37 				out.println("<html>");
38 				out.println("<head>");
39 				out.println("<title>Servlet searchServlet</title>");
40 				out.println("<link rel='stylesheet' href='mycss.css' type='text/css' />");
41 				out.println("</head>");
42 				out.println("<body><section>");
43 				out.println("<table class='producttable'>");
44 				out.println("<tr><td>product id</td><td>name</td><td>price</td><td>image</td><td>detail</td><td>action</td></tr>");
45 				for (Plant plant : list) {
46 					out.println("<tr>");
47 					out.println("<td>" + plant.getId() + "</td>");
48 					out.println("<td>" + plant.getName() + "</td>");
49 					out.println("<td>" + plant.getPrice() + "</td>");
50 					out.println("<td><img src='" + plant.getImgpath() + "' class='product'/></td>");
51 					out.println("<td><a href='#!'>view detail</a></td>");
52 					out.println("<td><a href='#!'>add to your cart</a></td>");
53 					out.println("</tr>");
54 				}
55 				out.println("</table></section>");
56 				out.println("</body>");
57 			}
58
59
60
61
62
63
64

```

Using control's names of the search form

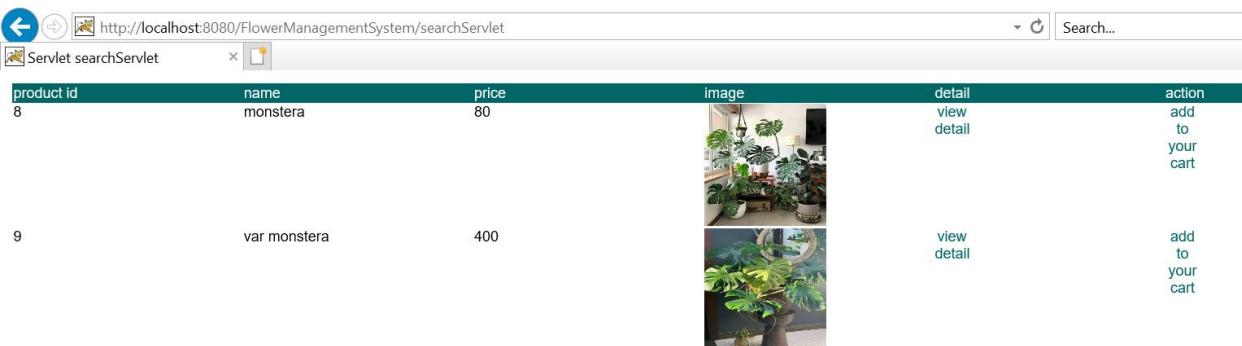
Step 6: run your app to get the result



http://localhost:8080/FlowerManagementSystem/index.html

TODO supply a title

Home Register Login mon by name search



Servlet searchServlet

product id	name	price	image	detail	action
8	monstera	80		view detail	add to your cart
9	var monstera	400		view detail	add to your cart

Congratulations!!!! you have finished workshop 2 ☺ ☺

Workshop 3: transfer from resources to others with/without data/objects?

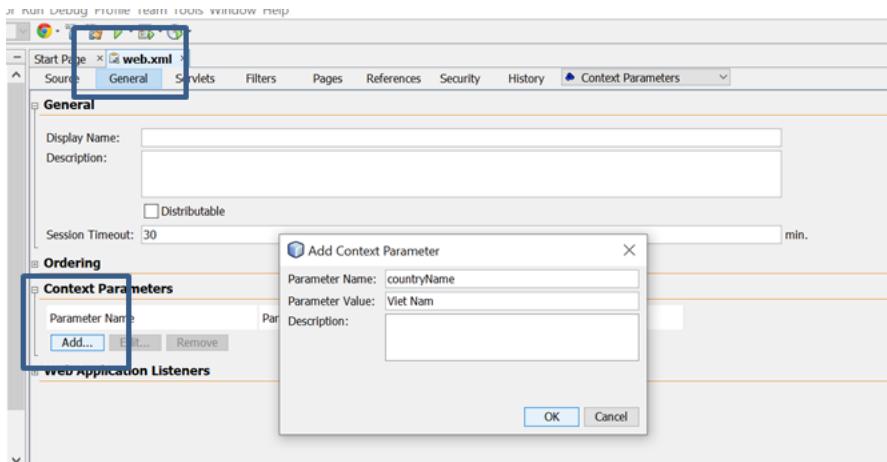
#initial parameters #scope # deploy # transfer

Learning outcome:

- Understand context and servlet initial parameters and how to define and access them from using web.xml file
- Understand the request, session, context scope term, and use memory suitably for storing data
- Understand transfer data from the resource to other resources

Part 1: Define the initial parameter in the context(web app), and display it in all servlets

Step 1: Open the file “web.xml”, do the followings



```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/:
    <context-param>
        <param-name>countryName</param-name>
        <param-value>Viet Nam</param-value>
    </context-param>
    <servlet>
```

Step 2: display the country name in the servlet “searchServlet”, or any servlet(if any)

NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects Files Services x Start Page x searchServlet.java x

```

40     String keyword = request.getParameter("txtsearch");
41     String searchby = request.getParameter("searchby");
42     ArrayList<Plant> list = PlantDAO.getPlants(keyword,searchby);
43
44     if (list != null && !list.isEmpty()) {
45         out.println("<!DOCTYPE html>");
46         out.println("<html>");
47         out.println("<head>");
48         out.println("<title>Servlet searchServlet</title>");
49         out.println(" <link rel='stylesheet' href='mycss.css' type='text/css' />");
50         out.println("</head>");
51         out.println("<body><section>");
52         ServletContext context=getServletContext();
53         String tmp=context.getInitParameter("countryName");
54         out.println("<p>The website is deploying in " + tmp);
55         out.println("<table class='producttable'>");
56         out.println("<tr><td>product id</td><td>name</td><td>price</td><td>image</td><td>detail</td><td>action</td></tr>");
57         for (Plant plant : list) {
58             out.println("<tr>");
59             out.println(" <td>" + plant.getId() + "</td>");
60             out.println(" <td>" + plant.getName() + "</td>");
61             out.println(" <td>" + plant.getPrice() + "</td>");
62             out.println(" <td><img src='"+ plant.getImgpath() +"' class='product'/></td>");
63             out.println(" <td><a href='#!'>view detail</a></td>");
64             out.println(" <td><a href='#!'>add to your cart</a></td>");
65             out.println("</tr>");
66         }
67         out.println("</table></section>");
68         out.println("</body>");
69         out.println("</html>");
70     } else out.println("nothing");
71 }

```

processRequest - Navigator x Members <empty>

searchServlet :: HttpServlet

- doGet(HttpServletRequest request, HttpServletResponse response)
- doPost(HttpServletRequest request, HttpServletResponse response)
- getServletInfo() : String
- processRequest(HttpServletRequest request, HttpServletResponse response)

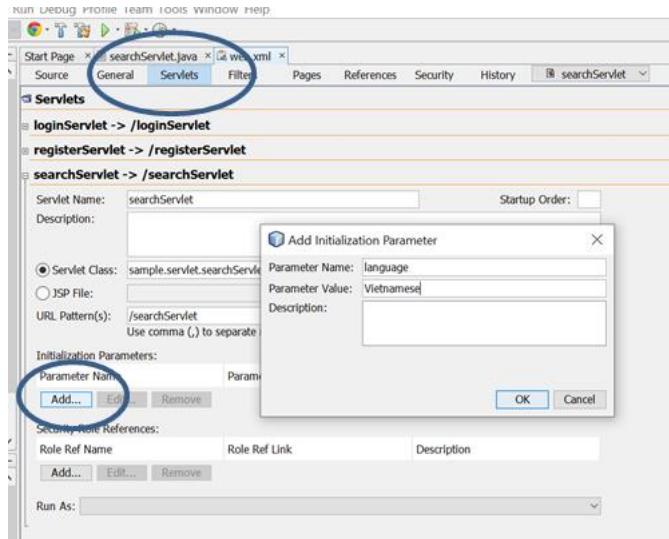
Output

Type here to search

Windows Taskbar: 86°F, ENG, 11/18/2021, 3:32 PM

Part 2: define only the initial parameter on the servlet only.

Step 1: Open the file “web.xml”, at “search servlet”, type the followings

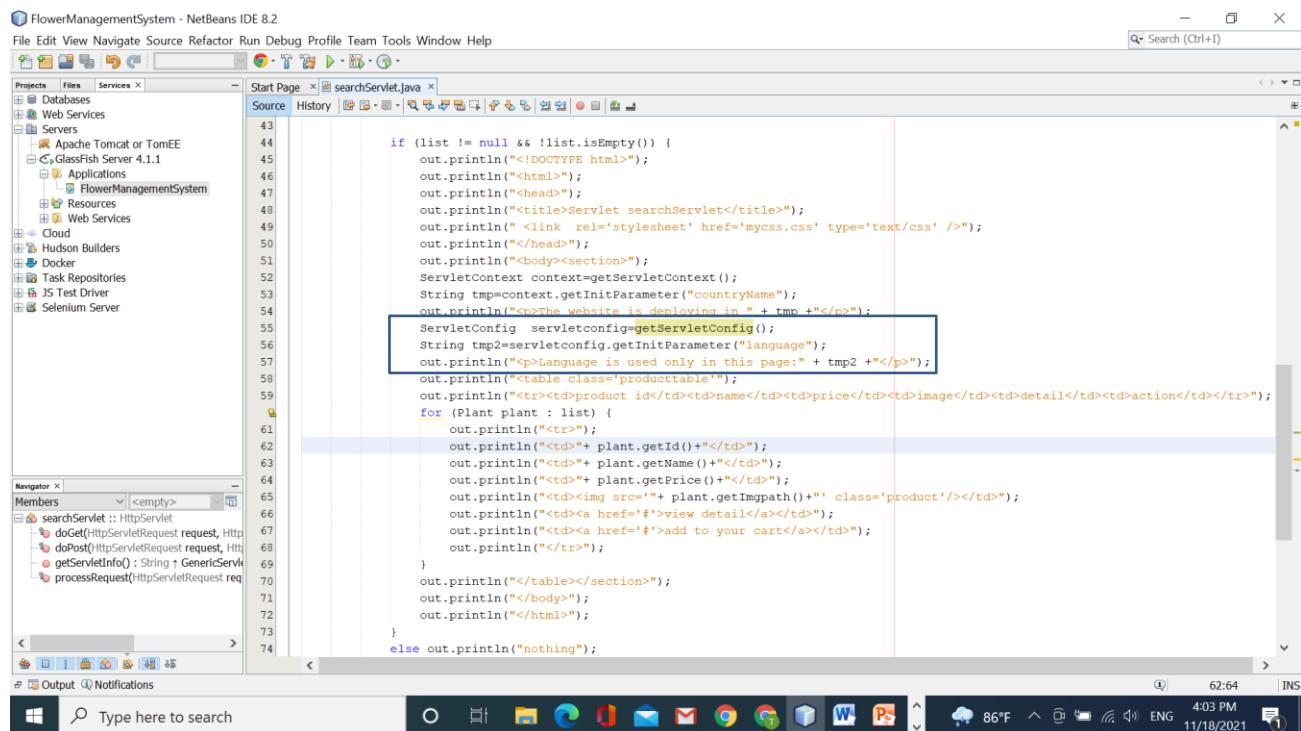


```

15 <servlet>
16     <servlet-name>searchServlet</servlet-name>
17     <servlet-class>sample.servlet.searchServlet</servlet-class>
18     <init-param>
19         <param-name>language</param-name>
20         <param-value>Vietnamese</param-value>
21     </init-param>
22 </servlet>

```

Step 2: Open the file “searchServlet”, and type

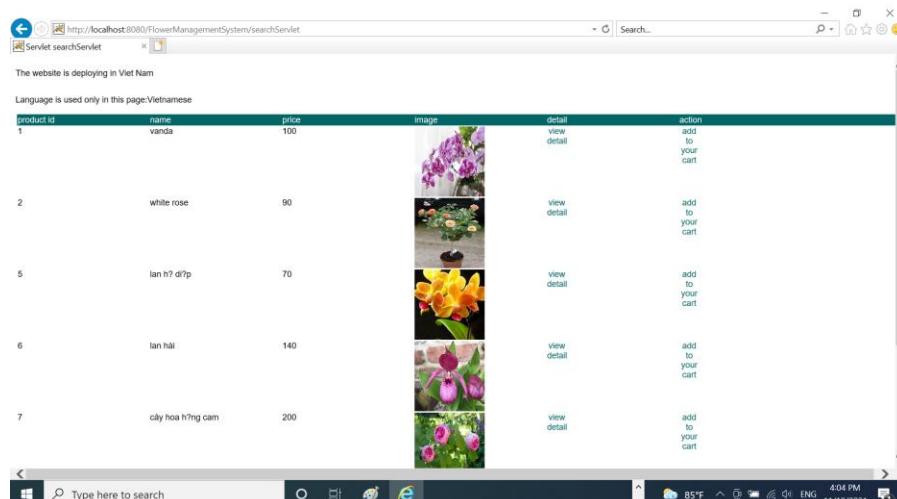


```

43 if (list != null && list.isEmpty()) {
44     out.println("<!DOCTYPE html>");
45     out.println("<html>");
46     out.println("<head>");
47     out.println("<title>Servlet searchServlet</title>");
48     out.println("<link rel='stylesheet' href='mycss.css' type='text/css' />");
49     out.println("</head>");
50     out.println("<body><section>");
51     ServletContext context=getServletContext();
52     String tmpc=context.getInitParameter("countryName");
53     out.println("<p>The website is deploying in " + tmpc + "</p>");
54     ServletConfig servletConfig=getServletConfig();
55     String tmp2=servletConfig.getInitParameter("language");
56     out.println("<p>Language is used only in this page:" + tmp2 + "</p>");
57     out.println("<table class='producttable'>");
58     out.println("<tr><td>product id</td><td>name</td><td>price</td><td>image</td><td>detail</td><td>action</td></tr>");
59     for (Plant plant : list) {
60         out.println("<tr>");
61         out.println("<td>" + plant.getId() + "</td>");
62         out.println("<td>" + plant.getName() + "</td>");
63         out.println("<td>" + plant.getPrice() + "</td>");
64         out.println("<td><img src='" + plant.getImgpath() + "' class='product' /></td>");
65         out.println("<td><a href='#!'>view detail</a></td>");
66         out.println("<td><a href='#!'>add to your cart</a></td>");
67         out.println("</tr>");
68     }
69     out.println("</table></section>");
70     out.println("</body>");
71     out.println("</html>");
72 }
73 else out.println("nothing");
74

```

The result:



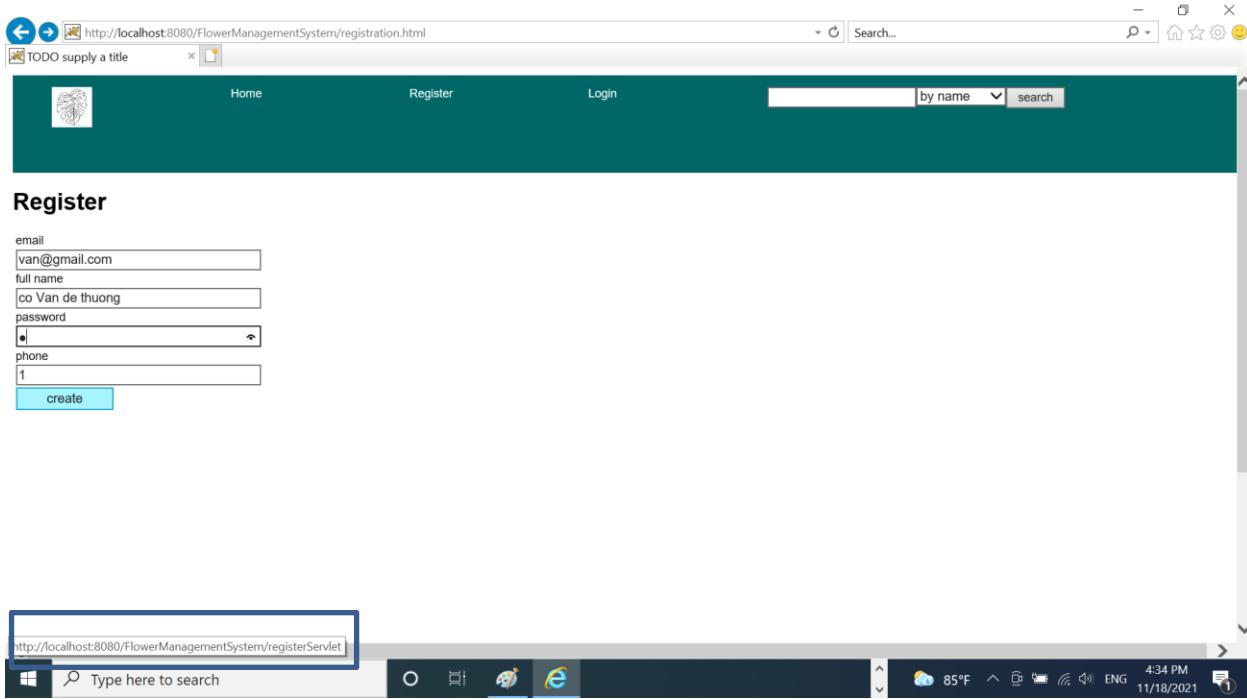
product_id	name	price	image	detail	action
1	vanda	100		view detail	add to your cart
2	white rose	90		view detail	add to your cart
5	lan h? đ?ip	70		view detail	add to your cart
6	lan h?i	140		view detail	add to your cart
7	cây hoa h?ng cam	200		view detail	add to your cart

Part 3: Transfer data from the servlet to other servlets by the method **forward**

Step 1: Open the file “registerServlet.java”, and edit the code

Step 2: In the package “sample. servlet”, create a new servlet named “sendOTP.java”

Step 3: run the web, click the link “register” to create a new account and you will see the result.



The screenshot shows a browser window with the URL <http://localhost:8080/FlowerManagementSystem/registerServlet>. The page title is "Servlet sendOTP at /FlowerManagementS". A callout box points to the URL in the address bar with the text "The content of the servlet 'sendOTP'". Below the title, a message says "please, check your email:van@gmail.com, a confirm code is sent to you."

Part 4: Transfer data from the servlet to other servlets by the method **include**

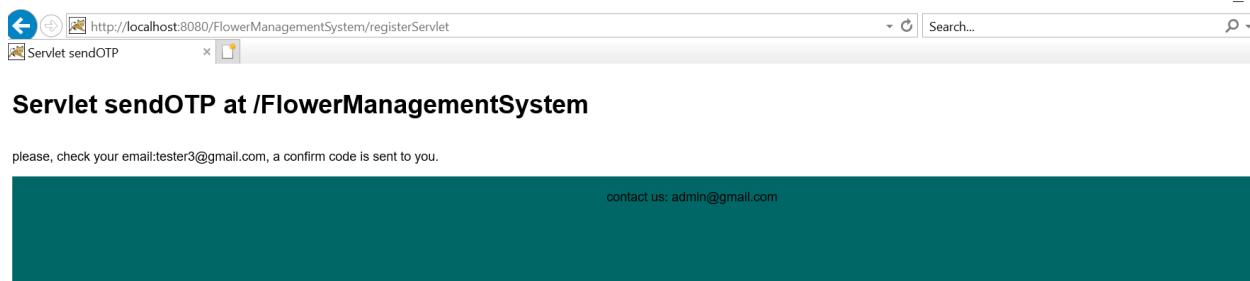
Step 1: In the package 'sample. servlet', create a new servlet named "contactServlet"

```
22
23     /**
24      * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
25      * methods.
26      *
27      * @param request servlet request
28      * @param response servlet response
29      * @throws ServletException if a servlet-specific error occurs
30      * @throws IOException if an I/O error occurs
31     */
32    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
33        throws ServletException, IOException {
34        response.setContentType("text/html;charset=UTF-8");
35        try (PrintWriter out = response.getWriter()) {
36            /* TODO output your page here. You may use following sample code. */
37            out.println("<footer>");
38            Account adminAcc=AccountDAO.getAccount("admin@gmail.com","admin");
39            if(adminAcc!=null){
40                out.println("<p align='center'>contact us: "+ adminAcc.getEmail() + "</p>");
41            }
42            out.println("</footer>");
43        }
44    }
45
46    HttpServlet methods. Click on the + sign on the left to edit the code.
47
48
49
50
51
52
53 }
```

Step 2: In the servlet “sendOTP”, edit the code:

```
25
26     /**
27      * @param request servlet request
28      * @param response servlet response
29      * @throws ServletException if a servlet-specific error occurs
30      * @throws IOException if an I/O error occurs
31     */
32    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
33        throws ServletException, IOException {
34        response.setContentType("text/html;charset=UTF-8");
35        try (PrintWriter out = response.getWriter()) {
36            /* TODO output your page here. You may use following sample code. */
37            out.println("<!DOCTYPE html>");
38            out.println("<html>");
39            out.println("<head>");
40            out.println("<title>Servlet sendOTP</title>");
41            out.println("<link rel='stylesheet' href='mycss.css' type='text/css' />");
42            out.println("</head>");
43            out.println("<body>");
44            out.println("<h1>Servlet sendOTP at " + request.getContextPath() + "</h1>");
45            String email=(String) request.getAttribute("email_newAccount");
46            out.println("<p>please, check your email: " + email + ", a confirm code is sent to you.</p>");
47            RequestDispatcher rd=request.getRequestDispatcher("contactServlet");
48            rd.include(request, response);
49            out.println("</body> ");
50            out.println("</html> ");
51        }
52
53    HttpServlet methods. Click on the + sign on the left to edit the code.
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
69 }
```

Step 3: Run the page register.html to create a new account, the result will be:



Congratulations!!!! you have finished workshop 3 ☺ ☺

Workshop 4: learn MVC2 design

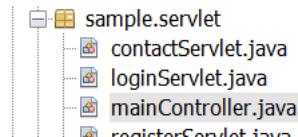
#session #JSP #Implicit objects #MVC2

Learning outcome:

- Study using a servlet to make the main controller that accepts all requests from the client
- Study using JSP pages to presentation content
- Using Session to store the user's full name after login
- Perform the user's all functions.

Part 1: Create a “mainController” using servlet . This servlet has a role like a gatekeeper. All requests should transfer to the mainController. It dispatch to decide upon the appropriate logic to handle the request

Step 1: in the package “sample.servlet”, create a new servlet named “mainController”



Step 2: You will create two new pages named “header.jsp”, “footer.jsp”.

header.jsp

```
<html>
  <head>
    <title>TODO supply a title</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="mycss.css" type="text/css" />
  </head>
  <body>
    <header>
      <nav>
        <ul>
          <li><a href=""></a> </li>
          <li><a href="">Home</a></li>
          <li><a href="">Register</a></li>
          <li><a href="">Login</a></li>
          <li><form action="" method="post" class="formsearch">
              <input type="text" name="txtsearch">
              <select name="searchby"><option value="byname">by name</option>
                <option value="bycate">by category</option>
              </select>
              <input type="submit" value="search" name="action">
            </form></li>
        </ul>
      </nav> </header>
    </body>
  </html>
```

Footer.jsp

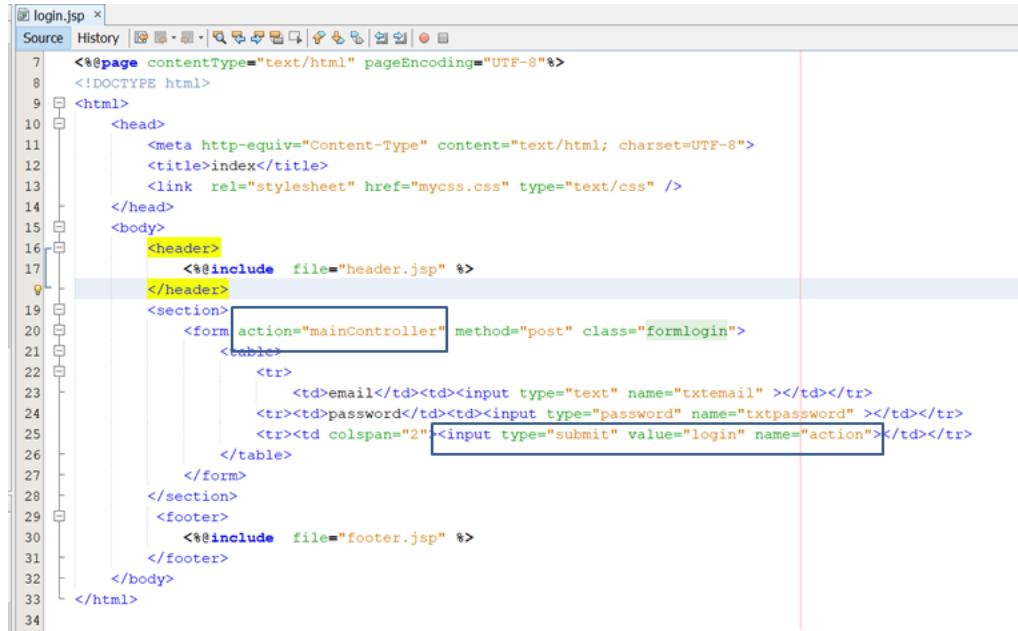
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
        <link rel="stylesheet" href="mycss.css" type="text/css" />
    </head>
    <body>
        <p style="color:white; text-align: center;">Copyright &copy; 2021</p>
    </body>
</html>
```

Step 3: create a new page named “index.jsp”

```
<%--  
1 Document : index  
2 Created on : Nov 19, 2021, 6:06:39 PM  
3 Author : user  
4 --%>  
5  
6  
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>  
8 <!DOCTYPE html>  
9 <html>  
10 <head>  
11     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
12     <title>index</title>  
13     <link rel="stylesheet" href="mycss.css" type="text/css" />  
14 </head>  
15 <body>  
16     <header>  
17         <%@include file="header.jsp" %>  
18     </header>  
19     <section>  
20  
21     </section>  
22     <footer>  
23         <%@include file="footer.jsp" %>  
24     </footer>  
25 </body>  
26 </html>  
27
```

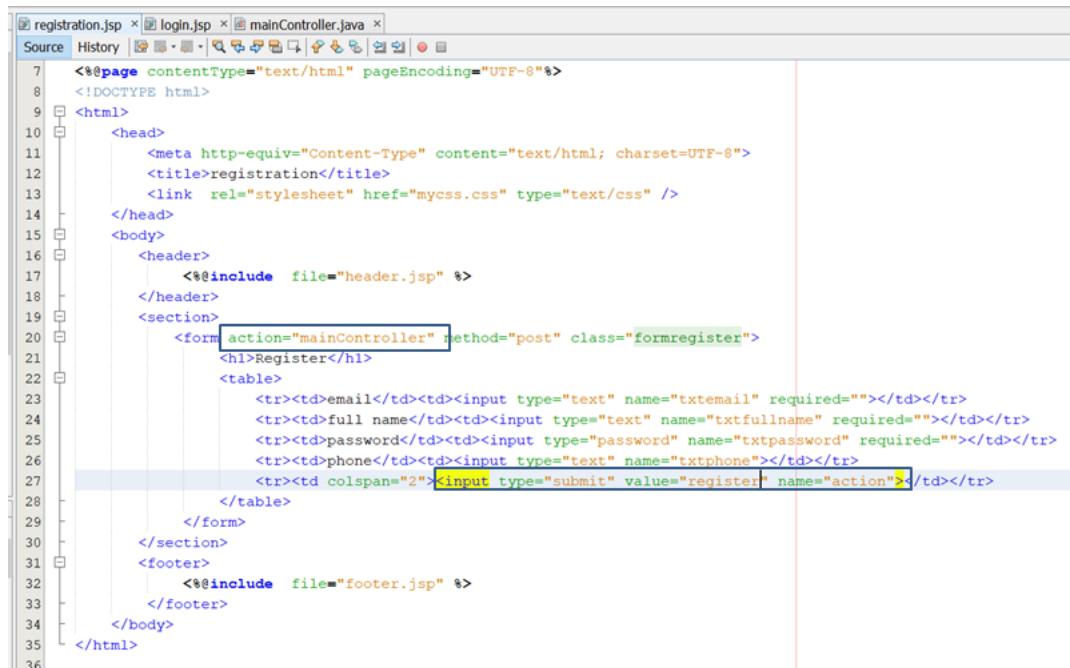
Step 4: do again some functions: login, registration using pages JSP, not HTML

Create a new page named "login.jsp"



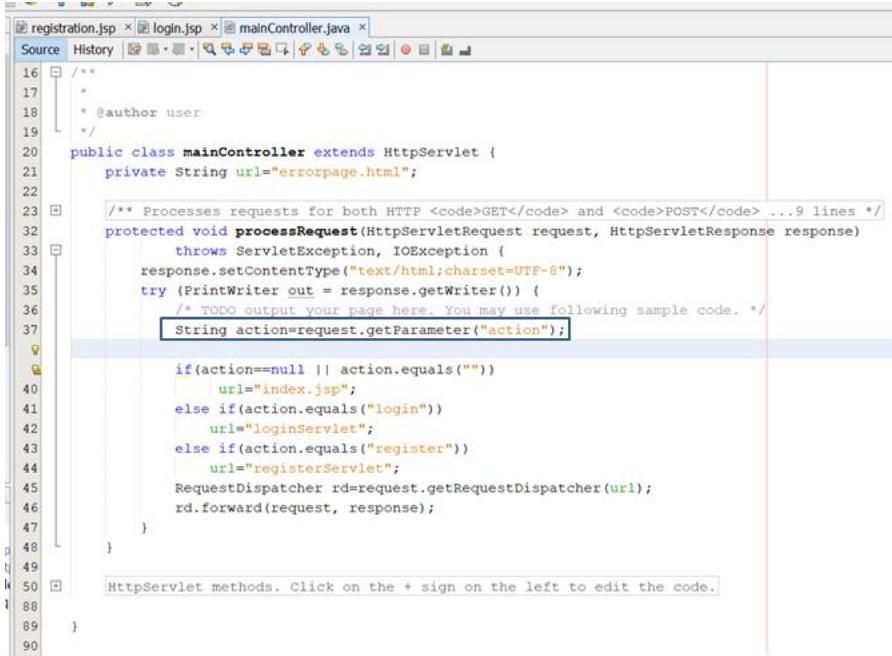
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>index</title>
        <link rel="stylesheet" href="mycss.css" type="text/css" />
    </head>
    <body>
        <header>
            <%@include file="header.jsp" %>
        </header>
        <section>
            <form action="mainController" method="post" class="formlogin">
                <table>
                    <tr>
                        <td>email</td><td><input type="text" name="txtemail" ></td></tr>
                    <tr><td>password</td><td><input type="password" name="txtpassword" ></td></tr>
                    <tr><td colspan="2"><input type="submit" value="login" name="action"></td></tr>
                </table>
            </form>
        </section>
        <footer>
            <%@include file="footer.jsp" %>
        </footer>
    </body>
</html>
```

Create a new page named "registration.jsp"



```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>registration</title>
        <link rel="stylesheet" href="mycss.css" type="text/css" />
    </head>
    <body>
        <header>
            <%@include file="header.jsp" %>
        </header>
        <section>
            <form action="mainController" method="post" class="formregister">
                <h1>Register</h1>
                <table>
                    <tr><td>email</td><td><input type="text" name="txtemail" required="" ></td></tr>
                    <tr><td>full name</td><td><input type="text" name="txtfullname" required="" ></td></tr>
                    <tr><td>password</td><td><input type="password" name="txtpassword" required="" ></td></tr>
                    <tr><td>phone</td><td><input type="text" name="txtphone" ></td></tr>
                    <tr><td colspan="2"><input type="submit" value="register" name="action"></td></tr>
                </table>
            </form>
        </section>
        <footer>
            <%@include file="footer.jsp" %>
        </footer>
    </body>
</html>
```

Step 5: In the servlet "mainController", edit the code:



```
16 /**
17 *
18 * @author user
19 */
20 public class mainController extends HttpServlet {
21     private String url="errorpage.html";
22
23     /** Processes requests for both HTTP <code>GET</code> and <code>POST</code> ...9 lines */
24     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
25             throws ServletException, IOException {
26         response.setContentType("text/html;charset=UTF-8");
27         try (PrintWriter out = response.getWriter()) {
28             /* TODO output your page here. You may use following sample code. */
29             String action=request.getParameter("action");
30
31             if(action==null || action.equals(""))
32                 url="index.jsp";
33             else if(action.equals("login"))
34                 url="loginServlet";
35             else if(action.equals("register"))
36                 url="registerServlet";
37             RequestDispatcher rd=request.getRequestDispatcher(url);
38             rd.forward(request, response);
39         }
40     }
41
42     //HttpServlet methods. Click on the + sign on the left to edit the code.
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
59
60
61
62
63
64
65
66
67
68
69
69
70
71
72
73
74
75
76
77
78
79
79
80
81
82
83
84
85
86
87
88
88
89
89
90
```

Step 7: Open the file web.xml, set up the page “index.jsp” is the first page, deploy your app. And run your app

Conclusion:

The key idea is all requests from the client using the format:

<input type="submit" name="action" value="....">

In the mainController, we get this action parameter to decide what servlet will be called.

Part 2: Using session to store the login user's name and create a new file JSP named “personalPage.jsp”

Step 1: Open the file “loginServlet.java”, edit the code

```

43 acc=AccountAO.getAccount(email, password);
44 System.out.println(acc.getEmail());
45 if(acc!=null){
46     //admin
47     if(acc.getRole()==1){
48         //chuyen qua admin home page
49     }
50     //user
51     else{
52         //chuyen qua welcome page
53         //response.sendRedirect("welcome.html");
54         HttpSession session=request.getSession(true);
55         if(session!=null){
56             session.setAttribute("name", acc.getFullname());
57             session.setAttribute("email", email);
58             response.sendRedirect("personalPage.jsp");
59         }
60     }
61 }
62 }
63 }
64 else response.sendRedirect("invalid.html");
65 } catch (Exception e) {
66     e.printStackTrace();
67 }
68 }
69 }
70 }
71 }
72 
```

HttpServlet methods. Click on the + sign on the left to edit the code.

Find:white No matches x

54.27 10:30 PM 11/19/2021 INS

Step 2: create two new pages named “header_loggedinUser.jsp”, “personalPage.jsp”.

```

6
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>
8 <!DOCTYPE html>
9 <html>
10    <head>
11        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12        <link rel="stylesheet" href="mycss.css" type="text/css" />
13    </head>
14    <body>
15        <nav>
16            <ul>
17                <li><a href="index.jsp">Home</a></li>
18                <li><a href="">change profile</a></li>
19                <li><a href="">completed orders</a></li>
20                <li><a href="">Canceled orders</a></li>
21                <li><a href="">processing orders</a></li>
22                <li>from<input type="date" name="from"> to <input type="date" name="to">
23                    <input type="submit" value="search">
24                </li>
25            </ul>
26        </nav>
27    </body>

```

```

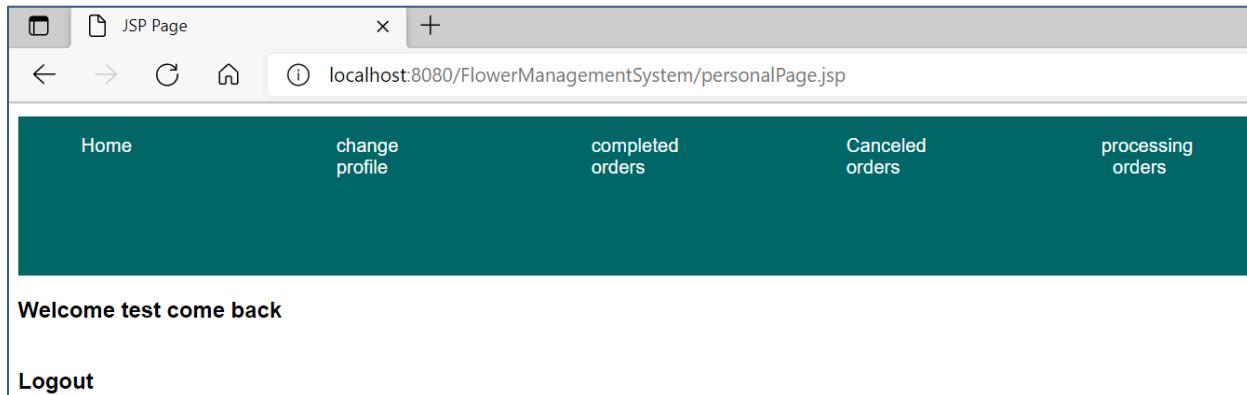
7  <%@page contentType="text/html" pageEncoding="UTF-8"%>
8  <!DOCTYPE html>
9  <html>
10 <head>
11     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12     <title>JSP Page</title>
13     <link rel="stylesheet" href="mycss.css" type="text/css" />
14 </head>
15 <body>
16     <%
17         String name = (String) session.getAttribute("name");
18         if (name == null) {
19             %>
20             <p><font color='red'>you must login to view personal page</font></p>
21             <p></p>
22             <%} else {
23             %>
24             <header>
25                 <%@include file="header_loggedInUser.jsp"%>
26             </header>
27             <section>
28                 <h3>Welcome <%= name%> come back </h3>
29                 <h3>Logout </h3>
30             </section>
31             <section> <%-- load all orders of the user at here --%> </section>
32             <footer>
33                 <%@include file="footer.jsp"%>
34             </footer>
35             <%}>
36         </body>
37     </html>

```

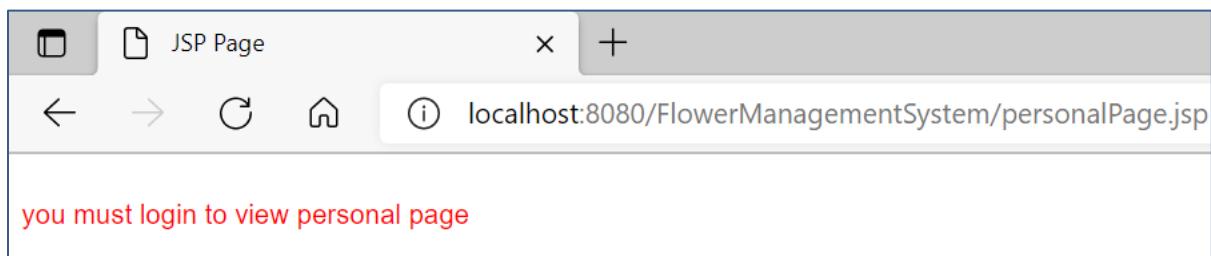
On the page “personalPage.jsp”, we will check user authentication. If the user is valid then open this page otherwise display a warning.

Step 4: run your app to get the result

Case 1: login successfully



Case 2: from the address bar, you type “<http://localhost:8080/FlowerManagementSystem/personalPage.jsp>”



The advanced challenge: You must finish the logout function. When the user clicks “logout”, the website will remove the attribute “name” in the session’s memory, and go to the page “index.jsp”.

personalPage.jsp

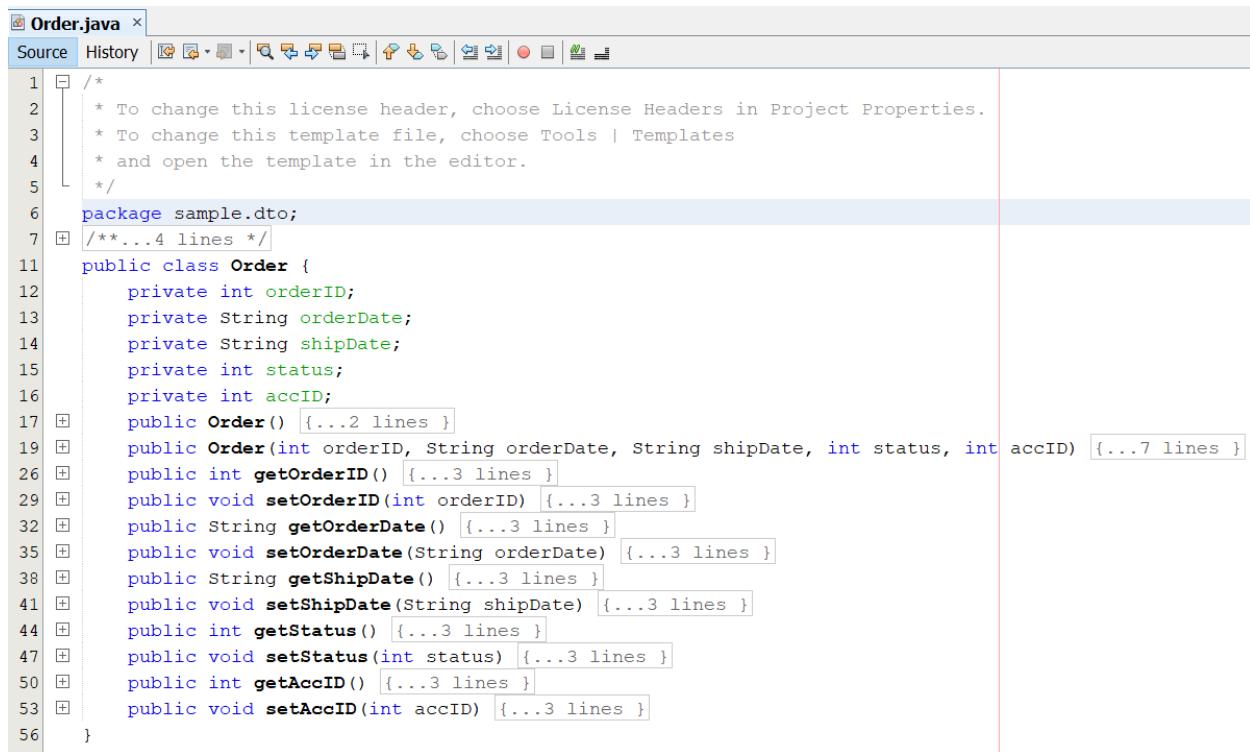
```
<h3>Welcome <%= name%> come back </h3>
<h3><a href="mainController?action=logout">Logout</a> </h3>
```

logoutServlet

```
HttpSession session=request.getSession();
session.invalidate();
response.sendRedirect("index.jsp");
```

Part 3: You will perform the function “view all orders”.

Step 1: Create a new class named “Order”



The screenshot shows a Java code editor with the file 'Order.java' open. The code defines a class 'Order' with various fields and methods. The code is heavily annotated with ellipses (...), indicating missing code. The IDE interface includes tabs for 'Source' and 'History', and a toolbar with various icons.

```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package sample.dto;
7  /**...4 lines */
8  public class Order {
9      private int orderID;
10     private String orderDate;
11     private String shipDate;
12     private int status;
13     private int accID;
14     public Order() {...2 lines}
15     public Order(int orderID, String orderDate, String shipDate, int status, int accID) {...7 lines}
16     public int getOrderID() {...3 lines}
17     public void setOrderID(int orderID) {...3 lines}
18     public String getOrderDate() {...3 lines}
19     public void setOrderDate(String orderDate) {...3 lines}
20     public String getShipDate() {...3 lines}
21     public void setShipDate(String shipDate) {...3 lines}
22     public int getStatus() {...3 lines}
23     public void setStatus(int status) {...3 lines}
24     public int getAccID() {...3 lines}
25     public void setAccID(int accID) {...3 lines}
26 }
```

Step 2: Create a new class named “OrderDAO”

OrderDAO.java

```

1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package sample.dao;
7
8  import java.sql.Connection;
9  import java.sql.PreparedStatement;
10 import java.sql.ResultSet;
11 import java.util.ArrayList;
12 import sample.dto.Account;
13 import sample.dto.Order;
14 import sample.dto.OrderDetail;
15 import sample.utils.DBUtils;
16 /*...4 lines */
17 public class OrderDAO {
18     public static ArrayList<Order> getOrders(String email){...31 lines }
19     public static ArrayList<OrderDetail> getOrderDetail(int orderID){...30 lines }
20 }

```

Step 3: Open the page "personalPage.jsp", edit the code:

```

<section> <!--load all orders of the user at here -->
<% 
    ArrayList<Order> list=OrderDAO.getOrders(email);
    String[] status={"","processing","completed","canceled"};
    if(list!=null && !list.isEmpty()){
        for (Order ord : list) { %>
            <table class="order">
                <tr><td>Order ID</td><td>Order Date</td><td>Ship Date</td><td>Order's status</td><td>action</td></tr>
                <tr><td><%= ord.getOrderId() %></td>
                    <td><%= ord.getOrderDate() %></td>
                    <td><%= ord.getShipDate() %></td>
                    <td><%= status[ord.getStatus()] %>
                        <br/><% if(ord.getStatus()==1) %><a href="#">cancel order</a>
                    </td>
                    <td><a href="orderDetail.jsp?orderid=<%= ord.getOrderId() %>">detail</a> </td></tr>
            </table>
        <% } 
    } 
    else 
%> 
        <p>You don't have any order</p>
</section>

```

Step 4: Now, you insert some tuples into tables: Orders, OrderDetails

DESKTOP-IRCVITT....hop - dbo.Orders SQLQuery1.sql - (I...antShop (sa (52))*

	OrderID	OrdDate	shipdate	status	AccID
1	2021-01-11	2021-10-11	2	1	
2	2021-11-23	NULL	1	1	
3	2021-10-01	NULL	3	1	
*	NULL	NULL	NULL	NULL	NULL

DESKTOP-IRCVITT....dbo.OrderDetails SQLQuery1.sql - (l...antShop (sa (52

	DetailId	OrderID	FID	quantity
▶	1	1	1	1
	2	1	2	2
	5	2	5	1
*	6	3	8	2
	7	3	9	1
*	NULL	NULL	NULL	NULL

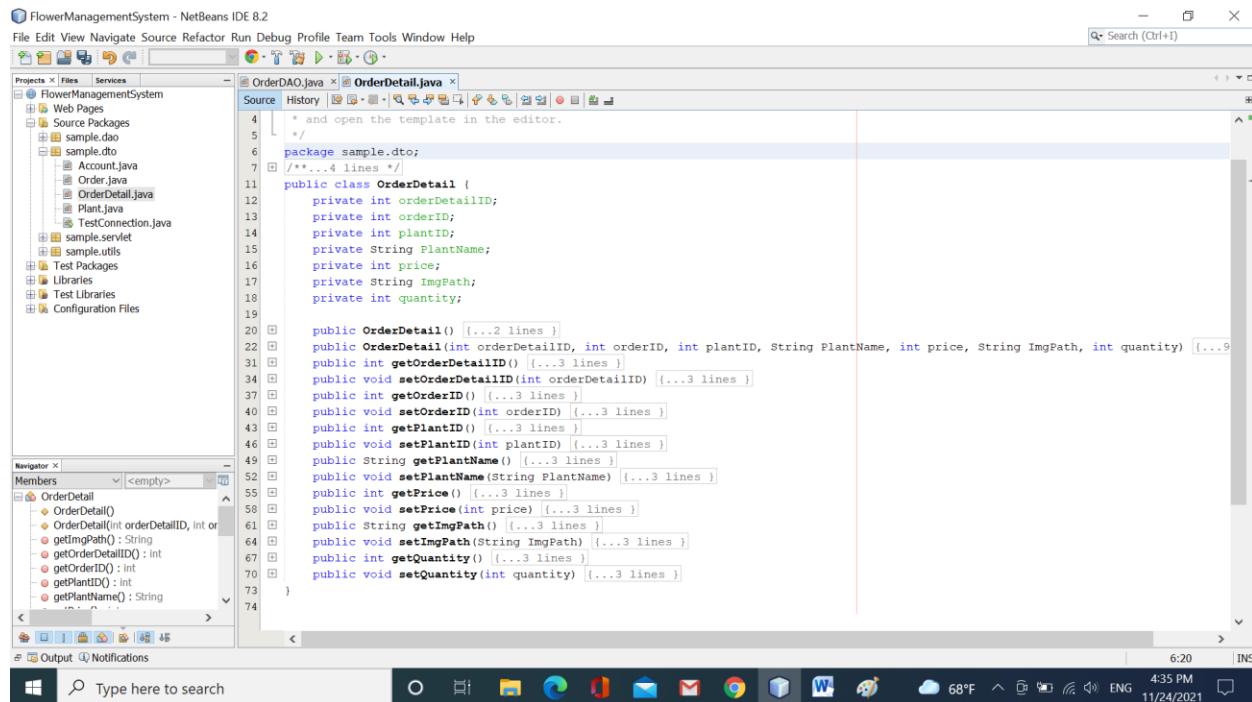
Step 5: run your app to get the result

Order ID	Order Date	Ship Date	Order's status	action
1	2021-01-11	2021-10-11	completed	detail
2	2021-11-23	null	processing	action
3	2021-10-01	null	canceled	cancel order



Part 4: View order detail of the order

Step 1: In the package “sample.dto”, create a new class named “OrderDetail.java”



Step 2: In the file OrderDAO.java, add more the method “getOrderDetail(int orderID)”

```

1 public class OrderDAO {
2     public static ArrayList<Order> getOrders(String email){...31 lines}
3     public static ArrayList<OrderDetail> getOrderDetail(int orderID){
4         Connection cn=null;
5         ArrayList<OrderDetail> list=new ArrayList<>();
6         try{
7             cn = DBUtils.makeConnection();
8             if(cn != null) {
9                 String sql = "select DetailID,OrderID,PID,PName,price,imgPath,quantity\n"
10                     + "from OrderDetails, Plants\n"
11                     + "where OrderID=? and OrderDetails.FID=Plants.PID";
12                 PreparedStatement pst = cn.prepareStatement(sql);
13                 pst.setInt(1, orderID);
14                 ResultSet rs = pst.executeQuery();
15                 if (rs != null) {
16                     while (rs.next()) {
17                         int detailID = rs.getInt("DetailID");
18                         int PlantID = rs.getInt("PID");
19                         String PlantName = rs.getString("PName");
20                         int price = rs.getInt("price");
21                         String imgPath=rs.getString("imgPath");
22                         int quantity=rs.getInt("quantity");
23                         OrderDetail orderdetail=new OrderDetail(detailID, orderID, PlantID, PlantName, price, imgPath, quantity);
24                         list.add(orderdetail);
25                     } } }
26             }catch(Exception e){ e.printStackTrace(); }
27             finally{ try {
28                 if(cn!=null) cn.close();
29             } catch (Exception e) { e.printStackTrace(); } }
30         }
31         return list;
32     }
33 }
```

Step 3: Create a new page named “OrderDetail.jsp”.

```
orderDetail.jsp x
Source History | Back Forward Stop Refresh Help
38     <h3>Logout </h3>
39     <a href="personalPage.jsp">view all orders</a>
40   </section>
41   <section> <!--load all orders of the user here -->
42     <% String orderid = request.getParameter("orderid");
43     if (orderid != null) {
44       int orderID = Integer.parseInt(orderid.trim());
45       ArrayList<OrderDetail> list = OrderDAO.getOrderDetail(orderID);
46       if (list != null && !list.isEmpty()) {
47         int money = 0;
48         for (OrderDetail detail : list) {
49           <table class="order">
50             <tr><td>Order ID:</td><td>Plant ID:</td><td>Plant Name:</td><td>Image:</td><td>quantity:</td></tr>
51             <tr><td><%= detail.getOrderID() %></td><td><%= detail.getPlantID() %></td>
52               <td><%= detail.getPlantName() %></td>
53               <td><img src='<%= detail.getImgPath() %>' class='plantimg' /> <br/> <%= detail.getPrice() %></td>
54               <td><%= detail.getQuantity() %></td>
55               <% money = money + detail.getPrice() * detail.getQuantity(); %>
56             </tr>
57           </table>
58         } //end for %
59         <h3> Total money: <%= money %></h3>
60       } //end if
61       else{
62         <p>You don't have any order</p>
63       }
64     }
65   } //end if
66   <%>
67 </section>
68 <footer>
69   <%@include file="footer.jsp" %>
70 
```

Step 5: run your app. On the page “personalPage.jsp”, click the link “detail” to view this order’s detail.

http://localhost:8080/FlowerManagementSystem2/orderDetail.jsp?orderId=3

JSP Page

Home change profile completed orders Canceled orders processing orders

from to search

Welcome test come back

[Logout](#)

[view all orders](#)

Order ID	Plant ID	Plant Name	Image	Price	quantity
3	8	monstera		80	2

Order ID	Plant ID	Plant Name	Image	Price	quantity
3	9	var monstera		400	1

Total money: 560

Copyright © 2021



Part 5: Load all products to the index.jsp, use index.jsp to display the result of searching.

When the user opens the page “index.jsp”, all products will be displayed.

When the user searches for the product by name/by category, only appropriate products are displayed.

Step 1: Open the page “header.jsp”, edit the code:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
        <link rel="stylesheet" href="mycss.css" type="text/css" />
    </head>
    <body>
        <nav>
            <ul>
                <li><a href="index.jsp"></a></li>
                <li><a href="mainController?action=Home">Home</a></li>
                <li><a href="registration.jsp">Register</a></li>
                <li><a href="login.jsp">Login</a></li>
                <li><form action="mainController" method="post" class="formsearch">
                    <input type="text" name="txtsearch">
                    <select name="searchby">
                        <option value="byname">by name</option>
                        <option value="bycate">by category</option>
                    </select>
                    <input type="submit" value="search" name="action">
                </form></li>
            </ul>
        </nav>
    </body>
</html>
```

Step 2: Open the file “mainController”, edit the code:

```
/*
 * 
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        /* TODO output your page here. You may use following sample code. */
        String action=request.getParameter("action");

        if(action==null || action.equals("")|| action.equals("search")){
            url="index.jsp";
        } else if(action.equals("login")){
            url="loginServlet";
        } else if(action.equals("register")){
            url="registerServlet";
        RequestDispatcher rd=request.getRequestDispatcher(url);
        rd.forward(request, response);
    }
}
HttpServlet methods. Click on the + sign on the left to edit the code.
```

Step 3: Open the page “index.jsp”, edit the code:

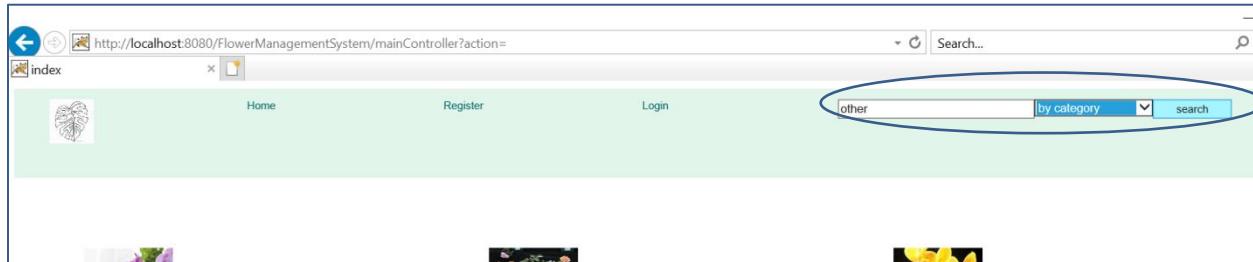
mainController.java | header.jsp | index.jsp

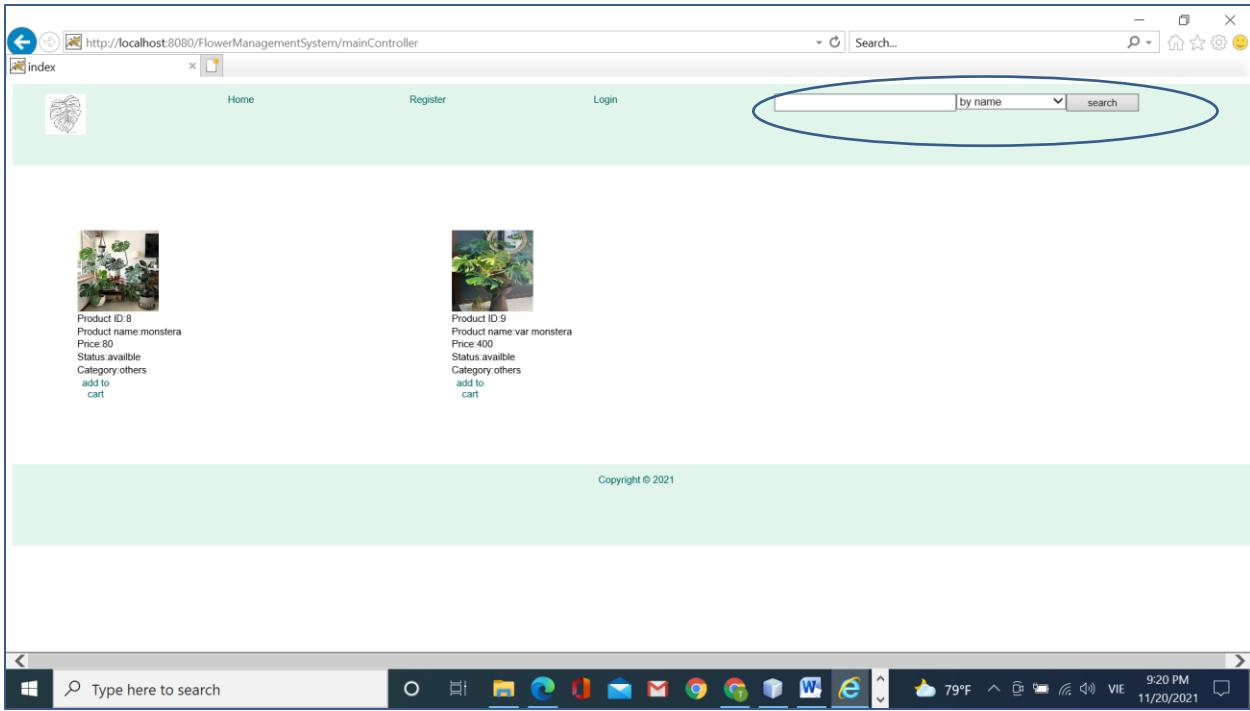
```

22         </header>
23         <section>
24             <%
25                 String keyword=request.getParameter("txtsearch");
26                 String searchby=request.getParameter("searchby");
27                 ArrayList<Plant> list;
28                 String [] tmp={"out of stock","available"};
29                 if(keyword==null && searchby==null) // when the page is loaded, all products will be displayed
30                     list=PlantDAO.getPlants("", "");
31                 else
32                     list=PlantDAO.getPlants(keyword, searchby); // only get products by name/category
33                 if(list!=null && !list.isEmpty()){
34                     for (Plant p : list) {
35                         <table class='product'>
36                             <tr>
37                                 <td><img src='<% p.getImgpath() %>' class='plantimg' /></td>
38                                 <td>Product ID:<% p.getId() %></td>
39                                 <td>Product name:<% p.getName() %></td>
40                                 <td>Price:<% p.getPrice() %></td>
41                                 <td>Status:<% tmp[p.getStatus()] %></td>
42                                 <td>Category:<% p.getCatename() %></td>
43                                 <td><a href="#">add to cart</a></td>
44                             </tr>
45                         </table>
46                     }
47                 }
48             </section>
49             <footer>
50                 <%@include file="footer.jsp" %>
51             </footer>
52         </body>
53     
```

Step 4: run your app

The sample: the user inputs “other”, and click the button “search”





After searching, the appropriate products are displayed. However, ***the input text control can not keep*** the user's search. So, we should edit the page "header.jsp" to overcome this.

Step 5: Open the page "header.jsp", edit the code

```

mainController.java x Index.jsp x header.jsp x
Source History
4
5
6
7 "text/html" pageEncoding="UTF-8">
8
9
10
11 <?xml version="1.0" encoding="UTF-8"?>
12 <?xml-stylesheet type="text/css" href="mycss.css"?>
13
14
15
16
17
18 <a href="index.jsp"></a> </li>
19 <a href="mainController?action=Home">Home</a></li>
20 <a href="registration.jsp">Register</a></li>
21 <a href="login.jsp">Login</a></li>
22 <form action="mainController" method="post" class="formsearch">
23 <input type="text" name="txtsearch" value="<%=(request.getParameter("txtsearch")==null)? "" : request.getParameter("txtsearch")%>">
24 <select name="searchby"><option value="byname">by name</option>
25 <option value="bycate">by category</option>
26 </select>
27 <input type="submit" value="search" name="action">
28 </form></li>
29

```

Step 6: Run your app again.

The screenshot shows a web browser window with the URL <http://localhost:8080/FlowerManagementSystem/mainController>. The page has a header with links for Home, Register, and Login. A search bar is located at the top right, containing the text "other" and a dropdown menu set to "by name". A blue oval highlights this search bar area. Below the header, there are two product cards. The first card shows a potted plant and details: Product ID 8, Product name: monstera, Price: 80, Status: available, Category: others, with a link to add it to the cart. The second card shows a potted plant and details: Product ID 9, Product name: var monstera, Price: 400, Status: available, Category: others, with a link to add it to the cart. At the bottom of the page, a green footer bar displays the text "Copyright © 2021".



Now, the user's searches are kept and rewrote to the input text control.

The advanced challenge:

- 1) You must perform the function view completed orders, processing orders, canceled orders.
- 2) You must perform the function order again on canceled orders.
- 3) You must perform the change your profile function (only update full name, phone).

Congratulations!!!! you have finished workshop 4 ☺ ☺

Workshop 5: Using Cookie, Session

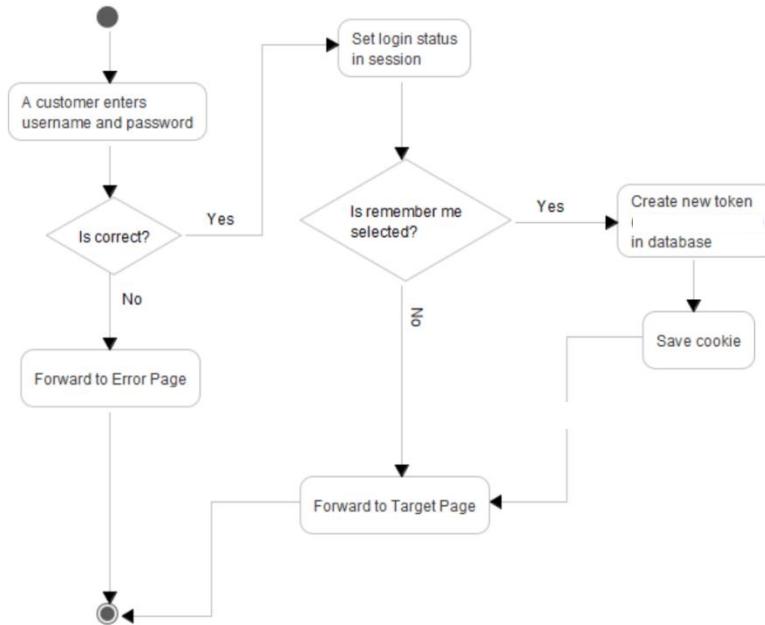
RewriteURL #HiddenField #Cookie # Session #Exception Handling

Learning outcome:

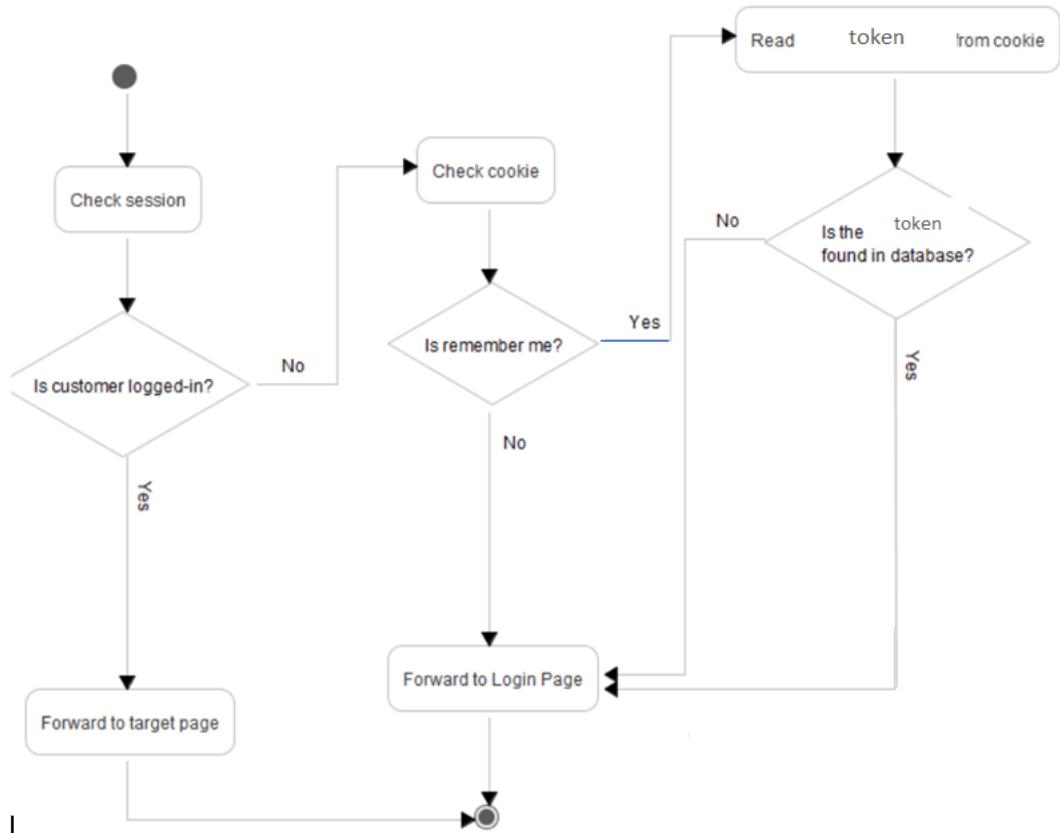
- Know to use the cookie to save login information
- Know to use the session to create the shopping cart
- How to handle exceptions

Part 1: Save the user's login information.

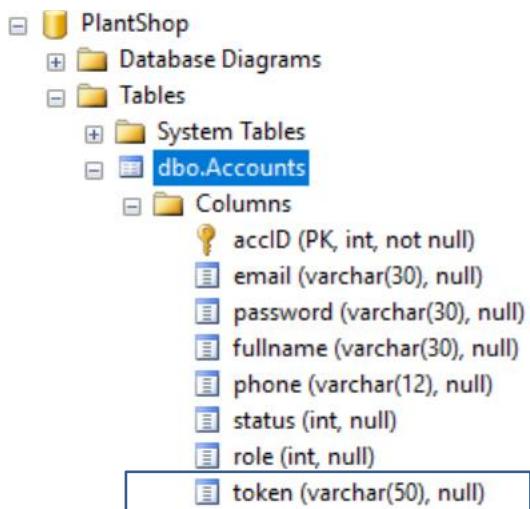
Describe the login function



Using the cookie to login



Step 1: add one more column “token” in the table Accounts.



Step 2: Add one more checkbox control in the page “login.jsp”

The screenshot shows a Java IDE interface with the file `login.jsp` open. The code is a JSP page with the following structure:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>index</title>
        <link rel="stylesheet" href="mycss.css" type="text/css" />
    </head>
    <body>
        <header>
            <%@include file="header.jsp" %>
        </header>
        <section>
            <form action="mainController" method="post" class="form">
                <table>
                    <tr><td>email</td><td><input type="text" name="txtemail" ></td></tr>
                    <tr><td>password</td><td><input type="password" name="txtpassword" ></td></tr>
                    <tr><td colspan="2"><input type="submit" value="login" name="action"></td></tr>
                    <tr><td colspan="2"><input type="checkbox" value="savelogin" name="savelogin">Stayed signed in</td></tr>
                </table>
            </form>
        </section>
        <footer>
            <%@include file="footer.jsp" %>
        </footer>
    </body>
</html>
```

Step 3: On the page “loginServlet”, edit the code:

The screenshot shows a Java IDE interface with the file `loginServlet.java` open. The code is a servlet with the following structure:

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        /* TODO output your page here. You may use following sample code. */
        String email=request.getParameter("txtemail");
        String password=request.getParameter("txtpassword");
        String save=request.getParameter("savelogin");
        Account acc=null;
        try {
            if(email ==null || email.equals("") || password==null || password.equals("")){
                Cookie[] c=request.getCookies();
                String token="";
                if (c != null) {
                    for (Cookie aCookie : c) {
                        if (aCookie.getName().equals("selector")) {
                            token = aCookie.getValue();
                        }
                    }
                }
                if(!token.equals(""))
                    response.sendRedirect("personalPage.jsp");
                else response.sendRedirect("errorpage.html");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        else{
            acc = AccountDAO.getAccount(email, password);
            if (acc != null) {
                //admin
                if (acc.getRole() == 1) {
                    //chuyen qua admin home page
                    response.sendRedirect("adminHome.jsp");
                } else {
                    response.sendRedirect("userHome.jsp");
                }
            }
        }
    }
}
```

FlowerManagementSystem - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+F)

Projects Files Services X Start Page x loginServlet.java x

Source History

```

60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91

```

processRequest - Navigator

Members <empty>

loginServlet :: HttpServlet

- doGet(HttpServletRequest request, HttpServletResponse response) : void
- doPost(HttpServletRequest request, HttpServletResponse response) : void
- getServletInfo() : String : GenericServlet
- processRequest(HttpServletRequest request, HttpServletResponse response) : void

Output Notifications

Type here to search

76:64 4:29 PM 88°F ENG 11/22/2021

Step 4: On the page “personalPage.jsp”, edit the code:

Start Page x loginServlet.java x personalPage.jsp x

```

19 </head>
20 <body>
21
22
23     String name = (String) session.getAttribute("name");
24     String email = (String) session.getAttribute("email");
25     Cookie[] c = request.getCookies();
26     boolean login=false;
27     if (name == null){
28         String token = "";
29         for (Cookie aCookie : c)
30             if (aCookie.getName().equals("selector")){
31                 token = aCookie.getValue();
32                 Account acc = AccountDAO.getAccount(token);
33                 if (acc != null){
34                     name = acc.getFullscreen();
35                     email = acc.getEmail();
36                     login=true;
37                 }
38             }
39     }
40     else
41         login=true;
42
43     //show content if login=true
44     if(login){
45         %>
46             <header>
47                 <%@include file="header_loggedinUser.jsp"%>
48             </header>
49             <section>
50                 <h3>Welcome <%= name%> come back </h3>
51                 <h3><a href="mainController?action=logout">Logout</a> </h3>

```

```
Start Page | loginServlet.java | personalPage.jsp | 
Source History | Back Forward | Find | Replace | Open | Save | 
43     //show content if login=true
44     if(login){
45         %>
46             <header>
47                 <%@include file="header_loggedInUser.jsp"%>
48             </header>
49             <section>
50                 <h3>Welcome <%= name%> come back </h3>
51                 <h3><a href="mainController?action=logout">Logout</a> </h3>
52             </section>
53             <section> <!--load all orders of the user at here -->
54             <%
55                 ArrayList<Order> list=OrderDAO.getOrders(email);
56                 String[] status={"","processing","completed","canceled"};
57                 if(list!=null && !list.isEmpty()){
58                     for (Order ord : list) { %>
59                         <table class="order">
60                             <tr><td>Order ID</td><td>Order Date</td><td>Ship Date</td><td>Order's status</td><td>action</td></tr>
61                             <tr><td><%= ord.getOrderID()%></td><td><%= ord.getOrderDate()%></td>
62                             <td><%= ord.getShipDate()%></td><td><%= status[ord.getStatus()]%></td>
63                             <td><a href="orderDetail.jsp?orderid=<%= ord.getOrderID()%>">detail</a></td></tr>
64                         </table>
65                     }
66                 }
67             <% else %>
68                 <p>You don't have any order</p>
69             </section>
70             <footer>
71                 <%@include file="footer.jsp" %>
72             </footer>
73         <%>
74     <%>
```

Step 5: run your app to test this function

Part 2: Using session to store the user's shopping cart.

We use `HashMap<String, Integer>` to store the customer's cart. The key is the productid and value is the quantity. Cart object is stored in the session memory on the server-side (you can change this using cookies to store the customer's cart on the client-side).

Step 1: Get the product id when the user hovers the link "add to cart".

Open the page "index.jsp", edit the code

FlowerManagementSystem - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+F)

Projects Services

Web Pages

- META-INF
- WEB-INF
- Images
- errorpage.html
- footer.jsp
- header.jsp
- header_joinedUser.jsp
- index.html
- index.jsp
- invalid.html
- login.html
- login.jsp
- mycss.css
- orderDetail.jsp
- personalPage.jsp
- registration.html
- registration.jsp
- testError.jsp
- viewCart.jsp
- welcome.html

Source Packages

Test Packages

Libraries

Test Libraries

Navigator

JSP

Filters:

Output Notifications

Type here to search

Windows Taskbar: 68°F 5:04 PM 11/24/2021

```

<td><a href="mainController?action=addtocart&pid=<%= p.getId() %>">add to cart</a></td>

```

Step 2: open the servlet “mainController”, edit the code:

FlowerManagementSystem - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+F)

Projects Services

Source History

mainController.java addToCartServlet.java viewCart.jsp updateCartServlets.java index.jsp mycss.css

HttpServlet methods. Click on the + sign on the left to edit the code.

```

protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException
{
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        /* TODO output your page here. You may use following sample code. */
        String action=request.getParameter("action");
        if(action==null || action.equals("")|| action.equals("search"))
            url="index.jsp";
        else if(action.equals("login"))
            url="LoginServlet";
        else if(action.equals("register"))
            url="registerServlet";
        else if(action.equals("logout"))
            url="LogoutServlet";
        else if(action.equals("addtocart"))
            url="addToCartServlet";
        else if(action.equals("viewcart"))
            url="viewCart.jsp";
        else if(action.equals("update"))
            url="updateCartServlets";
        else if(action.equals("delete"))
            url="deleteCartServlet";
        RequestDispatcher rd=request.getRequestDispatcher(url);
        rd.forward(request, response);
    }
}

```

Output Notifications

Type here to search

Windows Taskbar: 82°F 8:24 PM 11/22/2021

Step 3: create new new servlet named “addToCartServlet”

FlowerManagementSystem - NetBeans IDE 8.2

```

protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    /* TODO output your page here. You may use following sample code. */
    //get the selected productid
    String pid=request.getParameter("pid");
    //get Session that is storing the shopping cart
    HttpSession session=request.getSession(true);
    if(session==null){
        HashMap<String,Integer> cart=(HashMap<String,Integer>) session.getAttribute("cart");
        if(cart==null){//chua co gio hang thi tao moi
            cart=new HashMap<>();
            cart.put(pid,1);
        }
        else{
            //kiem tra xem san pham co trong cart chua
            Integer tmp=cart.get(pid);
            if(tmp==null){
                cart.put(pid,1);
            }
            else{
                tmp++;
                cart.put(pid, tmp);
            }
        }
        session.setAttribute("cart", cart);
        response.sendRedirect("index.jsp");
    }
}

```

Output Notifications 64:6 INS

Step 4: View the shopping cart.

Open the page “header.jsp”, add one more the link “view cart”

FlowerManagementSystem - NetBeans IDE 8.2

```

Author : user
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSF Page</title>
        <link rel="stylesheet" href="mycss.css" type="text/css" />
    </head>
    <body>
        <nav>
            <ul>
                <li><a href="index.jsp"></a> </li>
                <li><a href="mainController?action=>Home</a></li>
                <li><a href="registration.jsp">Register</a></li>
                <li><a href="login.jsp">Login</a></li>
                <li><a href="mainController?action=viewcart">view cart</a></li>
            </ul>
        </nav>
        <form action="mainController" method="post" class="formsearch">
            <input type="text" name="txtsearch" value="<%=(request.getParameter("txtsearch")==null)?": request.getParameter("txtsearch")%>">
            <select name="searchby"><option value="byname">by name</option>
                <option value="bycate">by category</option>
            </select>
            <input type="submit" value="search" name="action">
        </form></li>
    </body>
</html>

```

Output Notifications 22:60/1:8 INS

Create a new page named “viewCart.jsp”

FlowerManagementSystem - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+F)

Source History ▾

mainController.java addToCartServlet.java viewCart.jsp updateCartServlets.java index.jsp mycss.css

Projects Files Services

Navigator

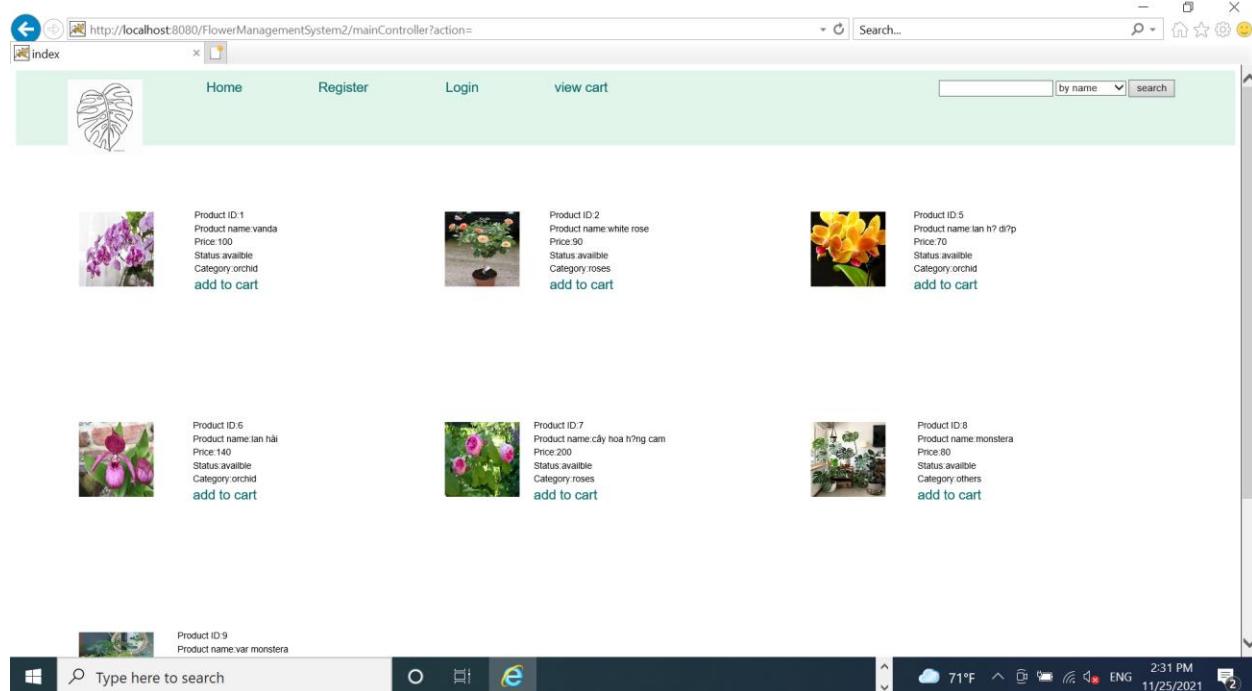
```
17 <header>
18     <%@include file="header.jsp" %>
19 </header>
20 <section>
21
22     <table width="100%" class="shopping">
23         <tr><td>Product id</td><td>quantity</td><td>action</td></tr>
24
25     <%>
26     HashMap<String, Integer> cart=(HashMap)session.getAttribute("cart");
27     if(cart!=null){
28         Set<String> pids=cart.keySet();
29         for (String pid : pids) {
30             int quantity=cart.get(pid);
31
32             <form action="mainController" method="post" >
33                 <tr> <td><input type="hidden" value=<%= pid %> name="pid"><a href="#"><%= pid %></a></td>
34                 <td><input type="number" value=<%= quantity %> name="quantity"></td>
35                 <td><input type="submit" value="update" name="action" >
36                 <input type="submit" value="delete" name="action" ></td>
37
38             </tr>
39             </form>
40         }
41     }
42     else {
43
44         <tr><td>Your cart is empty</td></tr>
45
46     }
47     <tr><td>Total money: </td></tr>
48     </table>
```

Output Notifications

Type here to search

Windows Start Task View File Explorer File Manager Mail Internet Explorer Google Chrome Microsoft Word Microsoft Excel Microsoft Powerpoint 34:81 8:23 PM 82°F ENG 11/22/2021

Step 5: Run your app



Click “view cart” from the header

The screenshot shows a JSP page titled "JSP Page" at the URL <http://localhost:8080/FlowerManagementSystem/mainController?action=viewcart>. The page has a header with links for Home, Register, Login, and view cart. It features a search bar with dropdown options "by name" and "search". Below the header, there's a section for a shopping cart. It displays two items: Product ID 1 with quantity 2 and Product ID 2 with quantity 1. Each item has "update" and "delete" buttons next to it. A message says "Your cart is empty" and "Total money:". At the bottom, there's a copyright notice "Copyright © 2021".

Go back to the home page and click some links “add to cart”

This screenshot shows the same JSP page after adding items to the cart. Now, the shopping cart section lists two items: Product ID 1 with quantity 2 and Product ID 2 with quantity 1. Each item row includes "update" and "delete" buttons. The rest of the page structure remains the same, including the header, search bar, and footer.



Part 3: change the quantity of the product on the shopping cart.

When the user clicks the button “update” on the page “viewcart.jsp”, we call “mainController” and transfer three data: product id, new quantity, and action=”update”. And then, “mainController” transfer this data to the updateCartServlet to process.

Sep 1: Create a new servlet named updateCartServlet

FlowerManagementSystem - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+F)

Source History Navigator Projects Files Services

```

27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96

```

HttpServlet methods. Click on the + sign on the left to edit the code.

Output Notifications 39:48 INS

Type here to search

82°F 8:33 PM 11/22/2021

Step 2: Run your app

http://localhost:8080/FlowerManagementSystem/mainController?action=viewcart

Search...

Home Register Login view cart

Product id quantity action

1	<input type="text" value="2"/>	<input type="button" value="update"/> <input type="button" value="delete"/>
2	<input type="text" value="2"/>	<input type="button" value="update"/> <input type="button" value="delete"/>

Total money:

Copyright © 2021



Part 4: Exception handling

During executing your app, if any exception has happened then the container will ***return immediately to the client exception obj that is presented based on HTML format***. We can use many ways to handle exceptions in your app.

- Use the file descriptor “web.xml” to setup: which pages will be opened when the error happens
 - Use some methods : sendError(), setStatus(),....of response object to send error to the client
 - Use the log file
 - Use your ways yourself

For example 1: validate the user's email on the client-side.

Open the page “register.jsp”

```
<tr><td>email</td><td><input type="text" name="txtemail" required="" pattern="^((\w)+@[a-zA-Z]+([.](\w)+){1,2})"></td><
```

```
the client.

mainController.java x registerServlet.java x registration.jsp x
Source History
30     * @throws IOException if an I/O error occurs
31     */
32     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
33             throws ServletException, IOException {
34         response.setContentType("text/html;charset=UTF-8");
35         try (PrintWriter out = response.getWriter()) {
36             /* TODO output your page here. You may use following sample code. */
37             String email=request.getParameter("txtemail");
38             String fullname=request.getParameter("txtfullname");
39             String password=request.getParameter("txtpassword");
40             String phone=request.getParameter("txtphone");
41             if ( phone.matches("[a-zA-Z]"))
42             {
43                 request.setAttribute("txtemail",email);
44                 request.setAttribute("txtfullname",fullname);
45                 request.setAttribute("txtphone",phone);
46                 request.setAttribute("ERROR","the phone is invalid");
47                 request.getRequestDispatcher("registration.jsp").forward(request, response);
48             }
49             else{
50                 int status = 1; //default a new account's status is 1
51                 int role = 0;//defaulr a new account's role is 0
52                 if (AccountDAO.insertAccount(email, password, fullname, phone, status, role)) {
53                     //response.sendRedirect("index.html");
54                     request.setAttribute("email_newAccount", email);
55                     RequestDispatcher rd = request.getRequestDispatcher("sendOTP");
56                     rd.forward(request, response);
57                 }
58                 else
59                     response.sendRedirect("errorpage.html");
60             }
61         }
62     }
}
```

Open the page “register.jsp”, display the message if any

```

<form action="mainController" method="post" class="form">
    <h1>Register</h1>
    <table>
        <tr><td>email</td>
            <td><input type="text" name="txtemail" required="" value="<%=(request.getAttribute("txtemail")==null)? "" : request.getAttribute("txtemail")%>"></td>
        <tr><td>full name</td>
            <td><input type="text" name="txtfullname" required="" value="<%=(request.getAttribute("txtfullname")==null)? "" : request.getAttribute("txtfullname")%>"></td>
        <tr><td>password</td>
            <td><input type="password" name="txtpassword" required="" value="<%=(request.getAttribute("txtpassword")==null)? "" : request.getAttribute("txtpassword")%>"></td>
        <tr><td>phone</td>
            <td><input type="text" name="txtphone" value="<%=(request.getAttribute("txtphone")==null)? "" : request.getAttribute("txtphone")%>"><%=(request.getAttribute("ERROR")==null)? "" : request.getAttribute("ERROR") %></td>
        </tr>
        <tr><td colspan="2"><input type="submit" value="register" name="action"></td></tr>
    </table>
</form>

```

The screenshot shows a web browser window with the following details:

- Address Bar:** http://localhost:8080/FlowerManagementSystem2/mainController
- Title Bar:** registration
- Header:** Home, by name, search, Login, view cart
- Content Area:**
 - Form Fields:** email (vanntn@fpt.edu.vn), full name (van), password (empty), phone (a)
 - Error Message:** the phone is invalid
 - Buttons:** register
- Page Footer:** Copyright © 2021

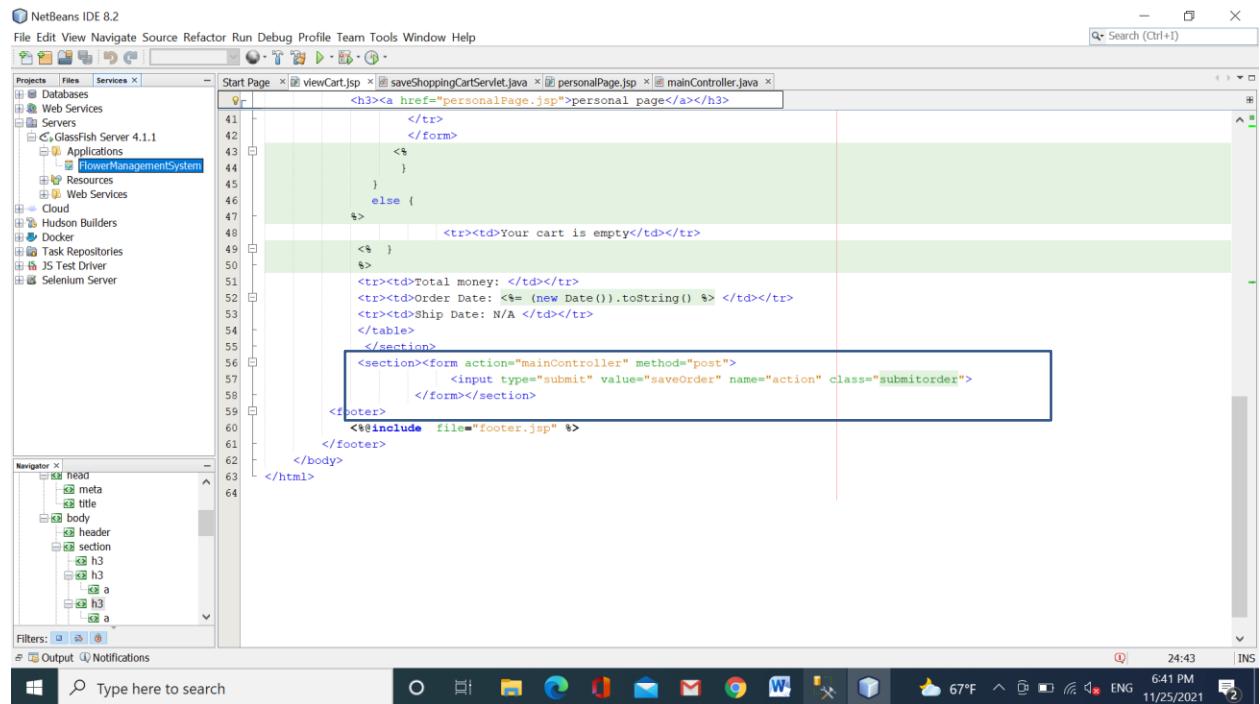


Part 5: Save shopping cart

save a new order of the customer. This function should be performed as a **transaction** that includes steps:

- ✓ Get the acld from table Accounts based on the email
- ✓ Insert a new Order to table Orders
- ✓ Insert each productid, quantity of the shopping cart to the table OrderDetails
- ✓ Commit transaction

Step 1: Open the page “viewCart.jsp”, edit the code



Step 2: Open the servlet “mainController”, edit the code:

```
else if(action.equals("saveOrder"))
    url="saveShoppingCartServlet";
RequestDispatcher rd=request.getRequestDispatcher(url);
rd.forward(request, response);
```

Step 3: Open the file “OrderDAO.java”, add more the code

```
28     public class OrderDAO {  
29         public static ArrayList<Order> getOrders(String email){...31 lines }  
30         public static ArrayList<OrderDetail> getOrderDetail(int orderID){...30 lines }  
31         public static boolean insertOrder(String email, HashMap<String, Integer> cart){  
32             Connection cn=null;  
33             boolean result=false;  
34             try{  
35                 cn = DBUtils.makeConnection();  
36                 if (cn != null) {  
37                     int accid=0,orderid=0;  
38                     cn.setAutoCommit(false); //off autocommit  
39                     //get account id by email  
40                     String sql = "select accID from Accounts where Accounts.email=?";  
41                     PreparedStatement pst = cn.prepareStatement(sql);  
42                     pst.setString(1, email);  
43                     ResultSet rs = pst.executeQuery();  
44                     if(rs!=null && rs.next()) accid = rs.getInt("accID");  
45                     //insert a new order  
46                     System.out.println("accountid:" + accid);  
47                     Date d = new Date(System.currentTimeMillis());  
48                     System.out.println("order date:" + d);  
49                     sql = "insert Orders(OrdDate,status,AccID) values(?, ?, ?)";  
50                     pst = cn.prepareStatement(sql);  
51                     pst.setDate(1, d);  
52                     pst.setInt(2, 1);  
53                     pst.setInt(3, accid);  
54                     pst.executeUpdate();  
55             }  
56         }  
57     }
```

```

114         //get order id that is the largest number
115         sql = "select top 1 orderID from Orders order by orderID desc ";
116         pst = cn.prepareStatement(sql);
117         rs = pst.executeQuery();
118         if(rs!=null && rs.next()) orderid = rs.getInt("orderID");
119         //insert order details
120         System.out.println("orderid:" + orderid);
121         Set<String> pids = cart.keySet();
122         for (String pid : pids) {
123             sql = "insert OrderDetails values(?, ?, ?) ";
124             pst = cn.prepareStatement(sql);
125             pst.setInt(1, orderid);
126             pst.setInt(2, Integer.parseInt(pid.trim()));
127             pst.setInt(3, cart.get(pid));
128             pst.executeUpdate();
129             cn.commit();
130             cn.setAutoCommit(true);
131         }
132         return true;
133     } else {
134         System.out.println("k chen order dc");
135     }
136
137 }catch(Exception e){
138     try{
139         if(cn!=null) cn.rollback();
140     }catch(SQLException ex){
141         ex.printStackTrace();
142     }
143     e.printStackTrace();
144     result=false;
145 }
146 finally{ try {
147     if(cn!=null) cn.close();
148 } catch (Exception e) { e.printStackTrace(); }
149 }
150
151     return result;
152 }
153 }
154

```

Step 3: New a servlet named “saveShoppingCartServlet”

Step 4: Open the page “viewCart.jsp”,add more the code:

The screenshot shows the NetBeans IDE interface with the following details:

- MenuBar:** File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help.
- Toolbar:** Standard Java IDE tools like New, Open, Save, Cut, Copy, Paste, Find, etc.
- Project Explorer:** Shows a project named "GlassFish Server 4.1.1" containing an application "FlowerManagementSystem". Other nodes include Databases, Web Services, Servers, Cloud, Hudson Builders, Docker, Task Repositories, JS Test Driver, and Selenium Server.
- Navigator:** Shows the current file structure for "viewCart.jsp", including sections, h3, a, font, and table elements.
- Code Editor:** Displays the JSF page source code. The code includes JSP tags, Java beans, and CSS styles. It features a welcome message, session attributes, and a shopping cart table. A red box highlights the line: `<%=(request.getAttribute("WARNING")==null)?":(String)request.getAttribute("WARNING")%>`.
- Status Bar:** Shows the current file as "JavaWeb_Setup_Workshop_Assignment_Plan - Microsoft Word (Product Activation Failed)".
- System Tray:** Shows the date and time as "7:05 PM 11/25/2021" and system icons.

Step 5: Open the page “login.jsp”, add more the code:

The screenshot displays the NetBeans IDE interface for the FlowerManagementSystem project. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, and Help. The left sidebar shows the Project Explorer with Web Pages, META-INF, WEB-INF, and Images sections containing files like errorpage.html, footer.jsp, header.jsp, header_loggedinUser.jsp, index.html, index.jsp, invalid.html, login.html, login.jsp, mycss.css, orderDetail.jsp, personalPage.jsp, registration.html, registration.jsp, testError.jsp, testCart.jsp, viewCart.jsp, and welcome.html. Below it is the Source Packages, Test Packages, Libraries, and Test Libraries section. The Navigator pane on the left provides a hierarchical view of the page structure, including nodes for 'td', 'input', 'tr', and 'footer'. The main workspace shows the 'Start Page' with the file 'login.jsp' open. The code editor contains JSP code with annotations like <%@page content-type="text/html" page-encoding="UTF-8" %>, <%@include file="header.jsp" %>, and <%>. It also includes HTML tags such as <head>, <body>, <header>, <section>, <form>, and <table>. A specific line of code, <%=(request.getAttribute("WARNING")==null)?:"":(String)request.getAttribute("WARNING")%>, is highlighted with a blue box. The bottom status bar shows the date as 11/25/2021 and the time as 6:59 PM.

Step 6: Run your app

Test case 1: Submit the order without logging

http://localhost:8080/FlowerManagementSystem2/mainController

JSP Page

Home Register Login view cart

you must login to finish the shopping

Product id

1	quantity 1	action update delete
2	1	update delete

Total money:
Order Date: Thu Nov 25 19:06:52 ICT 2021
Ship Date: N/A

saveOrder

Copyright © 2021

Test case 2: submit the order without any selected product

http://localhost:8080/FlowerManagementSystem2/mainController

JSP Page

Home Register Login view cart

your cart is empty

Product id
Your cart is empty

Total money:
Order Date: Thu Nov 25 19:08:37 ICT 2021
Ship Date: N/A

saveOrder

Copyright © 2021

Test case 3: submit the order successfully

http://localhost:8080/FlowerManagementSystem2/viewCart.jsp

JSP Page

Home Register Login view cart

Welcome test come back

Logout

personal page

Product id

1	quantity 1	action update delete
2	1	update delete

Total money:
Order Date: Thu Nov 25 19:09:56 ICT 2021
Ship Date: N/A

saveOrder

Copyright © 2021



Welcome test come back

[Logout](#)

personal page

save your cart sucessfully

Product id	quantity	action
Your cart is empty		
Total money:		
Order Date: Thu Nov 25 19:11:16 ICT 2021		
Ship Date: N/A		

saveOrder

Copyright © 2021



Click the link “personalPage” to view history

Welcome test come back

[Logout](#)

Order ID	Order Date	Ship Date	Order's status	action
1	2021-01-11	2021-10-11	completed	detail
2	2021-11-23	null	processing cancel order	detail
3	2021-10-01	null	cancel canceled	detail
35	2021-11-25	null	processing cancel order	detail
36	2021-11-25	null	processing cancel order	detail

Copyright © 2021



The advanced challenge:

- 1) You must finish the delete the product on the shopping cart
- 2) On page “viewCart.jsp”, you get the price, image of each product
- 3) Get the total money on the shopping cart

- 4) When the user clicks on the link in the column “product id”, your app will display the detail of this product.
- 5) Filter orders based on from..to. The result will be displayed on the page “personalPage.jsp”

Congratulations!!!! you have finished workshop 5 ☺ ☺

Workshop 6: JavaBean

JSP standard action # Expression language (EL)

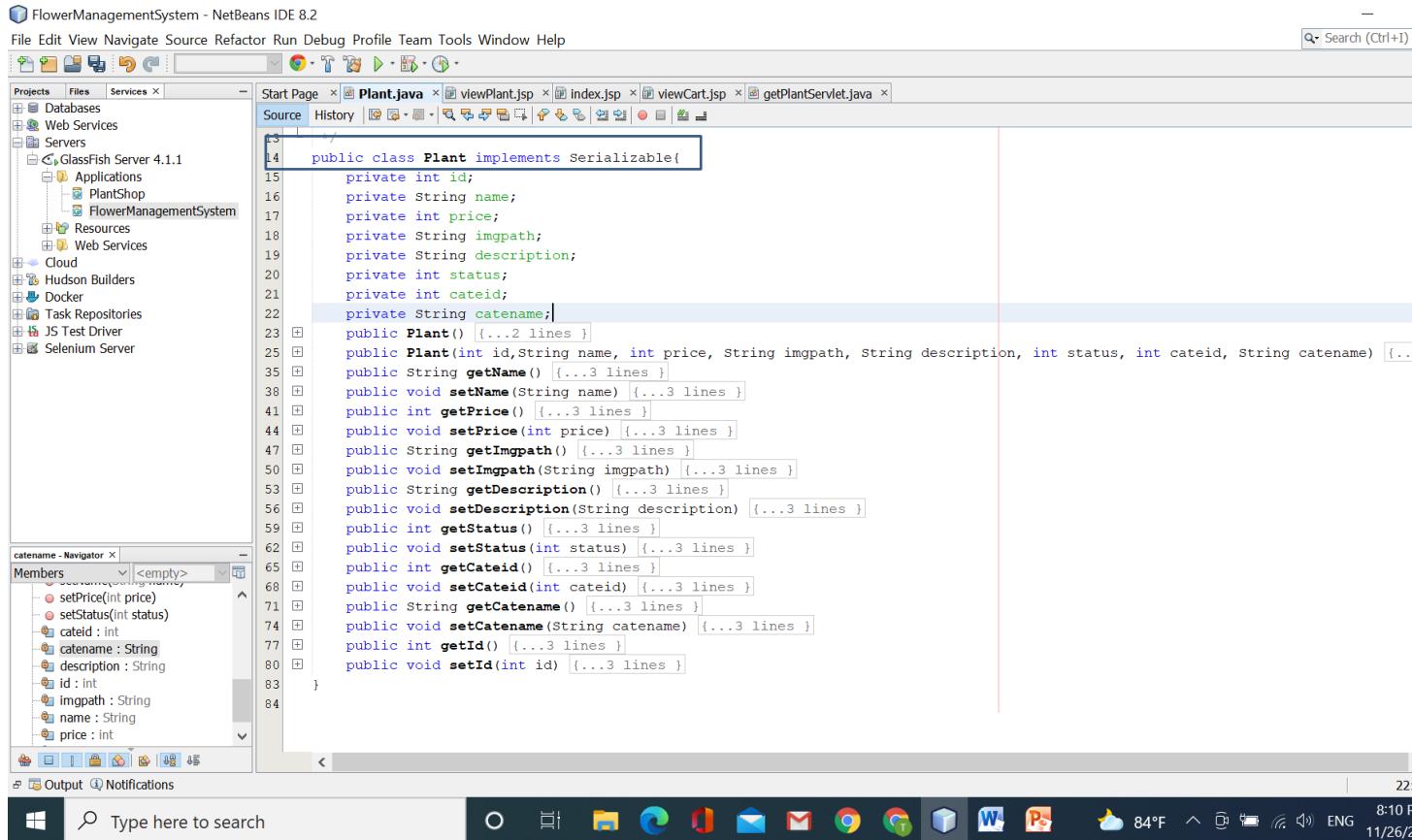
Learning outcome:

- Understand JavaBean and use JSP standard action to presentation
- Understand and apply EL in JSP pages

On the page “viewCart.jsp”, when the user clicks on the product’s id, its detail will be displayed.

Step 1: Open the file “Plant.java. the Plant class can be used as **Bean**, it must adapt the followings:

- Java Bean **implemented from Serializable**
- Bean class should **always use a package name**
- Bean class **must have a public no-argument constructor**
- The **properties of bean** (persistence) are **not declared “public”**. They are **accessed through getter and setter methods**.
- The public **getter** method is used to **retrieve the properties of a bean class**
- The public **setter** method is used to **set the properties of a bean class**
- The first character of each property should **name in lower case** then the **accessor methods** are used along with the property name **with the first character of each word in upper case (Ex: length – getLength and setLength)**
- The dataType of properties is **boolean** then the getter method is **isXxx** instead of **getXxx**



Step 2: Open the page “viewCart.jsp”, add more the code:

```
<form action="mainController" method="post" >
    <tr> <td><input type="hidden" value="<%= pid %>" name="pid"><a href="getPlantServlet?pid=<%= pid %>"><% pid %></a></td>
    <td><input type="number" value="<%= quantity %>" name="quantity"></td>
    <td><input type="submit" value="update" name="action" >
    <input type="submit" value="delete" name="action" ></td>
</tr>
</form>
```

Step 3: Open the file “PlantDAO.java”, add the method “getPlant(int pid)”. Return the plant object based on the given productid.

Start Page × PlantDAO.java ×

Source History

```
21 public static ArrayList<Plant> getPlants(String keyword, String searchby) { ... 41 lines ... }
22
23 public static Plant getPlant(int pid) {
24     Plant p=null;
25     Connection cn=null;
26     try {
27         cn=DBUtils.makeConnection();
28         if(cn!=null){
29             String sql = "select PID,PName,price,imgPath,description,status,Plants.CateID as cateID,CateName\n"
30                     + "from Plants, Categories\n"
31                     + "where Plants.CateID=Categories.CateID and PID=? ";
32             PreparedStatement pst=cn.prepareStatement(sql);
33             pst.setInt(1, pid);
34             ResultSet rs=pst.executeQuery();
35             if(rs!=null && rs.next())
36             {
37                 pid=rs.getInt("PID");
38                 String pname=rs.getString("PName");
39                 int price=rs.getInt("price");
40                 String imgPath=rs.getString("imgPath");
41                 String des=rs.getString("description");
42                 int status=rs.getInt("status");
43                 int cateid=rs.getInt("cateID");
44                 String cateName=rs.getString("CateName");
45                 p=new Plant(pid, pname, price, imgPath, des, status, cateid, cateName);
46             }
47         }
48     } catch (Exception e) {
49         e.printStackTrace();
50     }
51     finally{
52         try {
53             } catch (Exception e) {
54                 e.printStackTrace();
55             }
56     }
57 }
58
59 return p;
60 }
```

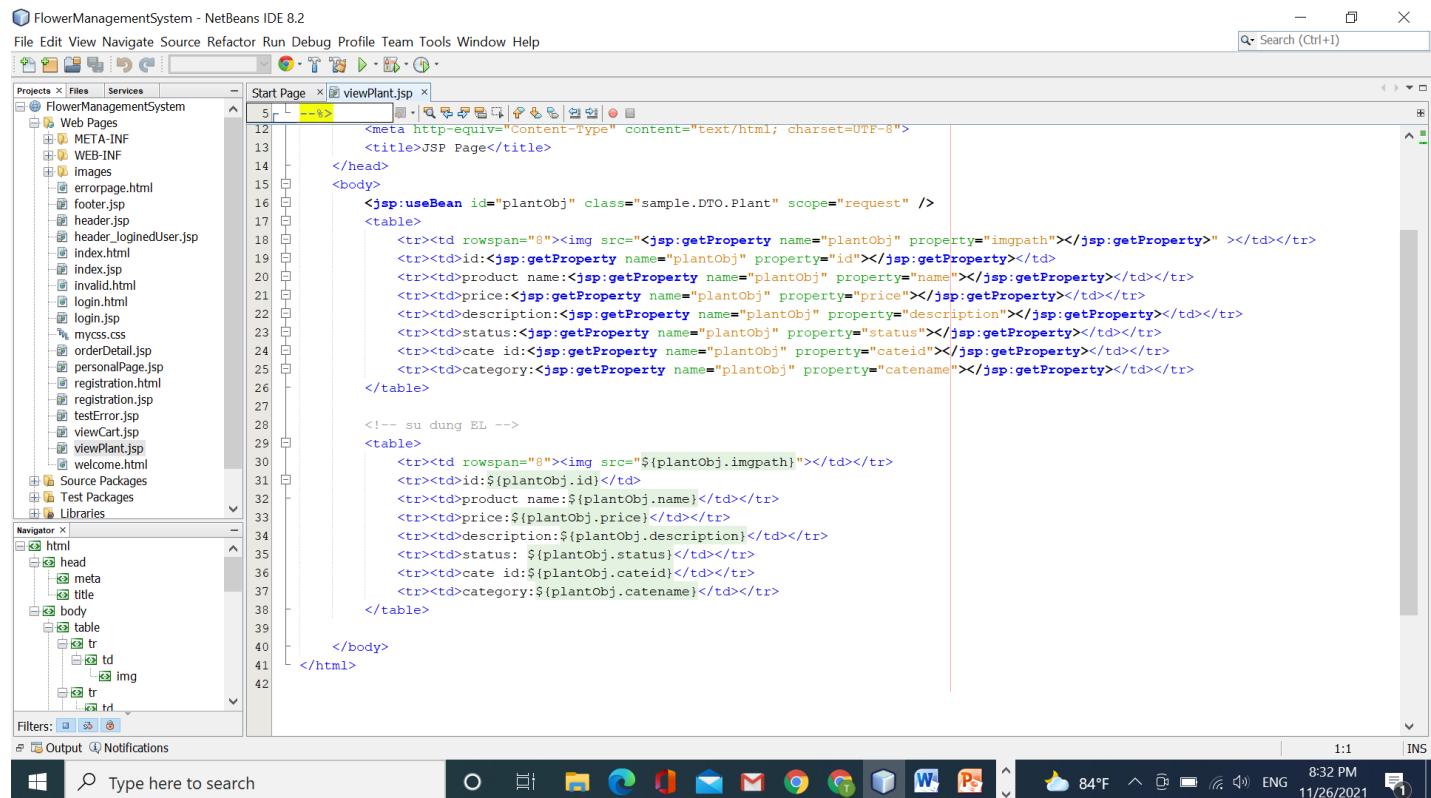
Step 4: In the package “sample.servlet”, create a new servlet named “getPlantServlet”.

```

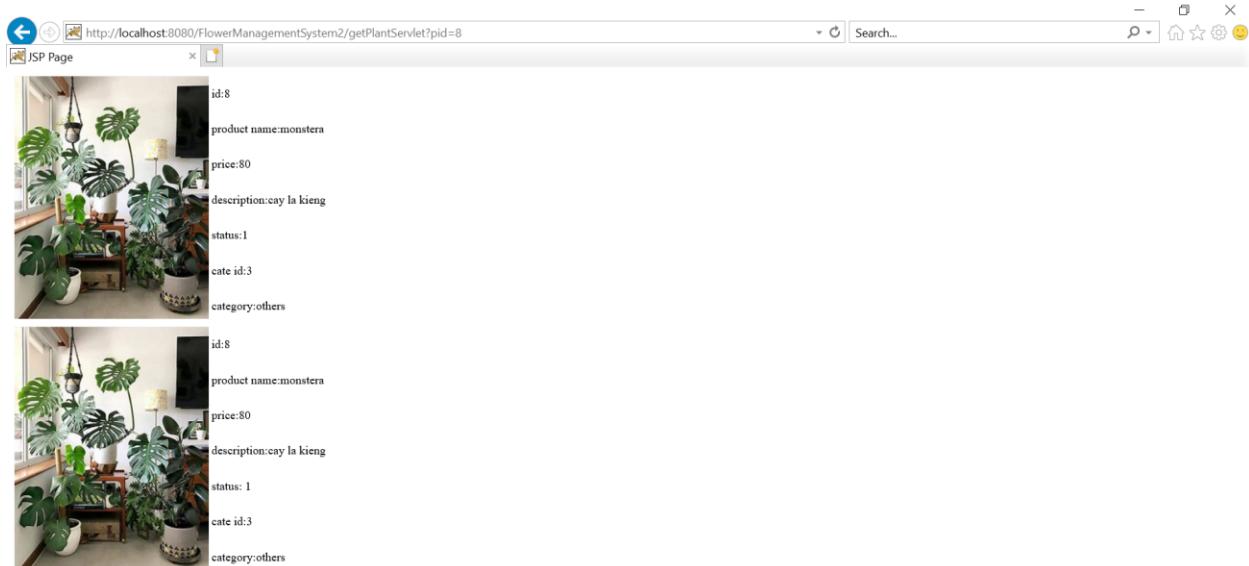
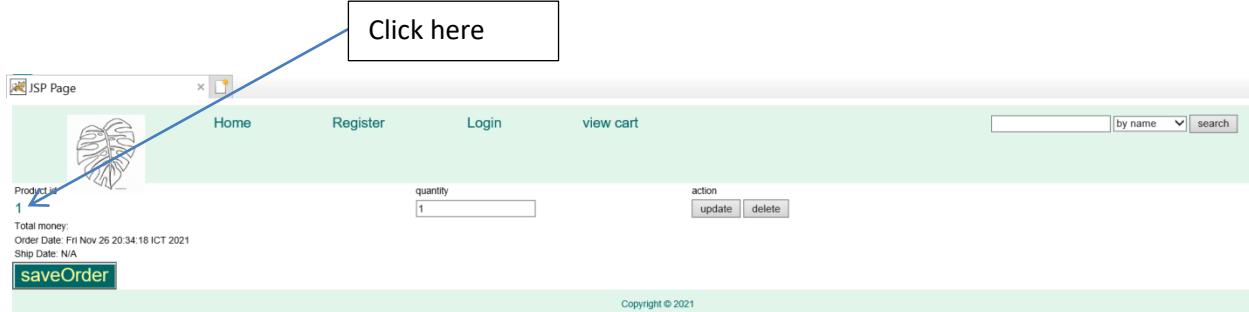
6   package sample.servlet;
7   import ...8 lines
15
16  /**
17   * ...4 lines */
18  public class getPlantServlet extends HttpServlet {
19
20      /**
21       * Processes requests for both HTTP <code>GET</code> and <code>POST</code> ...9 lines */
22      protected void processRequest(HttpServletRequest request, HttpServletResponse response)
23          throws ServletException, IOException {
24          response.setContentType("text/html;charset=UTF-8");
25          try (PrintWriter out = response.getWriter()) {
26              /* TODO output your page here. You may use following sample code. */
27              int pid=Integer.parseInt(request.getParameter("pid"));
28              Plant p=PlantDAO.getPlant(pid);
29              if(p!=null) {
30                  request.setAttribute("plantObj", p);
31                  request.getRequestDispatcher("viewPlant.jsp").forward(request, response);
32              }
33          }
34      }
35
36      /**
37       * HttpServlet methods. Click on the + sign on the left to edit the code.
38      */
39
40  }
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74

```

Step 5: Create a new page named “viewPlant.jsp”. For now, we use two ways to display information of the plant (<jsp standard action, EL)



Step 6: run your app



The advanced challenge: now, you should use JavaBean or EL to update all JSP pages in your app.

Hint: Before doing advanced challenges, you should back up your app to restore if any ☹☹.

Congratulations!!!! you have finished workshop 6 ☺ ☺

Workshop 7: JSP Standard Tag Library(JSTL)

#JSTL #CustomTag #Dynamic

Learning outcome:

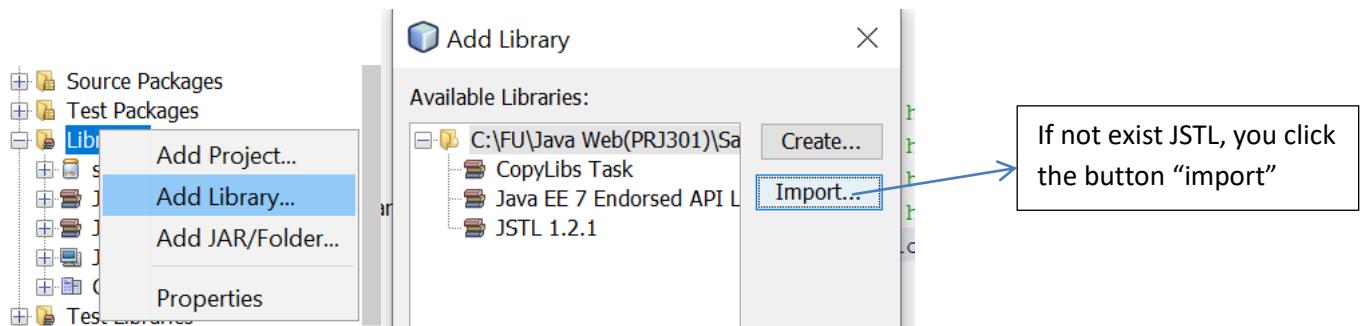
- How to apply JSTL to remove all java code in the JSP pages

Requirements: The admin functions after login:

- Manage accounts: view all accounts, block/unblock an account
- View all orders, filter orders between from date and to date, filter orders of any customer.

- Manage plants (view,create,update plants)
- Manage categories: view, create, update
- Log out

To begin to use the tag library, your project needs to do the following:



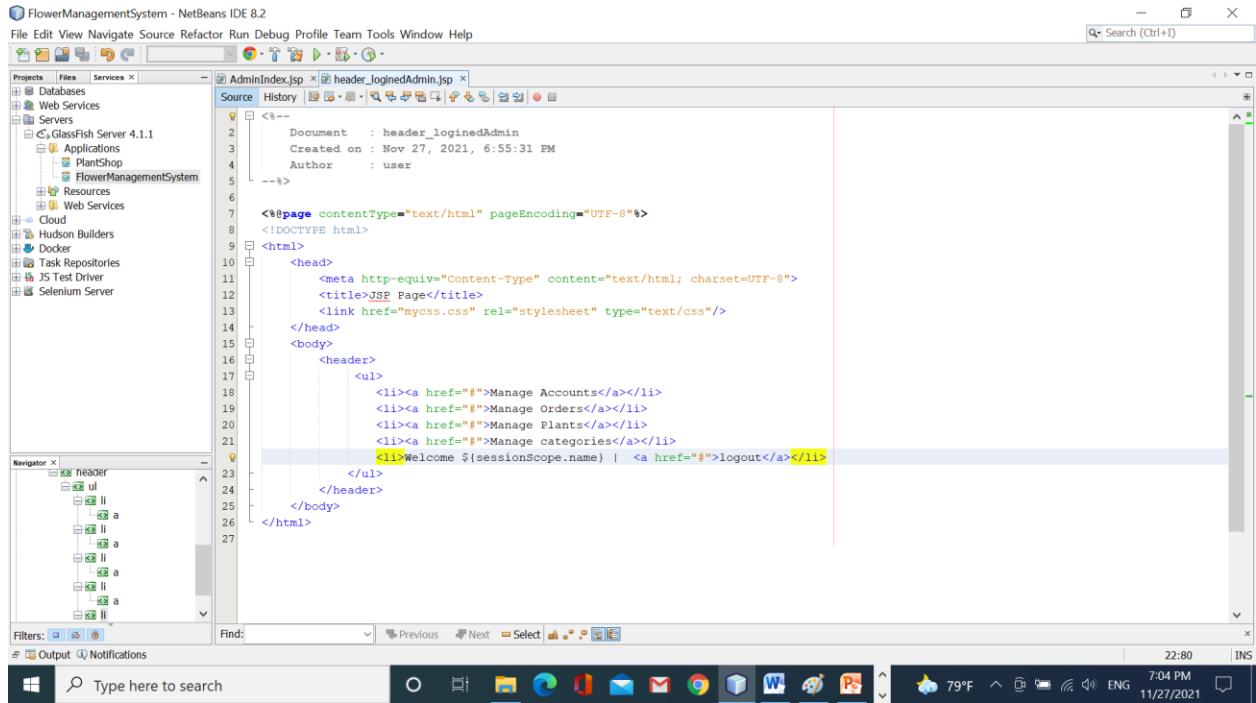
Part 1: Create a home page for the admin

Create a new page named “AdminIndex.jsp”. After login in the page “AdminIndex.jsp” will be shown for the admin.

Step 1: Open the servlet “loginServlet.java”, add more the code:

```
//admin
if (acc.getRole() == 1) {
    //chuyen qua admin home page
    HttpSession session = request.getSession(true);
    if (session != null) {
        session.setAttribute("name", acc.getFullname());
        session.setAttribute("email", email);
        //create a cookie and attach it to response object
        if (save != null) {
            String token = "ABC123"; //this is a sample to study.
            AccountDAO.updateToken(token, email);
            Cookie cookie = new Cookie("selector", token);
            cookie.setMaxAge(60 * 2); //this is a sample
            response.addCookie(cookie);
        }
        response.sendRedirect("AdminIndex.jsp");
    }
} //user
else {
    // ...
}
```

Step 2: Create a new page named “header_loggedinAdmin.jsp”



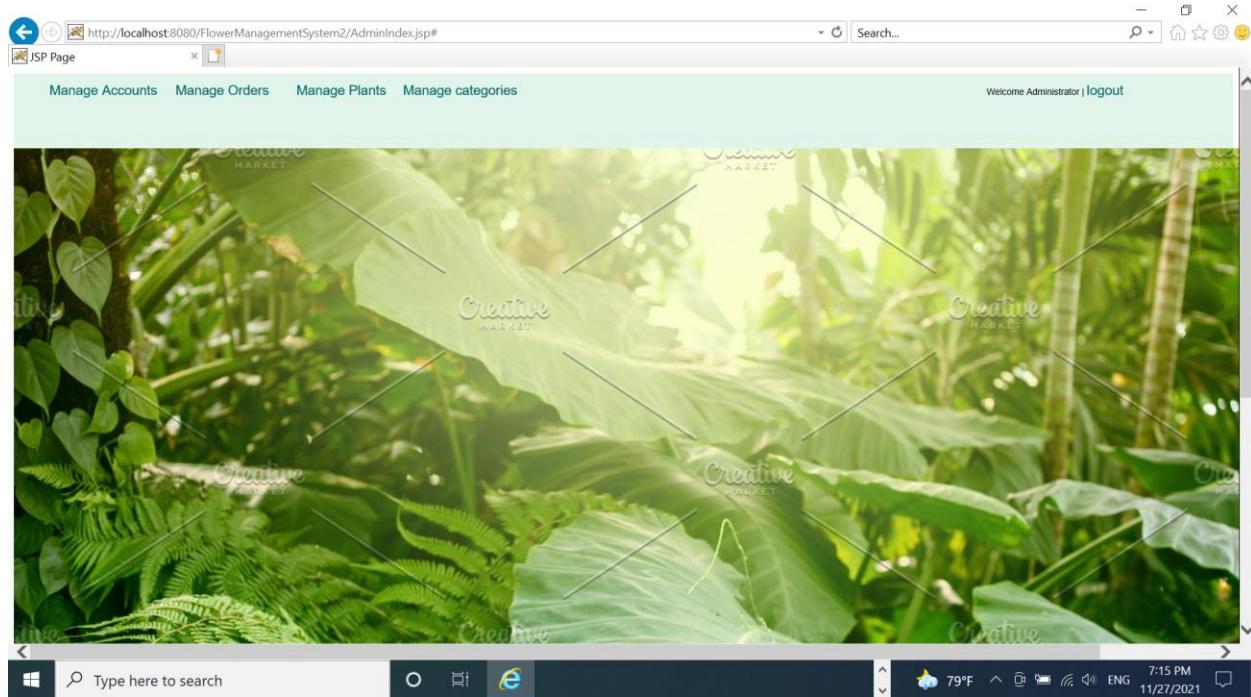
Step 3: Create a new page named “AdminIndex.jsp”

```

6
7   <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
8   <%@page contentType="text/html" pageEncoding="UTF-8"%>
9   <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
10  <!DOCTYPE html>
11  <html>
12    <head>
13      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
14      <title>JSP Page</title>
15      <link href="mycss.css" rel="stylesheet" type="text/css"/>
16    </head>
17    <body>
18      <c:import url="header_loggedinAdmin.jsp" />
19      <section class="right">
20        
21      </section>
22    </body>
23  </html>
24

```

Step 4: run your app to get the result



To finish this part, this index page should not be shown for illegal users. Review the page "personalPage.jsp" to do.

Part 2: Perform the function "manage Accounts"

When the admin clicks the link "manage Accounts", the page "manageAccount.jsp" will be shown. The admin will perform block/unblock account of the user (update the column status of the account whose role is 0)

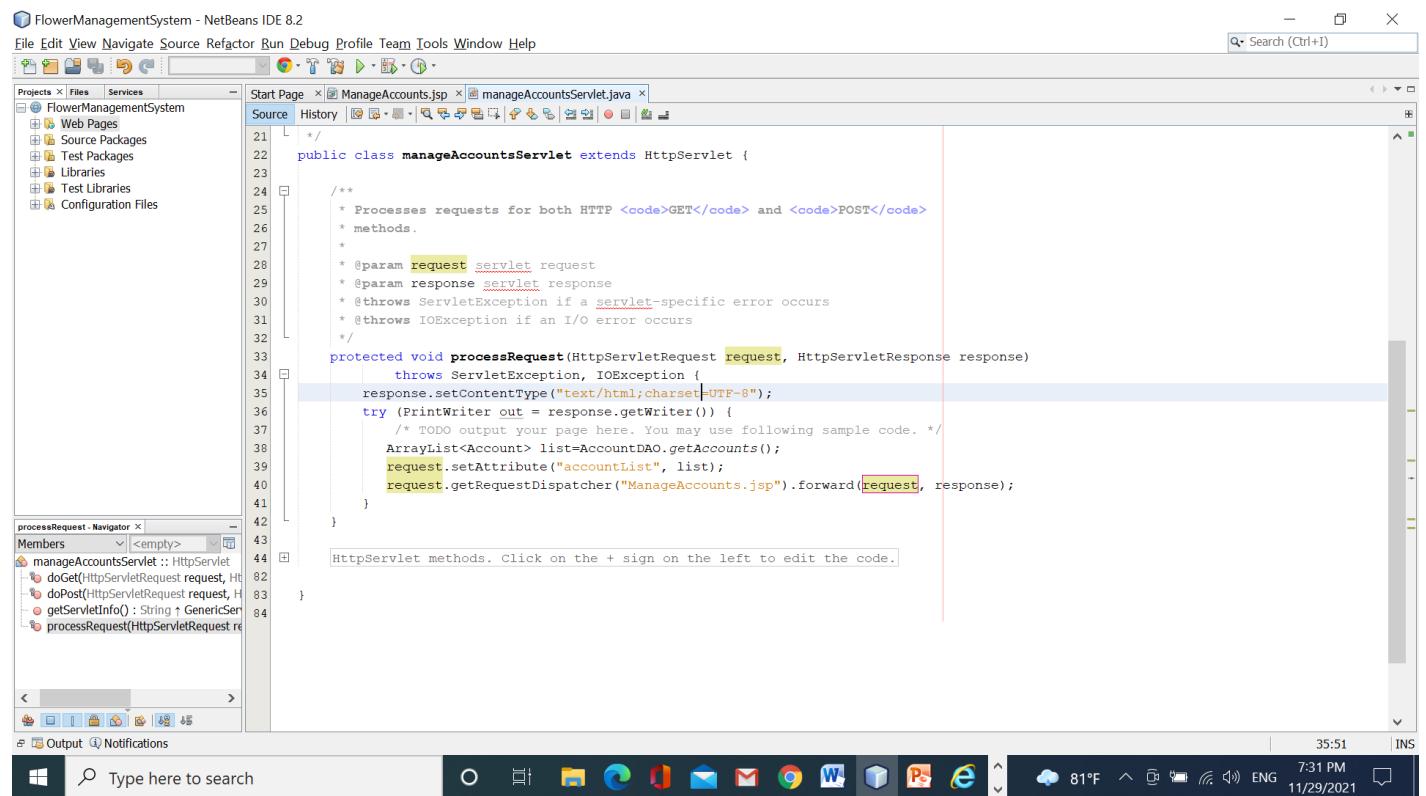
Step 1: Open the page "header_loggedinAdmin.jsp", add the code

```
<header>
  <ul>
    <li><a href="mainController?action=manageAccounts">Manage Accounts</a></li>
    <li><a href="#">Manage Orders</a></li>
    <li><a href="#">Manage Plants</a></li>
    <li><a href="#">Manage categories</a></li>
    <li>Welcome ${sessionScope.name} | <a href="#">logout</a></li>
  </ul>
</header>
```

Step 2: In the servlet “mainController”, add the code:

```
56     else if(action.equals("manageAccounts"))
57         url="manageAccountsServlet";
58         RequestDispatcher rd=request.getRequestDispatcher(url);
59         rd.forward(request, response);
60     }
61 }
```

Step 3: Create a new servlet named “manageAccountsServlet.java”



Step 4: Create a new page named “ManageAccounts.jsp”

Start Page × ManageAccounts.jsp ×

Source History | Back Forward | Find | Replace | Go To | Open In New Tab | Close

```
10  <%@page contentType="text/html" pageEncoding="UTF-8"%>
11  <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
12  <!DOCTYPE html>
13  <html>
14      <head>
15          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
16          <title>JSP Page</title>
17          <link href="mycss.css" rel="stylesheet" type="text/css"/>
18      </head>
19      <body>
20          <c:import url="header loggedinAdmin.jsp"/>
21          <form action="mainController" method="post">
22              <input type="text" name="txtSearch">
23              <input type="submit" value="searchAccount" name="action">
24          </form>
25          <h1></h1>
26          <table class="order">
27              <tr><th> ID</th>
28                  <th> Email</th>
29                  <th> Full name</th>
30                  <th> status</th>
31                  <th> phone</th>
32                  <th> role</th>
33                  <th> action</th>
34
35                  </tr>
36                  <c:forEach var="acc" items="${requestScope.accountList}">
37                      <tr><c:out value="${acc.getAccID()}"></c:out></td>
38                          <td><c:out value="${acc.getEmail()}"></c:out></td>
39                          <td><c:out value="${acc.getFullname()}"></c:out></td>
40                          <td><c:choose >
41                              <c:when test="${acc.getStatus() eq 1}"> active</c:when>
42                              <c:otherwise>inActive</c:otherwise>
43                          </c:choose>
44                          </td>
45                          <td><c:out value="${acc.getPhone()}"></c:out></td>
46                          <td>
47                              <c:choose >
48                                  <c:when test="${acc.getRole() eq 1}"> admin</c:when>
49                                  <c:otherwise>user</c:otherwise>
50                              </c:choose>
51                      </td>
52                      <c:if test="${acc.getRole() eq 0}">!-- only block/unblock account of the user-->
53                          <c:url var="mylink" value="mainController">
54                              <c:param name="email" value="${acc.getEmail()}"></c:param>
55                              <c:param name="status" value="${acc.getStatus()}"></c:param>
56                              <c:param name="action" value="updateStatusAccount"></c:param>
57                          </c:url>
58                          <a href="${mylink}">Block/UnBlock</a>
59                      </c:if>
60                  </tr>
61          </c:forEach>
62      </table>
63  </section>
64  </body>
65  </html>
```

Step 5: run your app to get the result

ID	Email	Full name	status	phone	role	action
1	test@gmail.com	test	active	123456	user	Block/UnBlock
2	admin@gmail.com	Administrator	active	123456	admin	Block/UnBlock

Part 3: Perform the block/unblock the account

When the admin clicks the links “block/unblock”, the account’s status will be changed

Step 1: Open the servlet “mainController”, add the code

```
else if(action.equals("updateStatusAccount"))
    url="updateStatusAccountServlet";
RequestDispatcher rd=request.getRequestDispatcher(url);
rd.forward(request, response);
```

Step 2: create a new servlet named “updateStatusAccountServlet.java”

```
16 /**
17 *
18 * @author user
19 */
20 public class updateStatusAccountServlet extends HttpServlet {
21
22     /** Processes requests for both HTTP <code>GET</code> and <code>POST</code> ... 9 lines */
23     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
24             throws ServletException, IOException {
25         response.setContentType("text/html; charset=UTF-8");
26         PrintWriter out = response.getWriter();
27         /* TODO output your page here. You may use following sample code. */
28         String email=request.getParameter("email");
29         int status=Integer.parseInt(request.getParameter("status"));
30         if(status==1)//active
31             AccountDAO.updateAccountStatus(email, 0);
32         else
33             AccountDAO.updateAccountStatus(email, 1);
34         request.getRequestDispatcher("mainController?action=manageAccounts").forward(request, response);
35     }
36
37     //HttpServlet methods. Click on the + sign on the left to edit the code.
38 }
39
40 }
```

Use this code to load all accounts again

Step 3: run your app

The screenshot shows a JSP page titled "Manage Accounts" with a table of account data. The table has columns: ID, Email, Full name, status, phone, role, and action. The "status" column contains two rows: one for ID 1 (Email: test@gmail.com) with status "inActive" and one for ID 2 (Email: admin@gmail.com) with status "active". The "status" column header is highlighted with a red border.

ID	Email	Full name	status	phone	role	action
1	test@gmail.com	test	inActive	123456	user	Block/UnBlock
2	admin@gmail.com	Administrator	active	123456	admin	

The advanced challenge:

- 1) On the page “manage accounts”, you must finish the search account function.
- 2) Perform the functions: manage Orders, Manage plants, manage categories
- 3) Finish the logout function (the same as the user’s logout)

Congratulations!!!! you have finished workshop 7 ☺ ☺

Workshop 8: Filter

Filter chain #Request/Response Wrapper class # Fileter as Controller

Learning outcome:

- Understand the role of filter in the JavaEE
- Understand the role of request/response wrapper class
- Know to apply filter as the main controller

Requirements: Student must perform all demos in the lecture.

Congratulations!!!! you have finished workshop 8 ☺ ☺