

## 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

The screenshot shows the Eclipse IDE interface with the 'workspaceStruts2 - RegistrationApplication/src/validators/DomainNameValidator.java' file open. The code defines a class 'DomainNameValidator' that extends 'FieldValidatorSupport'. It includes imports for ValidationException and FieldValidatorSupport, and an empty validate method.

```
1 package validators;
2
3 import com.opensymphony.xwork2.validator.ValidationException;
4 import com.opensymphony.xwork2.validator.validators.FieldValidatorSupport;
5
6 public class DomainNameValidator extends FieldValidatorSupport{
7
8     @Override
9     public void validate(Object arg0) throws ValidationException {
10         // TODO Auto-generated method stub
11     }
12 }
```

Create validators.xml file in the src folder

The screenshot shows the Eclipse IDE interface with the 'validators.xml' file open in the 'src' folder. The XML configuration defines a validation rule for the 'domainNameValidator' class.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE validators PUBLIC
"-//Apache Struts//XWork Validator Config 1.0//EN"
"http://struts.apache.org/dtds/xwork-validator-config-1.0.dtd">
<validators>
    <validation name="domainNameValidator" class="validators.DomainNameValidator" />
</validators>
```

## Add it in the registerAction-Validation.xml

```

<validators>
    <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>
    <validator type="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.validator.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");
    }
}

```

Registration Form

First name is required  
First name is required  
First Name:

Last name is required  
Last name is required  
Last Name:

Gender is required  
Gender is required  
Gender:  Male  Female

Age is required  
Age is required  
Age:

E-mail is required  
E-mail is required  
E-mail:

[Register](#)

Tomcat v9.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre1.8.0\_91\bin\javaw.exe (07-May-2019, 2:33:12 AM)  
INFO: Server startup in [11,828] milliseconds  
DomainNameValidator called

```

6  public class DomainNameValidator extends FieldValidatorSupport{
7
8    @Override
9    public void validate(Object arg0) throws ValidationException {
10      // TODO Auto-generated method stub
11      System.out.println("DomainNameValidator called");
12    }
13
14 }
15

```

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

```

1 package validators;
2
3 import com.opensymphony.xwork2.validator.ValidationException;
4
5 public class DomainNameValidator extends FieldValidatorSupport{
6
7   @Override
8   public void validate(Object object) throws ValidationException {
9     // TODO Auto-generated method stub
10    System.out.println("DomainNameValidator called");
11    String validDomain = "gmail.com";
12    String fieldName = getFieldName();
13    String email = (String)getFieldValue(fieldName,object);
14  }
15
16 }
17

```

fieldName :- email  
object :- RegisterAction object

```

1 package validators;
2
3 import com.opensymphony.xwork2.validator.ValidationException;
4
5 public class DomainNameValidator extends FieldValidatorSupport{
6
7   @Override
8   public void validate(Object object) throws ValidationException {
9     // TODO Auto-generated method stub
10    System.out.println("DomainNameValidator called");
11    String validDomain = "gmail.com";
12    String fieldName = getFieldName();
13    String email = (String)getFieldValue(fieldName,object);
14
15    if(!email.endsWith(validDomain))
16    {
17      addFieldError(fieldName, object);
18    }
19  }
20
21 }
22

```

**Age is required**

**Age is required**

Age:

**Email must have a valid domain name**

E-mail:  abc@fsafds.com

[English](#) [French](#) [German](#)

**Age is required**

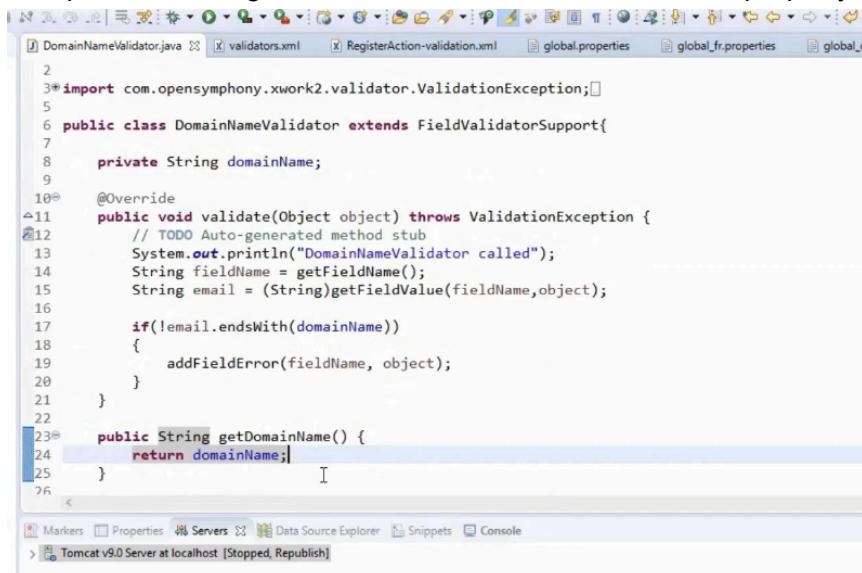
**Age is required**

Age:

E-mail:  abc@gmail.com

[English](#) [French](#) [German](#)

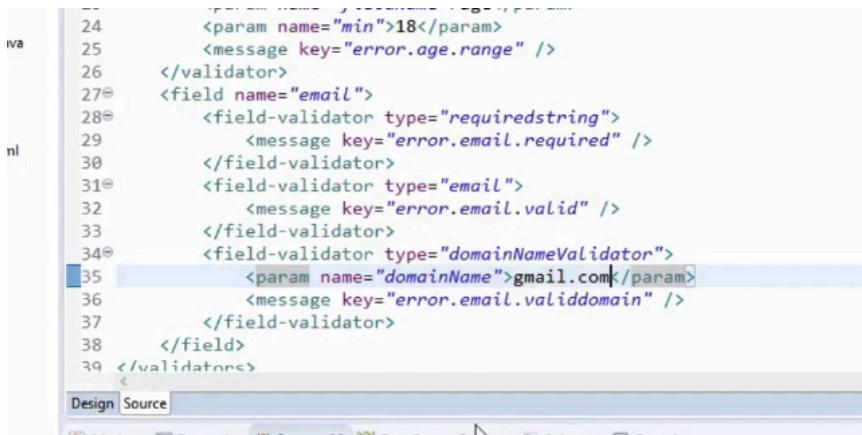
Lets parameterize gmail.com. For that add domainName property and its setters and getters.



```
1 package com.opensymphony.xwork2.validator.validators;
2
3 import com.opensymphony.xwork2.validator.ValidationException;
4
5 public class DomainNameValidator extends FieldValidatorSupport{
6
7     private String domainName;
8
9     @Override
10    public void validate(Object object) throws ValidationException {
11        // TODO Auto-generated method stub
12        System.out.println("DomainNameValidator called");
13        String fieldName = getFieldName();
14        String email = (String)getFieldValue(fieldName,object);
15
16        if(!email.endsWith(domainName))
17        {
18            addFieldError(fieldName, object);
19        }
20    }
21
22
23    public String getDomainName() {
24        return domainName;
25    }
26}
```

The screenshot shows the Eclipse IDE interface with the DomainNameValidator.java file open in the editor. The code defines a class extending FieldValidatorSupport with a private domainName field and an overridden validate method. It also includes a getter for domainName. The code is annotated with Javadoc-style comments and imports.

Now create same property in the registerAction-Validation.xml

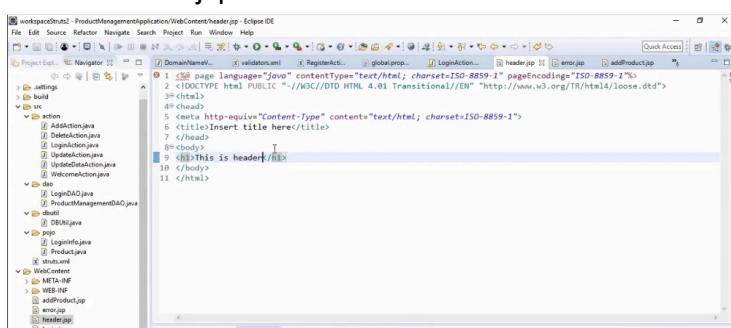


```
1 <validators>
2     <param name="min">18</param>
3     <message key="error.age.range" />
4 
```

The screenshot shows the Eclipse IDE interface with the registerAction-Validation.xml file open in the editor. The XML configuration includes validation rules for age and email fields, and a specific rule for the email field using the domainNameValidator defined in the Java code above. The domainNameValidator is configured with a parameter named "domainName" set to "gmail.com".

Create header panel:

Create header.jsp



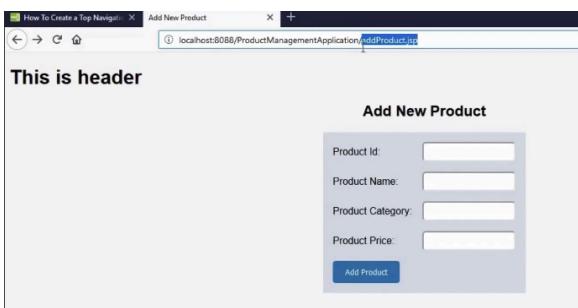
```
1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
3 <html>
4     <head>
5         <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
6         <title>Insert title here</title>
7     </head>
8     <body>
9         <h2>This is header!</h2>
10    </body>
11 </html>
```

The screenshot shows the Eclipse IDE interface with the header.jsp file open in the editor. The JSP page contains a simple HTML structure with a single heading element.

Include in every page except login.jsp.

The screenshot shows the Eclipse IDE interface. On the left is the Project Explorer view with files like UpdateAction.java, UpdateDataAction.java, WelcomeAction.java, LoginDAO.java, ProductManagementDAO.java, DBUtil.java, LoginInfo.java, Product.java, struts.xml, META-INF, WEB-INF, and several JSP files (addProduct.jsp, error.jsp, header.jsp, login.jsp, style.css, updateProduct.jsp, welcome.jsp). The code editor on the right contains JSP code with a line of code highlighted: <%@ include file="header.jsp" %>. Below the code editor is the Navigator view showing Tomcat v9.0 Server at localhost [Stopped].

```
9 <link rel="stylesheet" href="style.css">
10 </head>
11 <body>
12
13 <%@ include file="header.jsp" %>
14
15 <div align="center">
16     <h2>Add New Product</h2>
17     <:form action="addAction" class="formTable">
18         <s:textfield name="productId" label="Pr
19         <s:textfield name="productName" label="
20         <s:textfield name="productCategory" lab
21         <s:textfield name="productPrice" label=
22         <s:submit value="Add Product" class="ac
23     </:form>
24 </div>
```



The screenshot shows the Eclipse IDE interface with the code editor open to header.jsp. The code defines a top navigation bar with links to Home, Add Product, and Logout. The Logout link is highlighted.

```
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
10 <div class="topnav">
11     <a href="welcomeAction">Home</a>
12     <a href="addProduct.jsp">Add Product</a>
13     <a href="LogoutAction">Logout</a>
14 </div>
15
16 </body>
17 </html>
```

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

The screenshot shows the Eclipse IDE interface with the code editor open to header.jsp. The Logout link has been modified to include a style attribute: <a href="LogoutAction" style="float:right">Logout</a>. The modified line is highlighted.

```
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
10 <div class="topnav">
11     <a href="welcomeAction">Home</a>
12     <a href="addProduct.jsp">Add Product</a>
13     <a href="LogoutAction" style="float:right">Logout</a>
14 </div>
15
16 </body>
17 </html>
```

Now notice the header.

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

- Page Title: How To Create a Top Navigation Bar
- URL: localhost:8088/ProductManagementApplication/addProduct.jsp
- Content Area:
  - Form Title: Add New Product
  - Fields:
    - Product Id: [Input Field]
    - Product Name: [Input Field]
    - Product Category: [Input Field]
    - Product Price: [Input Field]
  - Buttons:
    - Add Product
- Top Right: Logout link

Session attributes:

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

- 1) LoginAction - Set session attribute
- 2) Login Interceptor
- 3) LogoutAction

For this download servlet jar and add it build path.

The screenshot shows the Maven Central Repository page for the Java Servlet API 3.0.1:

- Page Title: Java Servlet API » 3.0.1
- File Type: jar (83 KB)
- Repositories: Central, Java.net, Redhat GA
- Used By: 11,040 artifacts
- Dependencies (Listed):
  - log4j-api-2.11.1.jar - UNSTRUTS-2.0.20-MIN-110\struts-2.0.24
  - ognl-3.1.21.jar - D:\struts-2.5.20-min-lib\struts-2.5.20\lib
  - struts2-core-2.5.20.jar - D:\struts-2.5.20-min-lib\struts-2.5.20\lib
  - jaxws-api-2.3.1.jar - D:\struts-2.5.20-min-lib\struts-2.5.20\lib
  - JRE System Library [jre1.8.0\_91]

Add same in deployment assembly as well

The screenshot shows the Eclipse IDE interface with several open windows:

- New Assembly Directive**: A dialog box titled "Select Directive Type" with the sub-instruction "Add a new assembly directive." It lists several options:
  - Archive via Path Variable
  - Archives from File System
  - Archives from Workspace
  - Folder
  - Java Build Path Entry** (selected)
  - Project
- Java Build Path Entries**: A dialog box showing a single entry: "javax.servlet.jar - D:\servlet-3\_0-final-jar\_and\_schema".
- Code Editor**: Displays Java code for a LoginAction class:

```
1 package action;
2
3* import org.apache.struts2.ServletActionContext;*
4
5 public class LoginAction extends ActionSupport{
6
7     String userName;
8     String password;
9
10    public String execute() {
11        String statusCode = "";
12        boolean isValid = LoginDAO.isUserValid(new LoginInfo(userName, password));
13        ServletActionContext.getRequest().getSession().setAttribute("loggedinUser", userName);
14        if (isValid) {
15            statusCode = "success";
16        } else {
17            statusCode = "input";
18        }
19    }
20}
```

Add logout

```

1 package action;
2
3 import com.opensymphony.xwork2.ActionSupport;
4
5 public class LogoutAction extends ActionSupport{
6
7     public String execute( ) {
8         return "input";
9     }
10}
11

```

Because after logout, we want the user to go back to login.jsp page

In logout we will invalidate

```

1 package action;
2
3 import org.apache.struts2.ServletActionContext;
4
5 import com.opensymphony.xwork2.ActionSupport;
6
7 public class LogoutAction extends ActionSupport{
8
9     public String execute( ) {
10        ServletActionContext.getRequest().getSession().invalidate();
11        return "input";
12    }
13}
14

```

Add LogoutAction

```

4     "http://struts.apache.org/dtds/struts-2.5.dtd"
5
6<struts>
7    <package name="productManagementApp" extends="struts-default">
8        <action name="LoginAction" class="action.LoginAction">
9            <result name="success" type="redirect">welcomeAction</result>
10           <result name="input">/login.jsp</result>
11        </action>
12        <action name="LogoutAction" class="action.LogoutAction">
13            <result name="input">/login.jsp</result>
14        </action>
15        <action name="welcomeAction" class="action.WelcomeAction">
16            <result name="success">/welcome.jsp</result>
17        </action>

```

Create a login interceptor as shown below.

Eclipse IDE - workspaceStruts2 - ProductManagementApplication/src/interceptors/LoginInterceptor.java

```
6 import com.opensymphony.xwork2.interceptor.Interceptor;
7
8 public class LoginInterceptor implements Interceptor {
9
10    @Override
11    public void destroy() {
12        // TODO Auto-generated method stub
13        System.out.println("destroy() called");
14    }
15
16    @Override
17    public void init() {
18        // TODO Auto-generated method stub
19        System.out.println("init() called");
20    }
21
22    @Override
23    public String intercept(ActionInvocation arg0) throws Exception {
24        // TODO Auto-generated method stub
25        Object user = ServletActionContext.getRequest().getSession().getAttribute("loggedinUser");
26        if(user == null) {
27            return "input";
28        }
29        return null;
30    }
31
32
33
34
35
36}
```

l.jsp

```
23
24    @Override
25    public String intercept(ActionInvocation ai) throws Exception {
26        // TODO Auto-generated method stub
27        Object user = ServletActionContext.getRequest().getSession().getAttribute("loggedinUser");
28        if(user == null) {
29            if(ai.getAction().getClass().equals(LoginAction.class)) {
30                return ai.invoke();
31            }
32            return "input";
33        }
34        return ai.invoke();
35
36}
```

Eclipse IDE - workspaceStruts2 - ProductManagementApplication/src/header.jsp

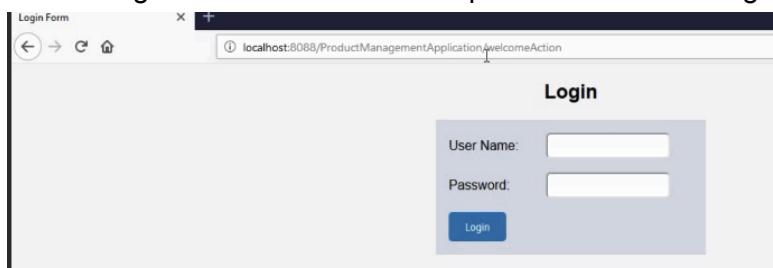
```
1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
3<html>
4<head>
5 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
6 <title>Insert title here</title>
7 </head>
8<body>
9<%
10 if(session.getAttribute("loggedinUser") == null) {
11     response.sendRedirect("login.jsp");
12 }
13 %>
14
15<div class="topnav">
16    <a href="welcomeAction">Home</a>
17    <a href="addProduct.jsp">Add Product</a>
18    <a href="logoutAction" style="float:right">Logout</a>
19 </div>
20
21 </body>
22 </html>
```

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

```
http://struts.apache.org/docs/struts-2.3.0.0.html

5
6<struts>
7  <package name="productManagementApp" extends="struts-default">
8    <interceptors>
9      <interceptor class="interceptors.LoginInterceptor"
10        name="LoginInterceptor" />
11      <interceptor-stack name="LoginStack">
12        <interceptor-ref name="LoginInterceptor" />
13        <interceptor-ref name="Attribute: name" />
14        <interceptor-ref name="Data Type: CDATA" />
15      </interceptor-stack>
16    </interceptors>
17    <default-interceptor-ref name="LoginStack"/>
18    <action name="loginAction" class="action.LoginAction">
19      <result name="success" type="redirect">welcomeAction</result>
20      <result name="input"/>/login.jsp</result>
21    </action>
22    <action name="logoutAction" class="action.LogoutAction">
23      <result name="input"/>/login.jsp</result>
24    </action>
25
26  </action>
27  <action name="welcomeAction" class="action.WelcomeAction">
28    <result name="success">/welcome.jsp</result>
29    <result name="input"/>/login.jsp</result>
30  </action>
```

Without login welcomeAction or addproduct will take to login.jsp



## Filter panel for our product list

1) Product Name  
2) Product Category  
3) Created Date (Product creation date)

Add New Product

Add this jar for date picker

The screenshot shows the Artifactory interface for the Struts 2 Dojo Plugin. At the top, there's a navigation bar with links like 'Home', 'Search', 'Artifacts', and 'Metrics'. Below it, a search bar contains the query 'Struts 2 Dojo Plugin'. The main content area displays the plugin's details: 'Struts 2 Dojo Plugin' by 'STRUTS', 'Apache 2.0' license, 'Tags: plugin, web-framework, apache', and 'Used By: 4 artifacts'. A table below lists 'Central (46)' artifacts, with one entry for '2.3.37' from 'Central' repository, dated 'Jan, 2019'.

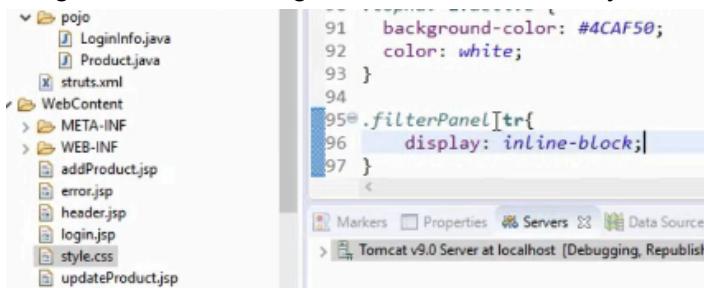
Add it in build path and deployment assembly

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

A screenshot of a Java code editor displaying a JSP file named 'welcome.jsp'. The code includes Struts tags such as <%@ page %>, <s:form>, <s:textfield>, and <sx:datetimepicker>. Line 11 contains <sx:head />, and line 22 contains <sx:datetimepicker name="createdDate" label="Create Date" displayFormat="dd-MMM-yyyy" />. The code is annotated with line numbers from 1 to 24.

A screenshot of a Java code editor displaying a JSP file named 'welcome.jsp'. The code includes Struts tags such as <%@ page %>, <s:form>, <s:textfield>, and <sx:datetimepicker>. Line 22 contains <sx:datetimepicker name="createdDate" label="Create Date" displayFormat="dd-MMM-yyyy" />. The code is annotated with line numbers from 14 to 28.

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



```

<%@ page contentType="text/html; charset=ISO-8859-1" %>
<%@ include file="header.jsp" %>
<html align="center">
<head>
<title>WelcomeWelcome

```

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

```
 10  
11 public class WelcomeAction extends ActionSupport{  
12  
13     private List<Product> products;  
14     private String productName;  
15     private String productCategory;  
16     private Date createdDate;  
17  
18     public void initializeProducts() {  
19         System.out.println("***** Filter Data *****");  
20         System.out.println(productName);  
21         System.out.println(productCategory);  
22         System.out.println(createdDate);  
23         products = ProductManagementDAO.getAllProducts();  
24     }  
25  
26     public String execute() {  
27         initializeProducts();  
28         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

The screenshot shows a web application titled "Welcome". At the top, there are three input fields: "Product Name" (Test product), "Product Category" (Test category), and "Create Date" (21-Aug-2019). To the right of these fields is a blue "Search Product" button. Below the search bar is a table header with columns: Product ID, Product Name, Product Category, Product Price, and Actions. The table body contains one row of data.

When filter page load

The screenshot shows the Eclipse IDE's Console tab. The title bar indicates "Tomcat v9.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre1.8.0\_91\bin\javaw.exe (08-Sep-2019, 1:48:31 AM)". The console output window displays the following truncated text:  
\*\*\*\*\* Filter Data \*\*\*\*\*  
Test product  
Test category  
Wed Aug 21 00:00:00 IST 2019 |

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            test              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     ROM product WHERE prod_id = ?);
38
39
40
41
42
43     String("prod_name"),rs.getString("prod_category"),rs.getInt("prod_price"),rs.getString("created_date"));
44
45
46
47
48
49

```

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

```

1 package action;
2
3@import com.opensymphony.xwork2.ActionSupport; []
4
5 public class UpdateAction extends ActionSupport{
6
7     private String productId;
8     private String productName;
9     private String productCategory;
10    private Integer productPrice;
11
12    public String execute() {
13        String statusCode = "";
14        Product product = new Product(productId, productName, productCategory, productPrice, null);
15        int recordUpdated = ProductManagementDAO.updateProduct(product);
16        if (recordUpdated == 1) {
17            statusCode = "success";
18        }
19    }

```

## Update addAction with createDate constructor

```

10 import pojo.LoginInfo;
11 import pojo.Product;
12
13 public class AddAction extends ActionSupport{
14
15     private String productId;
16     private String productName;
17     private String productCategory;
18     private Integer productPrice;
19
20    public String execute() {
21        String statusCode = "";
22        SimpleDateFormat formatter = new SimpleDateFormat("dd-MMM-yyyy");
23        String createdDateStr = formatter.format(new Date());
24        Product product = new Product(productId, productName, productCategory, productPrice, createdDateStr); []
25        int recordAdded = ProductManagementDAO.addProduct(product);
26        if (recordAdded == 1) {
27            statusCode = "success";

```

## And its corresponding ProductManagementDAO

```

57 }
58
59
60    public static int addProduct(Product product)
61    {
62        int status = 0;
63        try
64        {
65            Connection conn = DBUtil.getConnection();
66            PreparedStatement ps = conn.prepareStatement("INSERT INTO product VALUES(?,?,?,?,?)");
67            ps.setString(1, product.getProductId());
68            ps.setString(2, product.getProductName());
69            ps.setString(3, product.getProductCategory());
70            ps.setInt(4, product.getProductPrice()); []
71            ps.setString(5, product.getCreatedDate());
72            status = ps.executeUpdate();
73        }
74        catch(Exception e)

```

```

25     </div>
26
27     <table width="750" class="productTable" align="center">
28         <thead>
29             <tr>
30                 <th>Product ID</th>
31                 <th>Product Name</th>
32                 <th>Product Category</th>
33                 <th>Product Price</th>
34                 <th>Created Date</th>
35                 <th colspan="2">Actions</th>
36             </tr>
37         </thead>

```

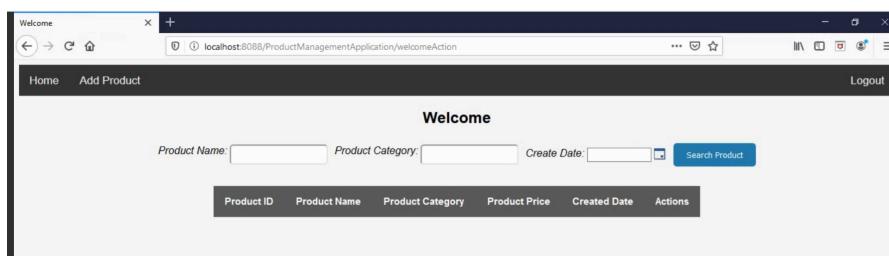
</td>	</td>	<s:property value="#product.productName"/>	</td>	<s:property value="#product.productCategory"/>	</td>	<s:property value="#product.productPrice"/>	</td>	<s:property value="#product.createdDate"/>	</td>	</td>
-------	-------	--	-------	--	-------	---	-------	--	-------	-------

```

7 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
8 <title>Update Product</title>
9 <link rel="stylesheet" href="style.css">
10 </head>
11 <body>
12
13     <%@ include file="header.jsp" %>
14
15     <div align="center">
16         <h2>Update Product</h2>
17         <:form action="updateAction" class="formTable">
18             <:textfield name="productId" label="Product Id" class="formTextField" readonly="true"/>
19             <:textfield name="productName" label="Product Name" class="formTextField"/>
20             <:textfield name="productCategory" label="Product Category" class="formTextField"/>
21             <:textfield name="productPrice" label="Product Price" class="formTextField"/>
22             <:textfield name="createdDate" label="Created Date" class="formTextField" readonly="true"/>
23             <:submit value="Update Product" class="actionBtn"/>
24         </:form>
25     </div>
26
27 </body>
28 </html>

```

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			
p001	Test product	Test category	100	05-May-2020	<b>Update</b>	<b>Delete</b>

```
*ProductManagementDAO.java */
10
11  public static List<Product> getAllProducts(String productName, String productCategory, String createdDate)
12  {
13      List<Product> productList = new ArrayList<Product>();
14      String whereClause = "";
15      if((productName == null) || productName.equals("")) && (productCategory == null) || productCategory.equals("")
16          whereClause = "";
17      }
18      else {
19          whereClause = " WHERE ";
20      }
21      int count = 0;
22      if(productName != null && !productName.equals("")){
23          count++;
24          if(count != 1) {
25              whereClause += " AND ";
26          }
27          whereClause += "prod_name = "+productName+"";
28      }
29      try {
30      {
31          Connection conn = DBUtil.getConnection();
32      }
33  }
```

```

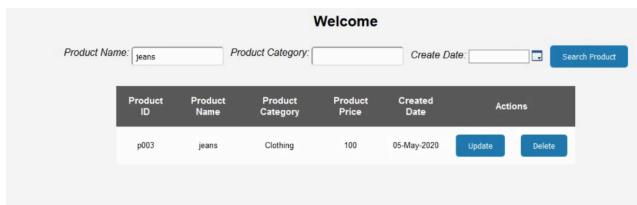
29     if(productCategory != null && !productCategory.equals("")) {
30         count++;
31         if(count != 1) {
32             whereClause += " AND ";
33         }
34         whereClause += "prod_category = '" + productCategory + "'";
35     }
36     if(createdDate != null && !createdDate.equals("")) {
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("prod_cat")

```

```

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();

```

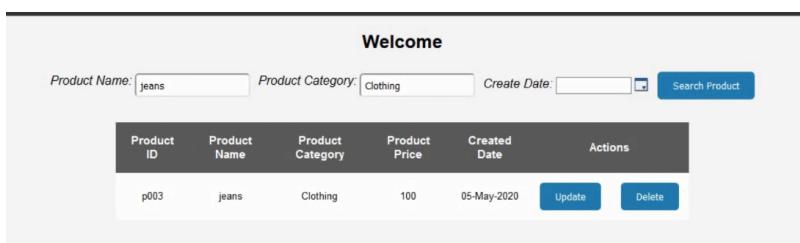


```

***** FILTER DATA *****
jeans

null
SELECT * FROM product WHERE prod_name = 'jeans'

```



```

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

```

Welcome

Product Name:  Product Category:  Create Date:

```
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'
```

The screenshot shows an IDE interface with three tabs at the top: `ProductManagementDAO.java`, `WelcomeAction.java`, and `AddAction.java`. The `ProductManagementDAO.java` tab is active, displaying the following Java code:

```
22     if(productName != null && !productName.equals("")) {
23         Struts2Tags.docx ++
24         int != 1) {
25             whereClause += " AND ";
26         }
27         whereClause += "prod_name = "+""+productName+"";
28     }
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     trv
```

debug / develop and explore