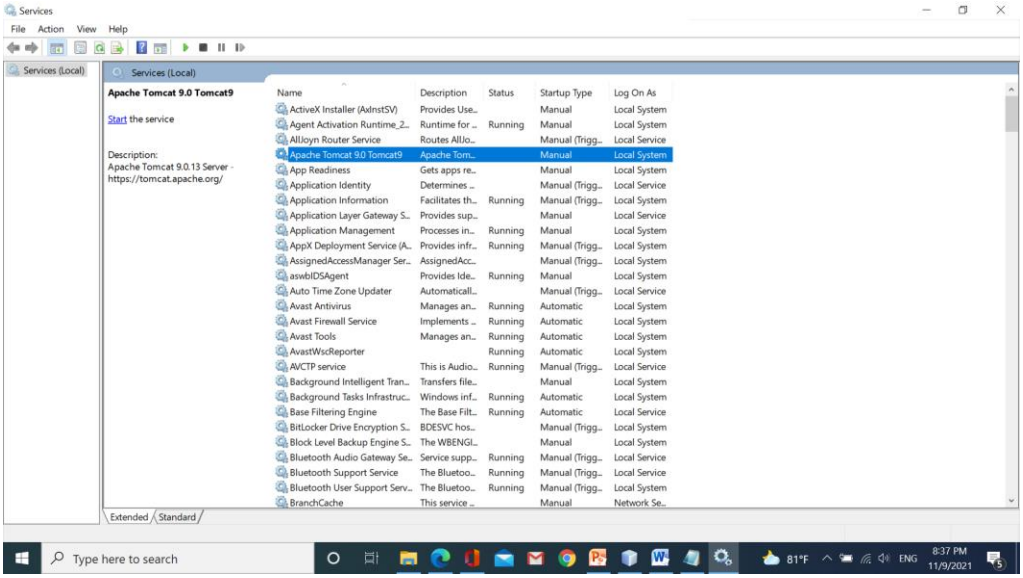
Guideline setup

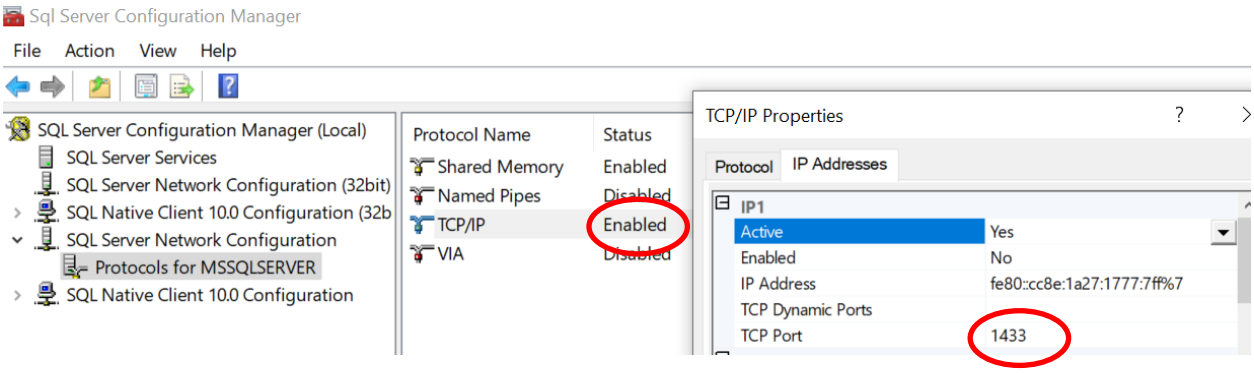
1) Download and install Webserver (Tomcat)
   How to download and install Tomcat 9 on windows - YouTube\
   Or you can use any webserver(for example: GlassFish server).
2) Start or Stop webserver by the service window
   Open the service window



   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

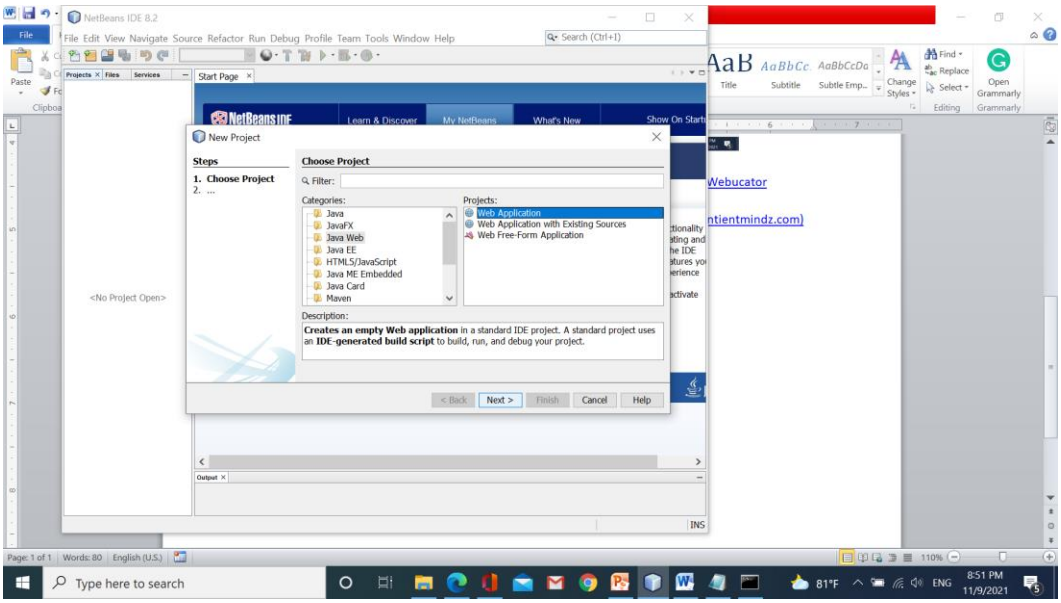


7) Create a database for doing all workshops and assignments. Run the code below

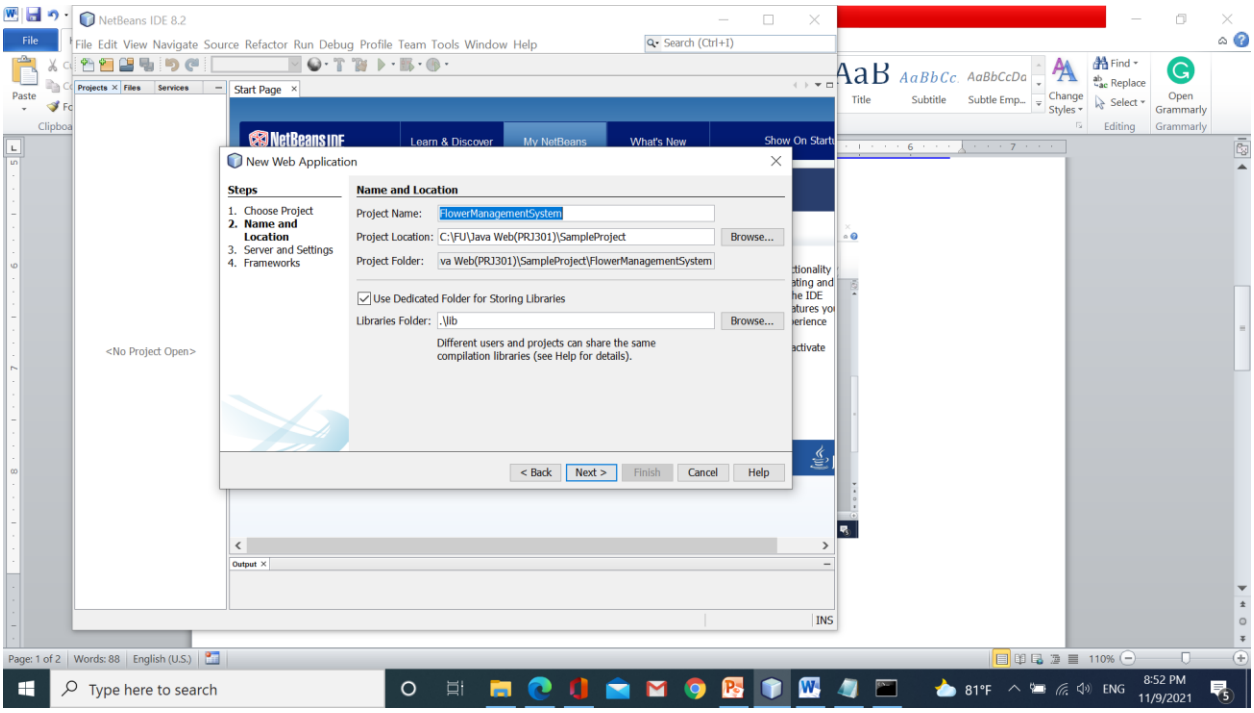
```
create database PlantShop
GO
use PlantShop
GO
create table Accounts(
   accID int identity(1,1)primary key,
   email varchar(30)  unique,
   password varchar(30),
   fullname varchar(30),
   phone varchar(12),
   status int check(status =1 or status=0),-- 1:active; 0:inactive
   role int check(role=1 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 JavaWeb Project

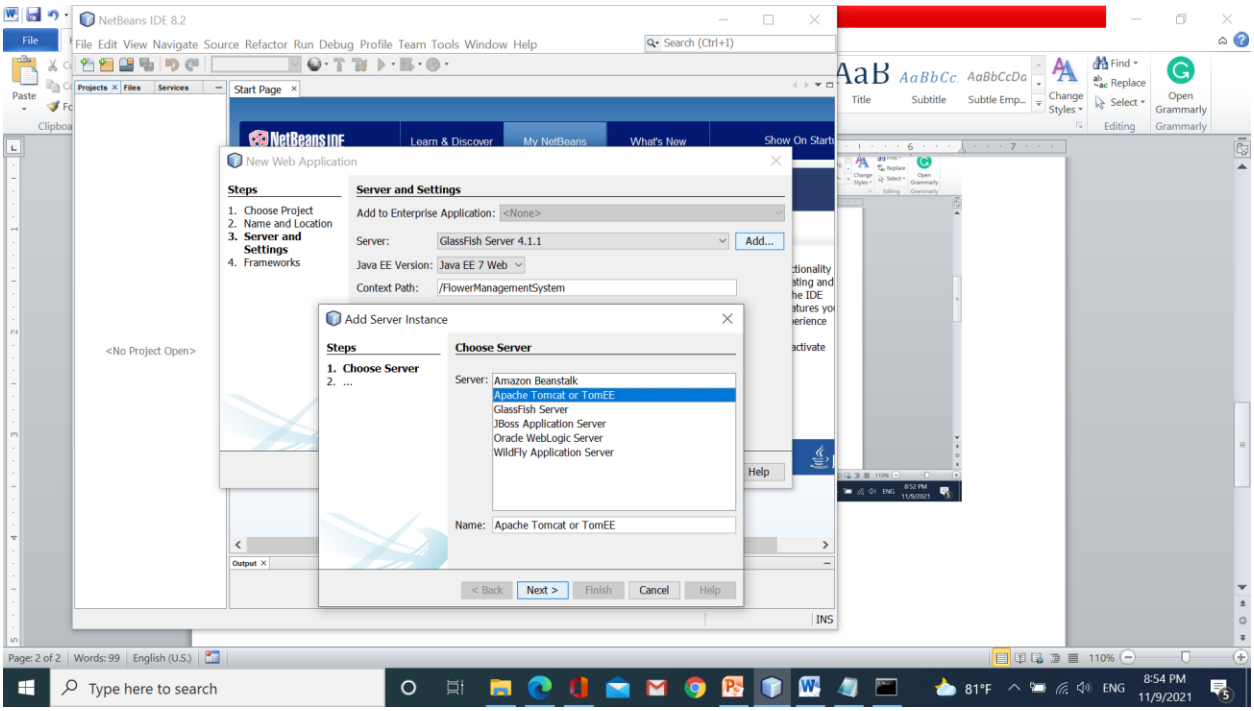
step 1: open Netbean/new Project/Java web/Web application



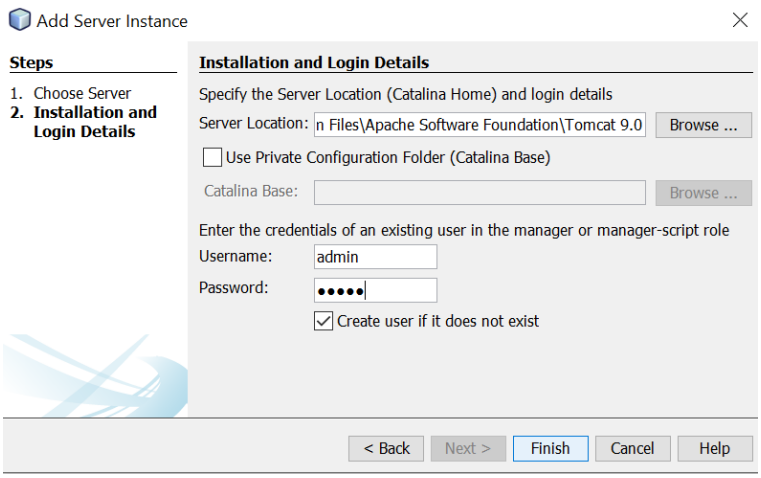
step 2: give a project name, project location and click Next



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 10^{th} week.
- All basic tasks are highlighted, others are advanced tasks
- The practical exam will be performed in the 9^{th} week.

Workshop 1: Learn JDBC, Servlet

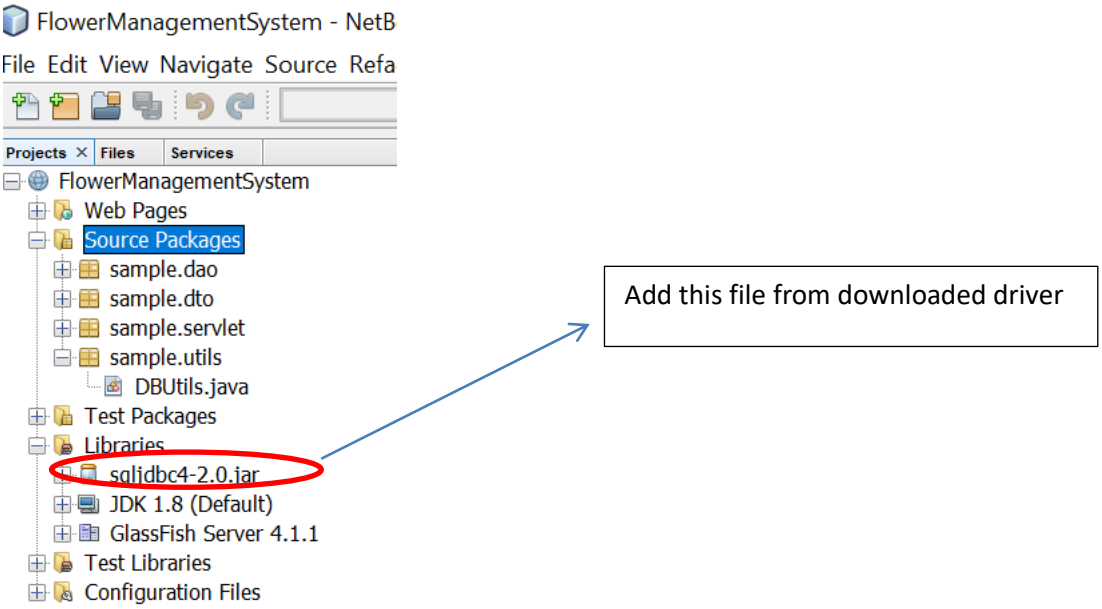
#HTML #Form # JDBC #Request Object #Response Object # Servlet

Learning outcome:

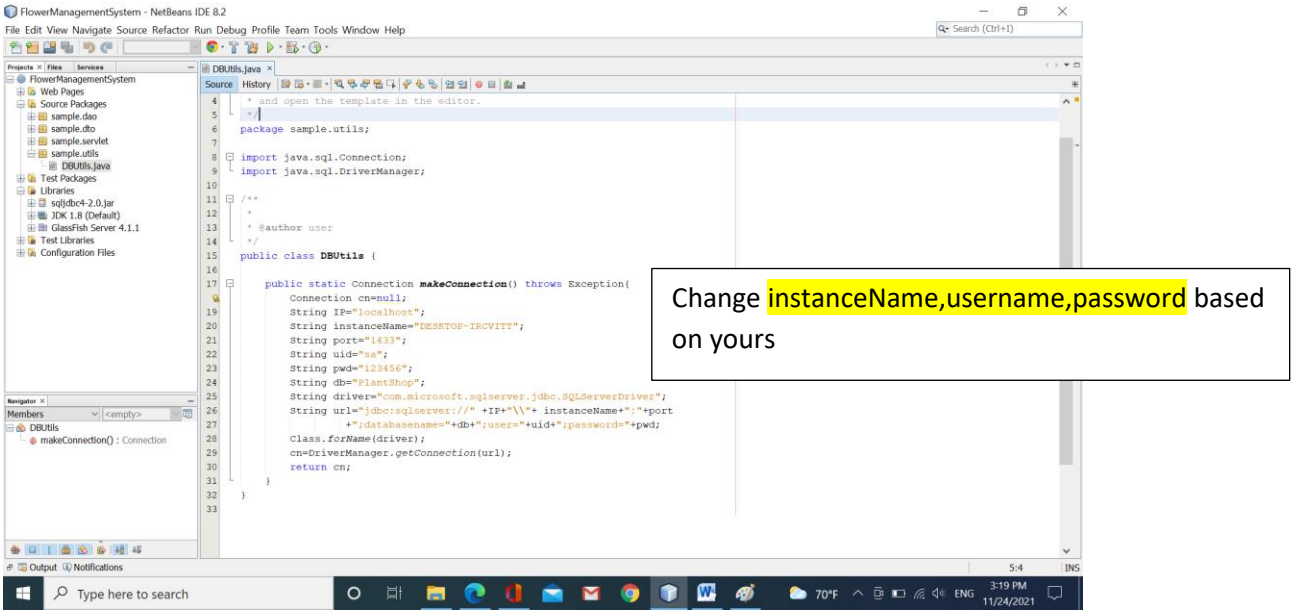
- Learn JDBC
- Learn servlet
- Perform the login function

Part 1: Learning JDBC

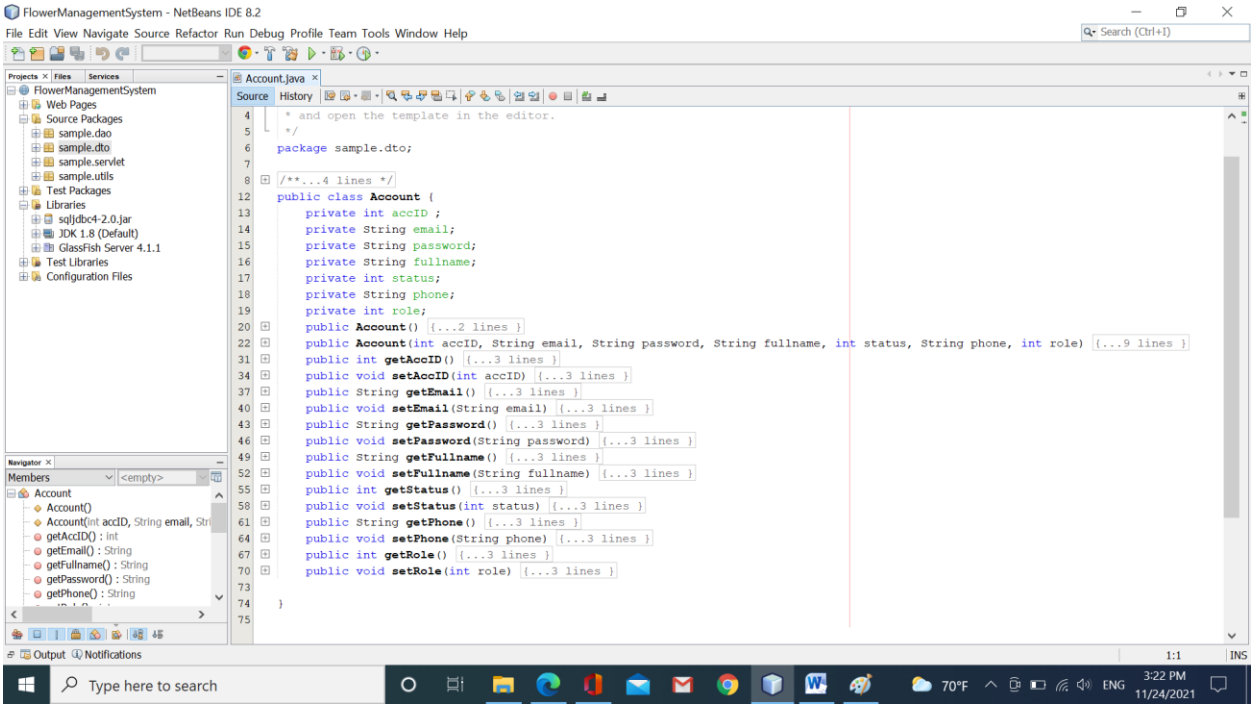
Step 1: The student creates some packages, files like the figure:



DBUtils.java

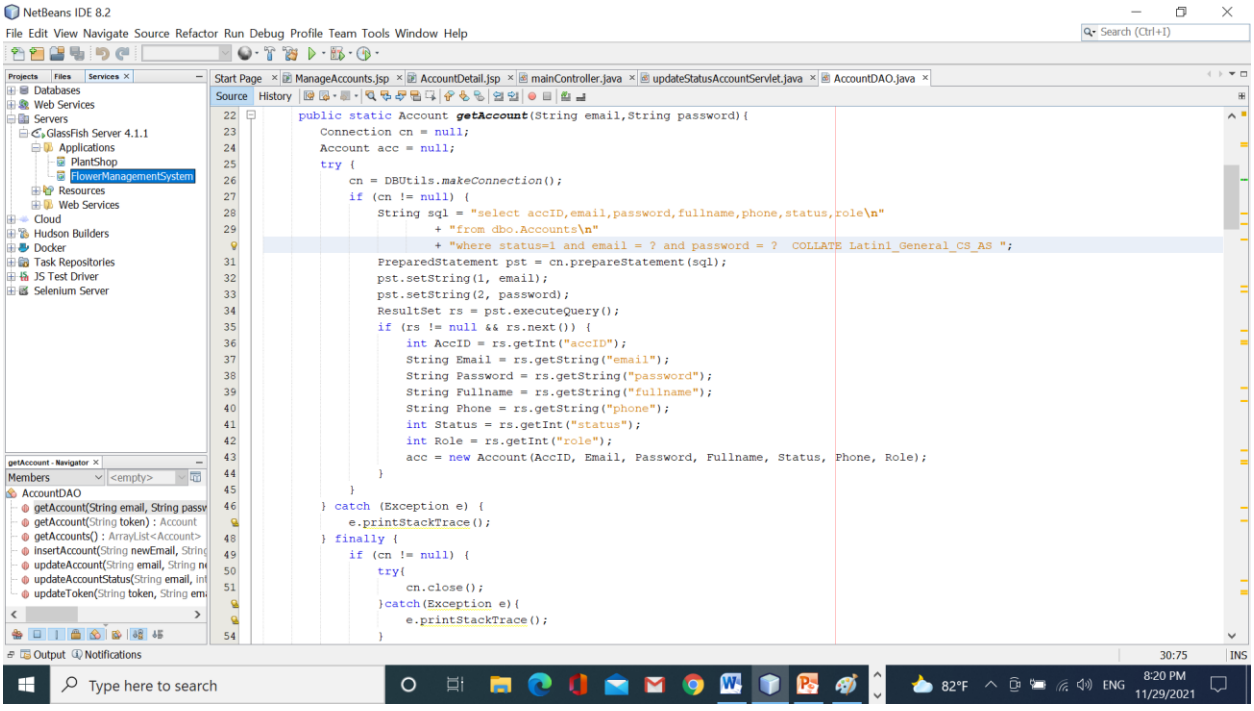


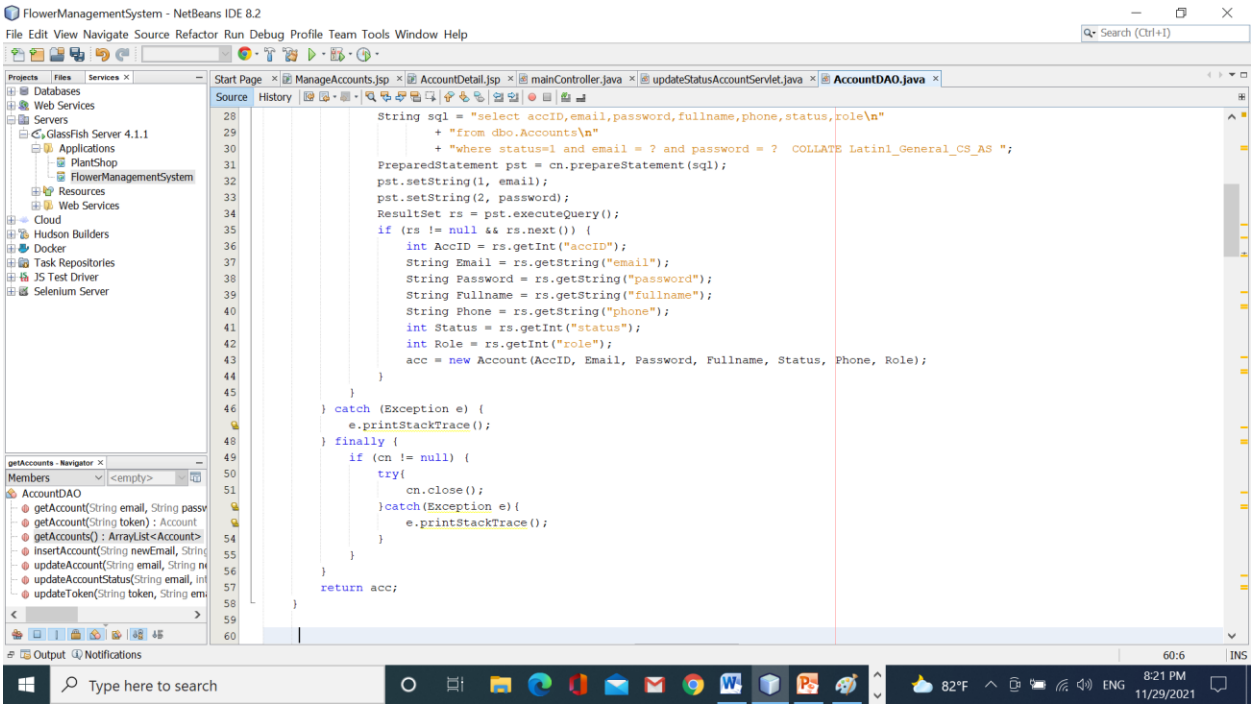
Account.java



AccountDAO.java

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





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:

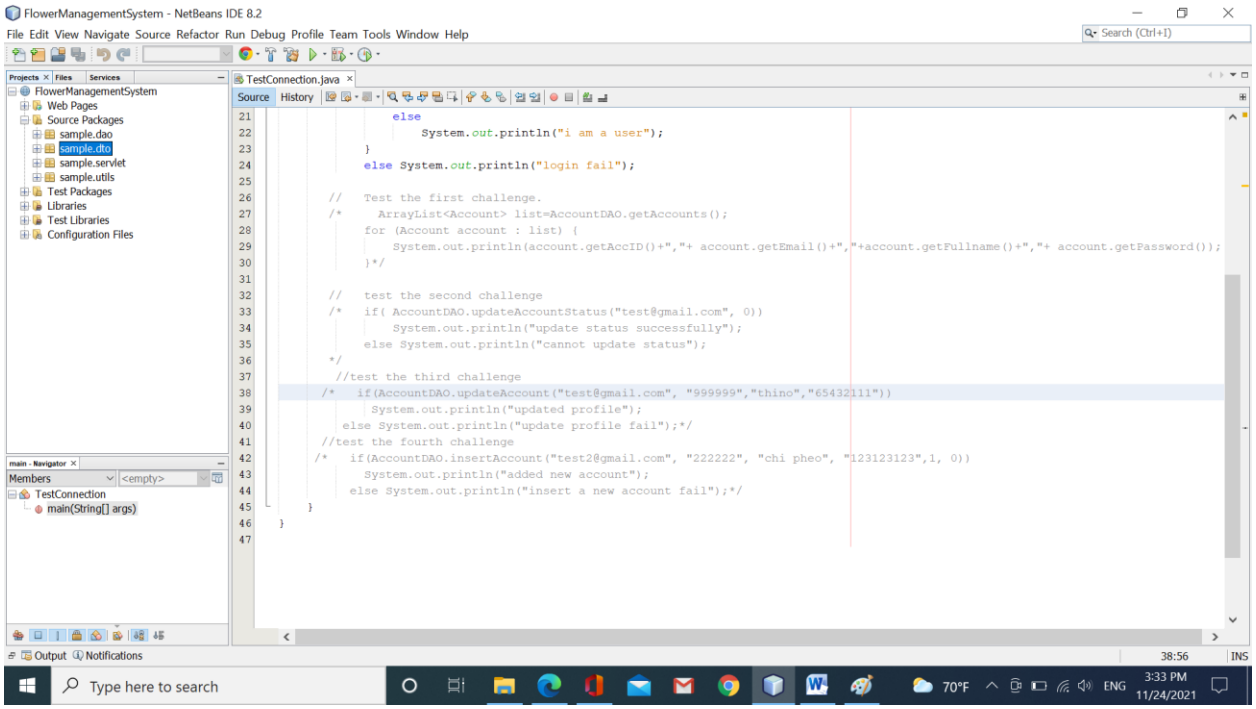
1) In the file AccountDAO, write the function public static ArrayList<Account> getAccounts(){…}. This function returns the list of accounts that are got from the table Account. You test this function in the main method.

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 newFullname, 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.

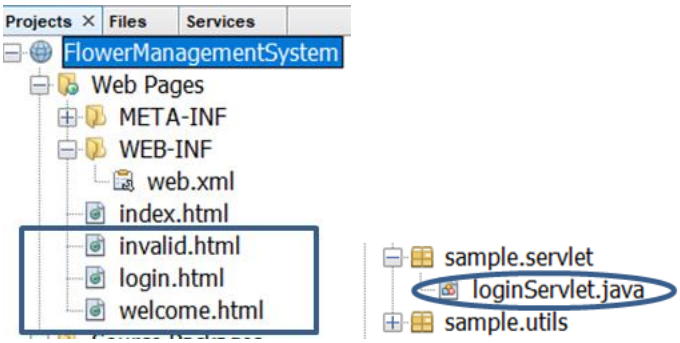
4) In the file AccountDAO, write a function named public static boolean insertAccount(String newEmail, String newPassword, String newFullname, String newPhone, int newSatus,int newRole). This function is used to insert a new account into the table Accounts and return true/false.

You can use the sample to test the methods above



Part 2: perform the login function.

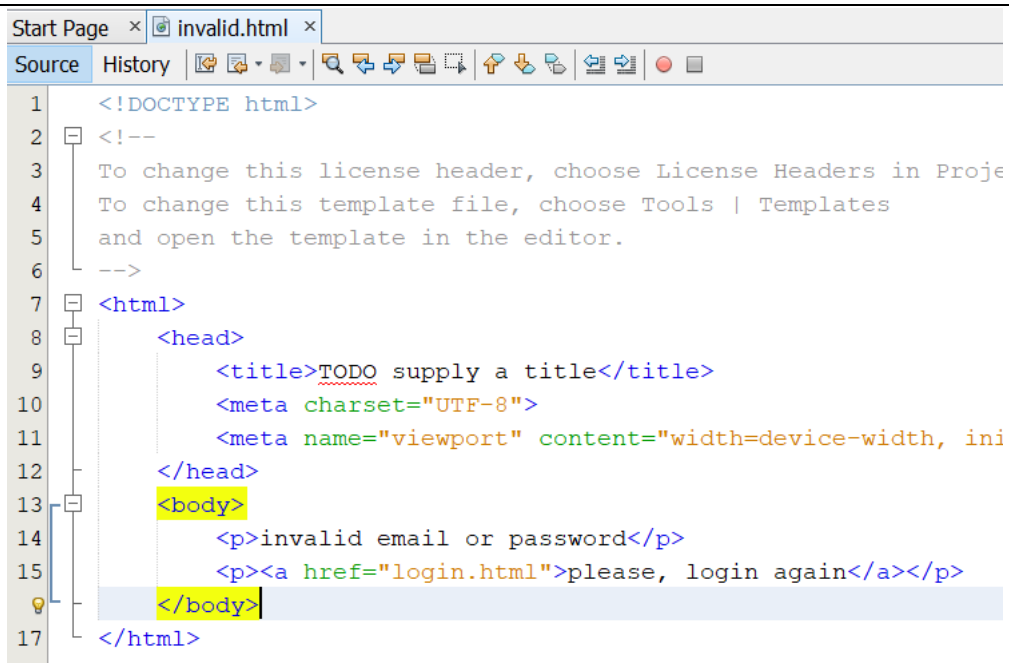
Step 1: You create three files HTML in the folder “Web Pages” and one file loginServlet in the package “sample.servlet” like the figure:



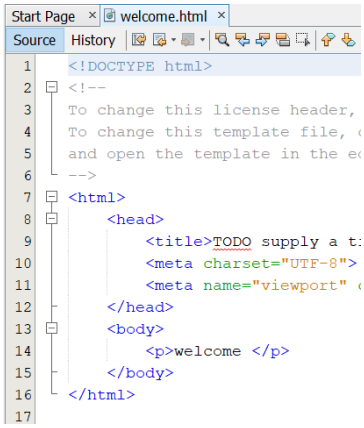
login.html

```
Start Page | login.html | loginServlet.java | web.xml | invalid.html
Source | History
 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>
            </form>
24      </body>
25  </html>
```

invalid.html

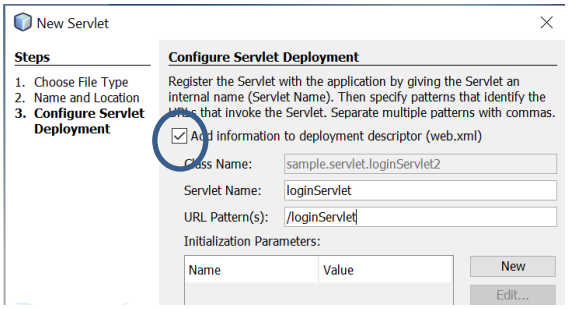


Welcome.html



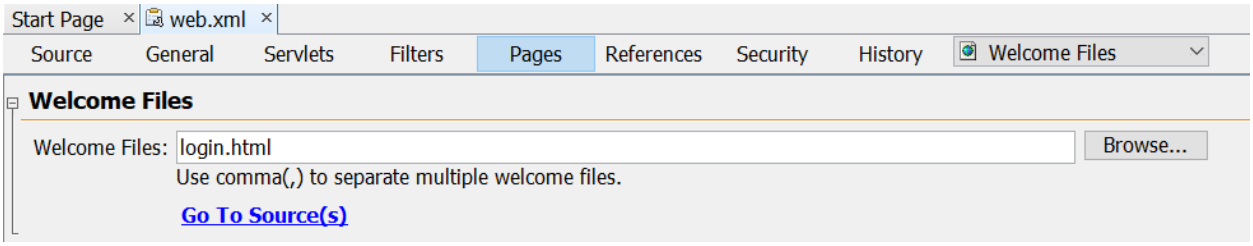
loginServlet.java

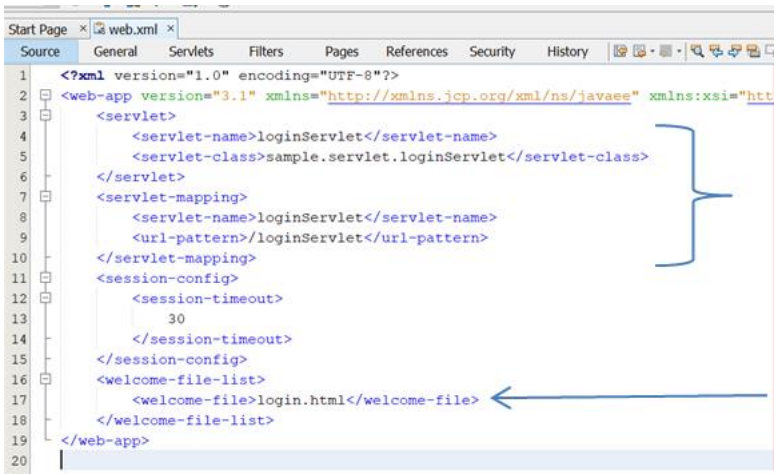
right-click in the package “sample.servlet”/new servlet/finish.



```java
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");
        Account acc=null;
         try {
            acc=AccountDAO.getAccount(email, password);
            if(acc!=null){
                //admin
                if(acc.getRole()==1){
                    //chuyen qua admin home page

                }
                //user
                else{
                    //chuyen qua welcome page
                    response.sendRedirect("welcome.html");

                }
            }
            else response.sendRedirect("invalid.html");
        } catch (Exception e) {
            e.printStackTrace();
        }

    }
}
```

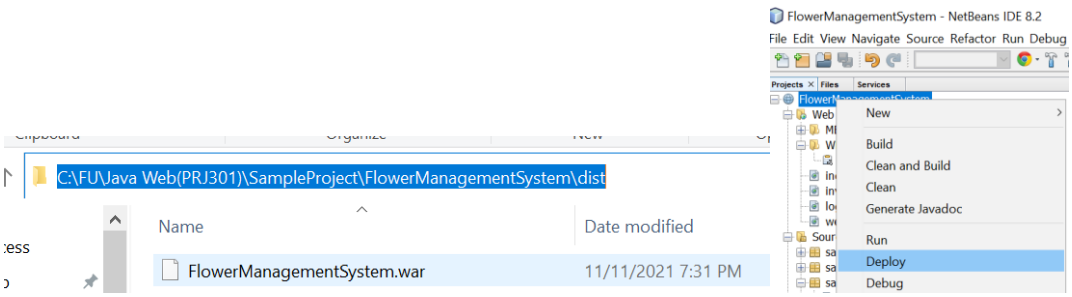
Open the file web.xml to set up the first page to run.



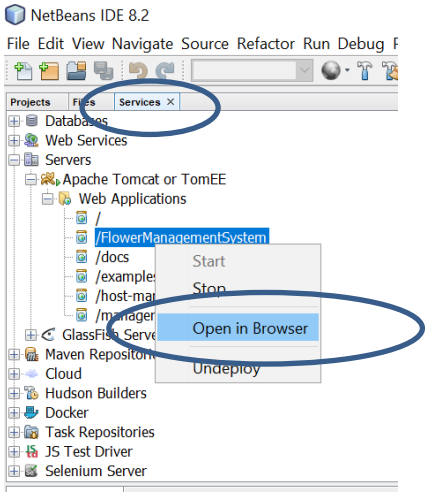


Now, you click “clean and build”/right-click on the project/deploy.

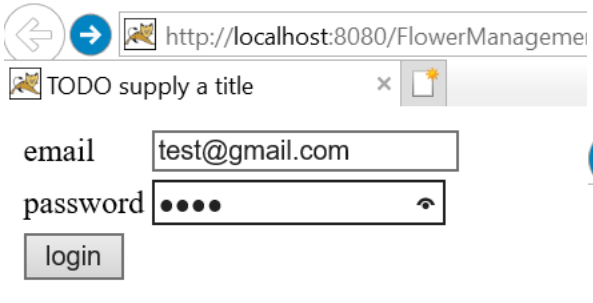
Deploy your app to copy *.war to the webserver. From now, you can run the web without needing Netbean.

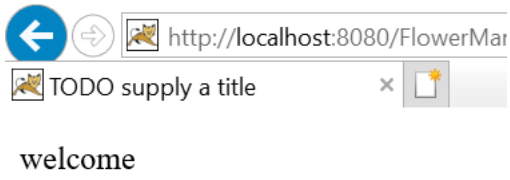


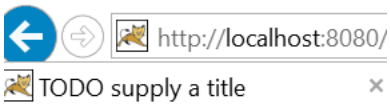
And then, click open in the browser



The result:







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=""><img src="images/logo.jpg"></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"

searchServlet.java | index.html | login.html | registration.html

Source | History

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

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

registration.html | registerServlet.java

Source | History

```
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'status is 1
41              int role=0;//defaulr a new account's role is 0
43              if(AccountDAO.insertAccount(email, password, fullname, phone, status, role)){
44                  response.sendRedirect("index.html");
45              }
46              else response.sendRedirect("errorpage.html");
47          }
48      }
49      HttpServlet methods. Click on the + sign on the left to edit the code.
87
88  }
89
```

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

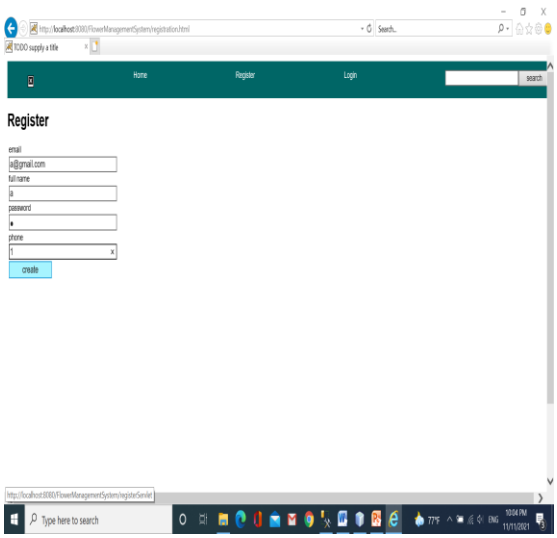
registration.html | registerServlet.java | errorpage.html

Source | History

```
<!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.
-->
<html>
    <head>
        <title>TODO supply a title</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <p>Something wrong, please click <a href="index.html">here</a> to return index page</p>
    </body>
</html>
```

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.

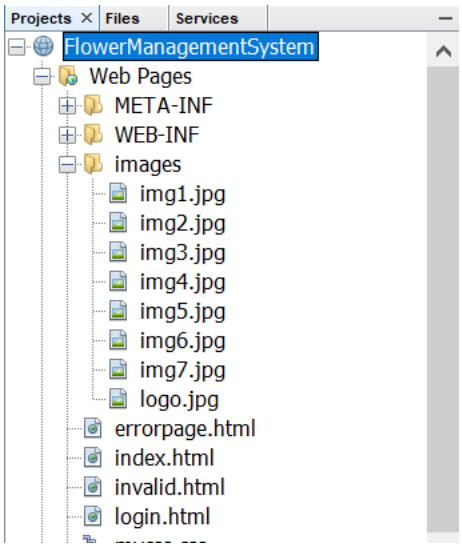
DESKTOP-IRCVITT....- dbo.Categories

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

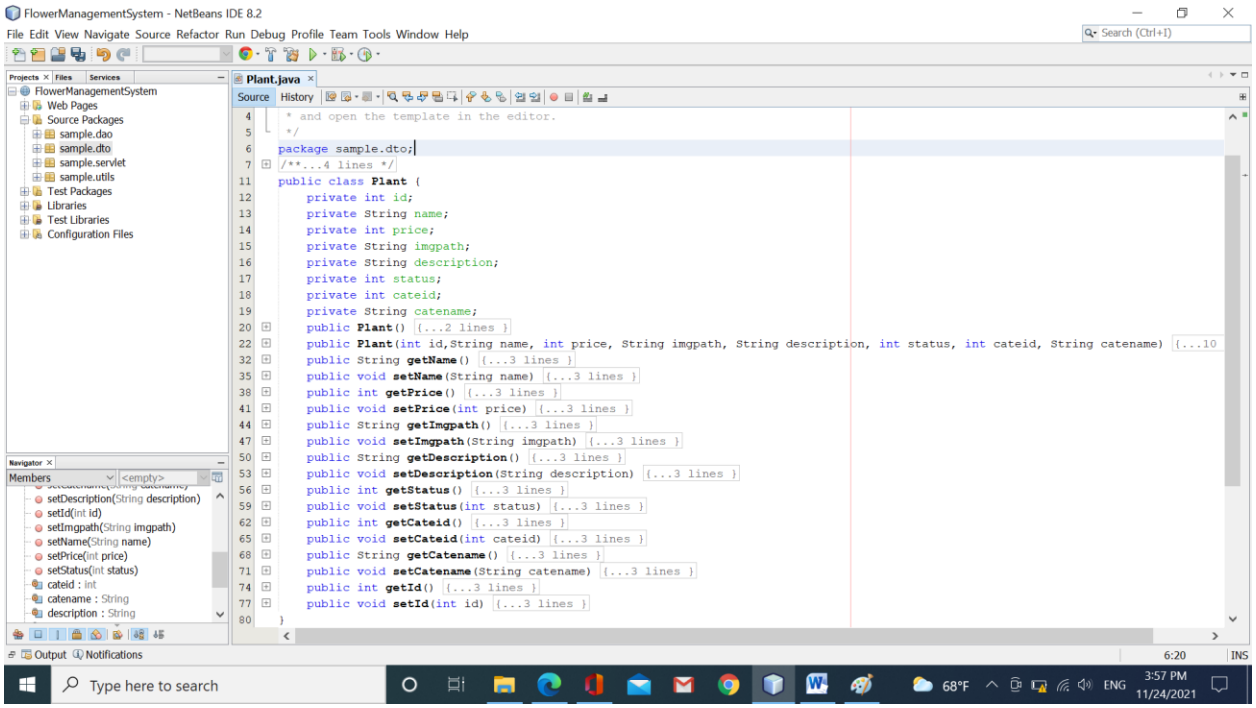
DESKTOP-IRCVITT....hop - dbo.Plants | DB_FlowerShop.sql...ntShop (sa (51))*

	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 hài	140	images/img3.jpg	hoa lan	1	1
	7	cây 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

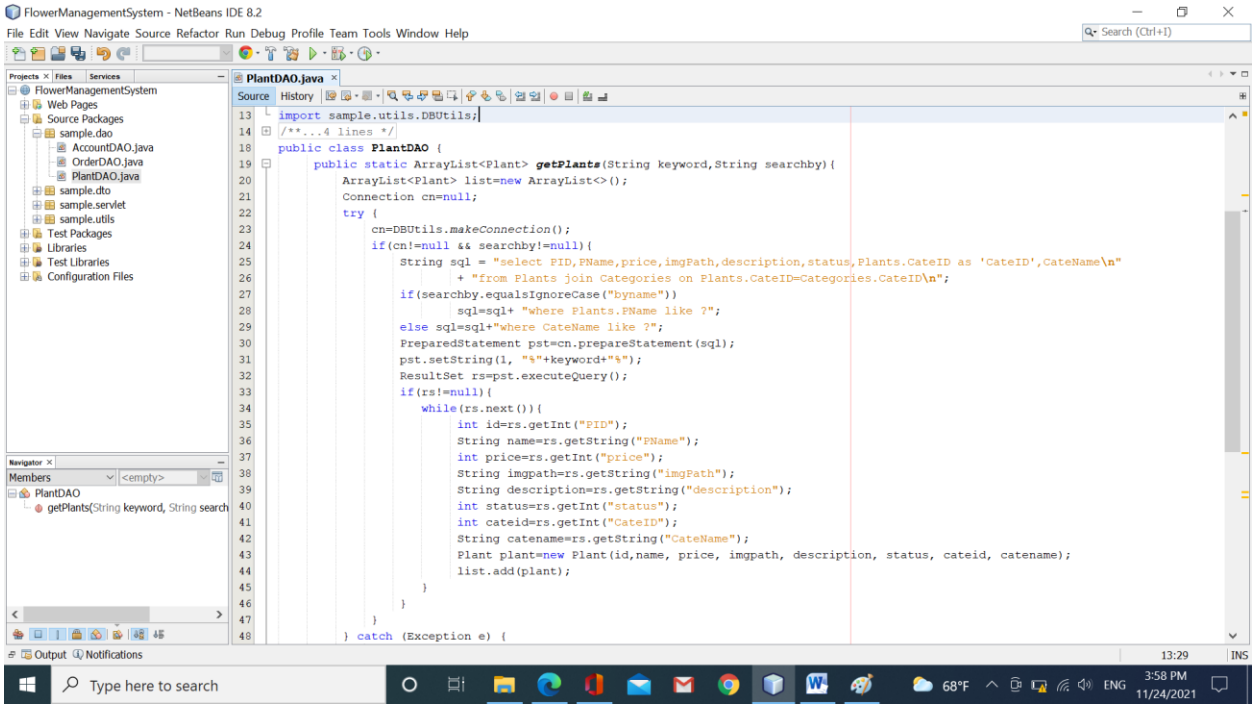
Step 2: Prepare your images



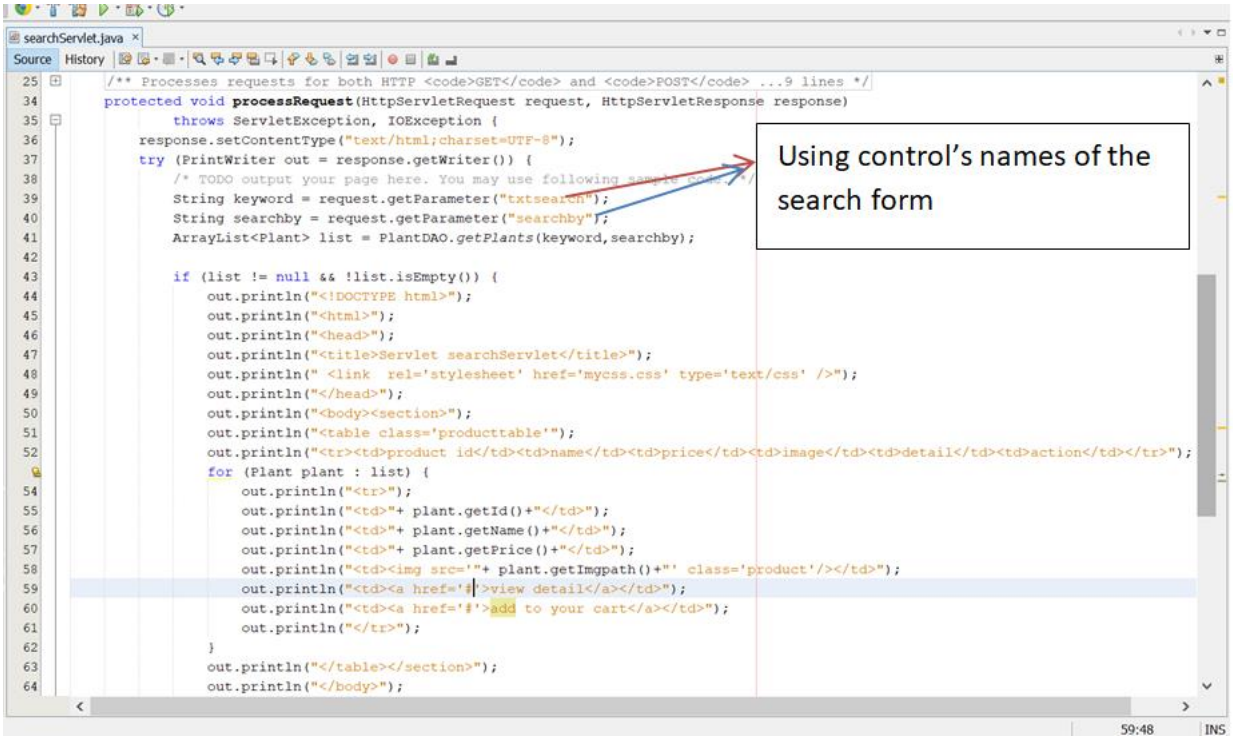
Step 3: In the package sample.dto, create a new file named “Plant.java”



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

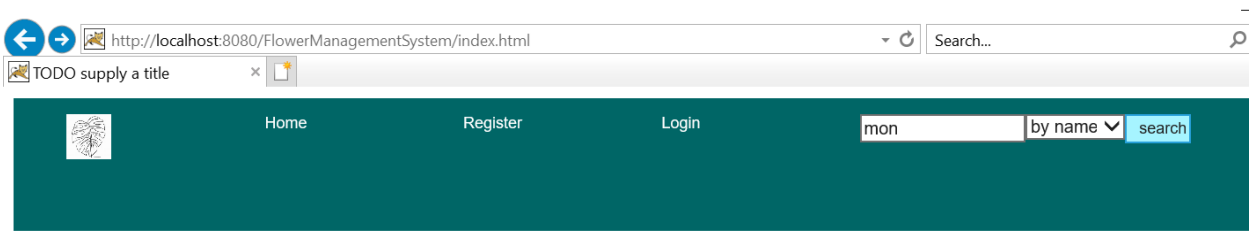


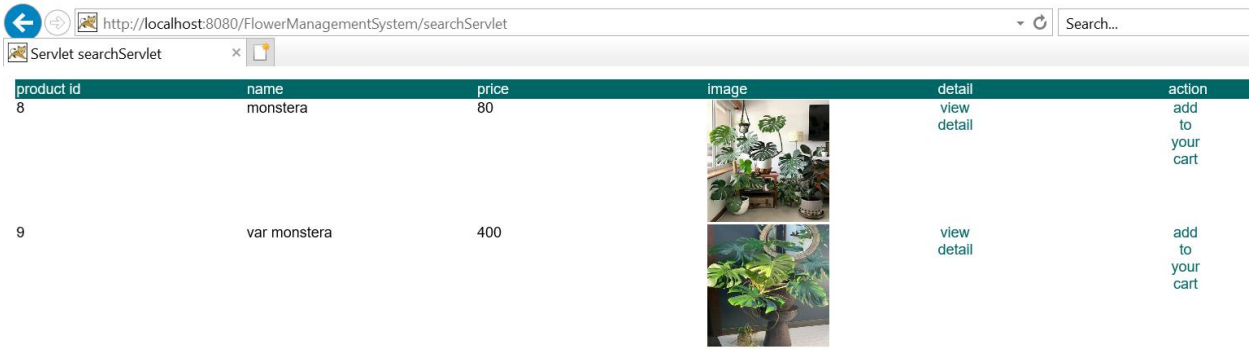
Step 5: In the package “sample.servlet, create a new file named “searchServlet.java”




```
/** Processes requests for both HTTP <code>GET</code> and <code>POST</code> ...9 lines */
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 keyword = request.getParameter("txtsearch");
        String searchby = request.getParameter("searchby");
        ArrayList<Plant> list = PlantDAO.getPlants(keyword,searchby);

        if (list != null && !list.isEmpty()) {
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet searchServlet</title>");
            out.println(" <link  rel='stylesheet' href='mycss.css' type='text/css' />");
            out.println("</head>");
            out.println("<body><section>");
            out.println("<table class='producttable'");
            out.println("<tr><td>product id</td><td>name</td><td>price</td><td>image</td><td>detail</td><td>action</td></tr>");
            for (Plant plant : list) {
                out.println("<tr>");
                out.println("<td>"+ plant.getId()+"</td>");
                out.println("<td>"+ plant.getName()+"</td>");
                out.println("<td>"+ plant.getPrice()+"</td>");
                out.println("<td><img src='"+ plant.getImgpath()+"' class='product'/></td>");
                out.println("<td><a href='#'>view detail</a></td>");
                out.println("<td><a href='#'>add to your cart</a></td>");
                out.println("</tr>");
            }
            out.println("</table></section>");
            out.println("</body>");
```

Step 6: run your app to get the result





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

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

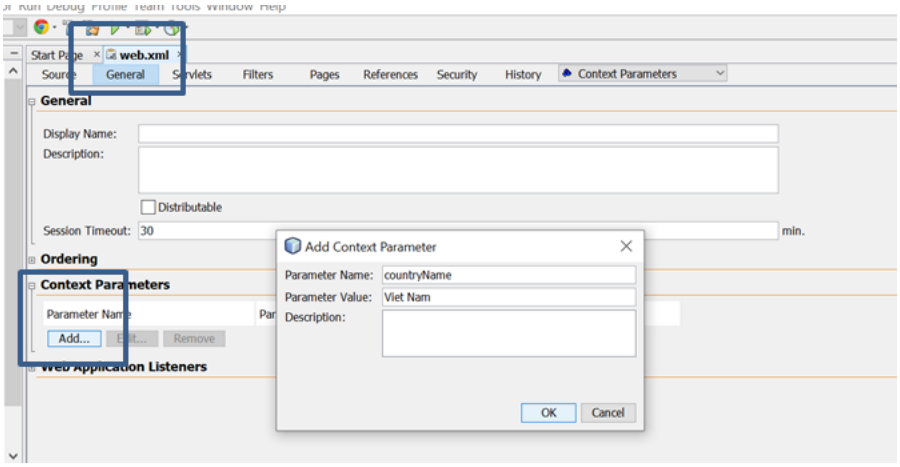
#initial parameters #scope # deploy # transfer

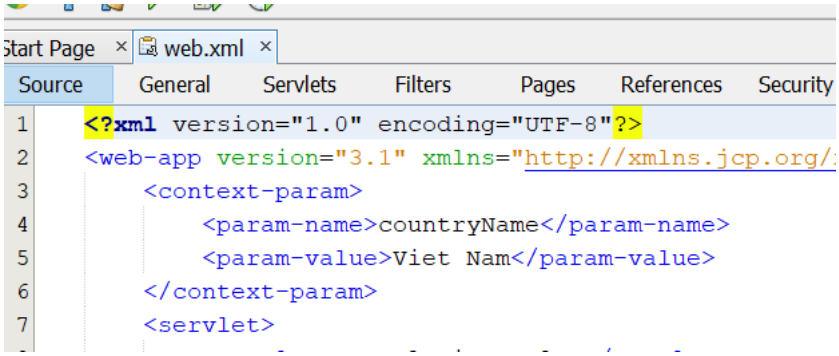
Learning outcome:

- 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

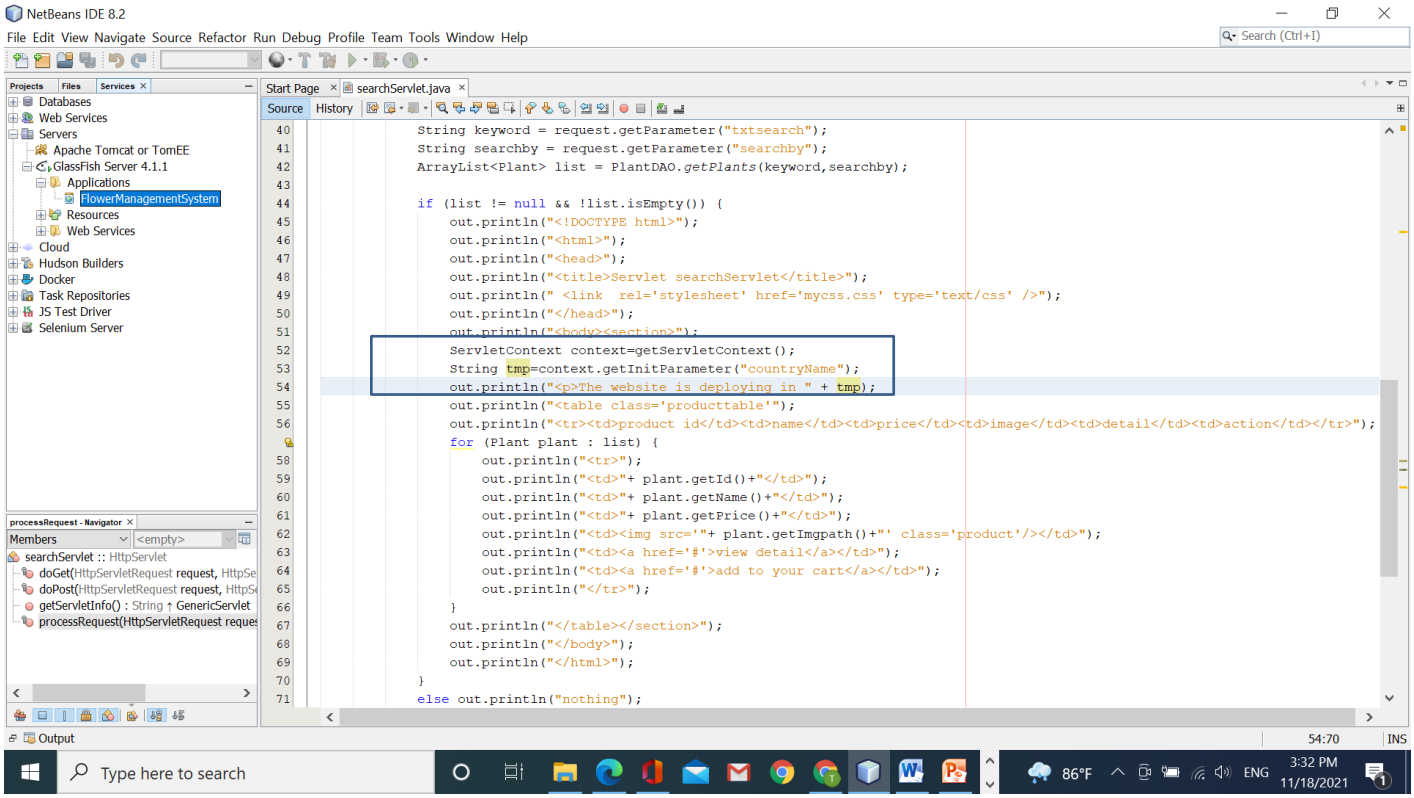
Part 1: Define the initial parameter in the context(web app), and display it in all servlets

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



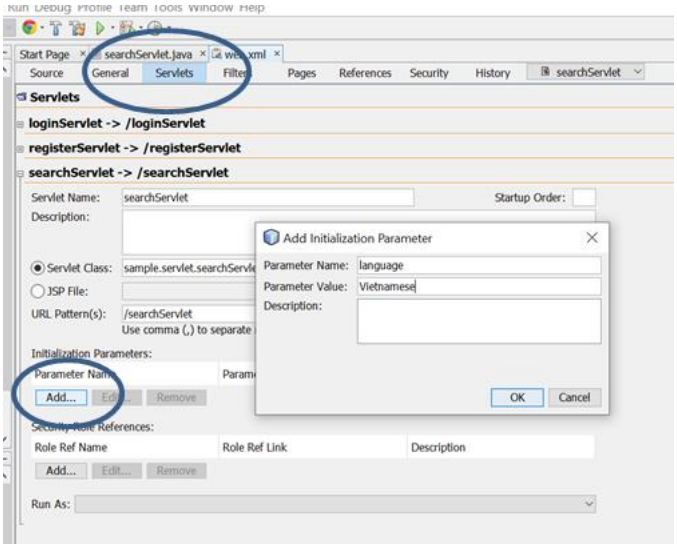


Step 2: display the country name in the servlet “searchServlet”, or any servlet(if any)



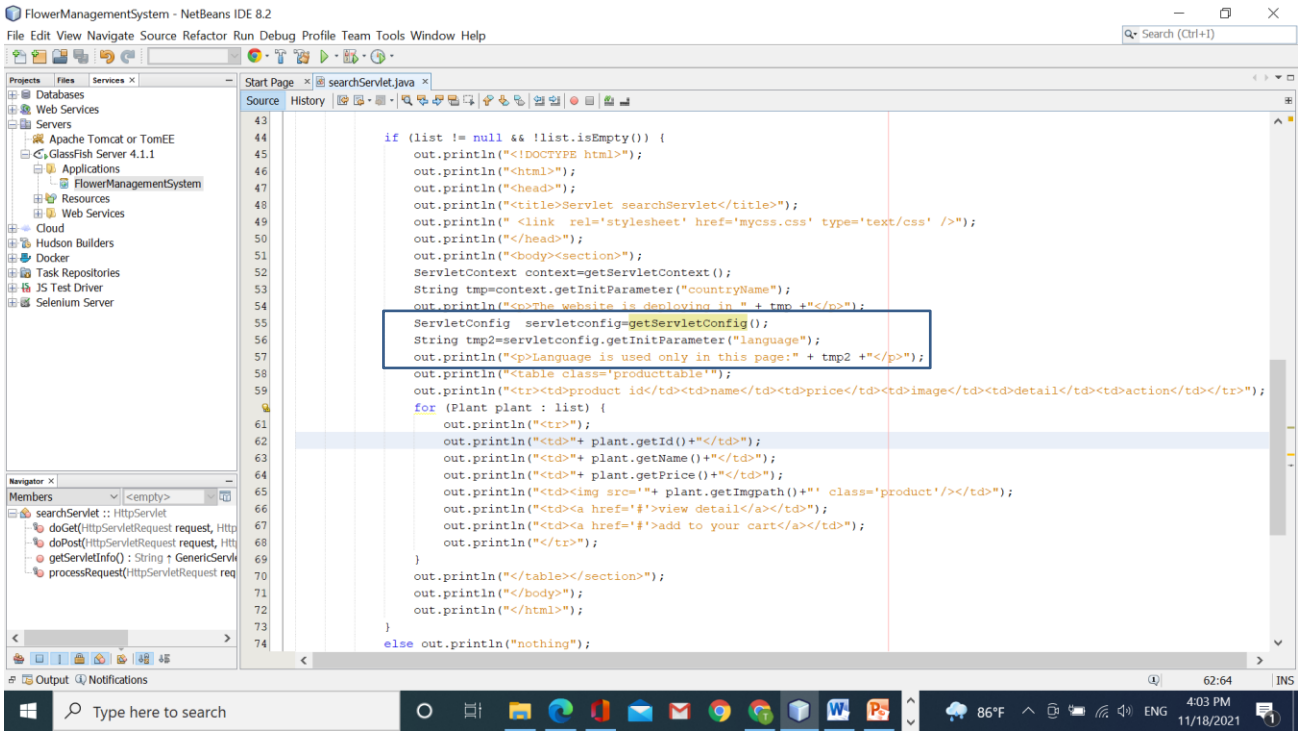
Part 2: define only the initial parameter on the servlet only.

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

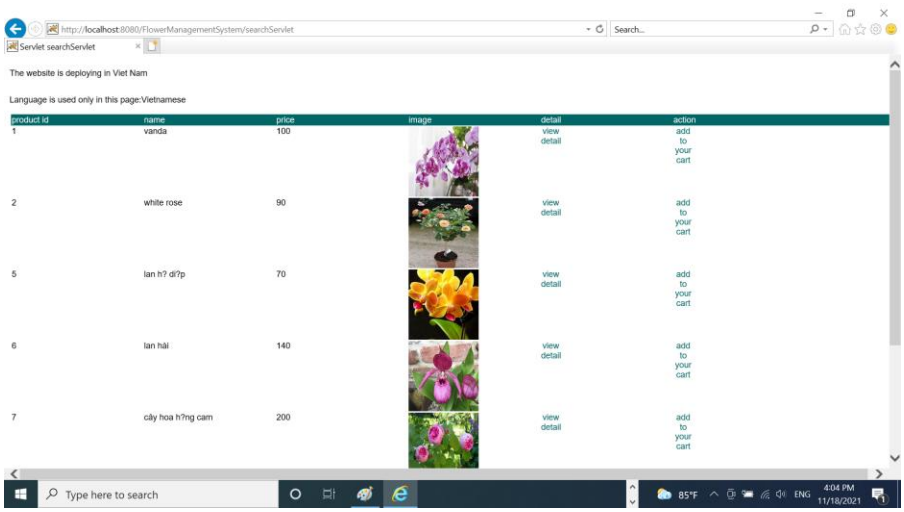


```
15      <servlet>
16          <servlet-name>searchServlet</servlet-name>
17          <servlet-class>sample.servlet.searchServlet</servlet-class>
18          <init-param>
19              <param-name>language</param-name>
20              <param-value>Vietnamese</param-value>
21          </init-param>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

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

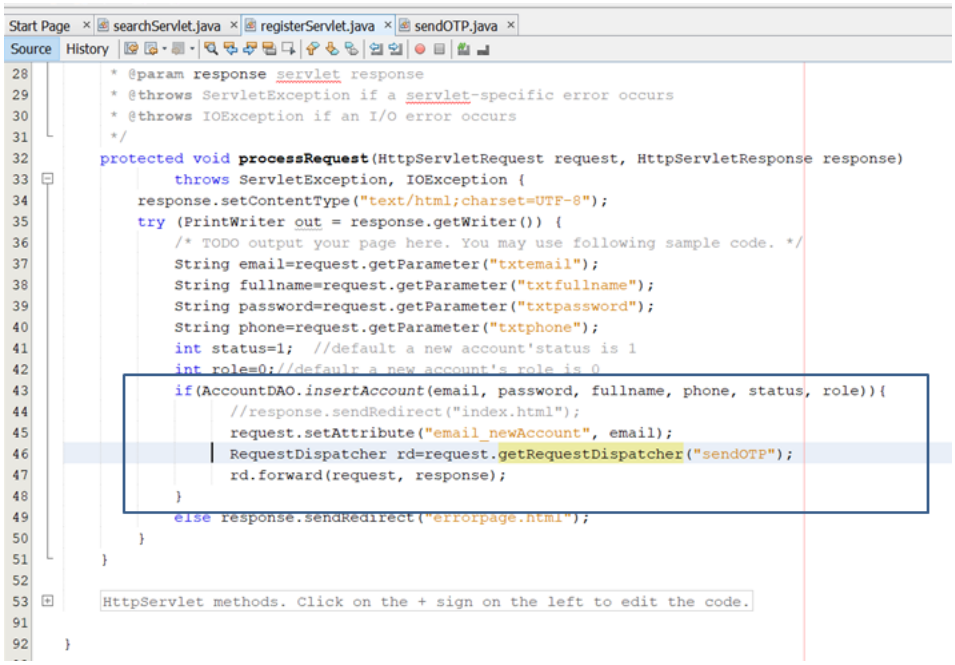


The result:



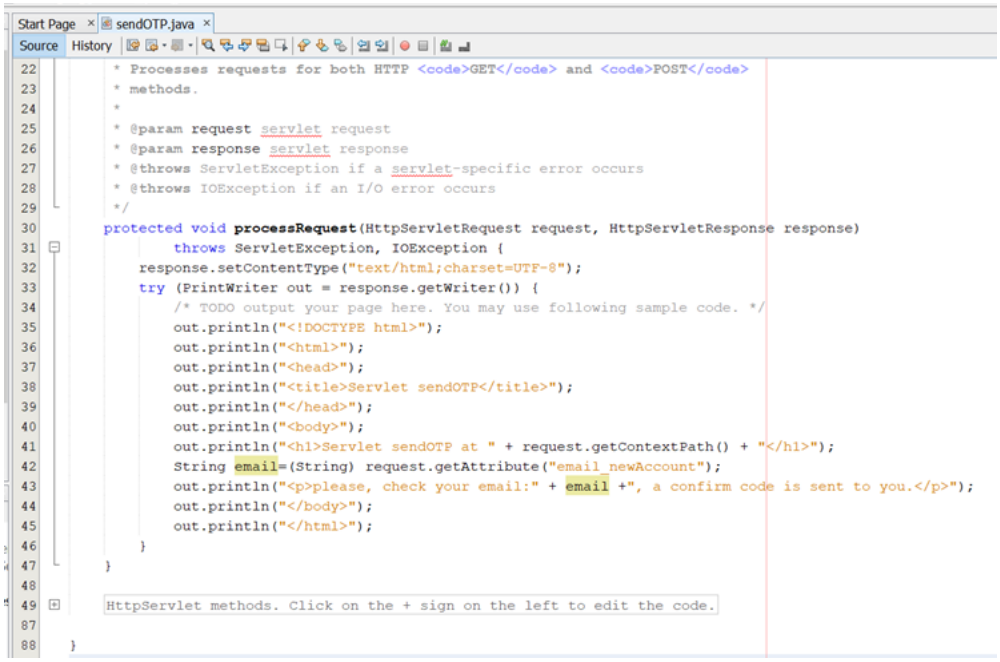
Part 3: Transfer data from the servlet to other servlets by the method **forward**

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



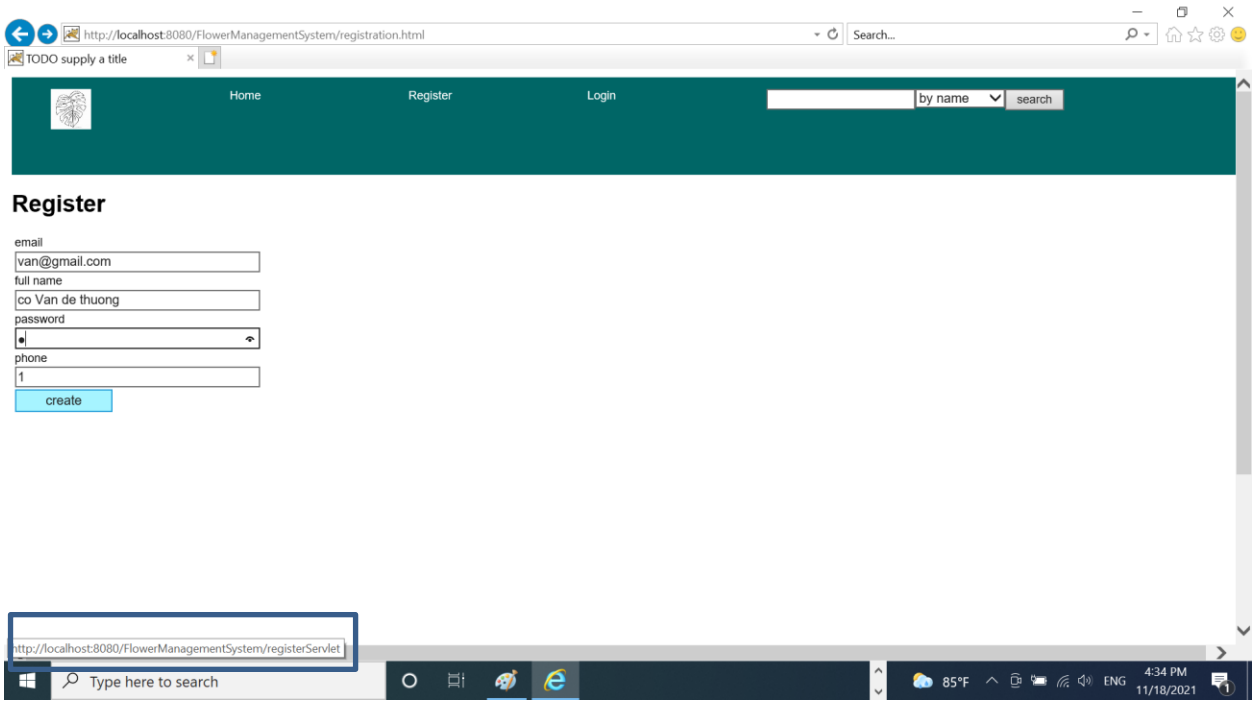
```
28          * @param response servlet response
29          * @throws ServletException if a servlet-specific error occurs
30          * @throws IOException if an I/O error occurs
31          */
32         protected void processRequest(HttpServletRequest request, HttpServletResponse response)
33                 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                 String email=request.getParameter("txtemail");
38                 String fullname=request.getParameter("txtfullname");
39                 String password=request.getParameter("txtpassword");
40                 String phone=request.getParameter("txtphone");
41                 int status=1;  //default a new account'status is 1
42                 int role=0;//defaulr a new account's role is 0
43                 if(AccountDAO.insertAccount(email, password, fullname, phone, status, role)){
44                     //response.sendRedirect("index.html");
45                     request.setAttribute("email_newAccount", email);
46                     RequestDispatcher rd=request.getRequestDispatcher("sendOTP");
47                     rd.forward(request, response);
48                 }
49                 else response.sendRedirect("errorpage.html");
50             }
51         }
52
53         HttpServlet methods. Click on the + sign on the left to edit the code.
91
92     }
```

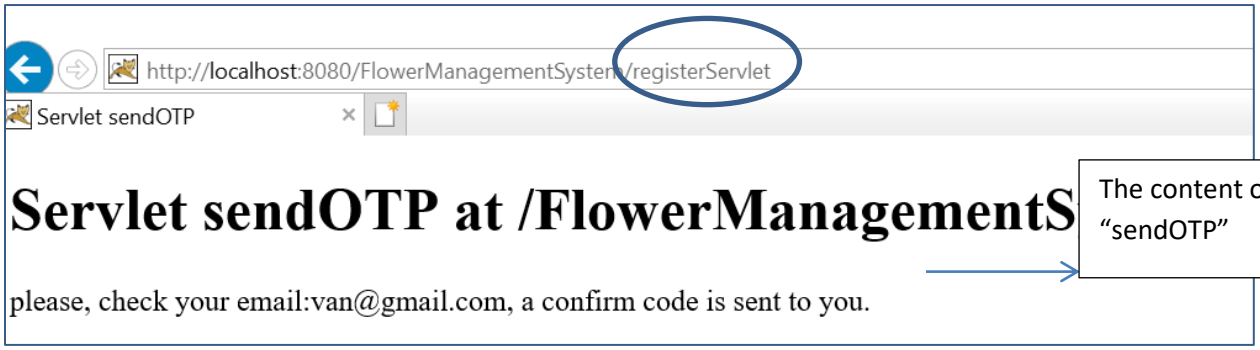
Step 2: In the package “sample. servlet”, create a new servlet named “sendOTP.java”



```
22          * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
23          * methods.
24          *
25          * @param request servlet request
26          * @param response servlet response
27          * @throws ServletException if a servlet-specific error occurs
28          * @throws IOException if an I/O error occurs
29          */
30         protected void processRequest(HttpServletRequest request, HttpServletResponse response)
31                 throws ServletException, IOException {
32             response.setContentType("text/html;charset=UTF-8");
33             try (PrintWriter out = response.getWriter()) {
34                 /* TODO output your page here. You may use following sample code. */
35                 out.println("<!DOCTYPE html>");
36                 out.println("<html>");
37                 out.println("<head>");
38                 out.println("<title>Servlet sendOTP</title>");
39                 out.println("</head>");
40                 out.println("<body>");
41                 out.println("<h1>Servlet sendOTP at " + request.getContextPath() + "</h1>");
42                 String email=(String) request.getAttribute("email_newAccount");
43                 out.println("<p>please, check your email:" + email +", a confirm code is sent to you.</p>");
44                 out.println("</body>");
45                 out.println("</html>");
46             }
47         }
48
49         HttpServlet methods. Click on the + sign on the left to edit the code.
87
88     }
```

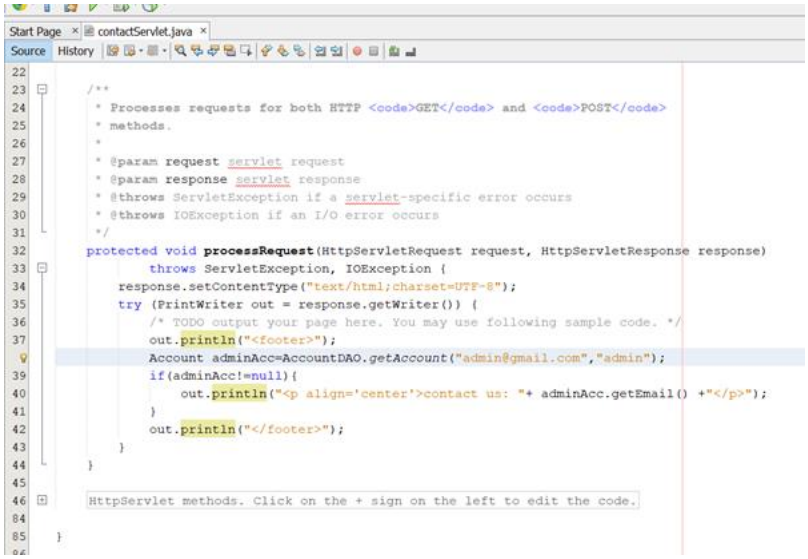
Step 3: run the web, click the link “register” to create a new account and you will see the result.



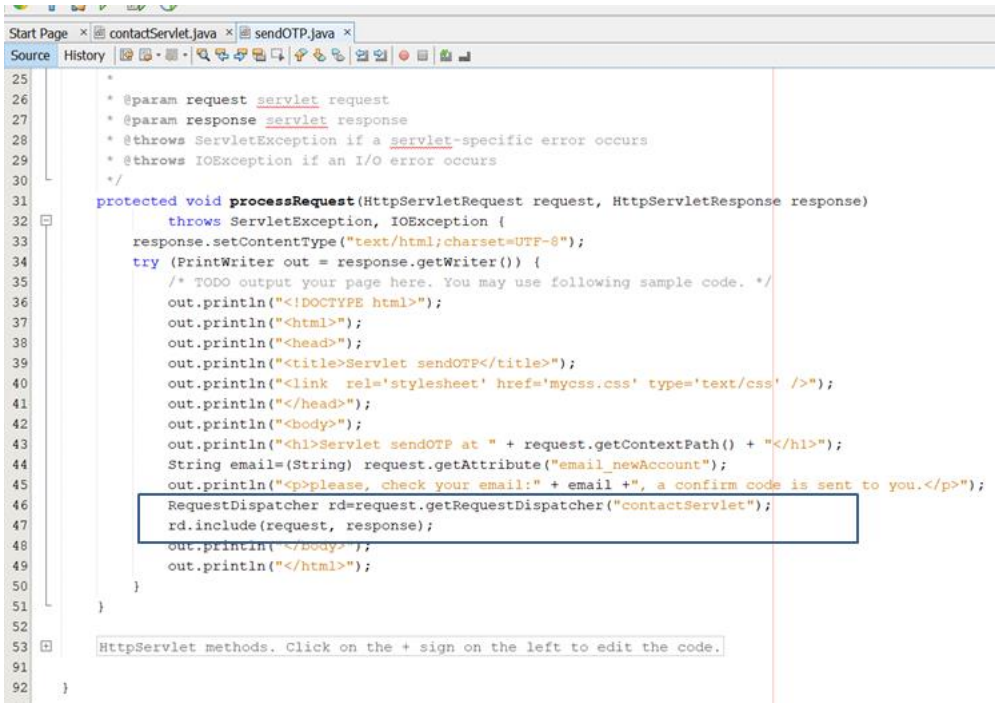


Part 4: Transfer data from the servlet to other servlets by the method **include**

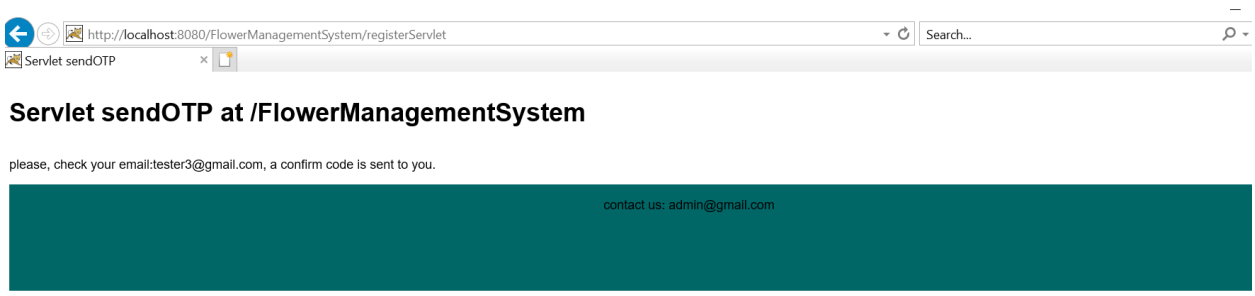
Step 1: In the package 'sample. servlet", create a new servlet named "contactServlet"



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



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

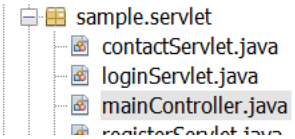
#session #JSP #Impicit objects #MVC2

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"



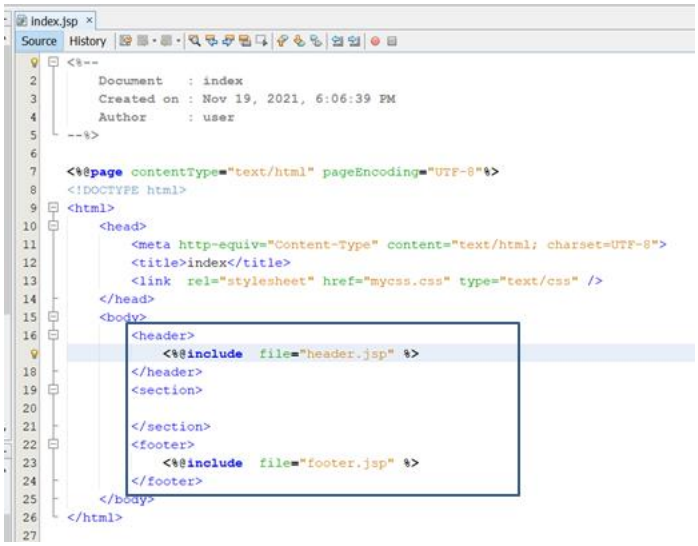
Step 2: You will create two new pages named "header.jsp", "footer.jsp".
header.jsp

```
<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=""><img src="images/logo.jpg" id="logo"></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

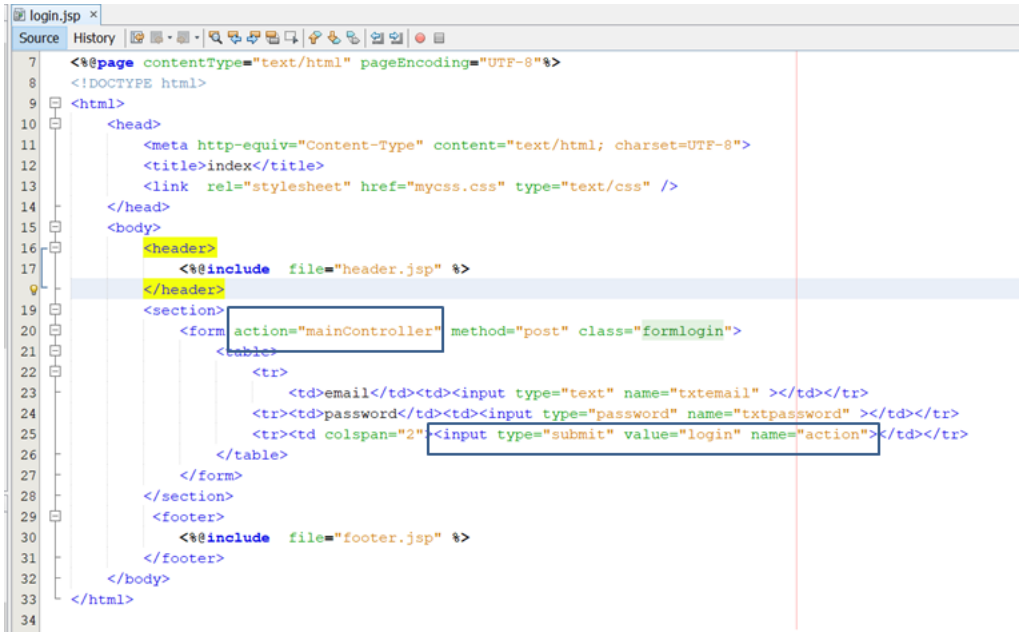
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
        <link  rel="stylesheet" href="mycss.css" type="text/css" />
    </head>
    <body>
        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

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

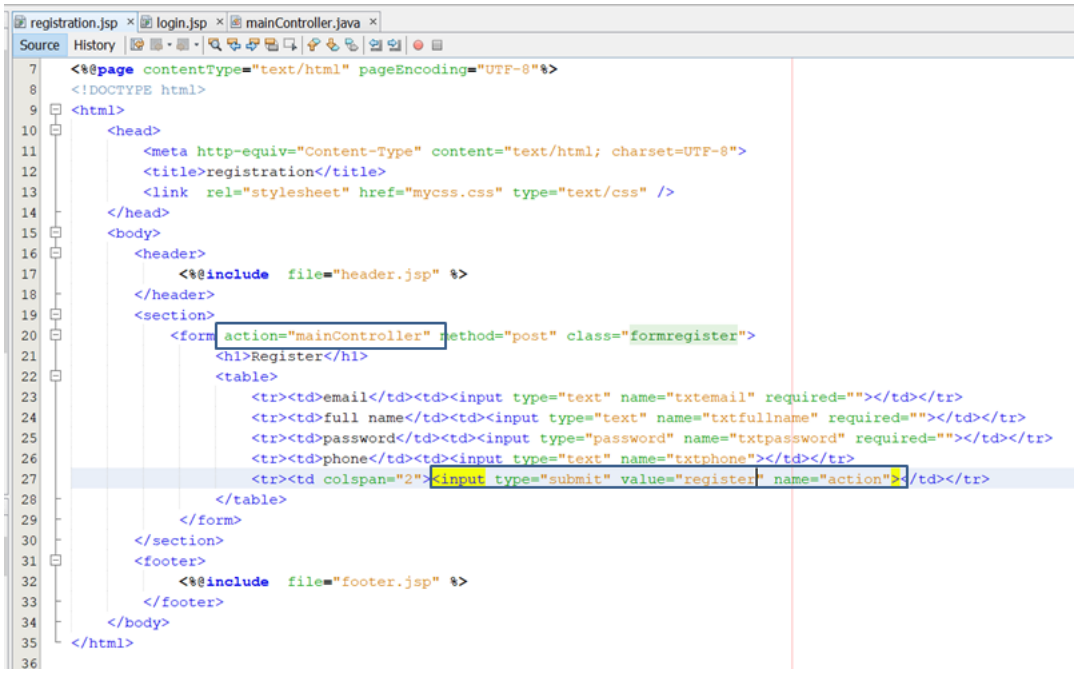


Step 4: do again some functions: login, registration using pages JSP, not HTML

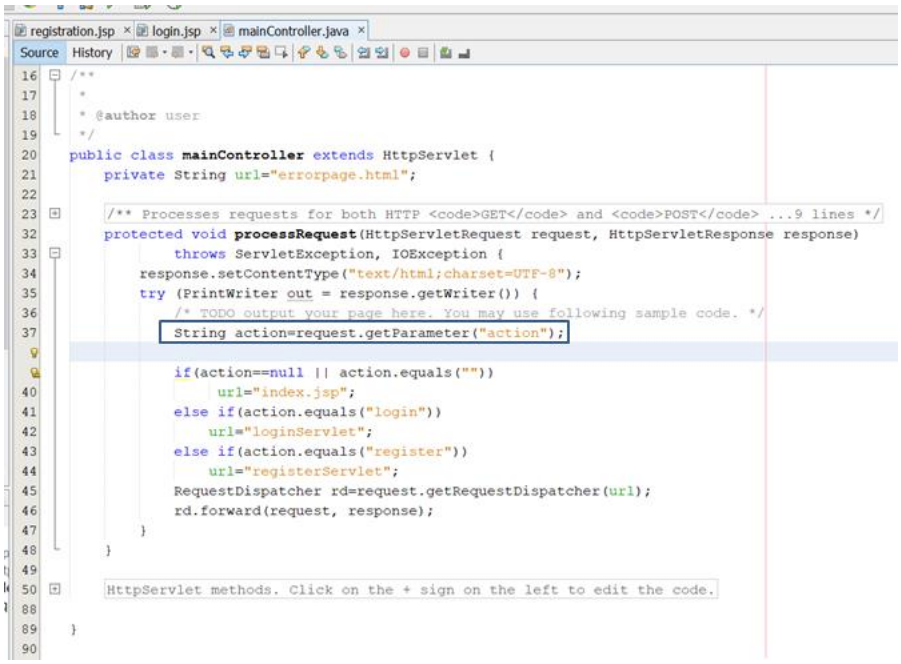
Create a new page named “login.jsp”



Create a new page named “registration.jsp”



Step 5: In the servlet “mainController”, edit the code:



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:

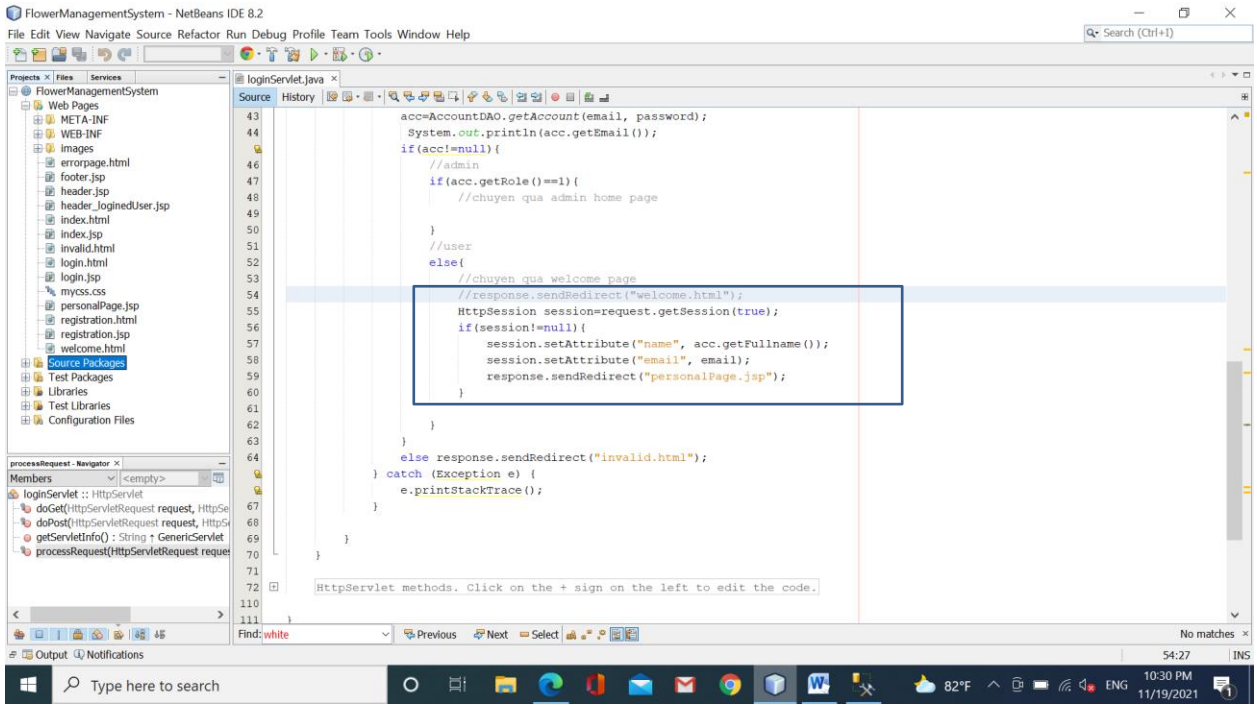
The key idea is all requests from the client using the format:

<input type=“submit” name=”action” value=”….”>

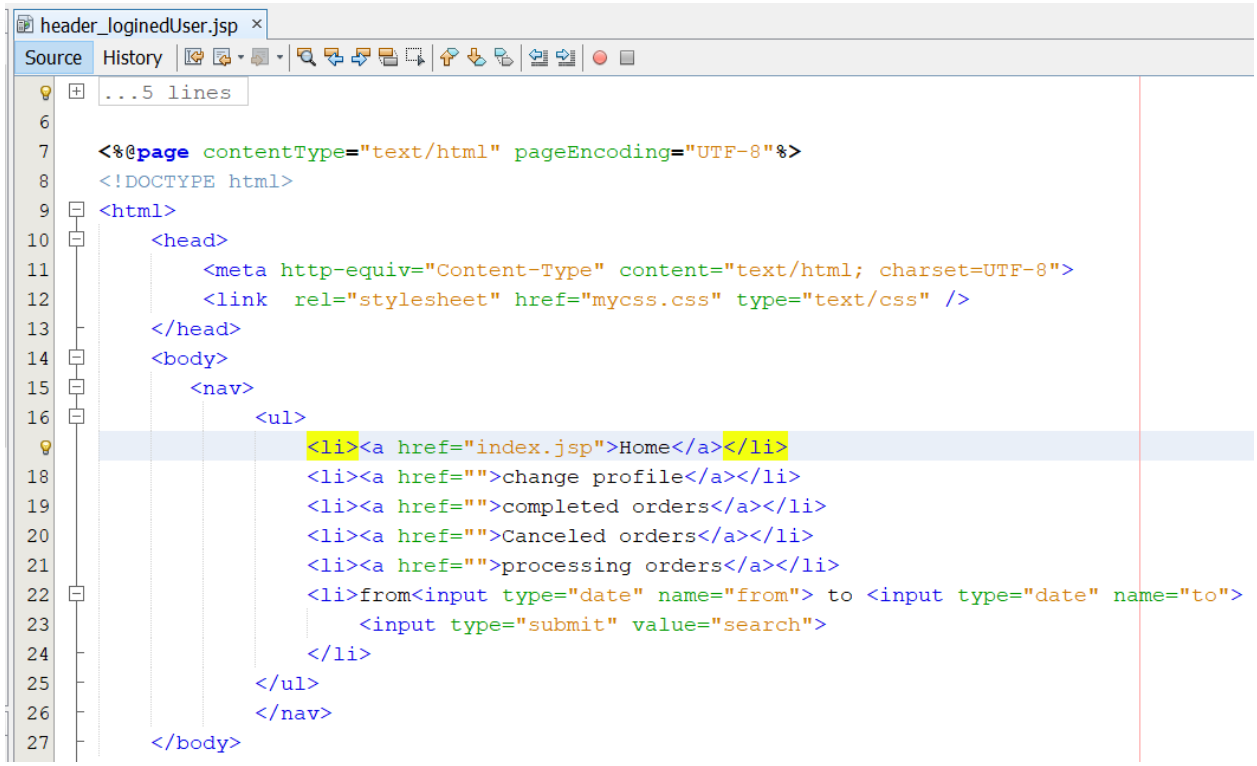
In the mainController, we get this action parameter to decide what servlet will be called.

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



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



personalPage.jsp × header_loginedUser.jsp ×

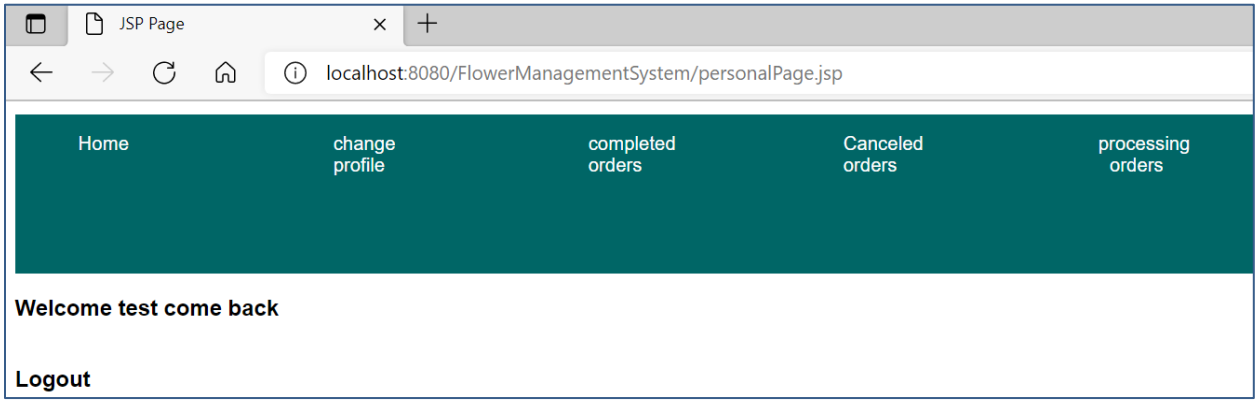
Source History

```
<%@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>
        <%
            String name = (String) session.getAttribute("name");
            if (name == null) {

        %>
        <p><font color='red'>you must login to view personal page</font></p>
        <p></p>
        <%} else {
        %>
    <header>
        <%@include  file="header_loginedUser.jsp"%>
    </header>
    <section>
        <h3>Welcome <%= name%> come back </h3>
        <h3>Logout </h3>
    </section>
    <section> <!--load all orders of the user at here --> </section>
    <footer>
        <%@include file="footer.jsp" %>
    </footer>
    <%}%>
</body>
</html>
```

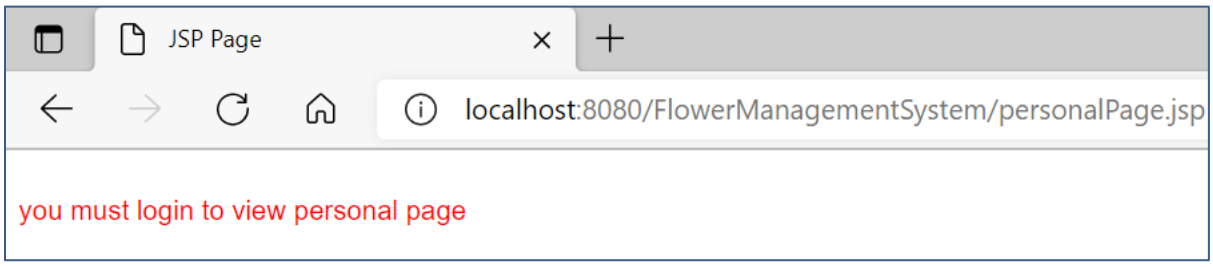
On the page “personalPage.jsp”, we will check user authentication. If the user is valid then open this page otherwise display a warning.

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>
```

logoutServet

```
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”

Order.java

```
/*
 * 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.
 */
package sample.dto;
/**...4 lines */
public class Order {
    private int orderID;
    private String orderDate;
    private String shipDate;
    private int status;
    private int accID;
    public Order() {...2 lines }
    public Order(int orderID, String orderDate, String shipDate, int status, int accID) {...7 lines }
    public int getOrderID() {...3 lines }
    public void setOrderID(int orderID) {...3 lines }
    public String getOrderDate() {...3 lines }
    public void setOrderDate(String orderDate) {...3 lines }
    public String getShipDate() {...3 lines }
    public void setShipDate(String shipDate) {...3 lines }
    public int getStatus() {...3 lines }
    public void setStatus(int status) {...3 lines }
    public int getAccID() {...3 lines }
    public void setAccID(int accID) {...3 lines }
}
```

Step 2: Create a new class named “OrderDAO”

OrderDAO.java

```java
/*
 * 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.
 */
package sample.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import sample.dto.Account;
import sample.dto.Order;
import sample.dto.OrderDetail;
import sample.utils.DBUtils;
/**...4 lines */
public class OrderDAO {
    public static ArrayList<Order> getOrders(String email){...31 lines }
    public static ArrayList<OrderDetail> getOrderDetail(int orderID){...30 lines }
}
```

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

```jsp
<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

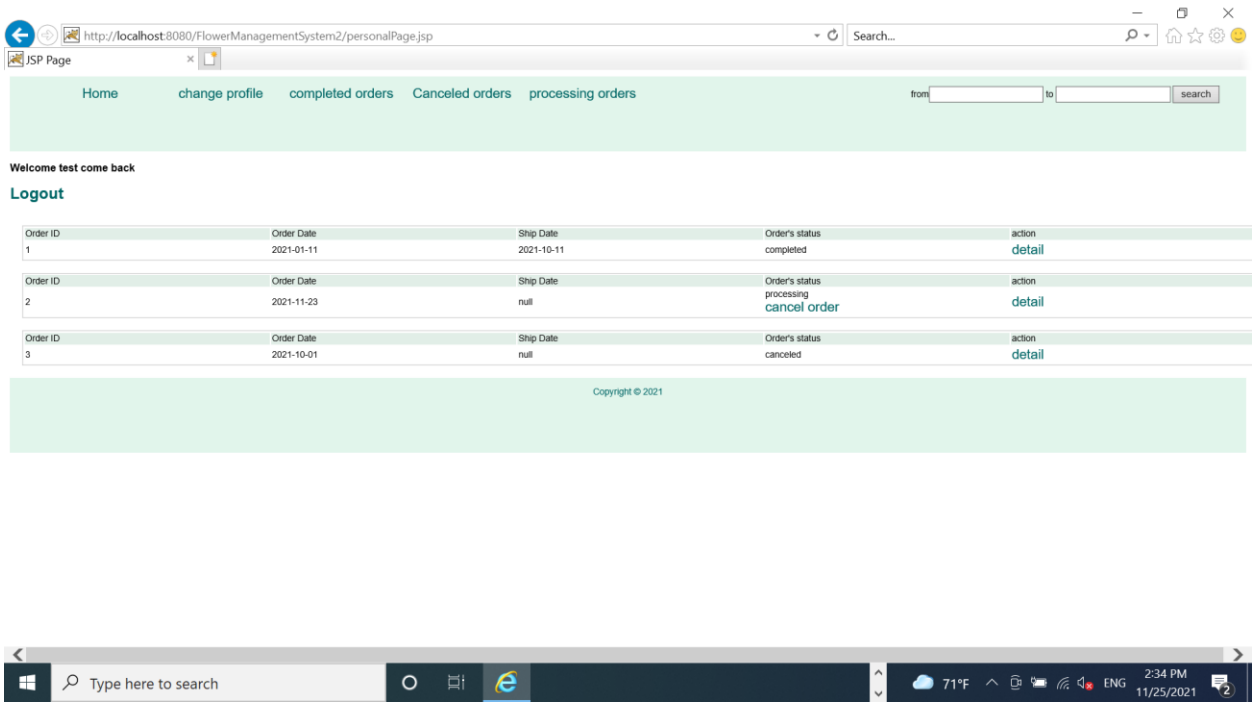
DESKTOP-IRCVITT....hop - dbo.Orders | SQLQuery1.sql - (l...antShop (sa (52))*

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

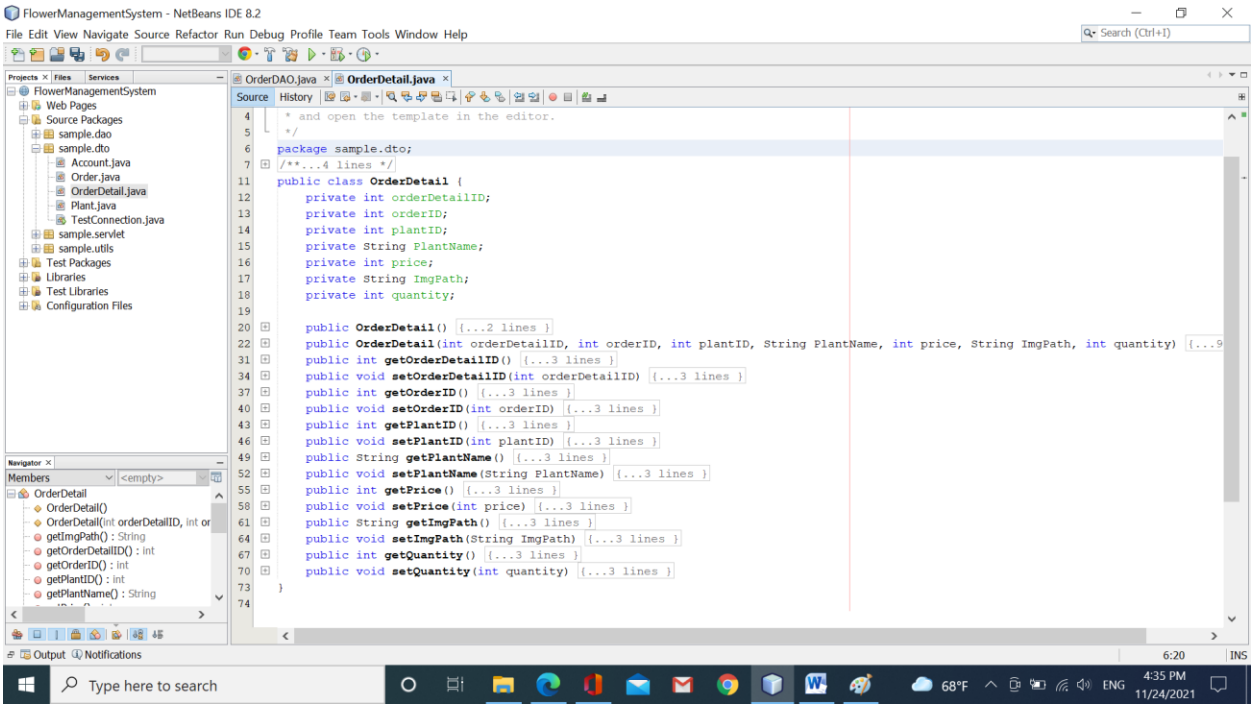
	DetailId	OrderID	FID	quantity
▶	1	1	1	1
	2	1	2	2
	5	2	5	1
	6	3	8	2
	7	3	9	1
*	NULL	NULL	NULL	NULL

Step 5: run your app to get the result



Part 4: View order detail of the order

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



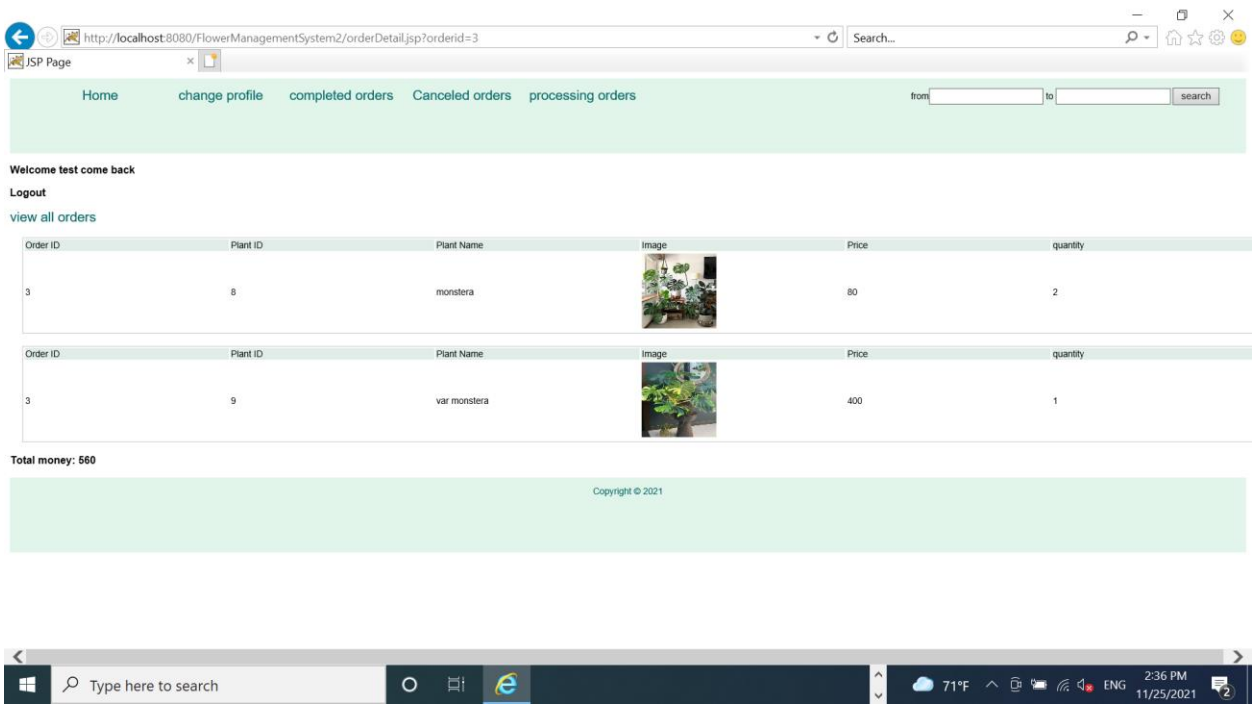
Step 2: In the file OrderDAO.java, add more the method “getOrderDetail(int orderID)”

```
public class OrderDAO {
    public static ArrayList<Order> getOrders(String email){...31 lines }
    public static ArrayList<OrderDetail> getOrderDetail(int orderID){
        Connection cn=null;
        ArrayList<OrderDetail> list=new ArrayList<>();
        try{
            cn = DBUtils.makeConnection();
            if (cn != null) {
                String sql = "select DetailId,OrderID,PID,PName,price,imgPath,quantity\n"
                        + "from OrderDetails, Plants\n"
                        + "where OrderID=? and OrderDetails.FID=Plants.PID";
                PreparedStatement pst = cn.prepareStatement(sql);
                pst.setInt(1, orderID);
                ResultSet rs = pst.executeQuery();
                if (rs != null) {
                    while (rs.next()) {
                        int detailID = rs.getInt("DetailId");
                        int PlantID = rs.getInt("PID");
                        String PlanName = rs.getString("PName");
                        int price = rs.getInt("price");
                        String imgPath=rs.getString("imgPath");
                        int quantity=rs.getInt("quantity");
                        OrderDetail orderdetail=new OrderDetail(detailID, orderID, PlantID, PlanName, price, imgPath, quantity);
                        list.add(orderdetail);
                    } } }
        }catch(Exception e){ e.printStackTrace();}
        finally{ try {
                if(cn!=null) cn.close();
            } catch (Exception e) { e.printStackTrace();      }
        }
        return list;
    }
}
```

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

```jsp
<h3>Logout </h3>
<a href="personalPage.jsp">view all orders</a>
</section>
<section> <!--load all orders of the user at here -->
<% String orderid = request.getParameter("orderid");
if (orderid != null) {
int orderID = Integer.parseInt(orderid.trim());
ArrayList<OrderDetail> list = OrderDAO.getOrderDetail(orderID);
if (list != null && !list.isEmpty()) {
int money = 0;
for (OrderDetail detail : list) {%>
<table class="order">
<tr><td>Order ID</td><td>Plant ID</td><td>Plant Name</td><td>Image</td><td>quantity</td></tr>
<tr><td><%= detail.getOrderID()%></td><td><%= detail.getPlantID()%></td>
<td><%= detail.getPlantName()%></td>
<td><img src='<%= detail.getImgPath()%>' class='plantimg' /> <br/> <%= detail.getPrice()%></td>
<td><%= detail.getQuantity()%></td>
<% money = money + detail.getPrice() * detail.getQuantity(); %>
</tr>
</table>
<% }//end for %>
<h3 > Total money: <%= money %></h3>
<% }//end if
else{
%>
<p>You don't have any order</p>
<%
}
}//end if
%>
</section>
<footer>
<%@include file="footer.jsp" %>
```

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

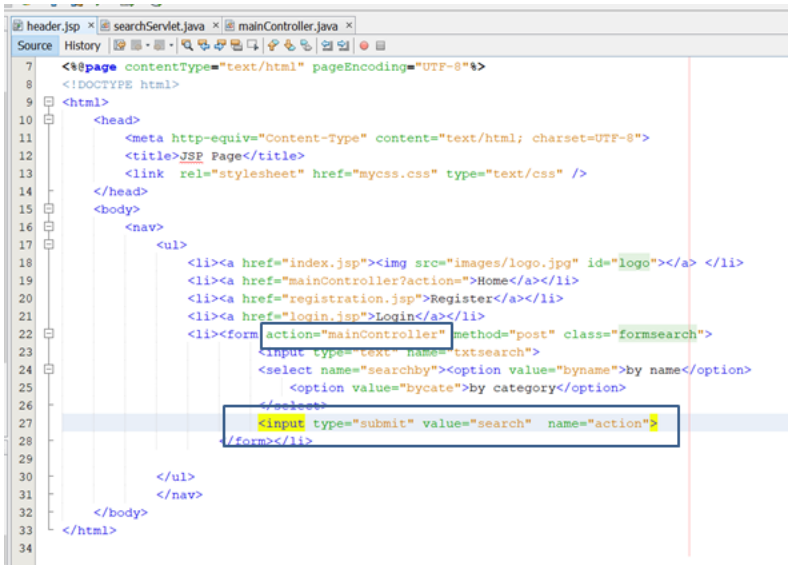


Part 5: Load all products to the index.jsp, use index.jsp to display the result of searching.

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

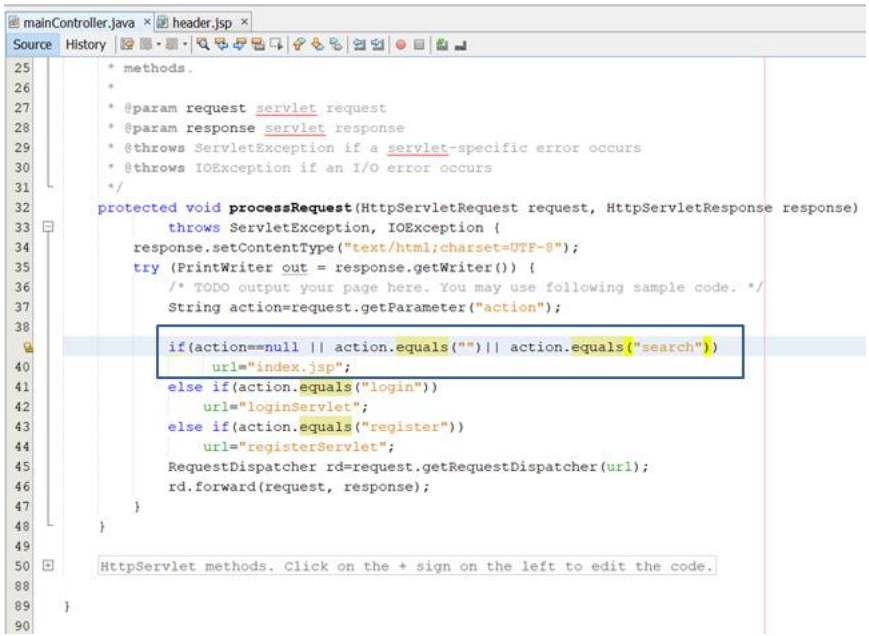
When the user searches for the product by name/by category, only appropriate products are displayed.

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"><img src="images/logo.jpg" id="logo"></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><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:



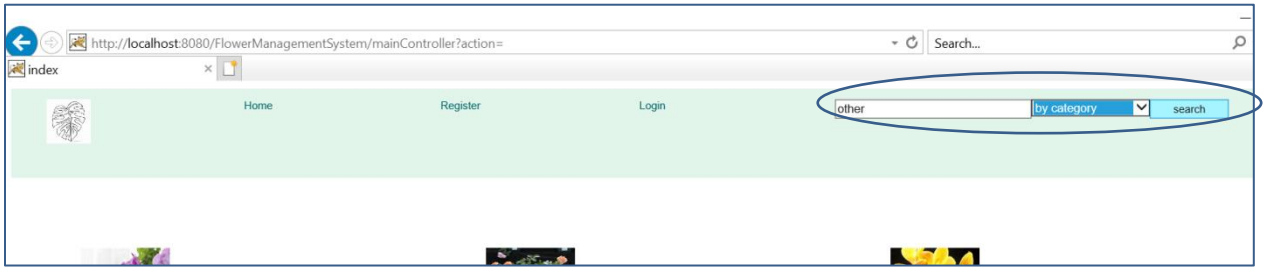

```
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse 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.

}
```

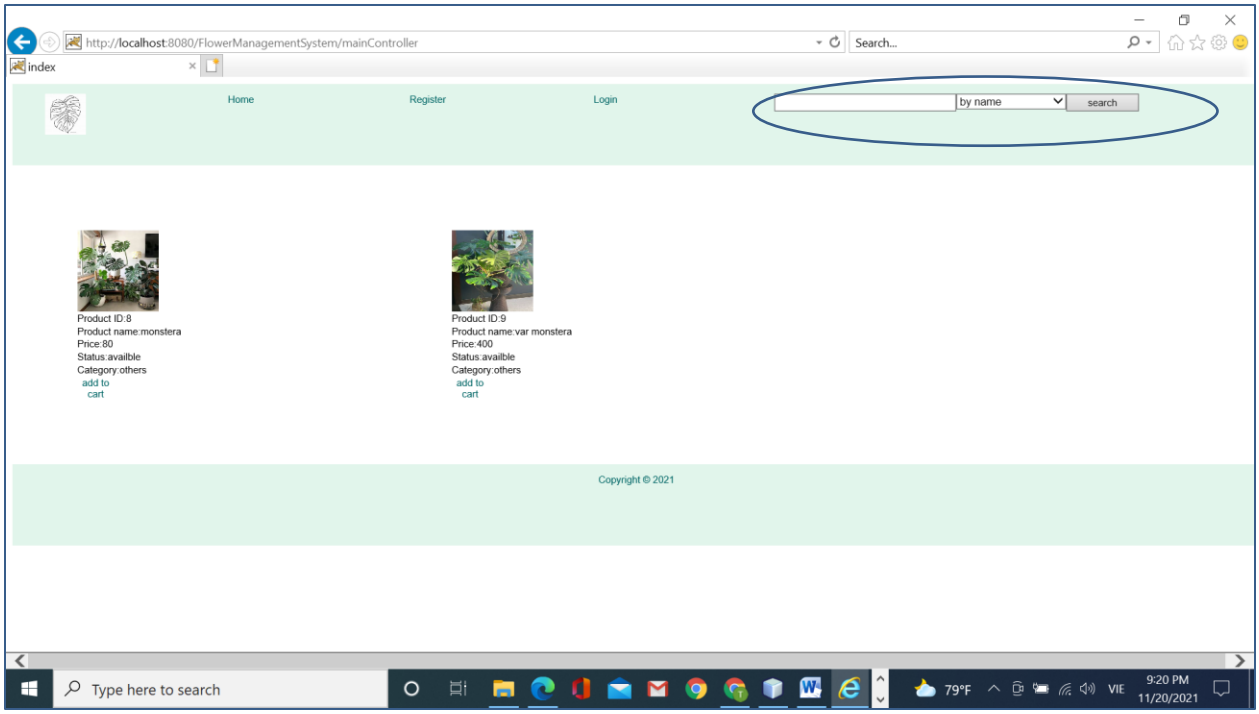
Step 3: Open the page “index.jsp”, edit the code:

```
mainController.java × header.jsp × index.jsp ×
Source  History
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","availble"};
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>
                                 <td>Category:<%= p.getCatename() %></td>
43                               <td><a href="">add to cart</a></td>
44                           </tr>
45                       </table>
46
47              <%          }
48                  }
49              %>
50          </section>
51          <footer>
52              <%@include  file="footer.jsp" %>
53          </footer>
54      </body>
```

Step 4: run your app

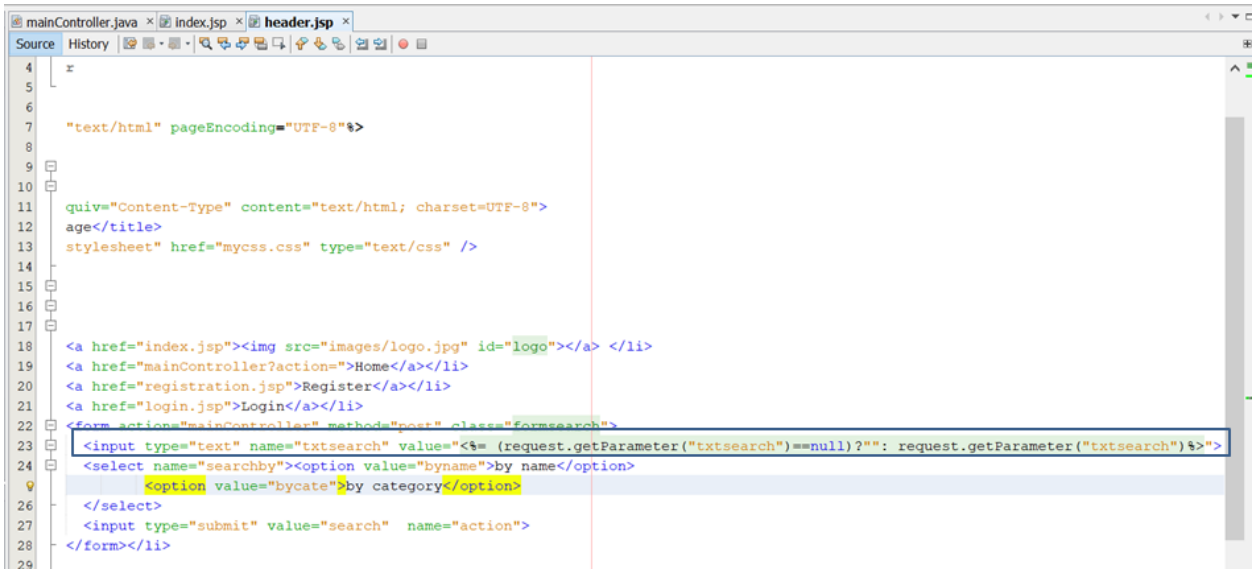
The sample: the user inputs “other”, and click the button “search”



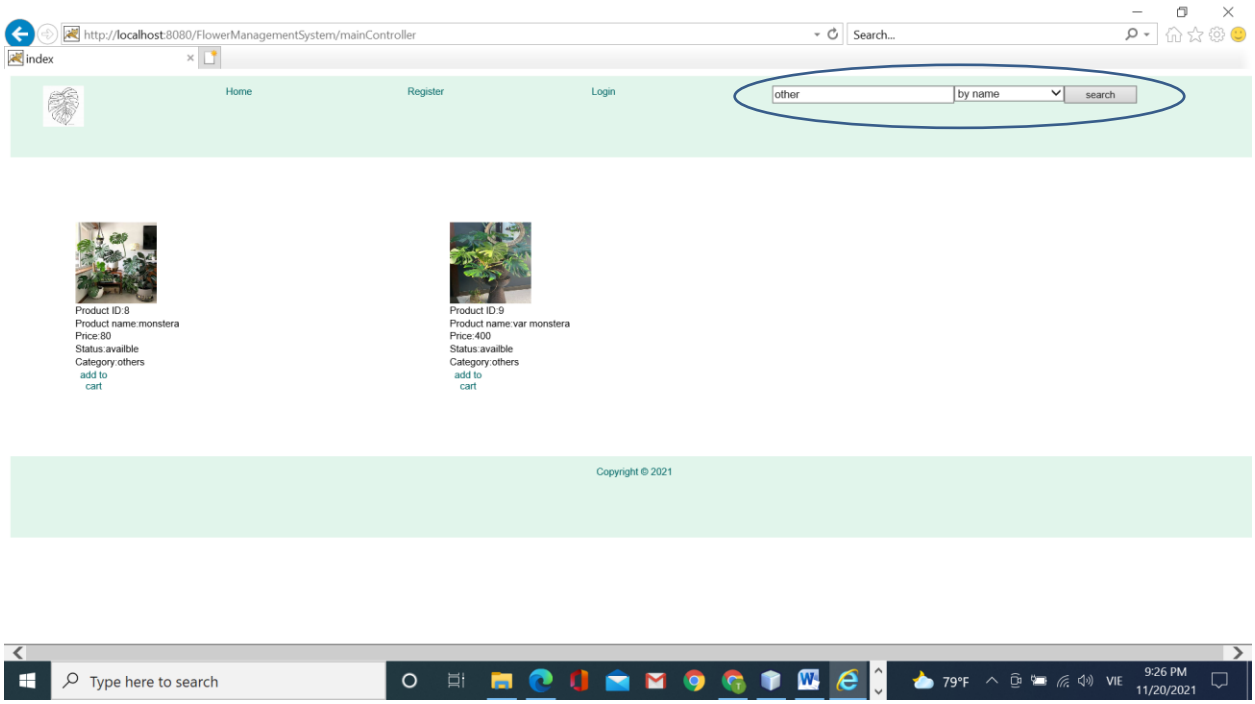


After searching, the appropriate products are displayed. However, ***the input text control can not keep*** the user's search. So, we should edit the page "header.jsp" to overcome this.

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



Step 6: Run your app again.



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

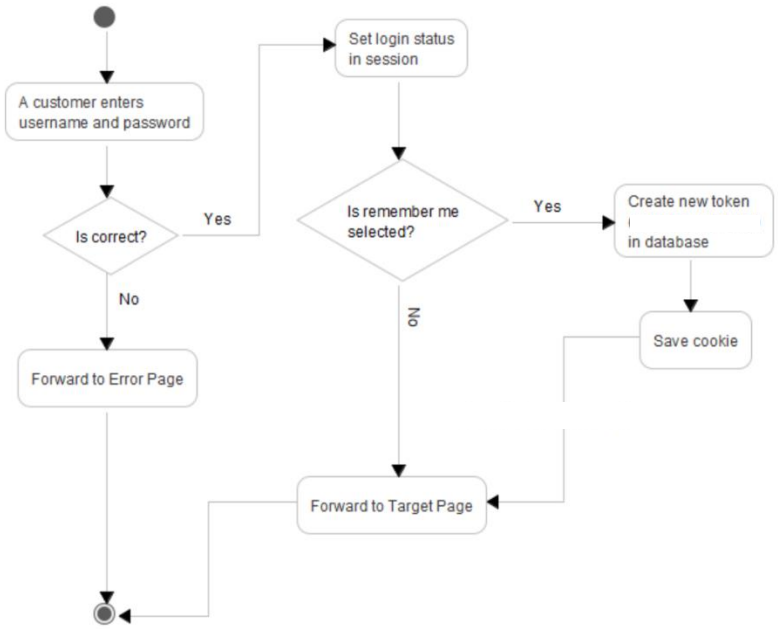
RewriteURL #HiddenField #Cookie # Session #Exception Handling

Learning outcome:

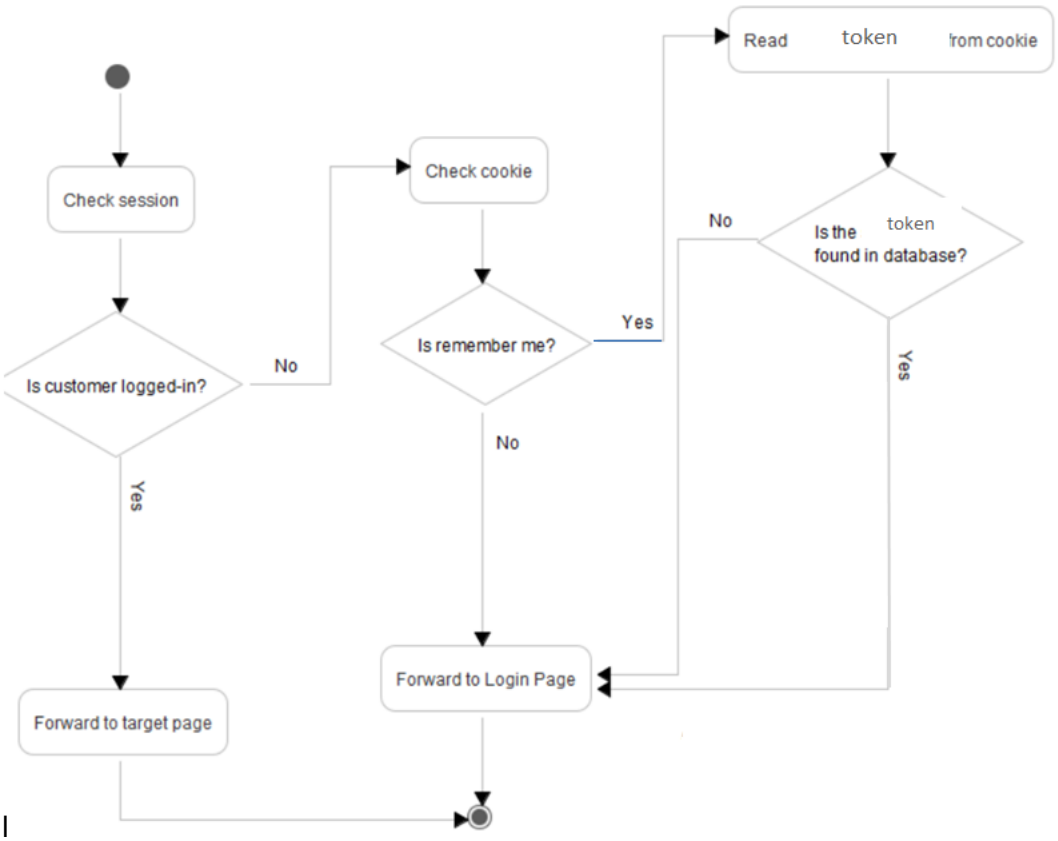
- Know to use the cookie to save login information
- Know to use the session to create the shopping cart
- How to handle exceptions

Part 1: Save the user's login information.

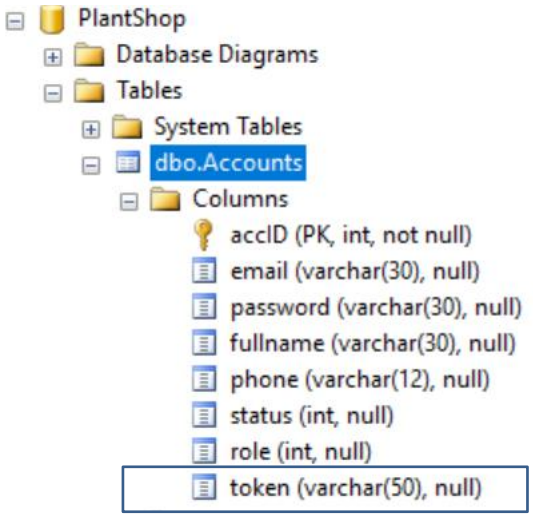
Describe the login function



Using the cookie to login



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

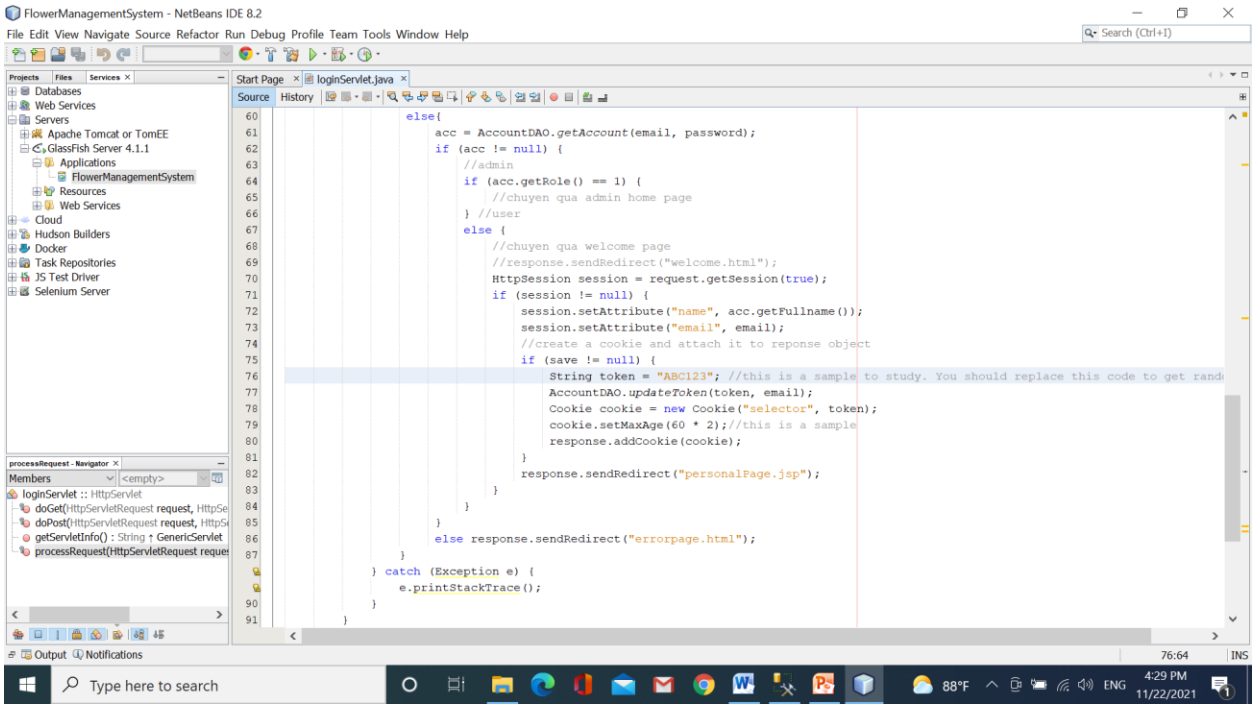


Step 2: Add one more checkbox control in the page “login.jsp”

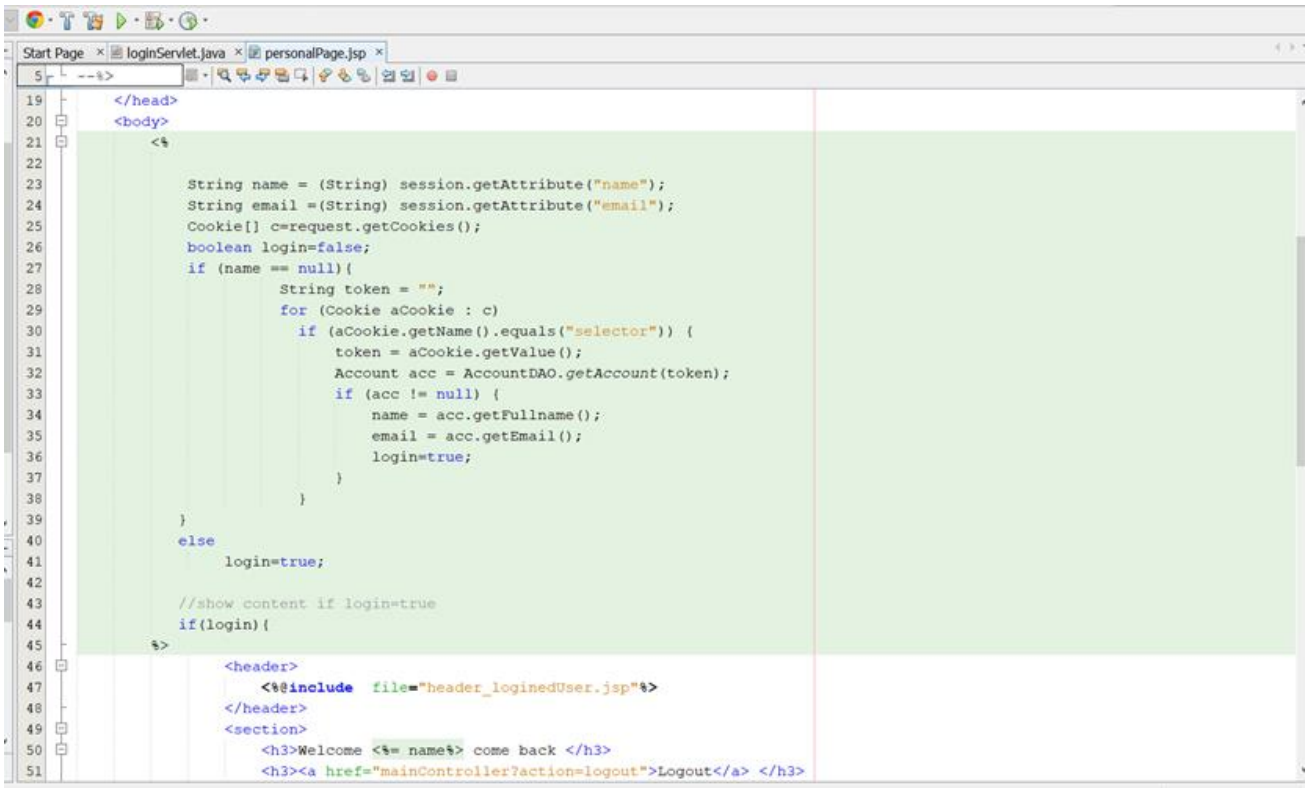
```jsp
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>index</title>
        <link  rel="stylesheet" href="mycss.css" type="text/css" />
    </head>
    <body>
        <header>
            <%@include  file="header.jsp" %>
        </header>
        <section>
            <form action="mainController" method="post" class="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>
```

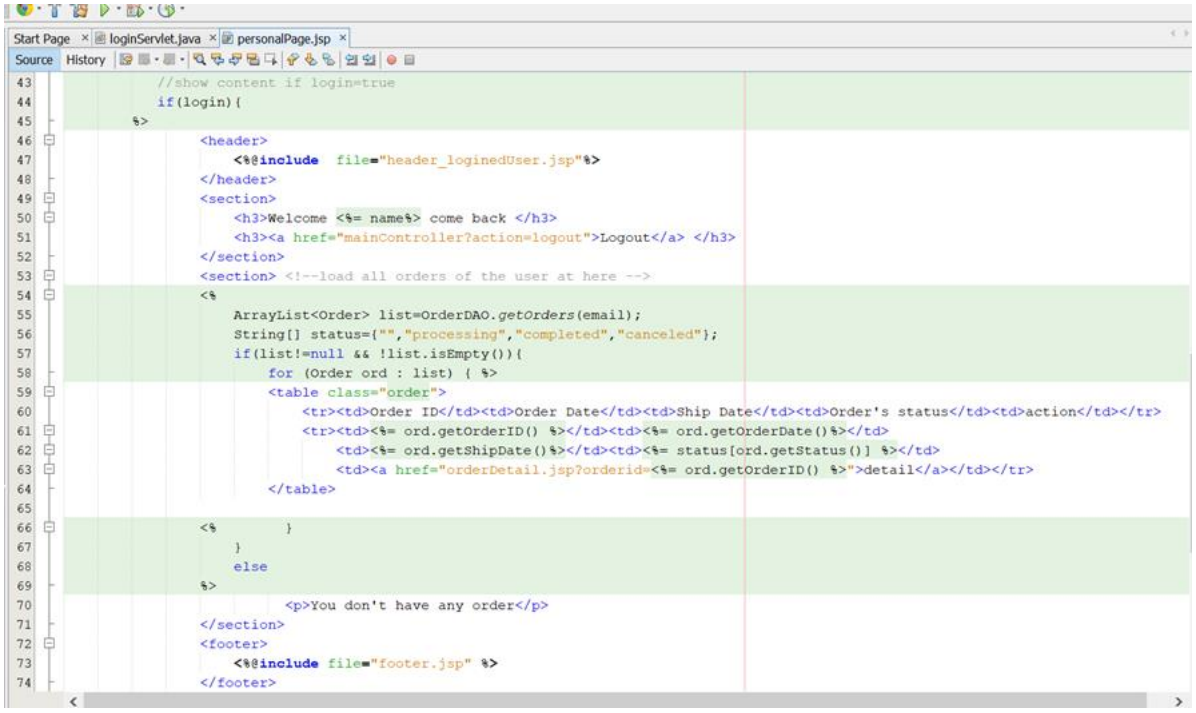
Step 3: On the page “loginServlet”, edit the code:

```jsp
     */
    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");
                }
                else{
                    acc = AccountDAO.getAccount(email, password);
                    if (acc != null) {
                        //admin
                        if (acc.getRole() == 1) {
                            //chuyen qua admin home page
```



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





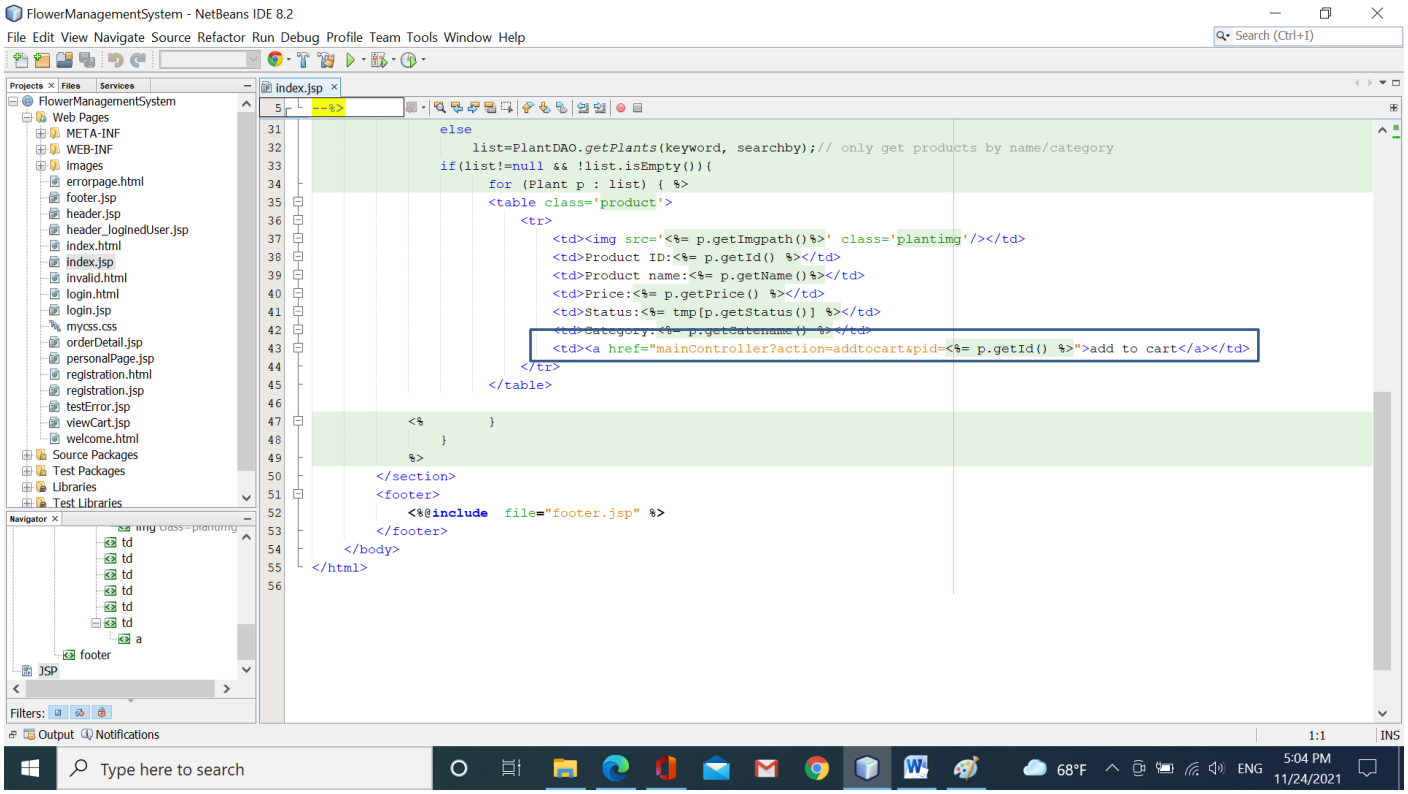
Step 5: run your app to test this function

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

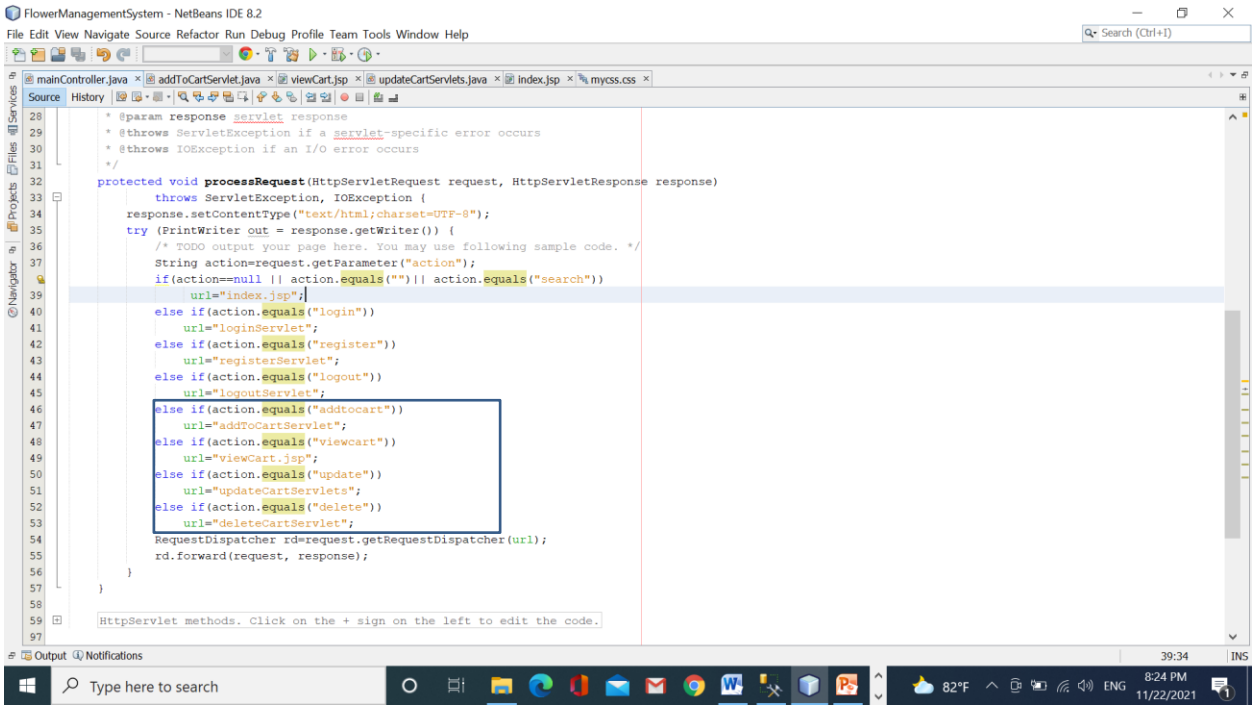
We use HashMap<String, Integer> to store the customer's cart. The key is the productid and value is the quantity. Cart object is stored in the session memory on the server-side (you can change this using cookies to store the customer's cart on the client-side).

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

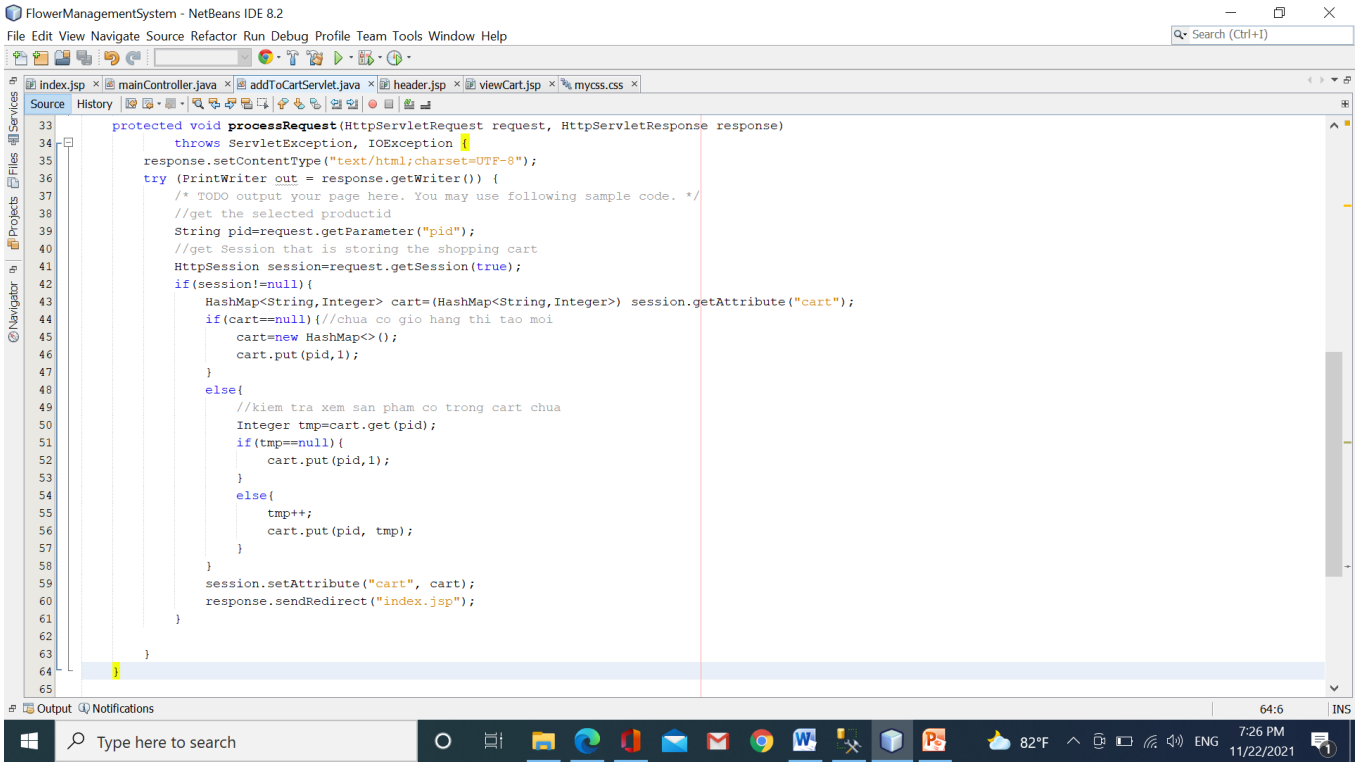
Open the page "index.jsp", edit the code



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

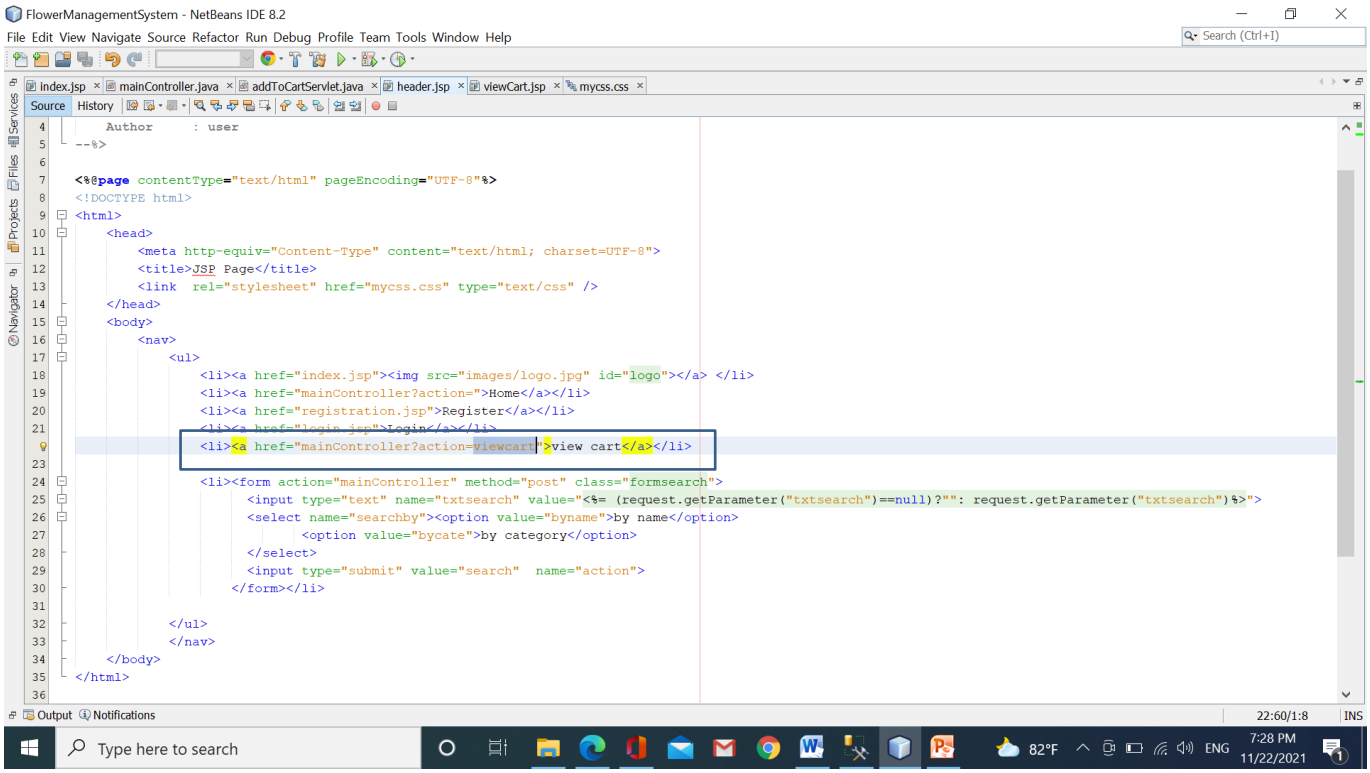


Step 3: create new new servlet named “addToCartServlet”

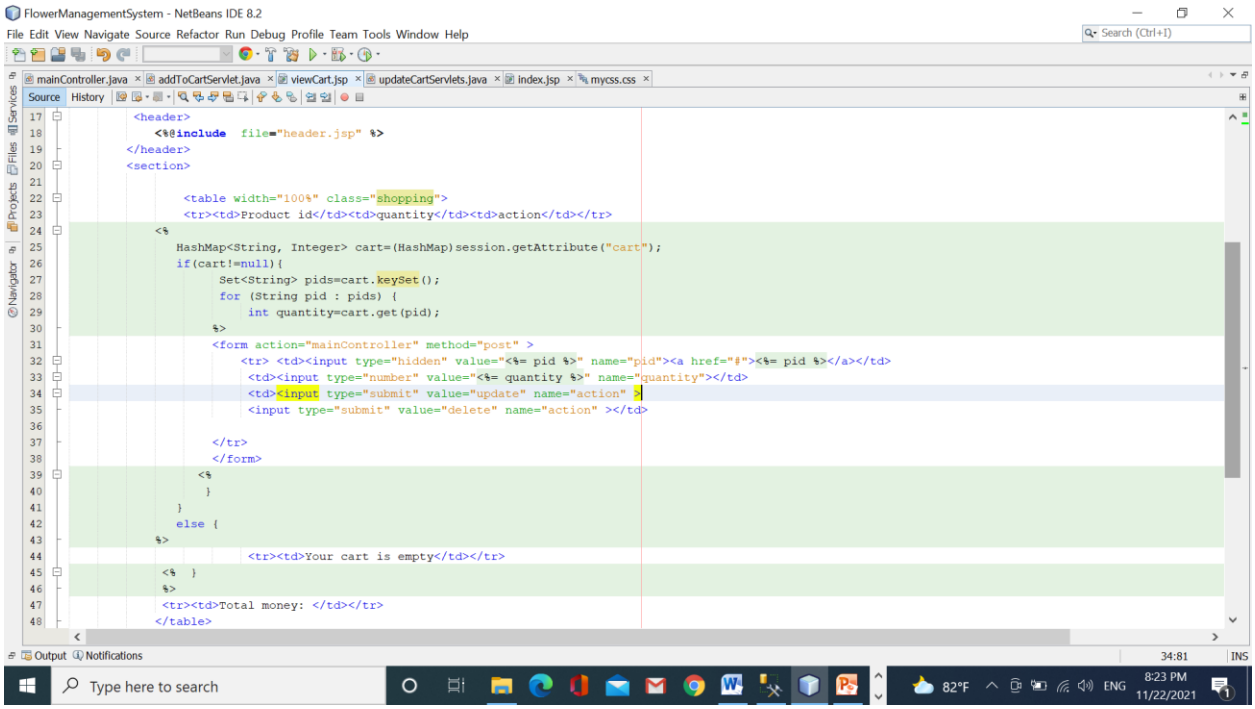


Step 4: View the shopping cart.

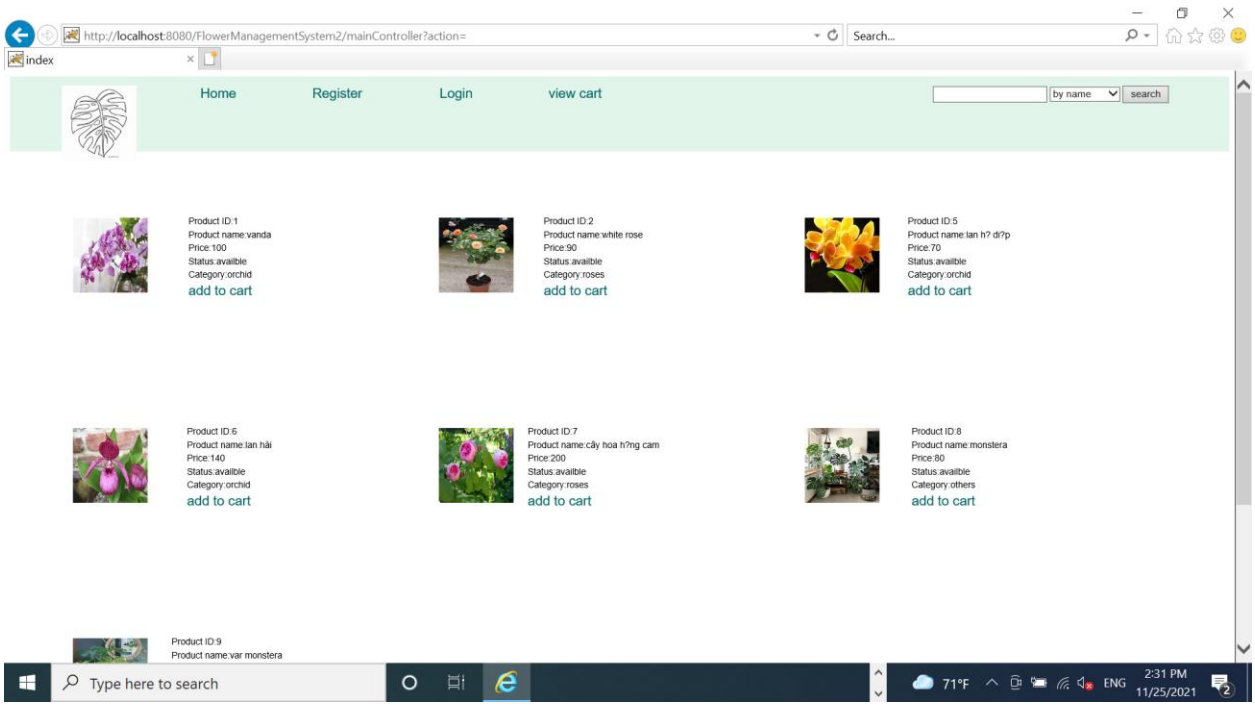
Open the page “header.jsp”, add one more the link “view cart”



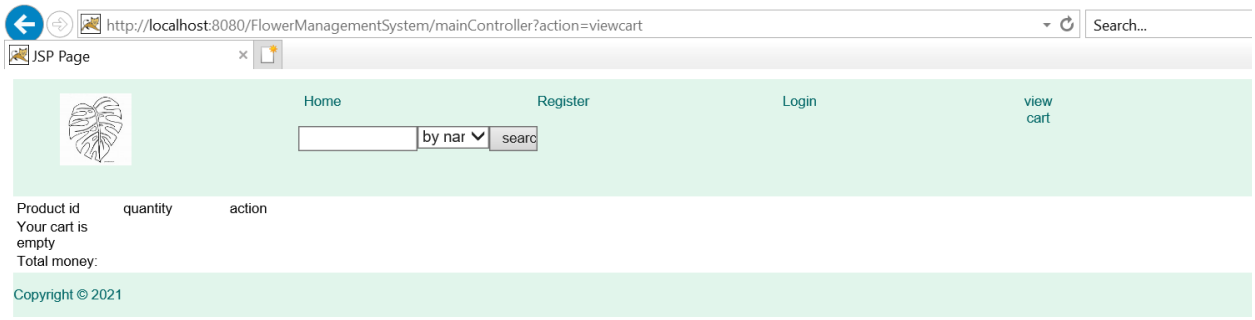
Create a new page named “viewCart.jsp”



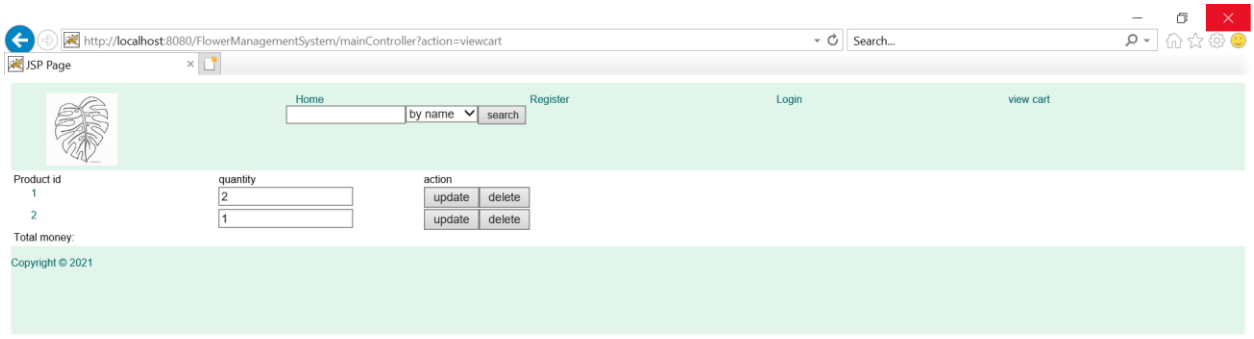
Step 5: Run your app



Click “view cart” from the header



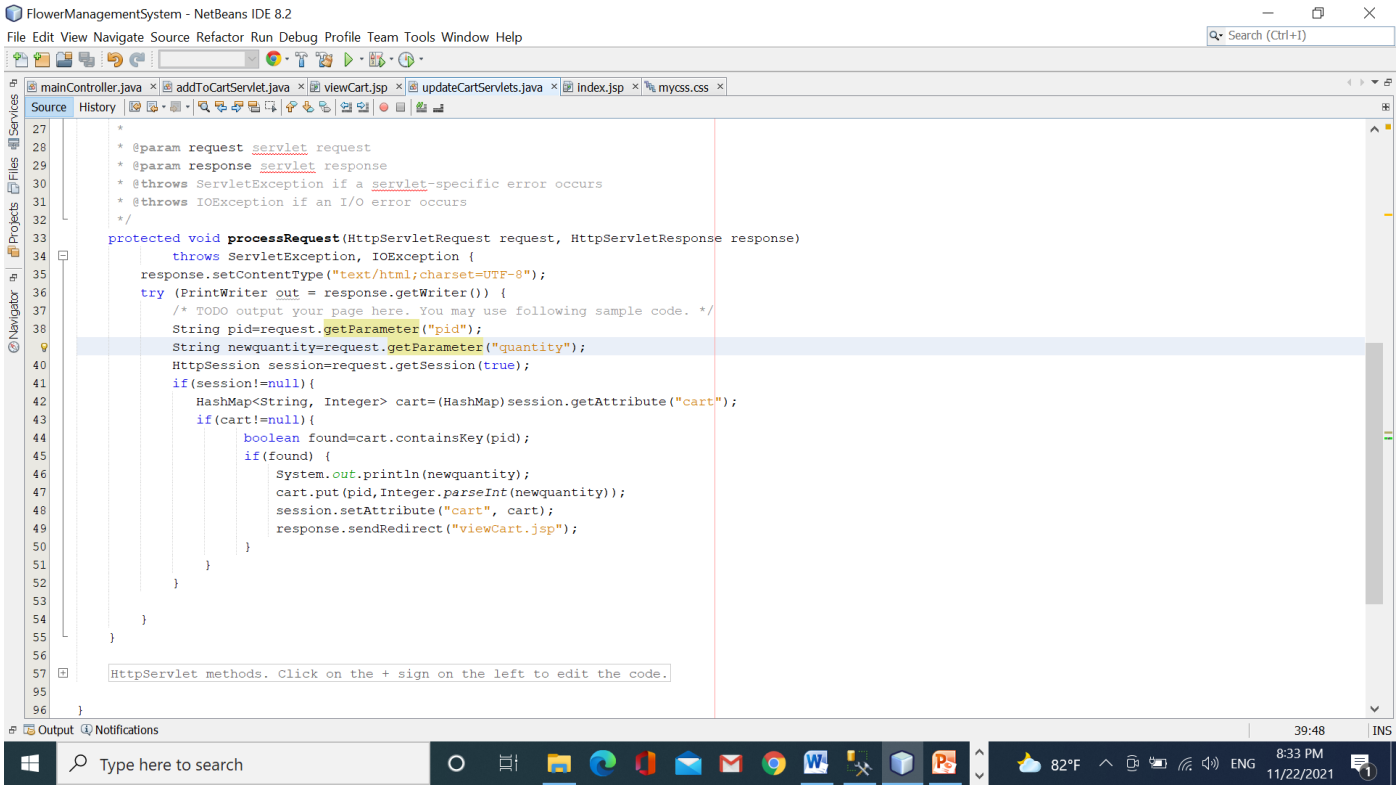
Go back to the home page and click some links “add to cart”



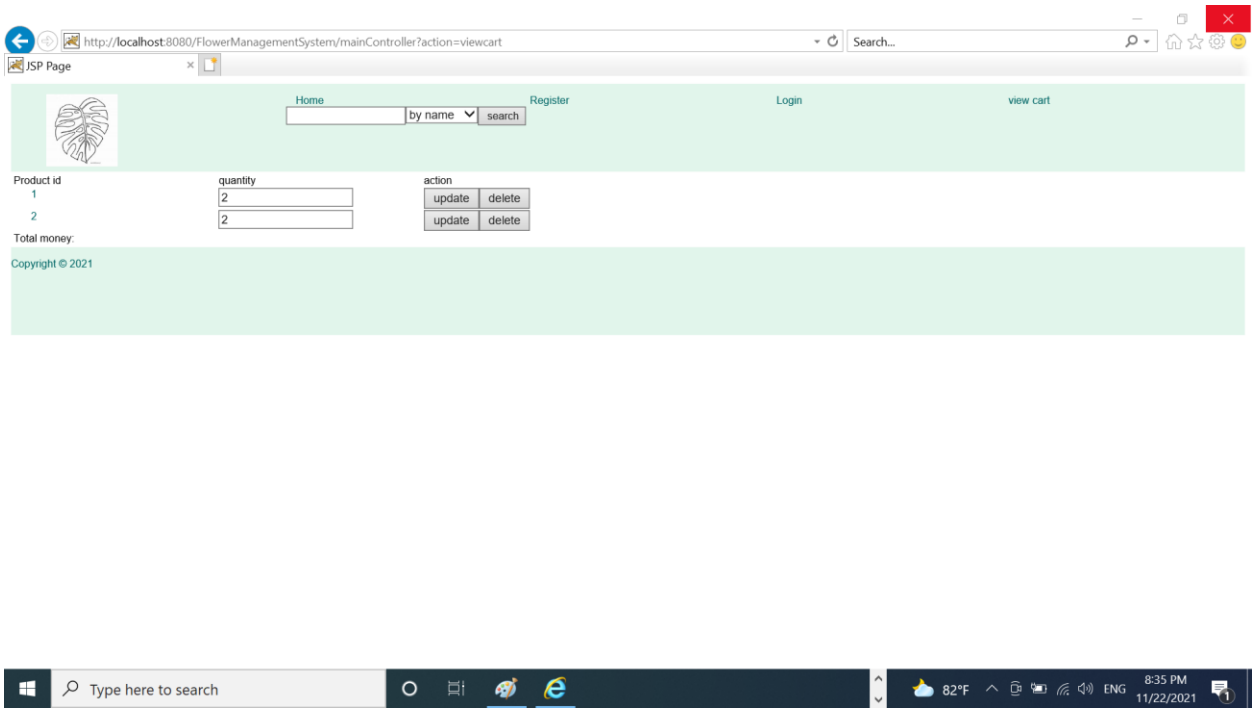
Part 3: change the quantity of the product on the shopping cart.

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



Step 2: Run your app



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><
```

For example 2: validate the user’s phone on the server-side, if there’s an error then a message is sent to the client.

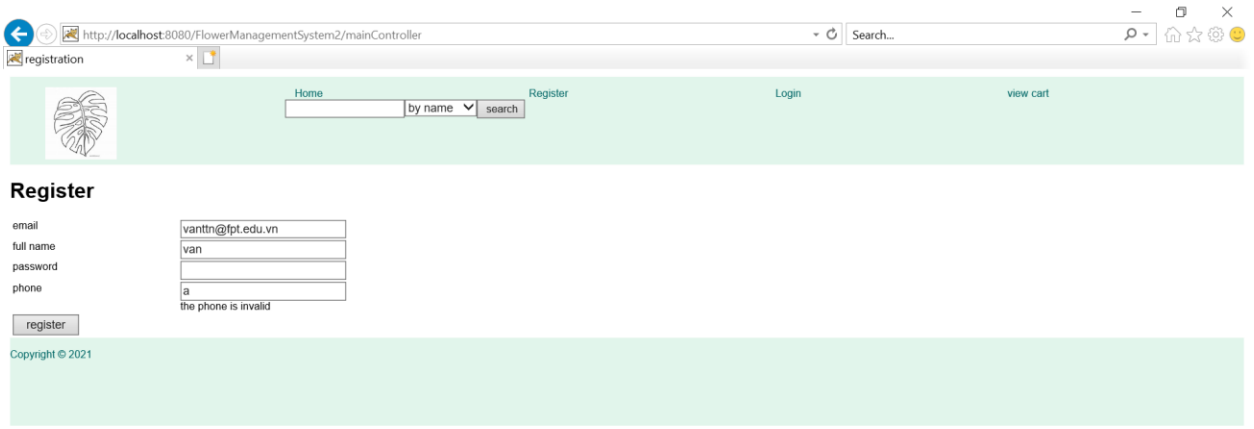
mainController.java | registerServlet.java | registration.jsp

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'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(
        <tr><td>full name</td>
            <td><input type="text" name="txtfullname" required=""  value="<%= (request.getAttribute("txtfullname")==null)?"":request.getAtt
        <tr><td>password</td>
            <td><input type="password" name="txtpassword" required=""></td></tr>
        <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>
```

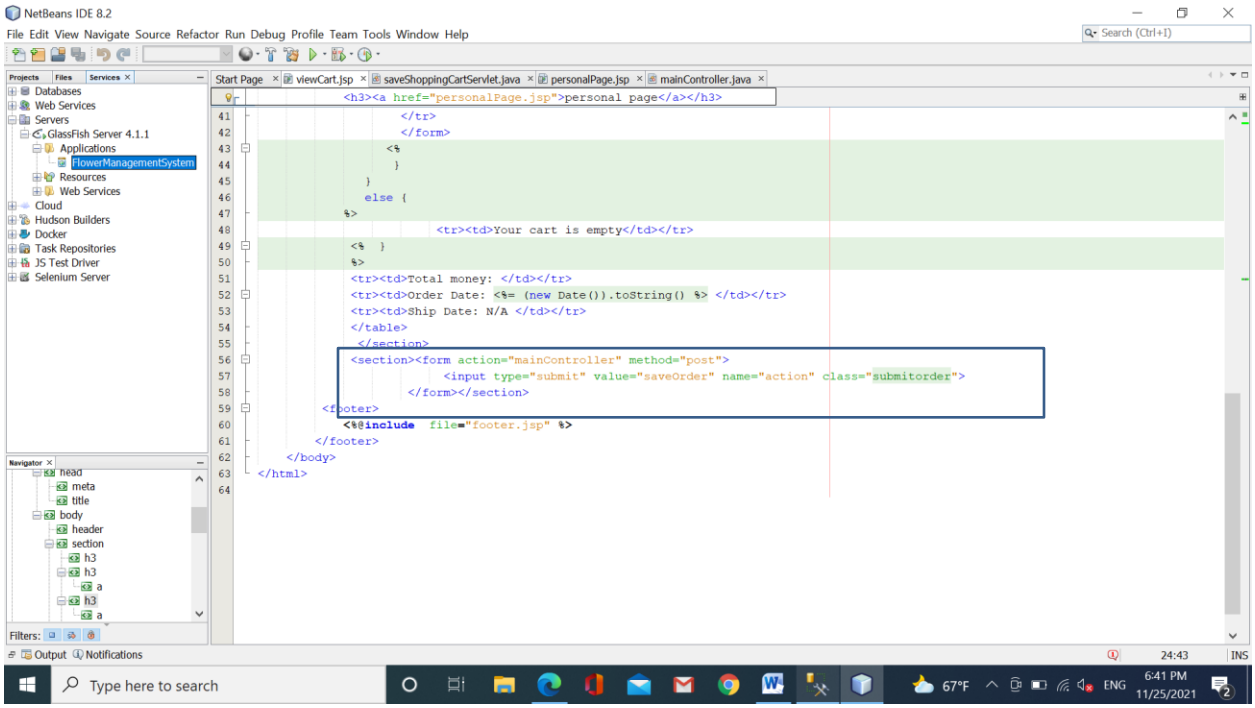


Part 5: Save shopping cart

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

- ✓ Get the accId 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:

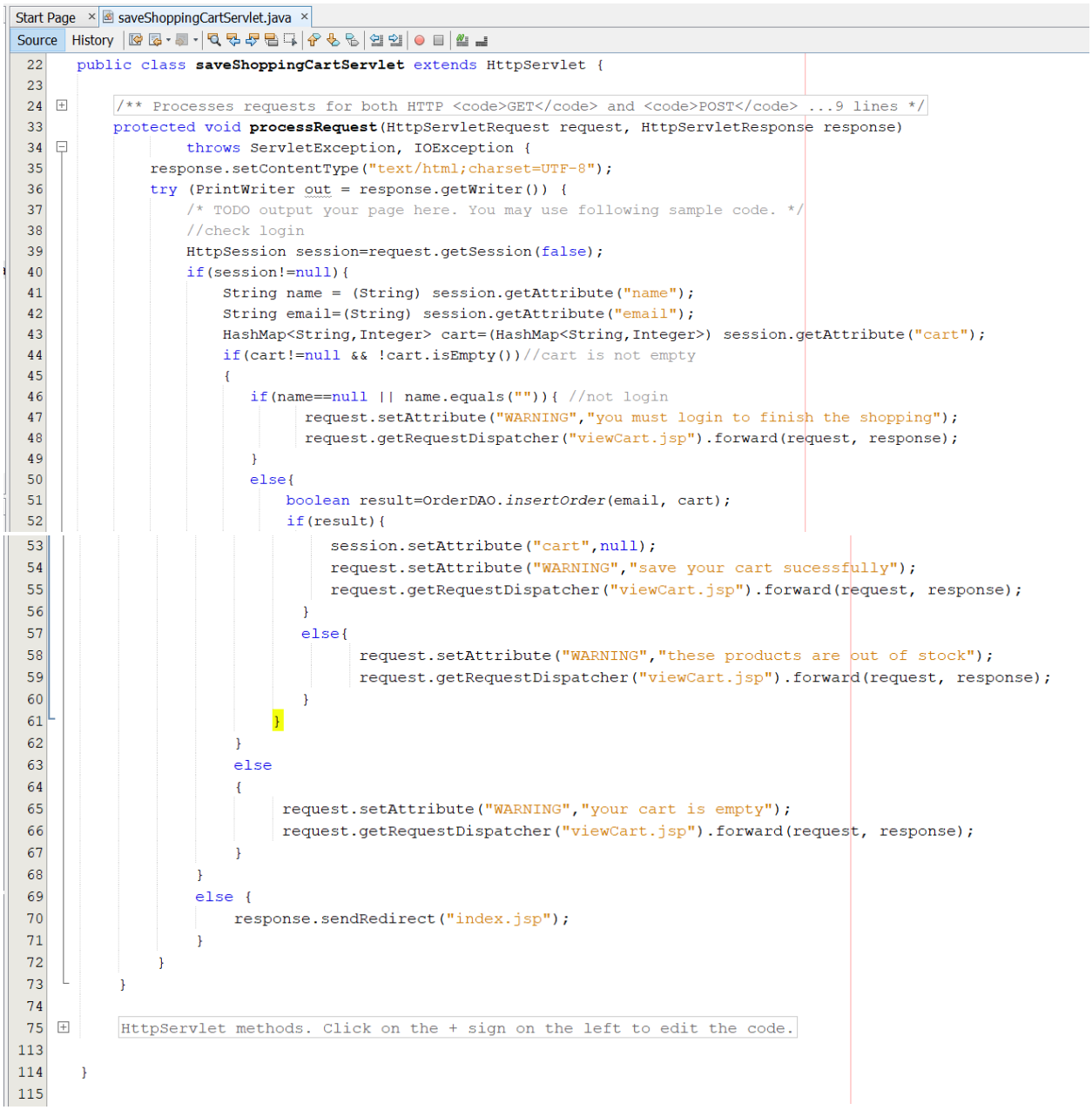
```
else if(action.equals("saveOrder"))
    url="saveShoppingCartServlet";
RequestDispatcher rd=request.getRequestDispatcher(url);
rd.forward(request, response);
```

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 }
60      public static ArrayList<OrderDetail> getOrderDetail(int orderID){...30 lines }
90      public static boolean insertOrder(String email, HashMap<String,Integer> cart){
91          Connection cn=null;
92          boolean result=false;
93          try{
94              cn = DBUtils.makeConnection();
95              if (cn != null) {
96                  int accid=0,orderid=0;
97                  cn.setAutoCommit(false);//off autocommit
98                  //get account id by email
99                  String sql = "select accID from Accounts  where Accounts.email=?";
100                 PreparedStatement pst = cn.prepareStatement(sql);
101                 pst.setString(1, email);
102                 ResultSet rs = pst.executeQuery();
103                 if(rs!=null && rs.next()) accid = rs.getInt("accID");
104                 //insert a new order
105                 System.out.println("accoutid:" + accid);
106                 Date d = new Date(System.currentTimeMillis());
107                 System.out.println("order date:" + d);
108                 sql = "insert Orders(OrdDate,status,AccID) values(?,?,?)";
109                 pst = cn.prepareStatement(sql);
110                 pst.setDate(1, d);
111                 pst.setInt(2, 1);
112                 pst.setInt(3, accid);
113                 pst.executeUpdate();
```

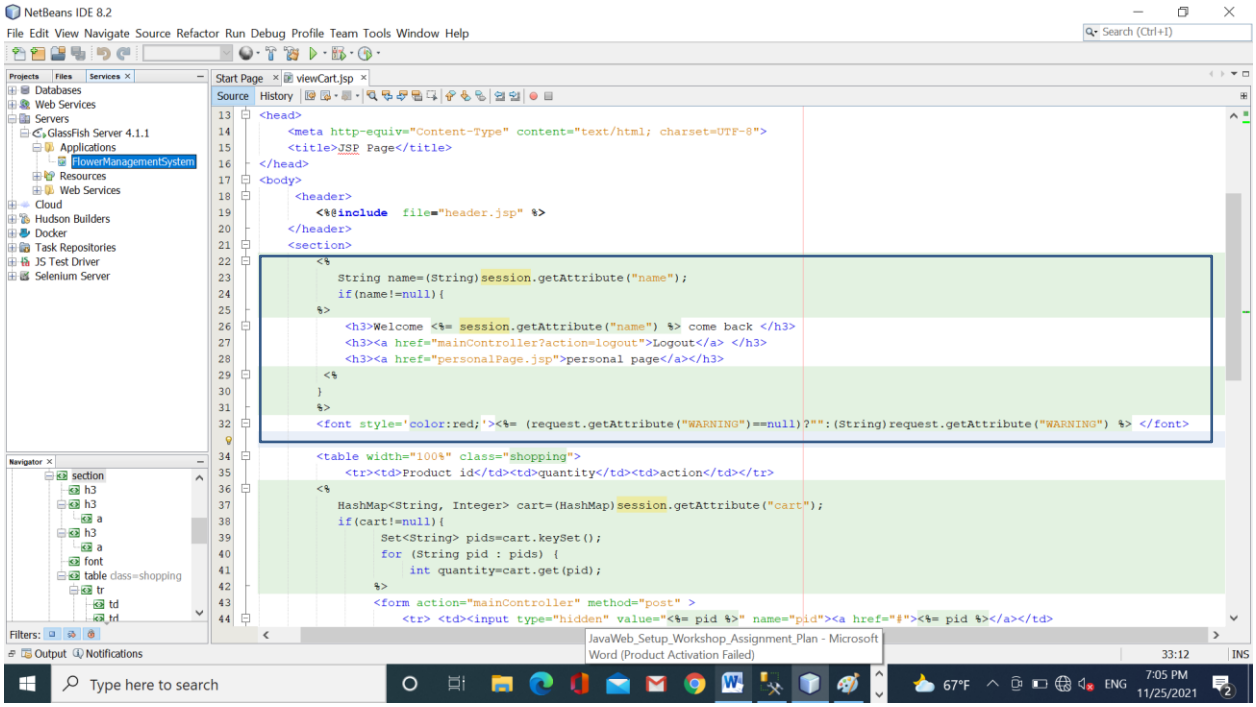
```
114                 //get order id that is the lagest number
115                 sql = "select top 1 orderID from Orders order by orderId desc "
116                 pst = cn.prepareStatement(sql);
117                 rs = pst.executeQuery();
118                 if(rs!=null && rs.next()) orderid = 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){
                    ex.printStackTrace();
142             }
                e.printStackTrace();
144             result=false;
145         }
146         finally{ try {
                    if(cn!=null) cn.close();
                } catch (Exception e) { e.printStackTrace();      }
149         }
150
151         return result;
152     }
153 }
154
```

Step 3: New a servlet named “saveShoppingCartServlet”

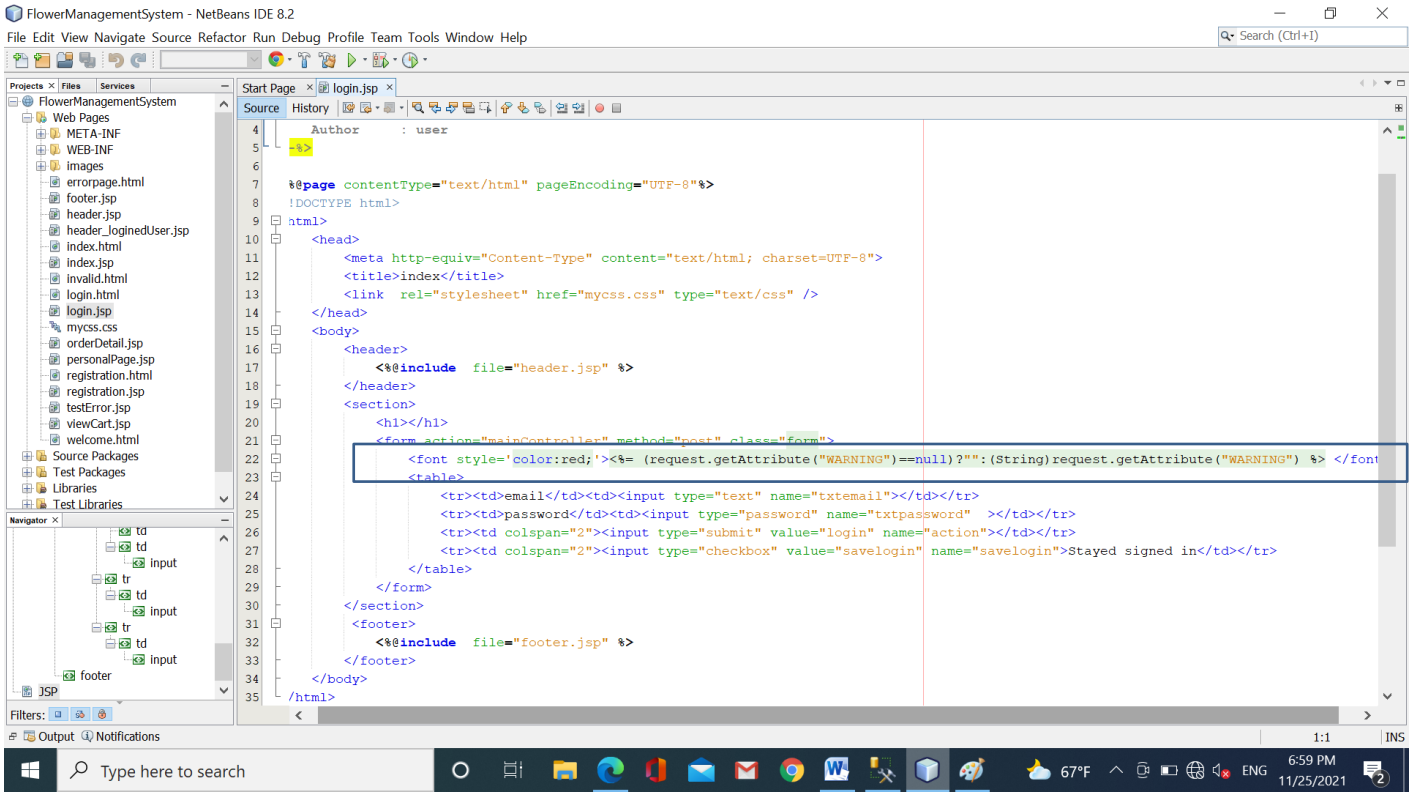


```
Start Page | saveShoppingCartServlet.java
Source | History
22  public class saveShoppingCartServlet extends HttpServlet {
23
24      /** Processes requests for both HTTP <code>GET</code> and <code>POST</code> ...9 lines */
33      protected void processRequest(HttpServletRequest request, HttpServletResponse response)
34              throws ServletException, IOException {
35          response.setContentType("text/html;charset=UTF-8");
36          try (PrintWriter out = response.getWriter()) {
37              /* TODO output your page here. You may use following sample code. */
38              //check login
39              HttpSession session=request.getSession(false);
40              if(session!=null){
41                  String name = (String) session.getAttribute("name");
42                  String email=(String) session.getAttribute("email");
43                  HashMap<String,Integer> cart=(HashMap<String,Integer>) session.getAttribute("cart");
44                  if(cart!=null && !cart.isEmpty())//cart is not empty
45                  {
46                     if(name==null || name.equals("")){ //not login
47                           request.setAttribute("WARNING","you must login to finish the shopping");
48                           request.getRequestDispatcher("viewCart.jsp").forward(request, response);
49                     }
50                     else{
51                         boolean result=OrderDAO.insertOrder(email, cart);
52                         if(result){
53                              session.setAttribute("cart",null);
54                              request.setAttribute("WARNING","save your cart sucessfully");
55                              request.getRequestDispatcher("viewCart.jsp").forward(request, response);
56                          }
57                          else{
58                               request.setAttribute("WARNING","these products are out of stock");
59                               request.getRequestDispatcher("viewCart.jsp").forward(request, response);
60                          }
61                      }
62                  }
63                  else
64                  {
65                       request.setAttribute("WARNING","your cart is empty");
66                       request.getRequestDispatcher("viewCart.jsp").forward(request, response);
67                  }
68              }
69              else {
70                  response.sendRedirect("index.jsp");
71              }
72          }
73      }
74
75      HttpServlet methods. Click on the + sign on the left to edit the code.
113
114 }
115
```

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

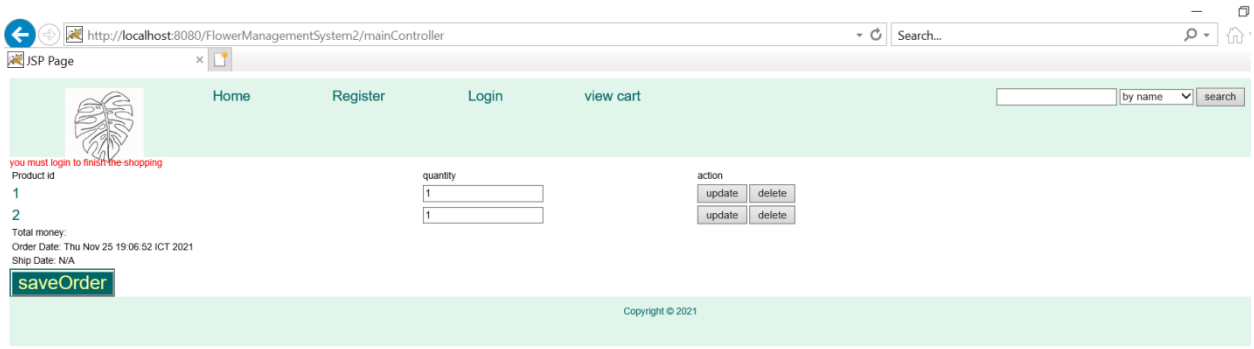


Step 5: Open the page “login.jsp”, add more the code:

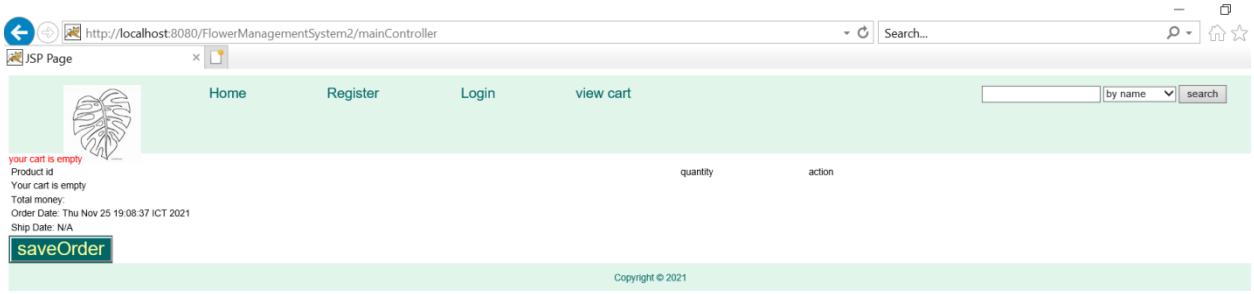


Step 6: Run your app

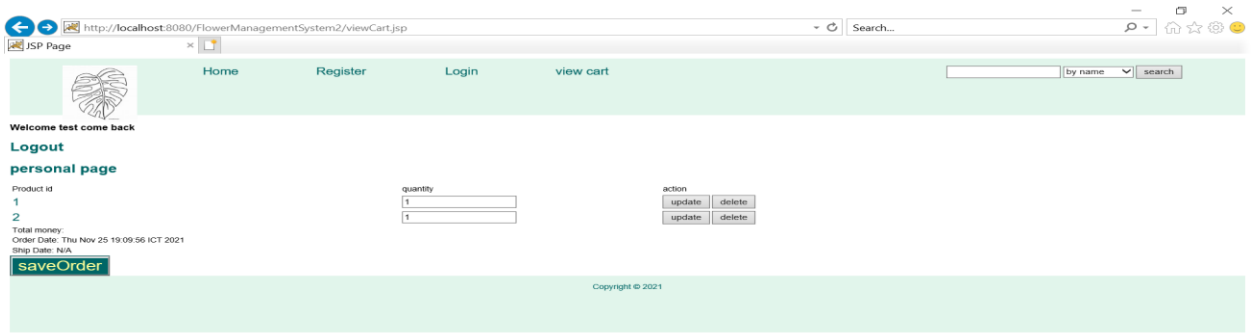
Test case 1: Submit the order without logging

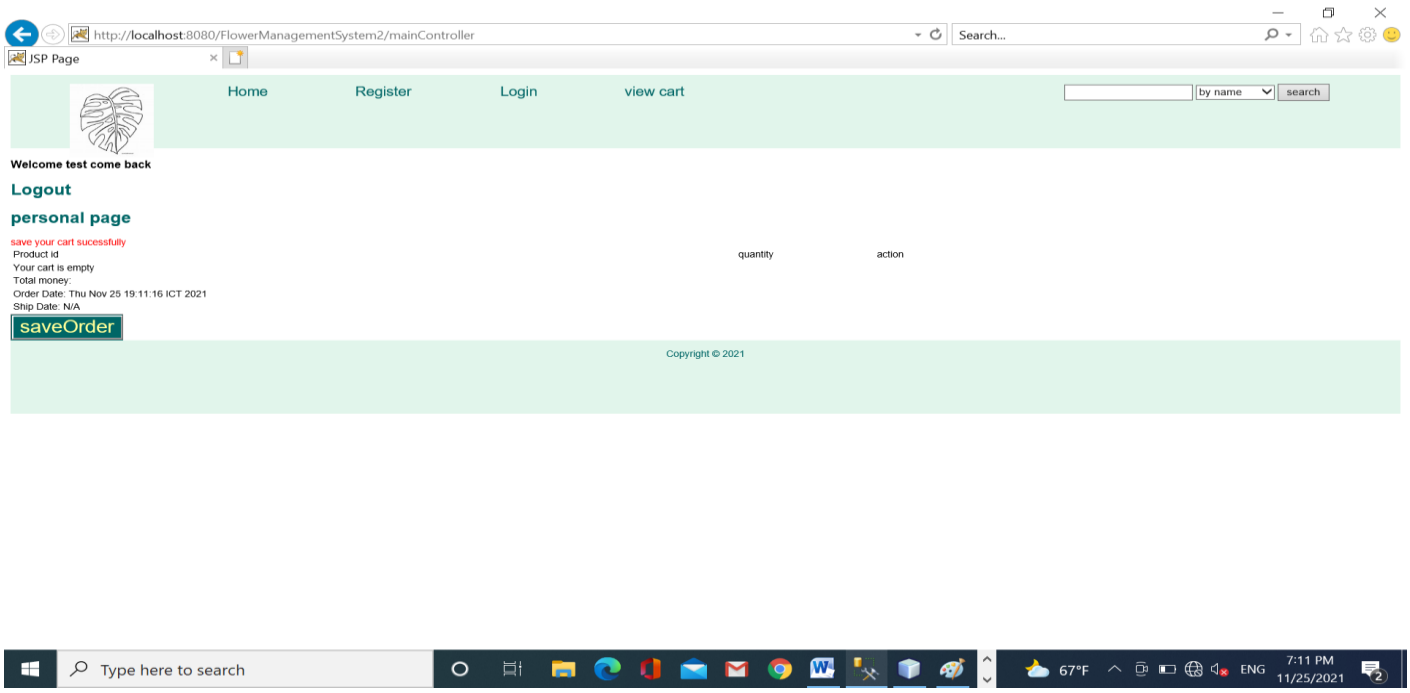


Test case 2: submit the order without any selected product

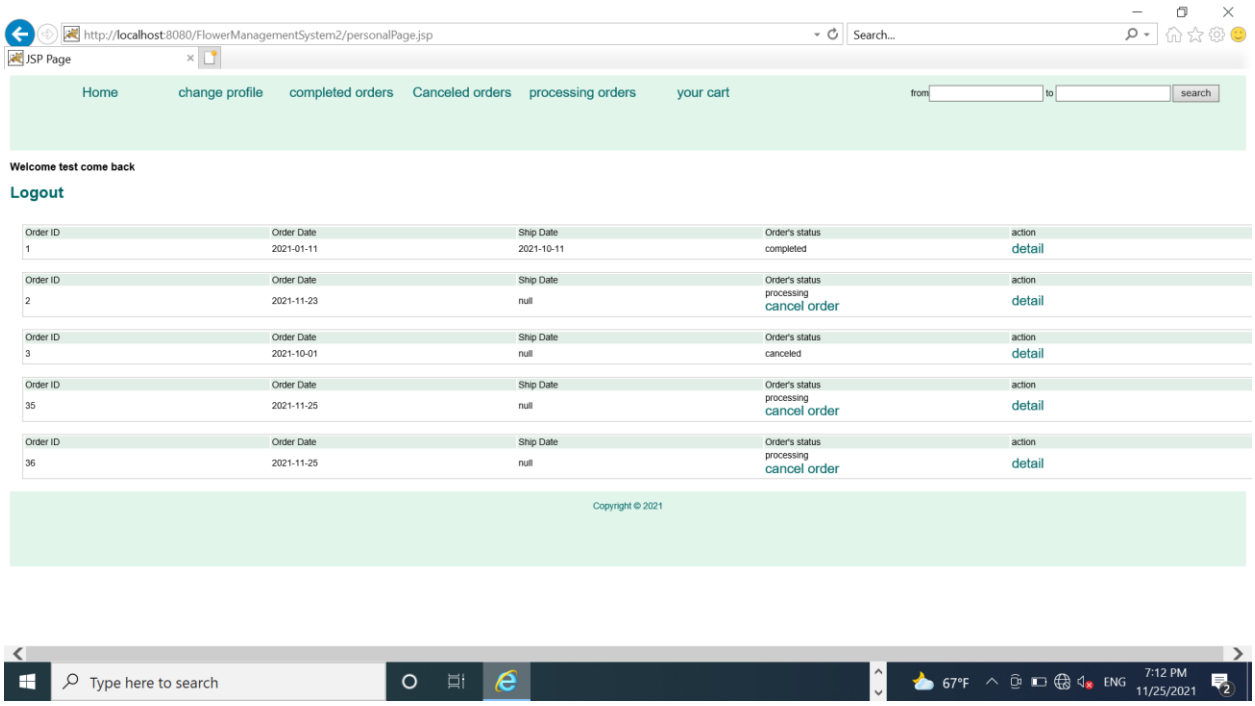


Test case 3: submit the order successfully





Click the link “personalPage” to view history



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)

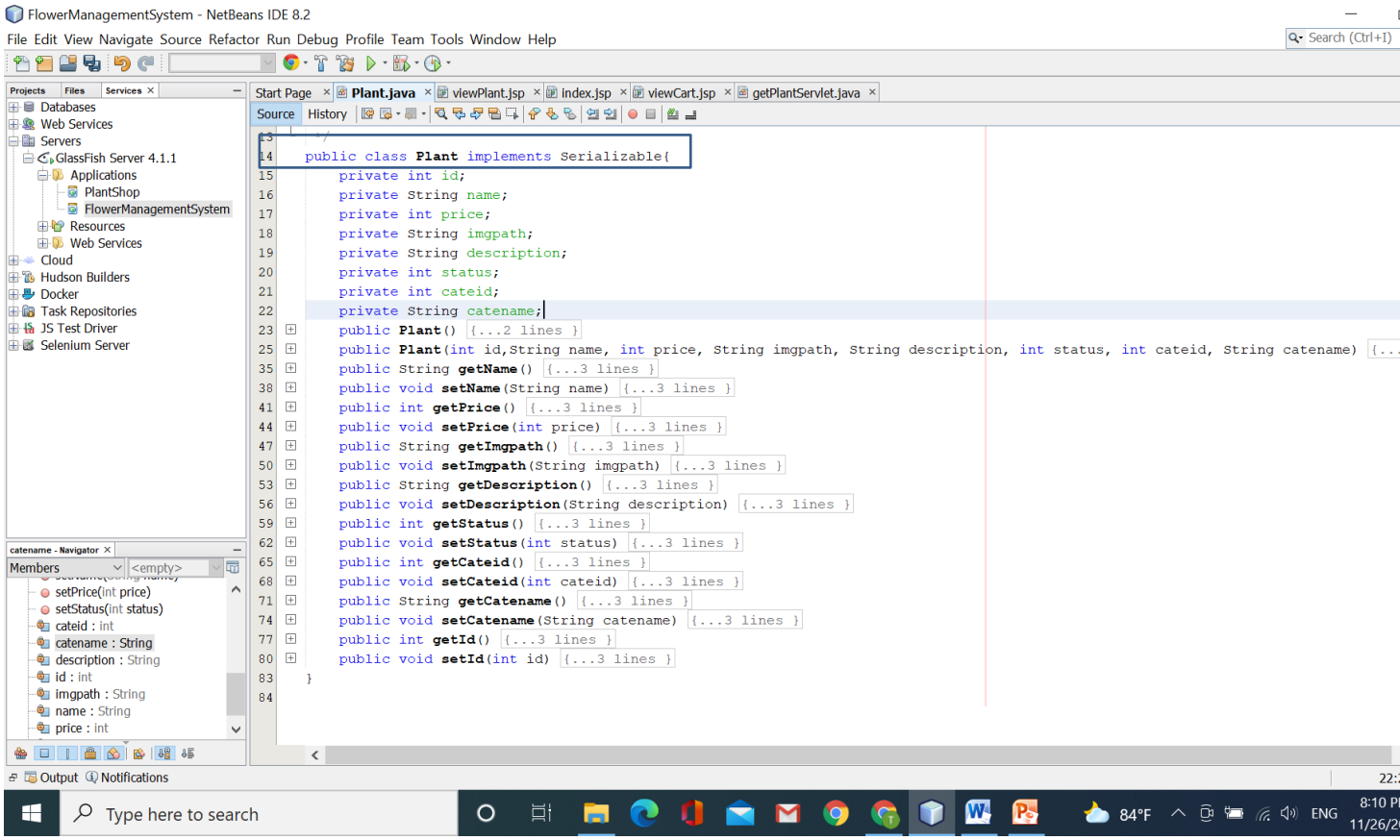
Learning outcome:

- 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

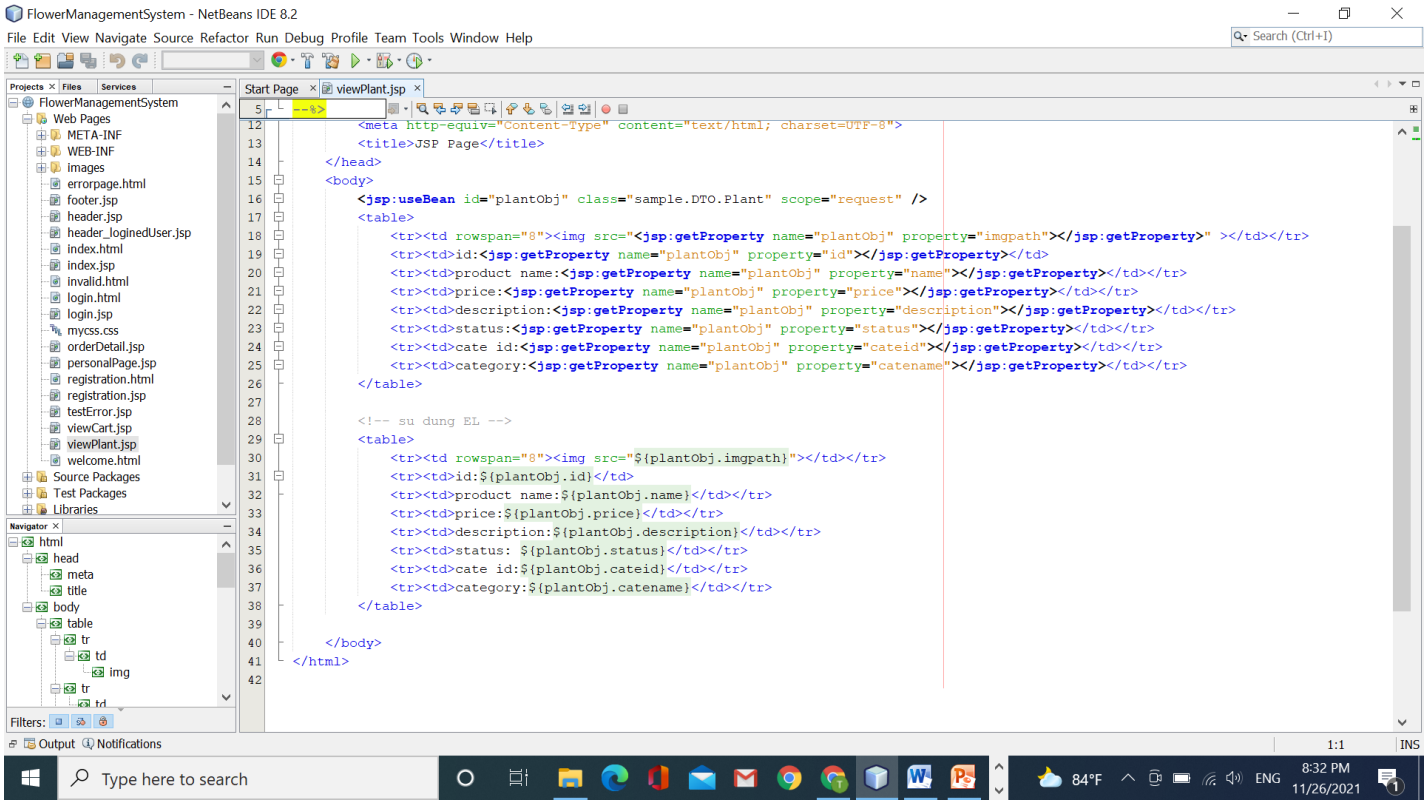
Source | History

```
21      public static ArrayList<Plant> getPlants(String keyword,String searchby){...41 lines }
62
63      public static Plant getPlant(int pid){
64          Plant p=null;
            Connection cn=null;
66          try {
67              cn=DBUtils.makeConnection();
68              if(cn!=null){
69                  String sql = "select PID,PName,price,imgPath,description,status,Plants.CateID as cateID,CateName\n"
70                          + "from Plants, Categories\n"
                            + "where Plants.CateID=Categories.CateID and PID=? ";
72                  PreparedStatement pst=cn.prepareStatement(sql);
73                  pst.setInt(1, pid);
74                  ResultSet rs=pst.executeQuery();
75                  if(rs!=null && rs.next())
76                  {
77                     pid=rs.getInt("PID");
78                     String pname=rs.getString("PName");
79                     int price=rs.getInt("price");
80                     String imgPath=rs.getString("imgPath");
81                     String des=rs.getString("description");
82                     int status=rs.getInt("status");
83                     int cateid=rs.getInt("cateID");
84                     String cateName=rs.getString("CateName");
85                     p=new Plant(pid, pname, price, imgPath, des, status, cateid, cateName);
86                  }
87              }
88          } catch (Exception e) {
                e.printStackTrace();
90          }
91          finally{
92              try {
93
94              } catch (Exception e) {
                    e.printStackTrace();
96              }
97          }
98          return p;
99      }
100 }
```

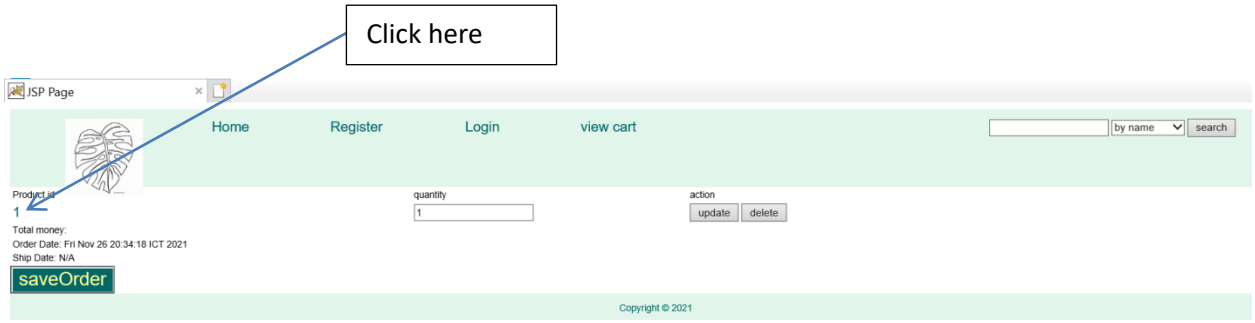
Step 4: In the package “sample.servlet”, create a new servlet named “getPlantServlet”.

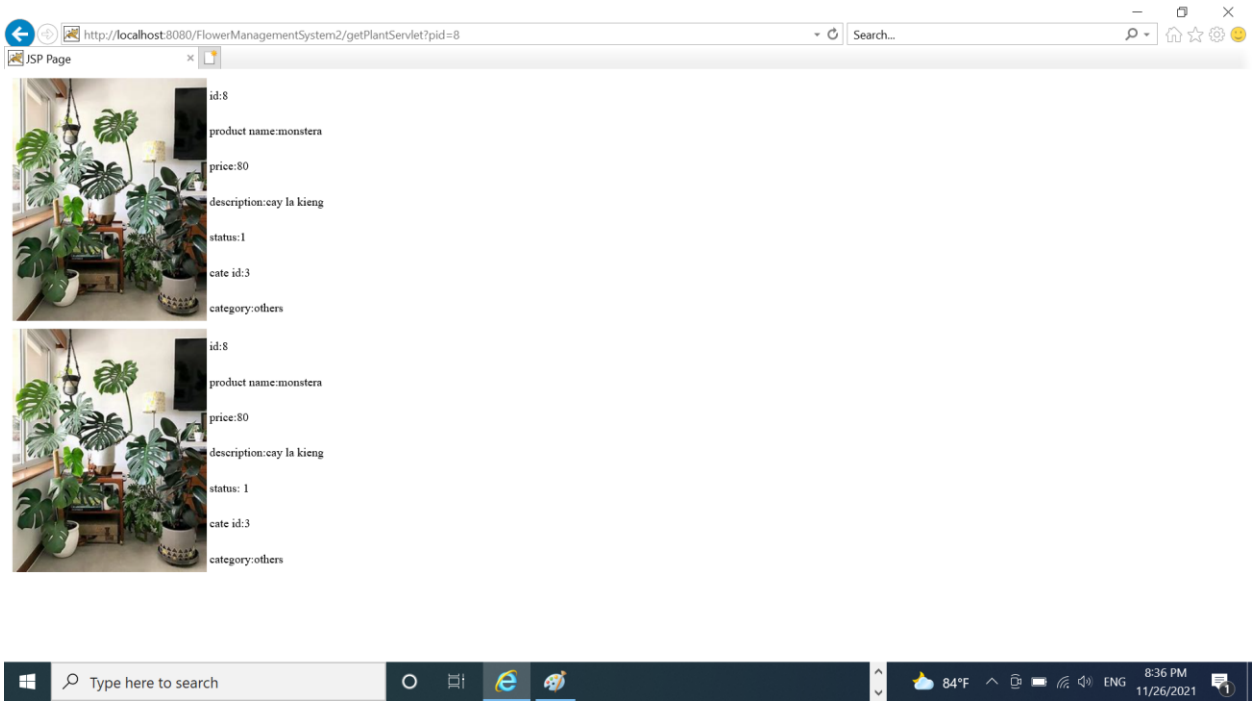
```java
package sample.servlet;
import ...8 lines

/**...4 lines */
public class getPlantServlet extends HttpServlet {

    /** Processes requests for both HTTP <code>GET</code> and <code>POST</code> ...9 lines */
    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. */
            int pid=Integer.parseInt(request.getParameter("pid"));
            Plant p=PlantDAO.getPlant(pid);
            if(p!=null) {
                request.setAttribute("plantObj", p);
                request.getRequestDispatcher("viewPlant.jsp").forward(request, response);
            }
        }
    }

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

}
```

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:

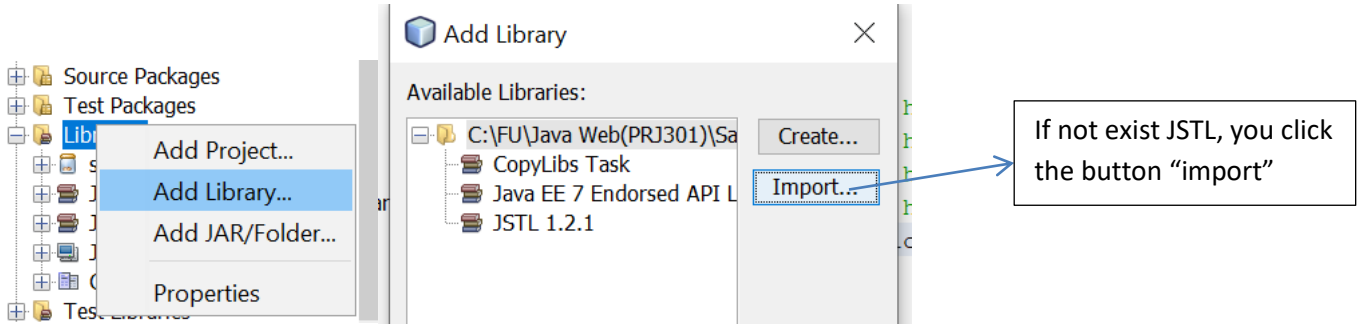
- How to apply JSTL to remove all java code in the JSP pages

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:



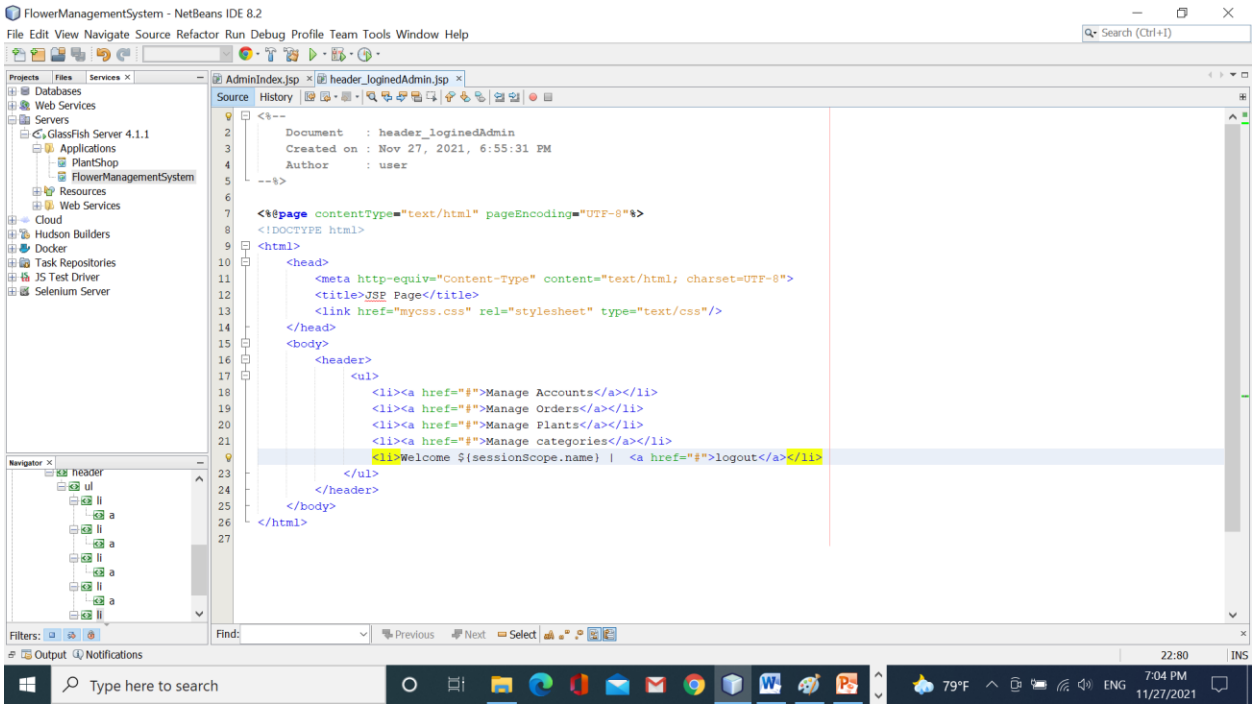
Part 1: Create a home page for the admin

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 reponse 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 {
```

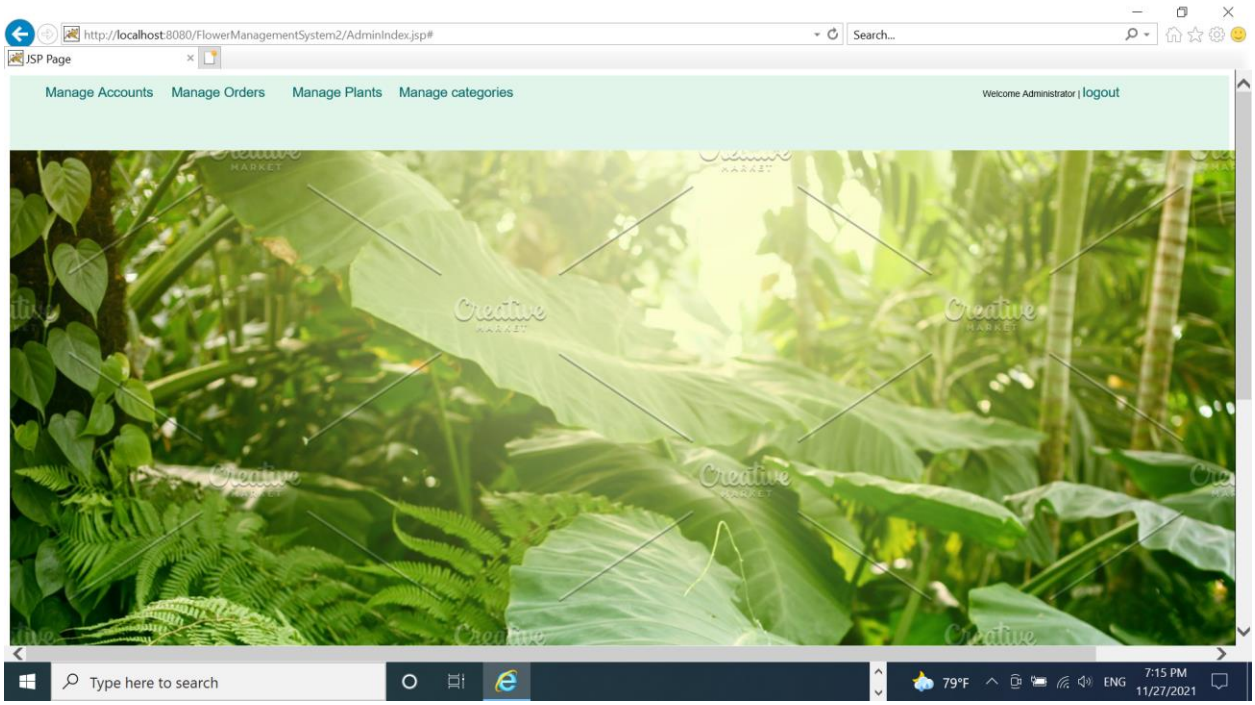
Step 2: Create a new page named “header_loginedAdmin.jsp”



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_loginedAdmin.jsp"></c:import>
19          <section class="right">
20              <img src="images/background.jpg"/>
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)

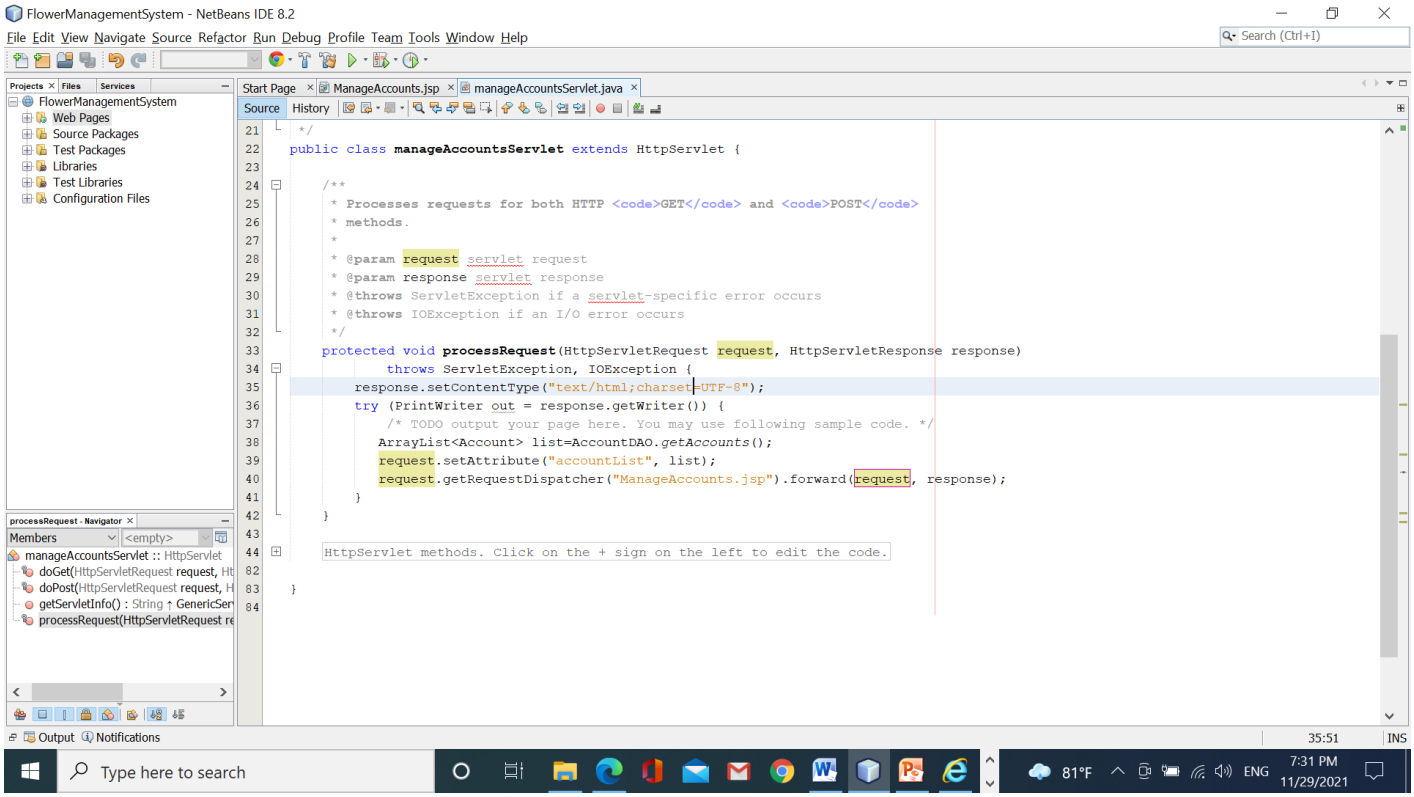
Step 1: Open the page "header_loginedAdmin.jsp", add the code

```
<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

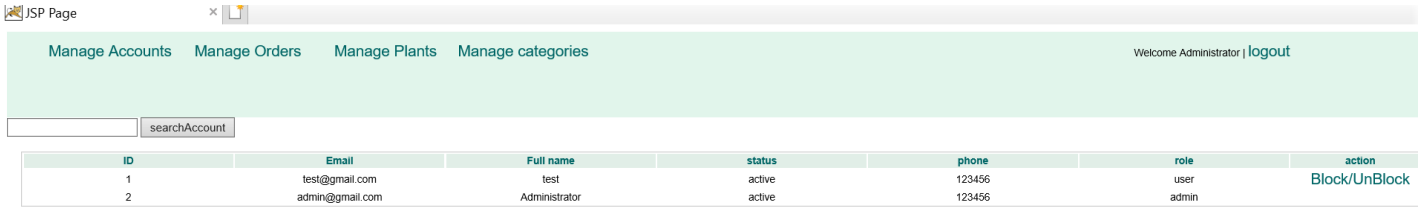
Source | History

```jsp
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib  uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
        <link href="mycss.css" rel="stylesheet" type="text/css"/>
    </head>
    <body>
        <c:import url="header_loginedAdmin.jsp"/>
        <form action="mainController" method="post">
            <input type="text" name="txtSearch">
            <input type="submit" value="searchAccount" name="action">
        </form>
        <h1></h1>
         <table class="order">
            <tr><th> ID</th>
                <th> Email</th>
                <th> Full name</th>
                <th> status</th>
                <th> phone</th>
                <th> role</th>
                <th> action</th>

            </tr>
            <c:forEach var="acc" items="${requestScope.accountList}">
                <tr><td><c:out value="${acc.getAccID()}"></c:out></td>
                    <td><c:out value="${acc.getEmail()}"></c:out></td>
                    <td><c:out value="${acc.getFullname()}"></c:out></td>
                    <td><c:choose >
                            <c:when test="${acc.getStatus() eq 1}"> active</c:when>
                            <c:otherwise>inActive</c:otherwise>
                        </c:choose>
                    </td>
                    <td><c:out value="${acc.getPhone()}"></c:out></td>
                    <td>
                        <c:choose >
                            <c:when test="${acc.getRole() eq 1}"> admin</c:when>
                            <c:otherwise>user</c:otherwise>
                        </c:choose>
                    </td>
                    <td><c:if test="${acc.getRole() eq 0}"><!-- only block/unblock account of the user-->
                            <c:url var="mylink" value="mainController">
                               <c:param name="email" value="${acc.getEmail()}"></c:param>
                               <c:param name="status" value="${acc.getStatus()}"></c:param>
                               <c:param name="action" value="updateStatusAccount"></c:param>
                            </c:url>
                            <a href="${mylink}">Block/UnBlock</a>
                        </c:if></td>
                </tr>
            </c:forEach>
        </table>
    </section>
    </body>
</html>
```

Search an account based on its full name

The link is used to change the account's status.

Step 5: run your app to get the result



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”

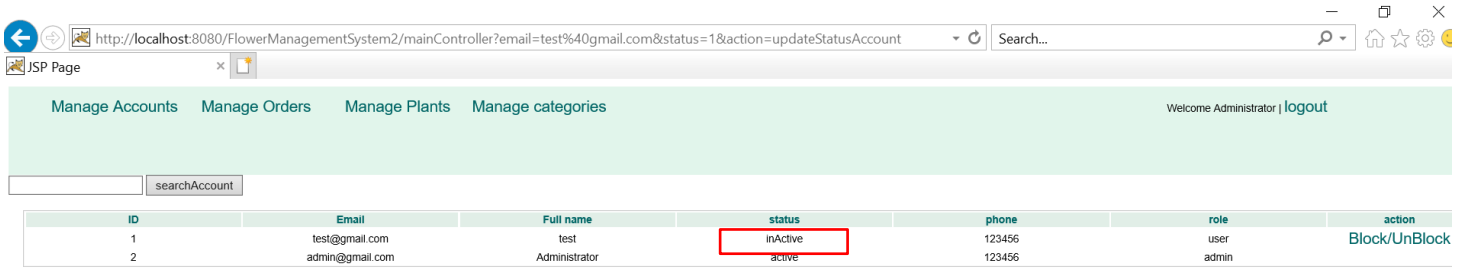
Start Page | ManageAccounts.jsp | mainController.java | updateStatusAccountServlet.java

Source | History

```
/**
 *
 * @author user
 */
public class updateStatusAccountServlet extends HttpServlet {

    /** Processes requests for both HTTP <code>GET</code> and <code>POST</code> ...9 lines */
    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("email");
            int status=Integer.parseInt(request.getParameter("status"));
            if(status==1)//active
                AccountDAO.updateAccountStatus(email, 0);
            else
                AccountDAO.updateAccountStatus(email, 1);
            request.getRequestDispatcher("mainController?action=manageAccounts").forward(request, response);
        }
    }

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

}
```

Use this code to load all accounts again

Step 3: run your app



The advanced challege:

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 ☺ ☺