This document contains Custom Validators and Struts Version 2 (common header and manager user session) and filter

Create a custom validator for email in the registration page.

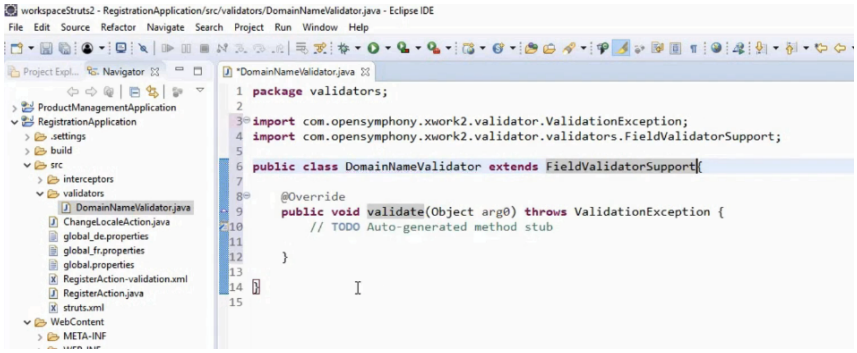
Domain name validator
- To validate the domain name of email

ValidatorSupport or FieldValidatorSupport

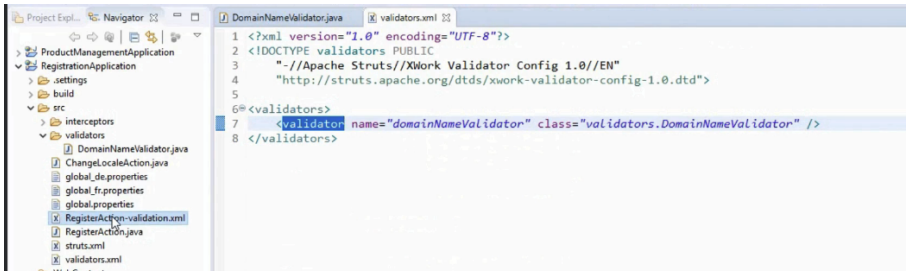
We will use our existing RegistrationApplication

1) Created a validator class
2) Declared the validator in validators.xml file
3) Added the validator in RegisterAction-validation.xml file

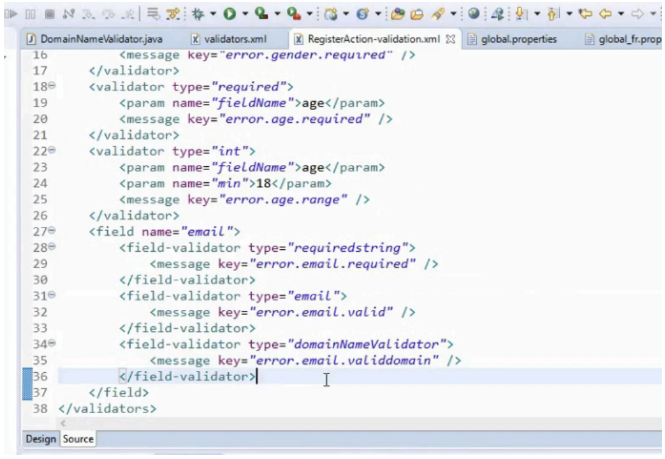
Create the above basic structure and below



Create validators.xml file in the src folder

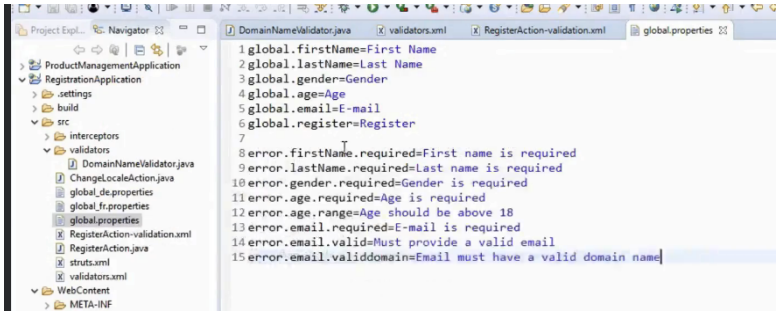


Add it in the registerAction-Validation.xml



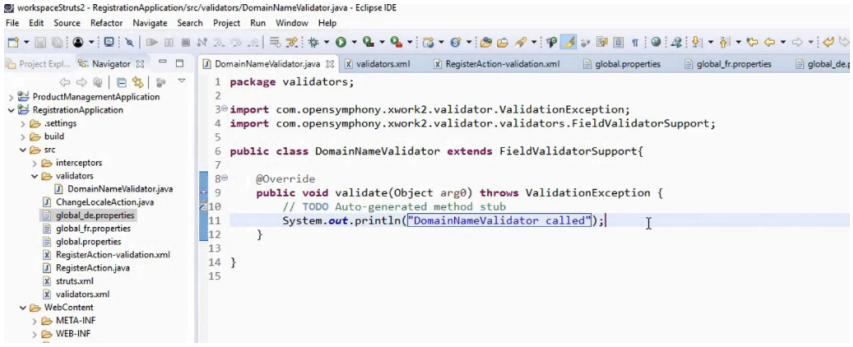
```
        <message key="error.gender.required" />
    </validator>
    <validator type="required">
        <param name="fieldName">age</param>
        <message key="error.age.required" />
    </validator>
    <validator type="int">
        <param name="fieldName">age</param>
        <param name="min">18</param>
        <message key="error.age.range" />
    </validator>
    <field name="email">
        <field-validator type="requiredstring">
            <message key="error.email.required" />
        </field-validator>
        <field-validator type="email">
            <message key="error.email.valid" />
        </field-validator>
        <field-validator type="domainNameValidator">
            <message key="error.email.validdomain" />
        </field-validator>
    </field>
</validators>
```

Add “error.email.validdomain” in all the properties file like fr , german and english

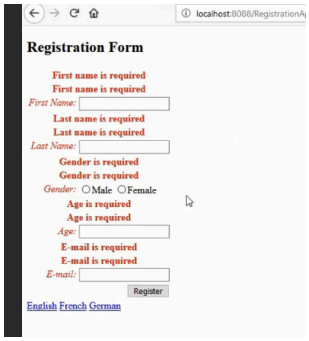


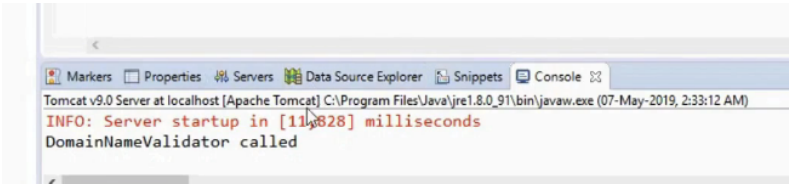
```
global.firstName=First Name
global.lastName=Last Name
global.gender=Gender
global.age=Age
global.email=E-mail
global.register=Register

error.firstName.required=First name is required
error.lastName.required=Last name is required
error.gender.required=Gender is required
error.age.required=Age is required
error.age.range=Age should be above 18
error.email.required=E-mail is required
error.email.valid=Must provide a valid email
error.email.validdomain=Email must have a valid domain name
```

Add simple sout in the java file for testing



```
package validators;

import com.opensymphony.xwork2.validator.ValidationException;
import com.opensymphony.xwork2.validators.FieldValidatorSupport;

public class DomainNameValidator extends FieldValidatorSupport{

    @Override
    public void validate(Object arg0) throws ValidationException {
        // TODO Auto-generated method stub
        System.out.println("DomainNameValidator called");
    }

}
```





```java
public class DomainNameValidator extends FieldValidatorSupport{

    @Override
    public void validate(Object arg0) throws ValidationException {
        // TODO Auto-generated method stub
        System.out.println("DomainNameValidator called");
    }

}
```

1) getFieldName - FieldValidatorSupport
2) getFieldValue(fieldName,object) - ValidatorSupport

DomainNameValidator.java | validators.xml | RegisterAction-validation.xml | global.properties | global_fr.properties | global_de.properties

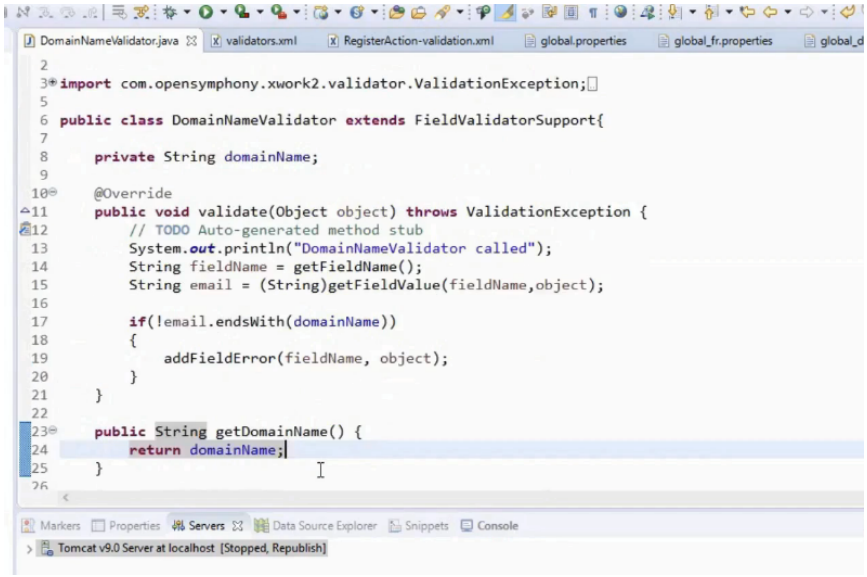
```java
package validators;

import com.opensymphony.xwork2.validator.ValidationException;

public class DomainNameValidator extends FieldValidatorSupport{

    @Override
    public void validate(Object object) throws ValidationException {
        // TODO Auto-generated method stub
        System.out.println("DomainNameValidator called");
        String validDomain = "gmail.com";
        String fieldName = getFieldName();
        String email = (String)getFieldValue(fieldName,object);
    }

}
```

fieldName :- email
object :- RegisterAction object

DomainNameValidator.java | validators.xml | RegisterAction-validation.xml | global.properties | global_fr.prope

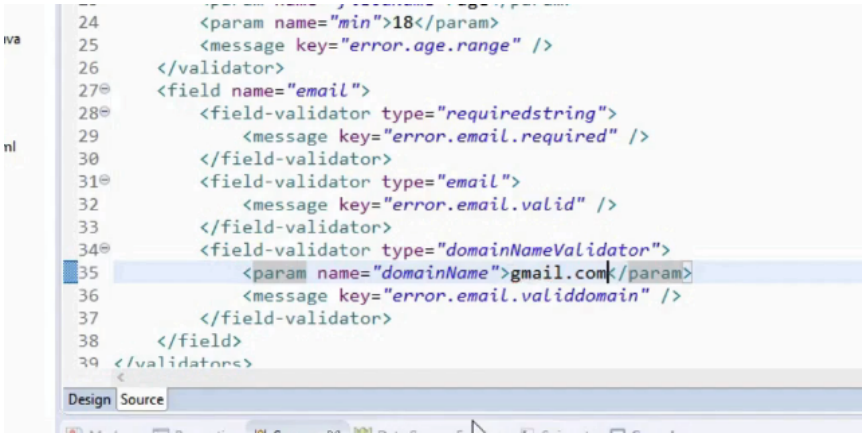
```java
package validators;

import com.opensymphony.xwork2.validator.ValidationException;

public class DomainNameValidator extends FieldValidatorSupport{

    @Override
    public void validate(Object object) throws ValidationException {
        // TODO Auto-generated method stub
        System.out.println("DomainNameValidator called");
        String validDomain = "gmail.com";
        String fieldName = getFieldName();
        String email = (String)getFieldValue(fieldName,object);

        if(!email.endsWith(validDomain))
        {
            addFieldError(fieldName, object);
        }
    }

}
```

Lets parameterize gmail.com. For that add domainName property and its setters and getters.



```
DomainNameValidator.java | validators.xml | RegisterAction-validation.xml | global.properties | global_fr.properties | global_d

2
3  import com.opensymphony.xwork2.validator.ValidationException;
5
6  public class DomainNameValidator extends FieldValidatorSupport{
7
8      private String domainName;
9
10     @Override
11     public void validate(Object object) throws ValidationException {
12         // TODO Auto-generated method stub
13         System.out.println("DomainNameValidator called");
14         String fieldName = getFieldName();
15         String email = (String)getFieldValue(fieldName,object);
16
17         if(!email.endsWith(domainName))
18         {
19             addFieldError(fieldName, object);
20         }
21     }
22
23     public String getDomainName() {
24         return domainName;
25     }
26

Markers | Properties | Servers | Data Source Explorer | Snippets | Console
Tomcat v9.0 Server at localhost [Stopped, Republish]
```

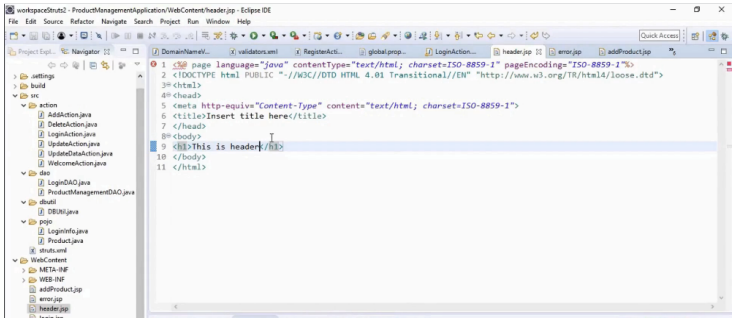
Now create same property in the registerAction-Validation.xml



```
iva
nl
24          <param name="min">18</param>
25          <message key="error.age.range" />
26      </validator>
27      <field name="email">
28          <field-validator type="requiredstring">
29              <message key="error.email.required" />
30          </field-validator>
31          <field-validator type="email">
32              <message key="error.email.valid" />
33          </field-validator>
34          <field-validator type="domainNameValidator">
35              <param name="domainName">gmail.com</param>
36              <message key="error.email.validdomain" />
37          </field-validator>
38      </field>
39  </validators>

Design | Source
```

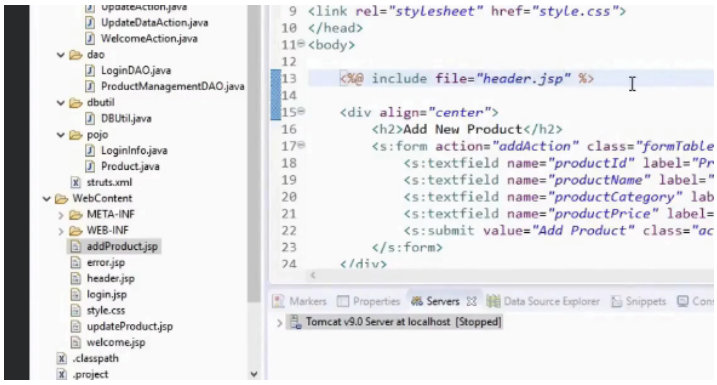
Create header panel:

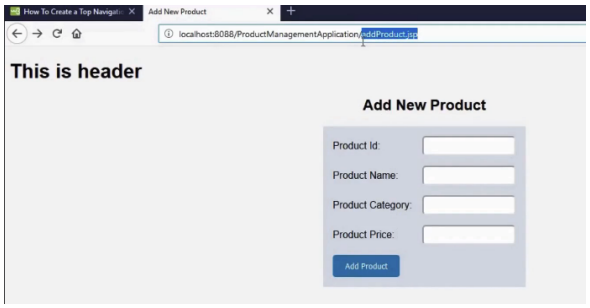
Create header.jsp

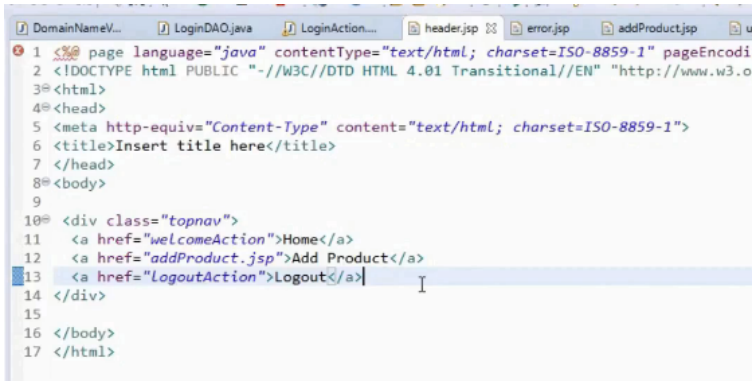


```
workspaceStruts2 - ProductManagementApplication/WebContent/header.jsp - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

1  <%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
3  <html>
4  <head>
5  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
6  <title>Insert title here</title>
7  </head>
8  <body>
9  <h1>This is header</h1>
10 </body>
11 </html>
```

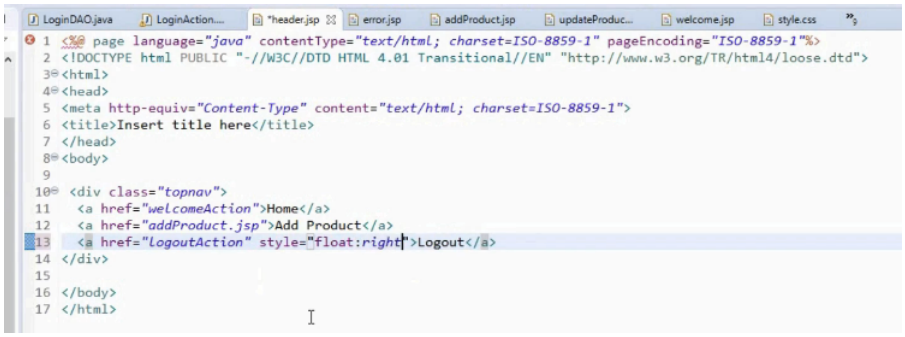
Include in every page except login.jsp.



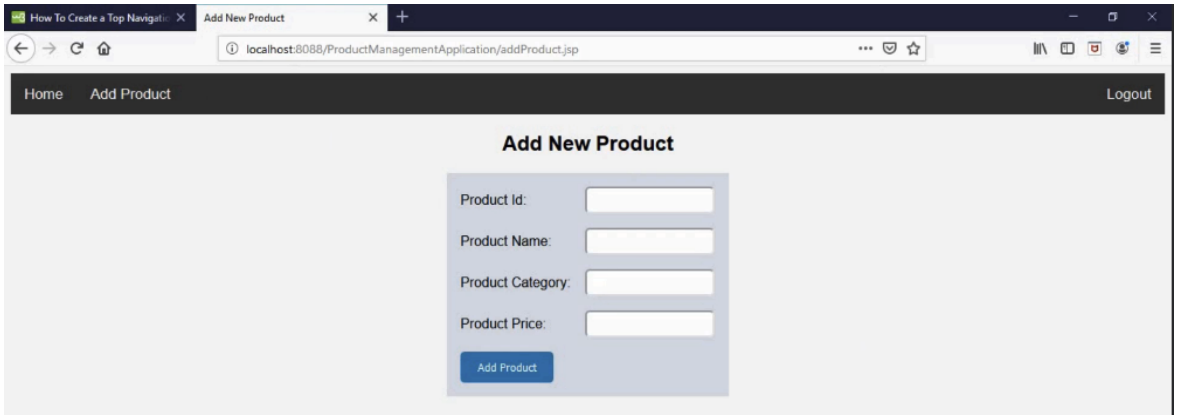




Now change the header.jsp content. In href we have to provide either Action or jsp.

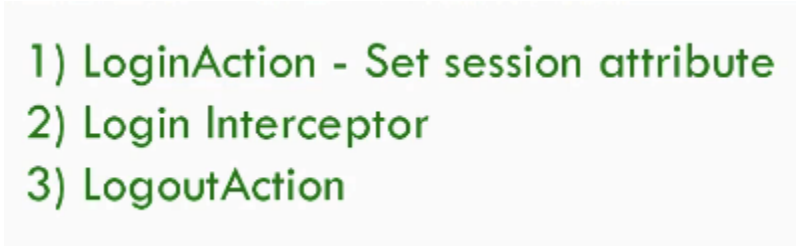


Now notice the header.



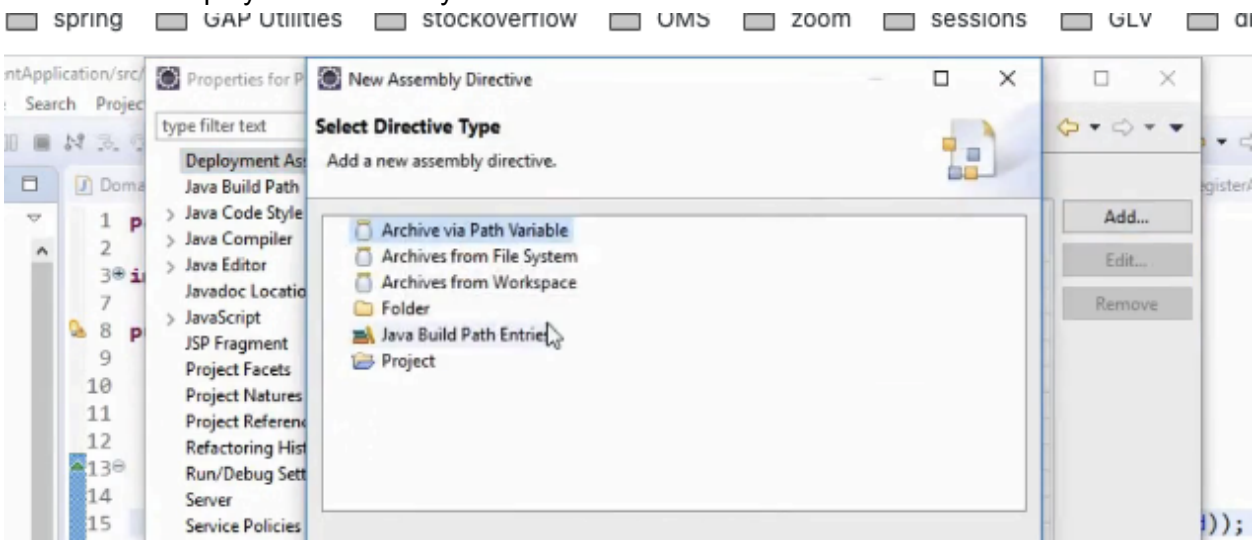
Session attributes:

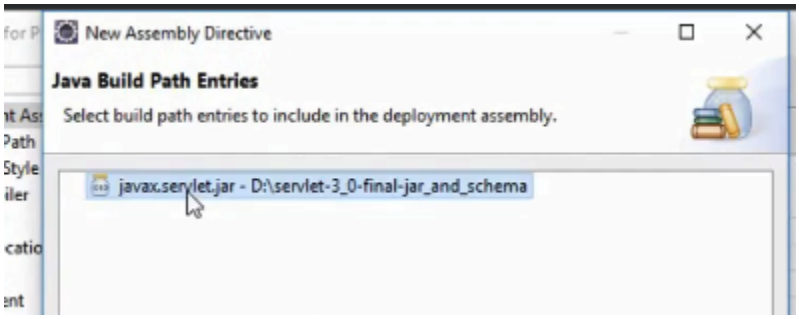
Without login we are able to access welcome, add new products and other pages which is a functional flaw.

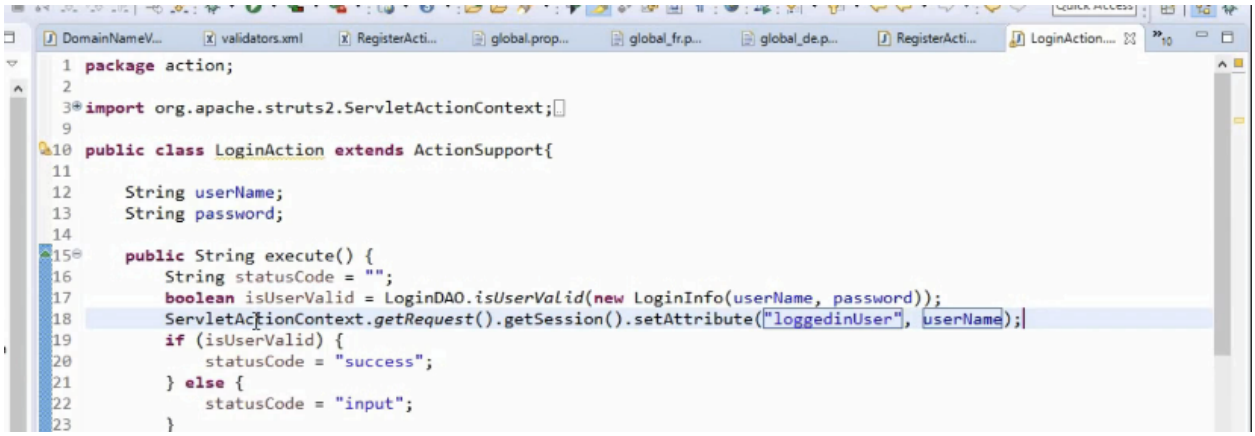


For this download servlet jar and add it build path.

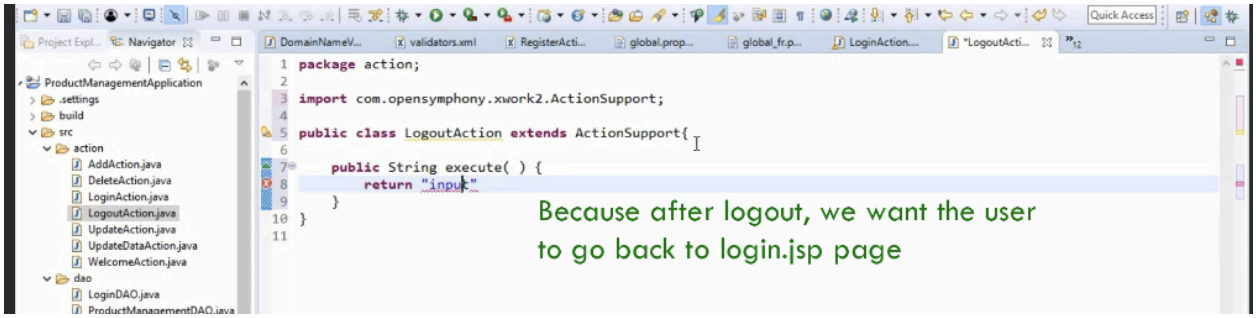
Add same in deployment assembly as well





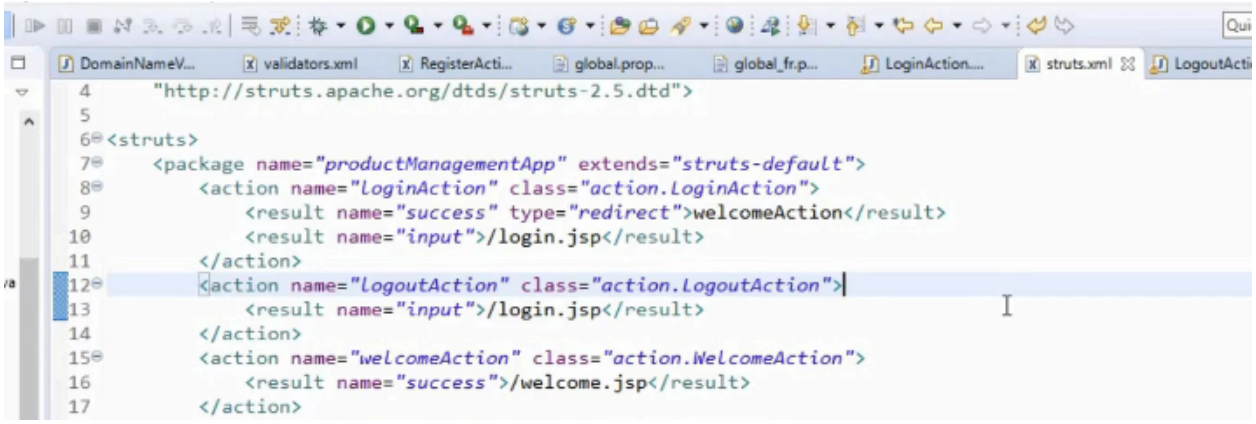


Add logout

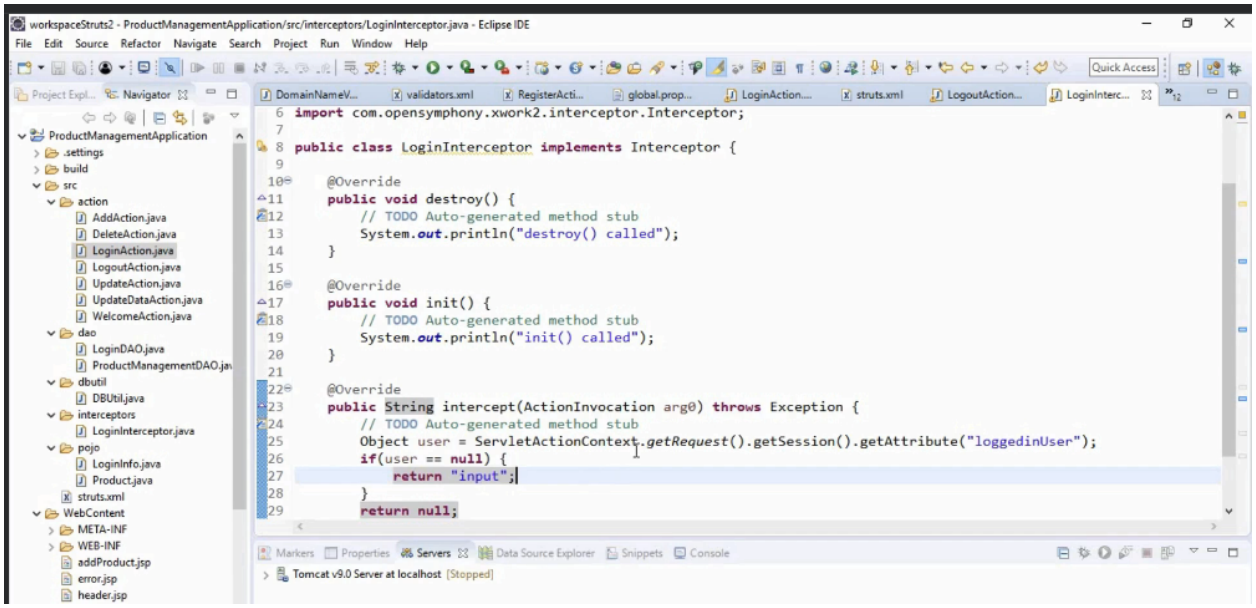


In logout we will invalidate

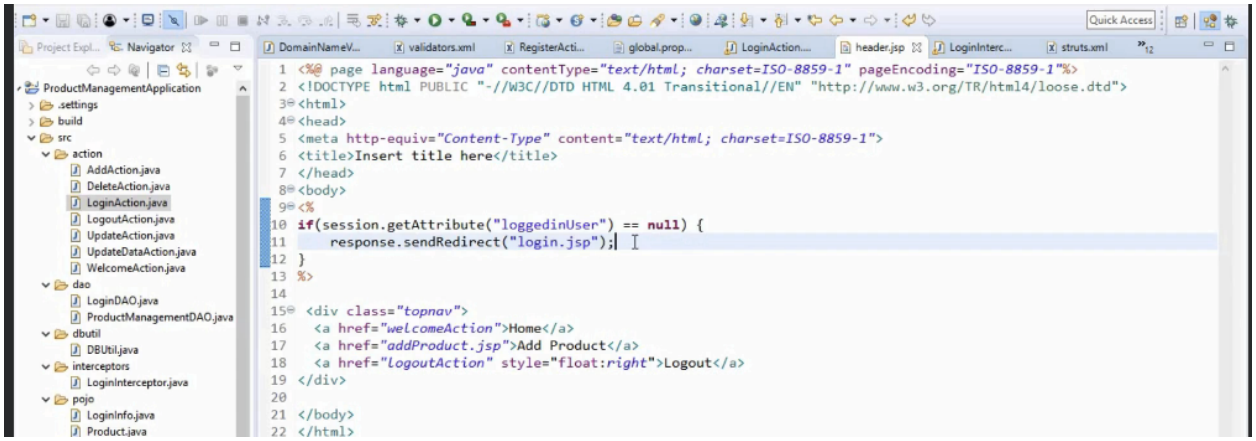
Add LogoutAction



Create a login interceptor as shown below.


workspaceStruts2 - ProductManagementApplication/src/interceptors/LoginInterceptor.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
ProductManagementApplication
.settings
build
src
action
AddAction.java
DeleteAction.java
LoginAction.java
LogoutAction.java
UpdateAction.java
UpdateDataAction.java
WelcomeAction.java
dao
LoginDAO.java
ProductManagementDAO.java
dbutil
DBUtil.java
interceptors
LoginInterceptor.java
pojo
LoginInfo.java
Product.java
struts.xml
WebContent
META-INF
WEB-INF
addProduct.jsp
error.jsp
header.jsp
import com.opensymphony.xwork2.interceptor.Interceptor;
public class LoginInterceptor implements Interceptor {
@Override
public void destroy() {
// TODO Auto-generated method stub
System.out.println("destroy() called");
}
@Override
public void init() {
// TODO Auto-generated method stub
System.out.println("init() called");
}
@Override
public String intercept(ActionInvocation arg0) throws Exception {
// TODO Auto-generated method stub
Object user = ServletActionContext.getRequest().getSession().getAttribute("loggedinUser");
if(user == null) {
return "input";
}
return null;
Markers Properties Servers Data Source Explorer Snippets Console
Tomcat v9.0 Server at localhost [Stopped]

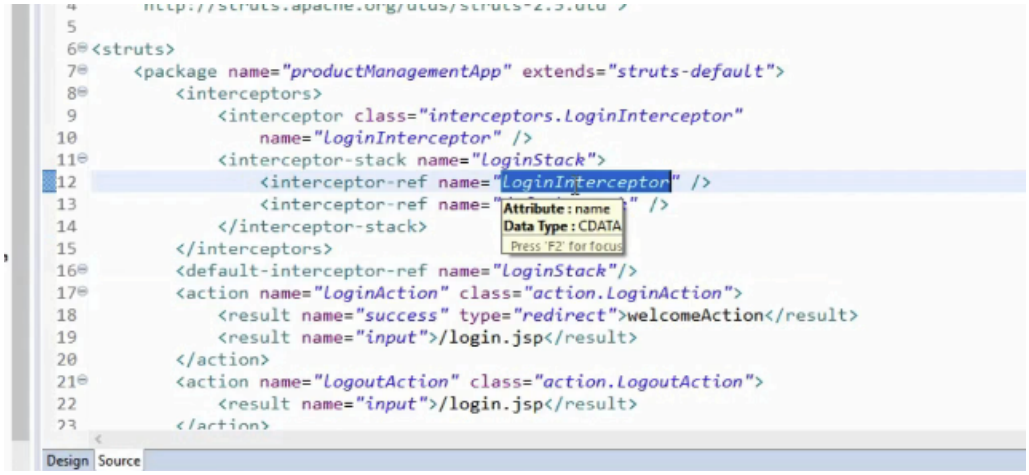

@Override
public String intercept(ActionInvocation ai) throws Exception {
// TODO Auto-generated method stub
Object user = ServletActionContext.getRequest().getSession().getAttribute("loggedinUser");
if(user == null) {
if(ai.getAction().getClass().equals(LoginAction.class)) {
return ai.invoke();
}
return "input";
}
return ai.invoke();
}

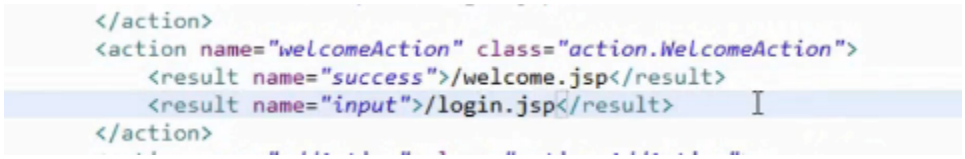

ProductManagementApplication
.settings
build
src
action
AddAction.java
DeleteAction.java
LoginAction.java
LogoutAction.java
UpdateAction.java
UpdateDataAction.java
WelcomeAction.java
dao
LoginDAO.java
ProductManagementDAO.java
dbutil
DBUtil.java
interceptors
LoginInterceptor.java
pojo
LoginInfo.java
Product.java
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
if(session.getAttribute("loggedinUser") == null) {
response.sendRedirect("login.jsp");
}
%>

Home
Add Product
Logout

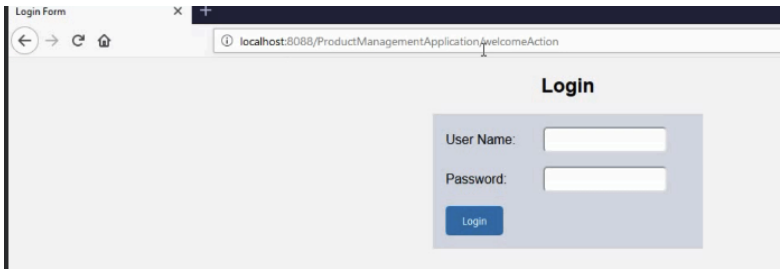
</body>
</html>

Add this interceptor inside struts.xml which will call this interceptor before every action.

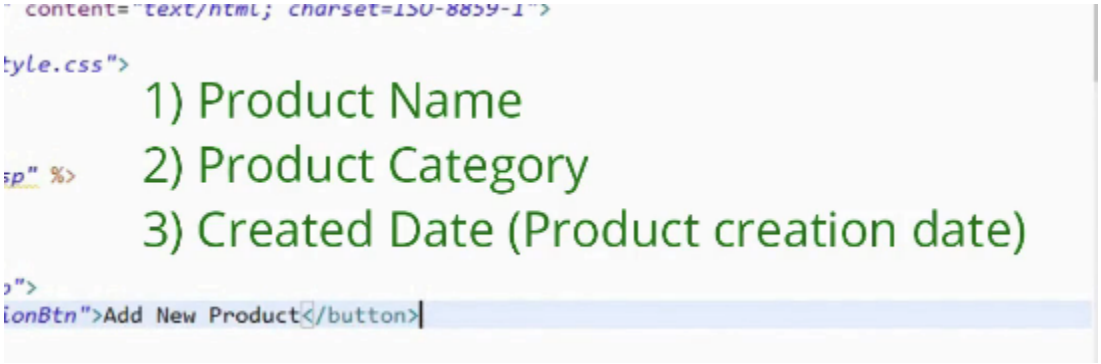




Without login welcomeAction or addproduct will take to login.jsp



Filter panel for our product list



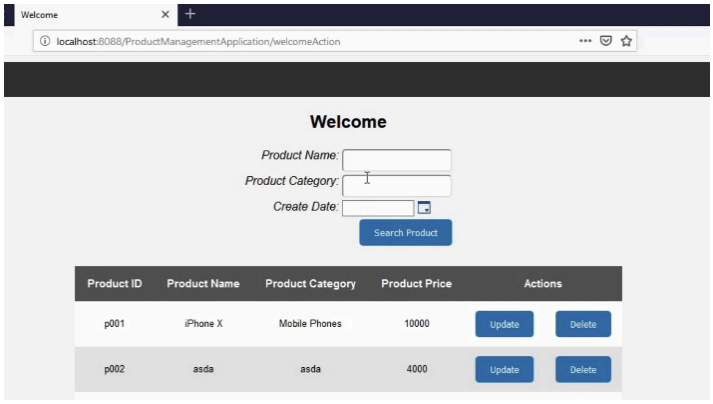
Add this jar for date picker

Add it in build path and deployment assembly

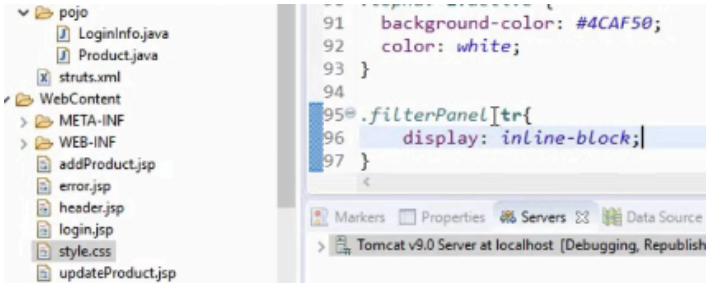
In order to have we need to have xs tags in line numbers 4, 11 and 22

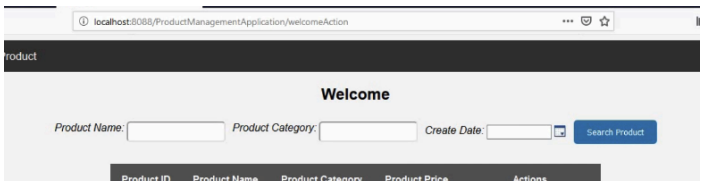
```
1  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2      pageEncoding="ISO-8859-1"%>
3  <%@ taglib prefix="s" uri="/struts-tags" %>
4  <%@ taglib prefix="sx" uri="/struts-dojo-tags" %>
5  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
6  <html>
7  <head>
8  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
9  <title>Welcome</title>
10 <link rel="stylesheet" href="style.css">
11 <sx:head />
12 </head>
13 <body>
14
15     <%@ include file="header.jsp" %>
16
17     <div align="center">
18         <h2>Welcome</h2>
19         <s:form>
20             <s:textfield name="productName" label="Product Name" class="formTextField"/>
21             <s:textfield name="productCategory" label="Product Category" class="formTextField"/>
22             <sx:datetimepicker />
23         </s:form>
24     </div>
```

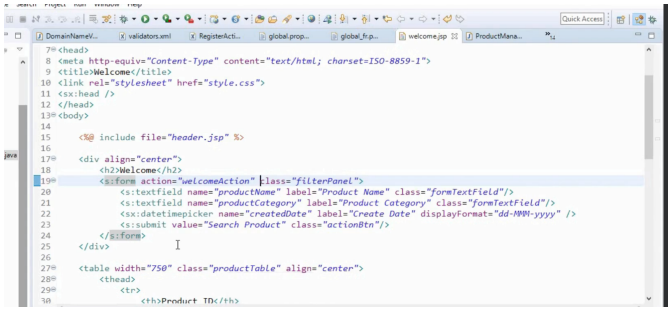
```
14
15     <%@ include file="header.jsp" %>
16
17     <div align="center">
18         <h2>Welcome</h2>
19         <s:form>
20             <s:textfield name="productName" label="Product Name" class="formTextField"/>
21             <s:textfield name="productCategory" label="Product Category" class="formTextField"/>
22             <sx:datetimepicker name="createdDate" label="Create Date" displayFormat="dd-MMM-yyyy" />
23         </s:form>
24     </div>
25
26     <table width="750" class="productTable" align="center">
27         <thead>
28             <tr>
```



Bring the search in single line add this in the style.css







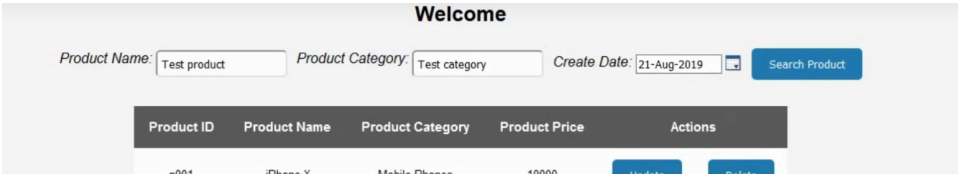
Because in the welcome page only we will display our filter data so welcome action in turn calls welcome action only.

Create setters and getters for these variables

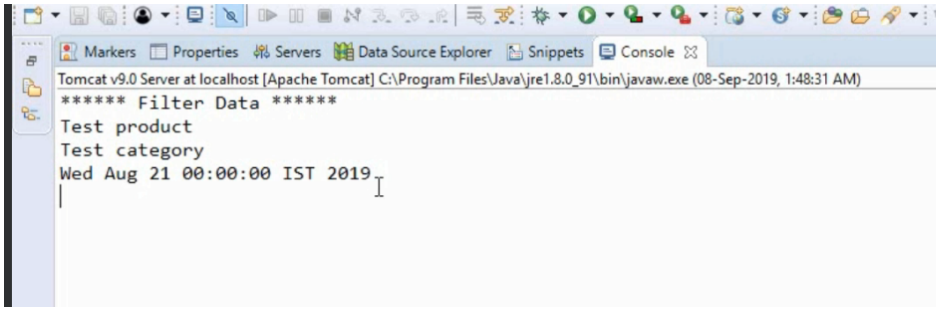
```
import java.util.Date;

public class WelcomeAction extends ActionSupport{

    private List<Product> products;
    private String productName;
    private String productCategory;
    private Date createdDate;

    public void initializeProducts() {
        System.out.println("****** Filter Data ******");
        System.out.println(productName);
        System.out.println(productCategory);
        System.out.println(createdDate);
        products = ProductManagementDAO.getAllProducts();
    }

    public String execute() {
        initializeProducts();
        return "success";
```

```
Sep 08, 2019 1:48:48 AM org.apache.catalina.startup.Catalina start
INFO: Server startup in [15,423] milliseconds
****** Filter Data ******
null
null
null
```

Initial page load



When filter page load



Truncated the data

```
Copyright (c) 1982, 2014, Oracle.  All rights reserved.

SQL> connect
Enter user-name: system
Enter password:
Connected.
SQL> set linesize 200;
SQL> select * from product;

PROD_ID                        PROD_NAME                      PROD_CATEGORY                  PROD_PRICE
------------------------------ ------------------------------ ------------------------------ ----------
p001                           iPhone X                       Mobile Phones                       10000
p002                           asda                           asda                                 4000
p003                           Jeans                          Clothing                              500
p002                           tesd                           asda                                 4000

SQL> TRUNCATE TABLE product;

Table truncated.

SQL> select * from product;

no rows selected

SQL>
```

Introduce created date column

```
SQL> select * from product;

no rows selected

SQL> ALTER TABLE product ADD created_date varchar2(2);_ -- Size should be 20 over here
```

Refactor the product.class with setters and getters toString override and constructor.

Product.java

```
1  package pojo;
2
3  public class Product {
4
5      private String productId;
6      private String productName;
7      private String productCategory;
8      private Integer productPrice;
9      private String createdDate;
10
11     public Product() {
12         // TODO Auto-generated constructor stub
13     }
14
15     public Product(String productId, String productName, Strir
```

Update DAO accordingly.

Product.java | ProductManagementDAO.java

```
30
31
32
33
34
35
36
37
38
39
40
41
42
43 ROM product WHERE prod_id = ?");
44
45
46
47
48 String("prod_name"),rs.getString("prod_category"),rs.getInt("prod_price"),rs.getString("created_date"));
49
```

Added null - because we do not want created dated to be modified when updating a product

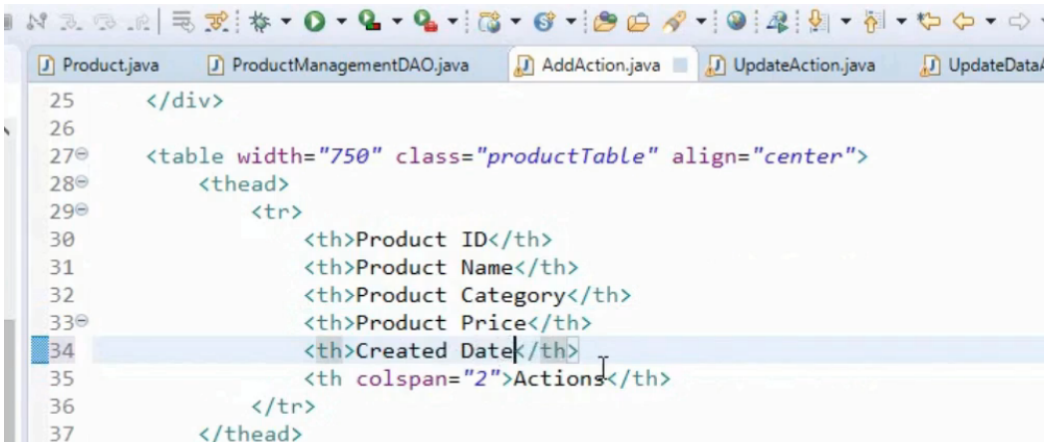
```java
package action;

import com.opensymphony.xwork2.ActionSupport;

public class UpdateAction extends ActionSupport{

    private String productId;
    private String productName;
    private String productCategory;
    private Integer productPrice;

    public String execute() {
        String statusCode = "";
        Product product = new Product(productId, productName, productCategory, productPrice, null);
        int recordUpdated = ProductManagementDAO.updateProduct(product);
        if (recordUpdated == 1) {
            statusCode = "success";
```

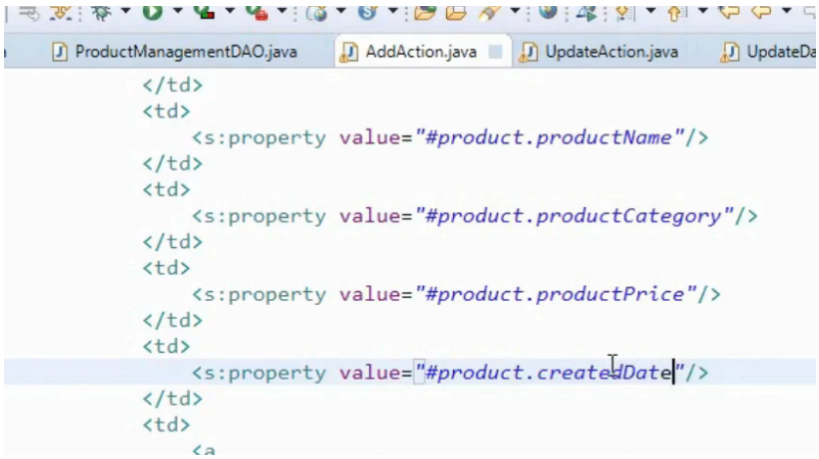
Update addAction with createDate constructor

```java
import pojo.LoginInfo;
import pojo.Product;

public class AddAction extends ActionSupport{

    private String productId;
    private String productName;
    private String productCategory;
    private Integer productPrice;

    public String execute() {
        String statusCode = "";
        SimpleDateFormat formatter = new SimpleDateFormat("dd-MMM-yyyy");
        String createdDateStr = formatter.format(new Date());
        Product product = new Product(productId, productName, productCategory, productPrice, createdDateStr);
        int recordAdded = ProductManagementDAO.addProduct(product);
        if (recordAdded == 1) {
            statusCode = "success";
```

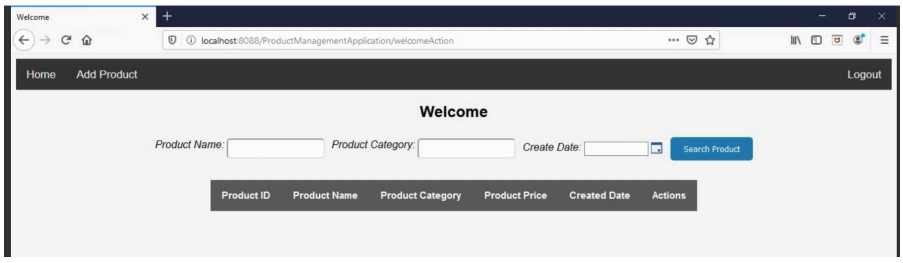
And its corresponding ProductManagementDAO

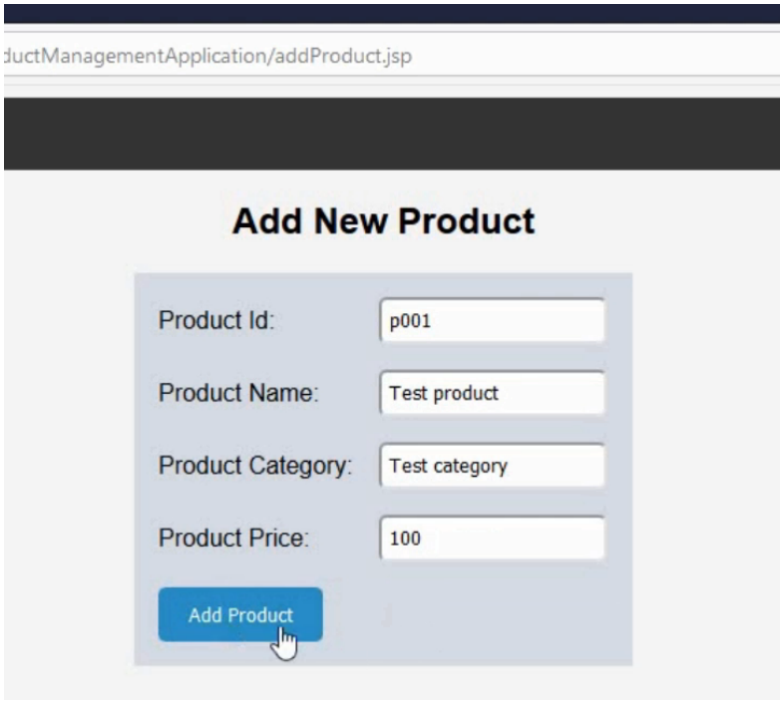
```java
    }


    public static int addProduct(Product product)
    {
        int status = 0;
        try
        {
            Connection conn = DBUtil.getConnection();
            PreparedStatement ps = conn.prepareStatement("INSERT INTO product VALUES(?,?,?,?,?)");
            ps.setString(1, product.getProductId());
            ps.setString(2, product.getProductName());
            ps.setString(3, product.getProductCategory());
            ps.setInt(4, product.getProductPrice());
            ps.setString(5, product.getCreatedDate());
            status = ps.executeUpdate();
        }
        catch(Exception e)
```

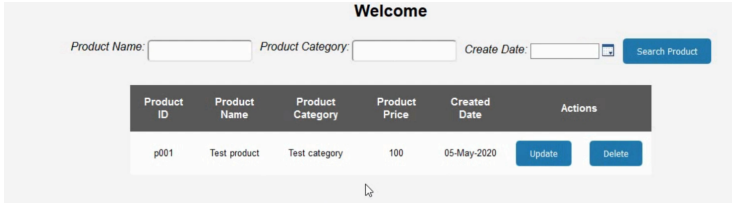


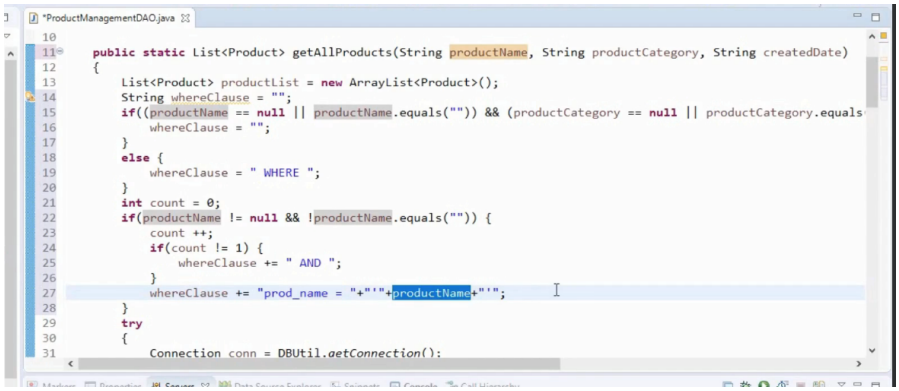


Disable createdData in update product jsp.




ductManagementApplication/addProduct.jsp
Add New Product
Product Id:
p001
Product Name:
Test product
Product Category:
Test category
Product Price:
100
Add Product


Welcome
Product Name:
Product Category:
Create Date:
Search Product
Product ID
Product Name
Product Category
Product Price
Created Date
Actions
p001
Test product
Test category
100
05-May-2020
Update
Delete


*ProductManagementDAO.java
public static List<Product> getAllProducts(String productName, String productCategory, String createdDate)
{
List<Product> productList = new ArrayList<Product>();
String whereClause = "";
if((productName == null || productName.equals("")) && (productCategory == null || productCategory.equals
whereClause = "";
}
else {
whereClause = " WHERE ";
}
int count = 0;
if(productName != null && !productName.equals("")) {
count ++;
if(count != 1) {
whereClause += " AND ";
}
whereClause += "prod_name = "+"'"+productName+"'";
}
try
{
Connection conn = DBUtil.getConnection();

```
*ProductManagementDAO.java
29        if(productCategory != null && !productCategory.equals("")) {
30            count ++;
31            if(count != 1) {
32                whereClause += " AND ";
33            }
34            whereClause += "prod_category = "+"'"+productCategory+"'";
35        }
36        if(createdDate != null && !createdDate.equals("")) {
37            count ++;
38            if(count != 1) {
39                whereClause += " AND ";
40            }
41            whereClause += "created_date = "+"'"+createdDate+"'";
42        }
43        try
44        {
45            Connection conn = DBUtil.getConnection();
46            Statement st= conn.createStatement();
47            ResultSet rs= st.executeQuery("SELECT * FROM product");
48            while(rs.next())
49            {
50                Product product = new Product(rs.getString("prod_id"),rs.getString("prod_name"),rs.getString("pr
```

```
18
19    public void initializeProducts() {
20        System.out.println("****** Filter Data ******");
21        System.out.println(productName);
22        System.out.println(productCategory);
23        System.out.println(createdDate);
24        String createdDateStr = "";
25        if(createdDate != null) {
26            SimpleDateFormat formatter = new SimpleDateFormat("dd-MMM-yyyy");
27            createdDateStr = formatter.format(createdDate);
28        }
29        products = ProductManagementDAO.getAllProducts(productName, productCategory, createdDateStr);
30    }
31
32    public String execute() {
33        initializeProducts();
```

Welcome

Product Name: jeans Product Category: Create Date: Search Product

Product ID	Product Name	Product Category	Product Price	Created Date	Actions
p003	jeans	Clothing	100	05-May-2020	Update Delete

```
****** Filter Data ******
jeans

null
SELECT * FROM product WHERE prod_name = 'jeans'
```

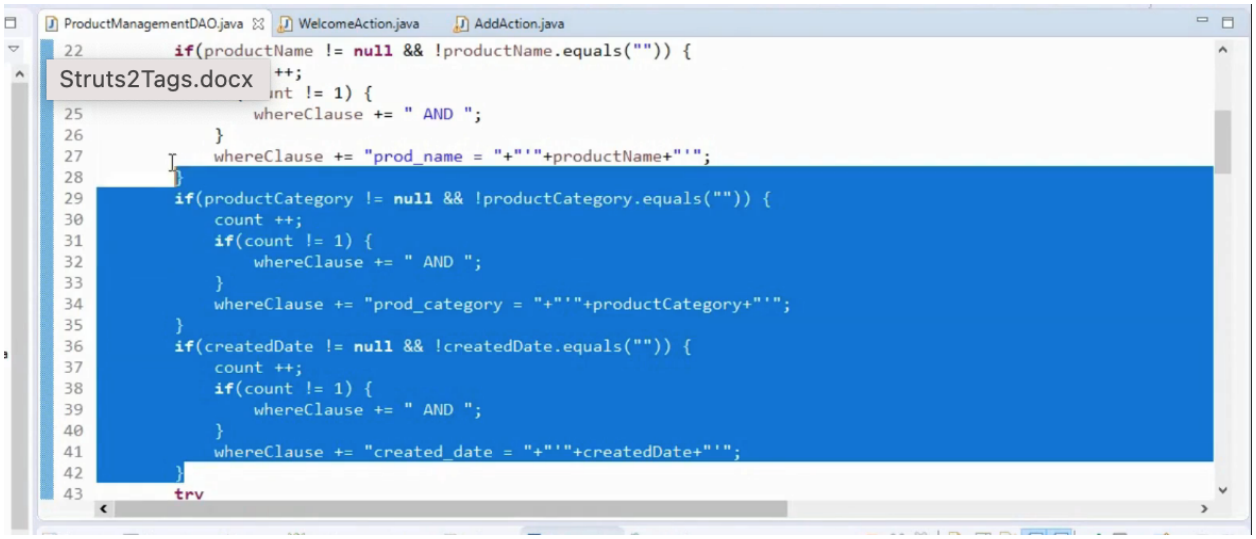
Welcome

Product Name: jeans Product Category: Clothing Create Date: Search Product

Product ID	Product Name	Product Category	Product Price	Created Date	Actions
p003	jeans	Clothing	100	05-May-2020	Update Delete

```
SELECT * FROM product WHERE prod_name = 'jeans'
****** Filter Data ******
jeans
Clothing
null
SELECT * FROM product WHERE prod_name = 'jeans' AND prod_category = 'Clothing'
```


```
jeans
Clothing
Tue May 05 00:00:00 IST 2020
SELECT * FROM product WHERE prod_name = 'jeans' AND prod_category = 'Clothing' AND created_date = '05-May-2020'
```



debug / develop and explore