

## Lesson 02 Demo 03

### Filters

**Objective:** To explore filters by creating a new filter and working on the login filter's response, pre-processing, and post-processing

**Tools Required:** Eclipse IDE

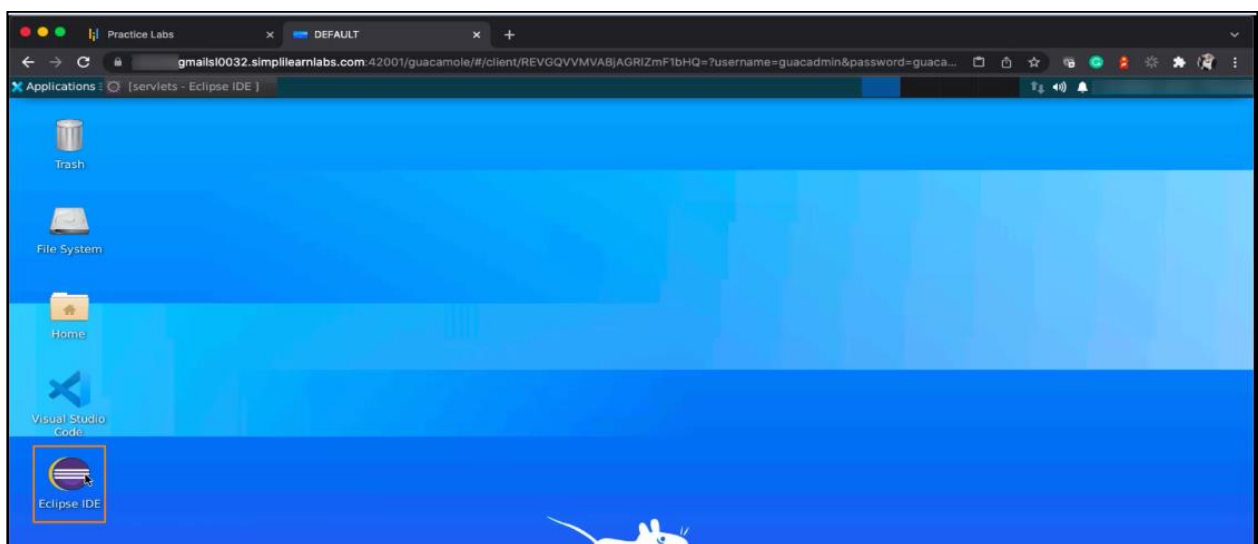
**Prerequisites:** None

#### Steps to be followed:

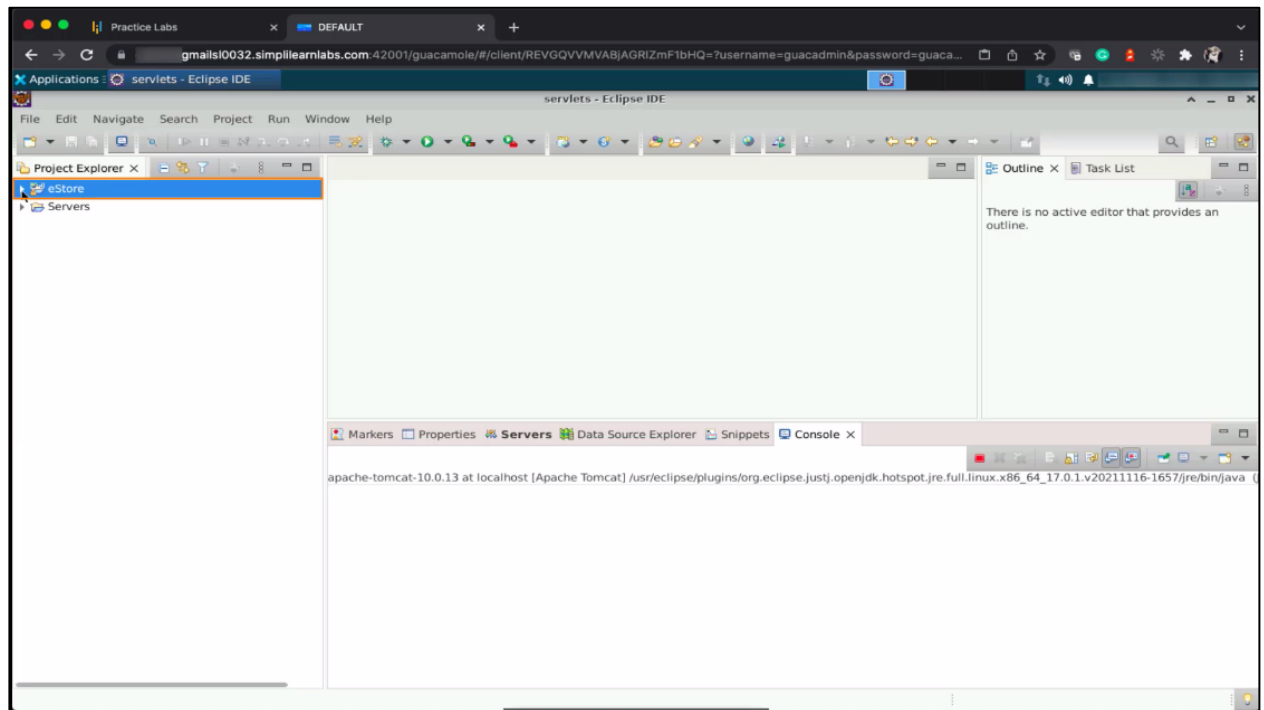
1. Create a new filter
2. Work with mandatory methods
3. Work with a filter mechanism
4. Work on the login filter's response
5. Make a filter for login parameters

#### Step 1: Create a new filter

##### 1.1 Open Eclipse IDE

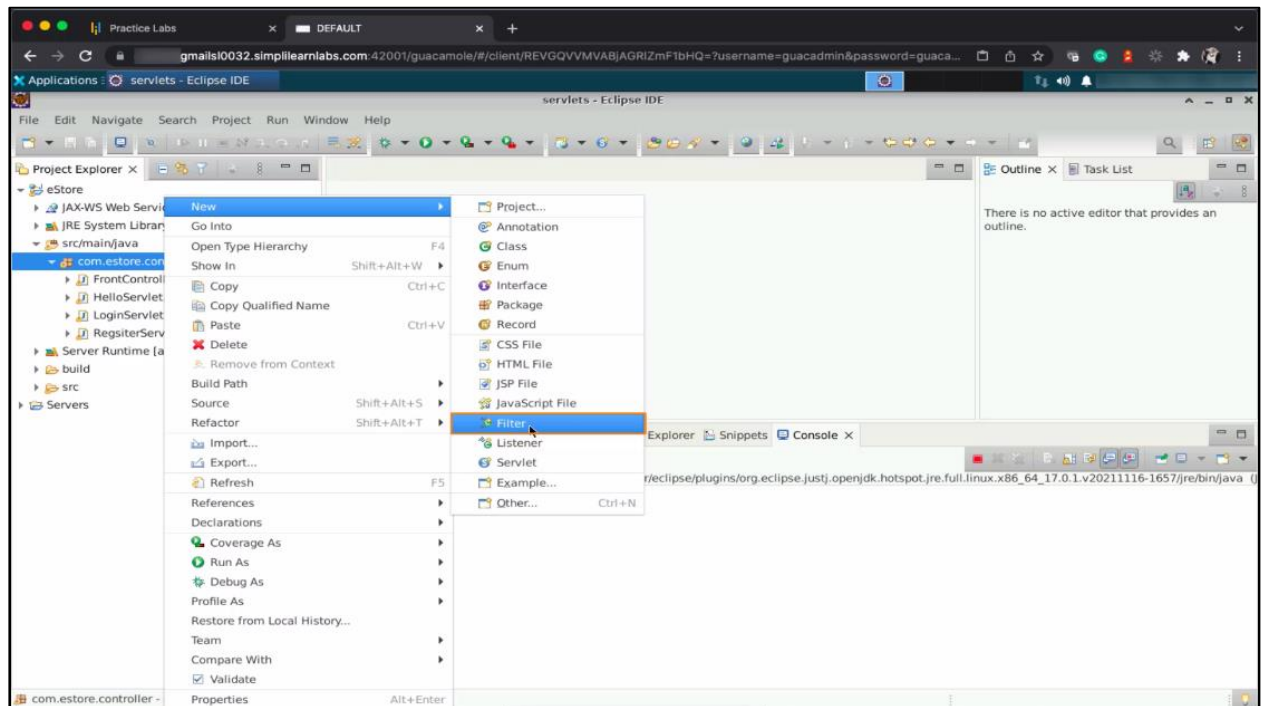


## 1.2 Navigate to the eStore project



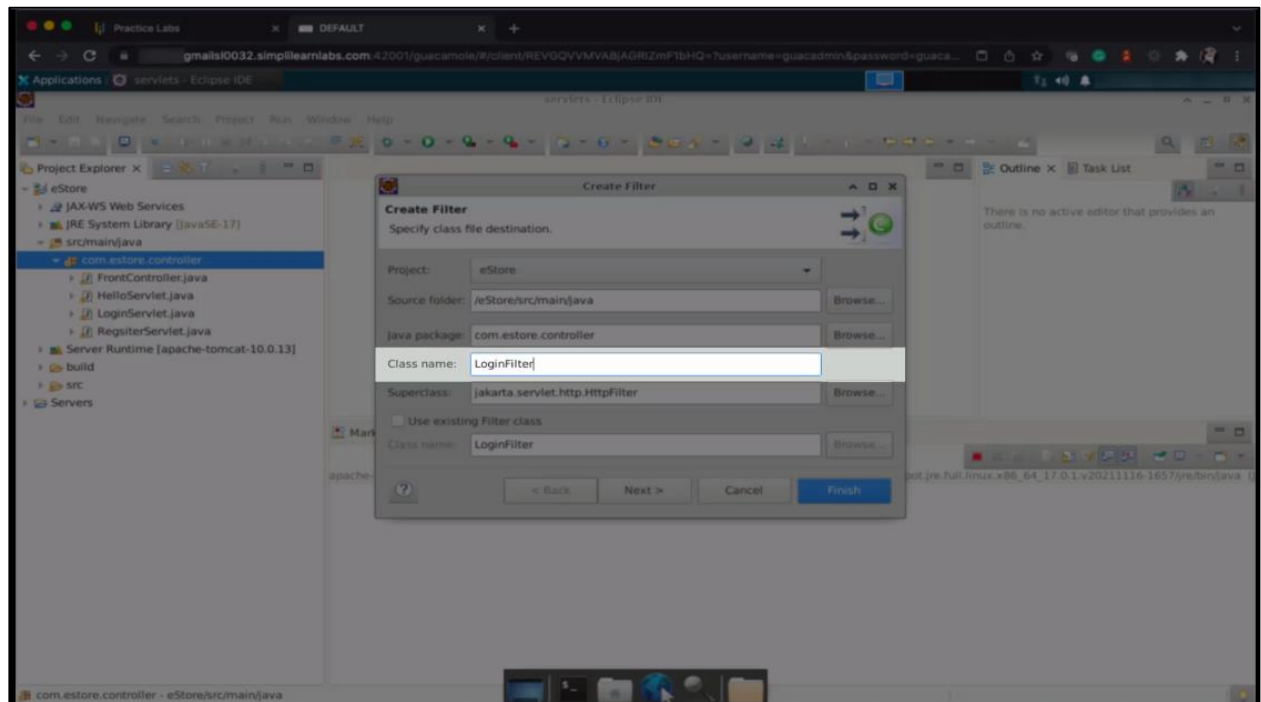
**Note:** Please refer to the previous demo on how to create the **eStore** project

### 1.3 Create a new filter by right-clicking on the project, selecting **New**, and clicking on **Filter**

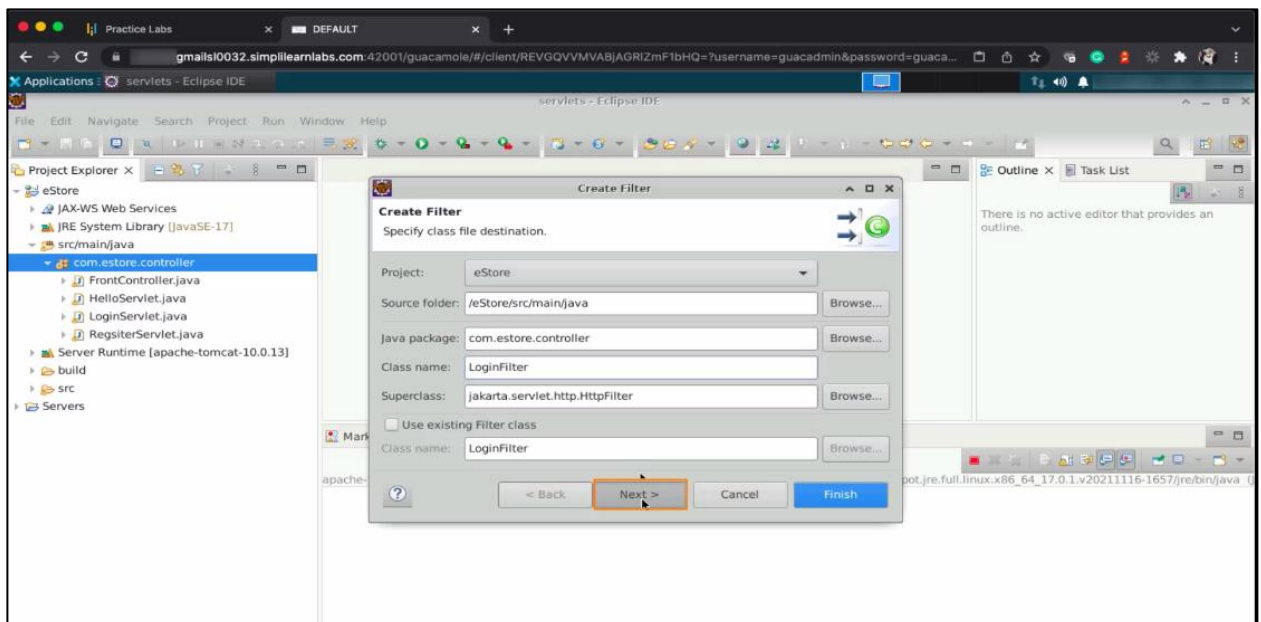


**Observation:** Rather than requests going directly to the servlet, the request will first navigate to the filter which will then send the request to the servlet.

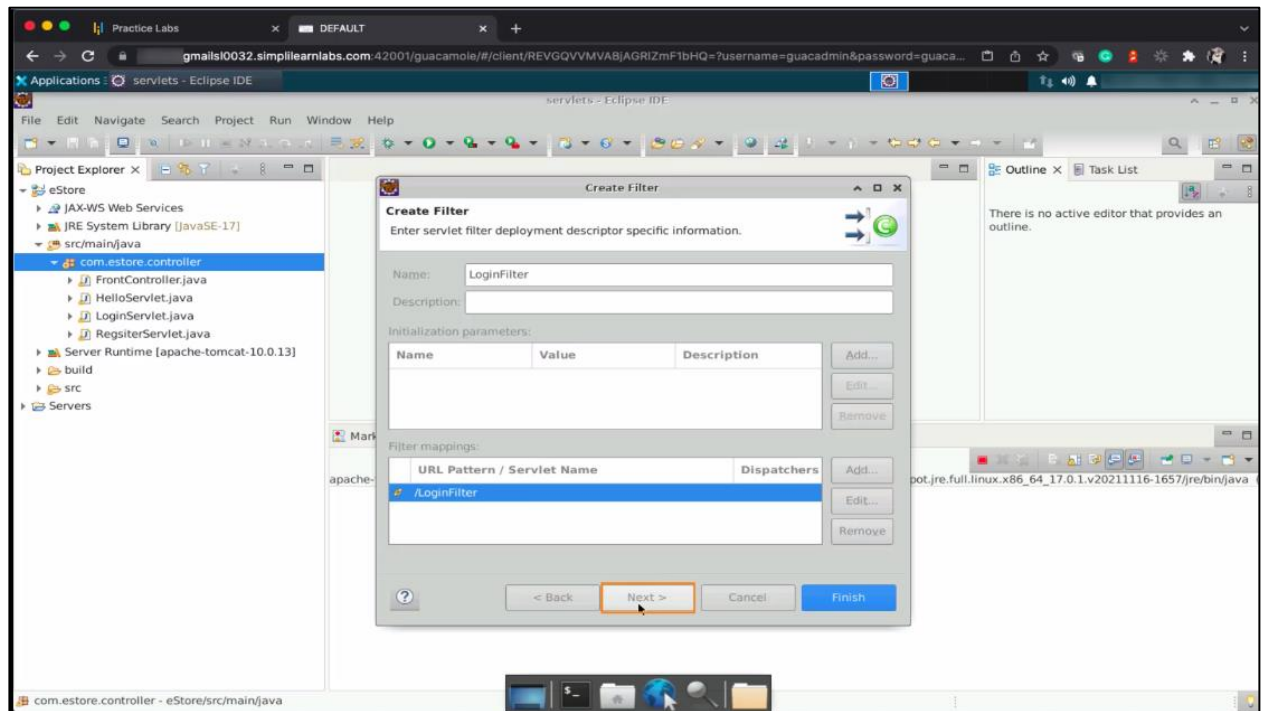
## 1.4 Name the filter as **LoginFilter**



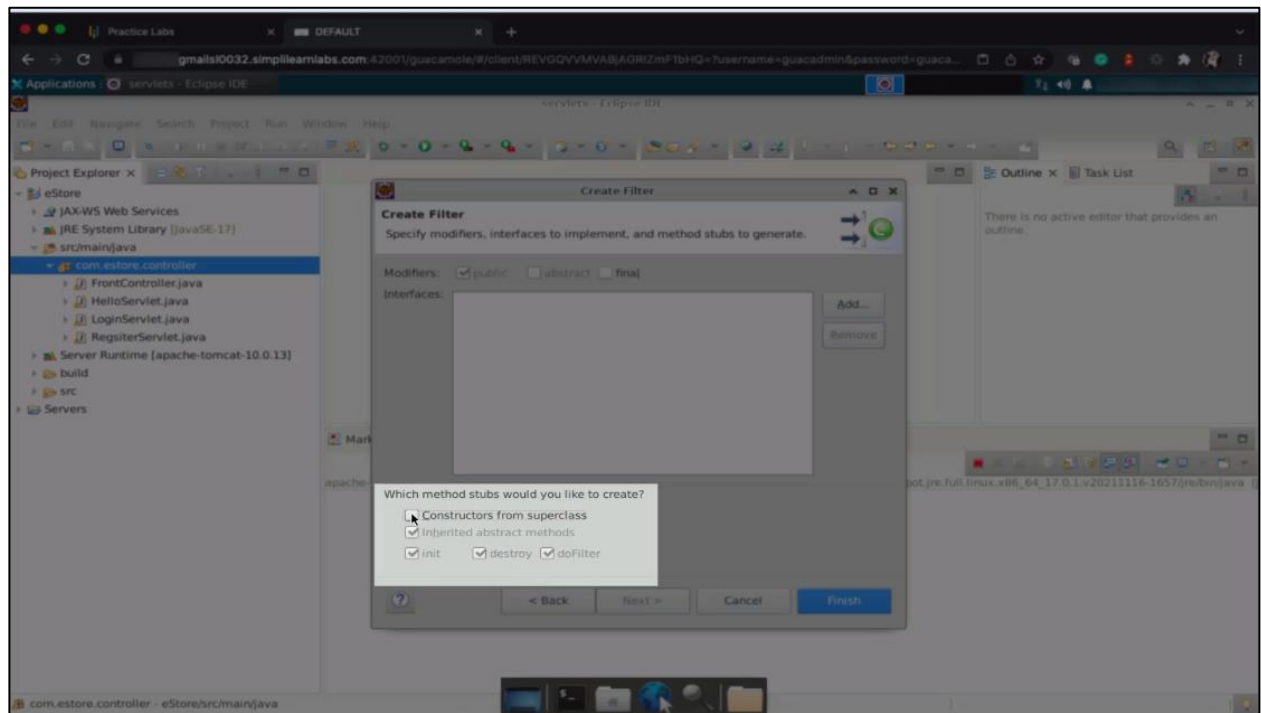
## 1.5 Click on **Next**



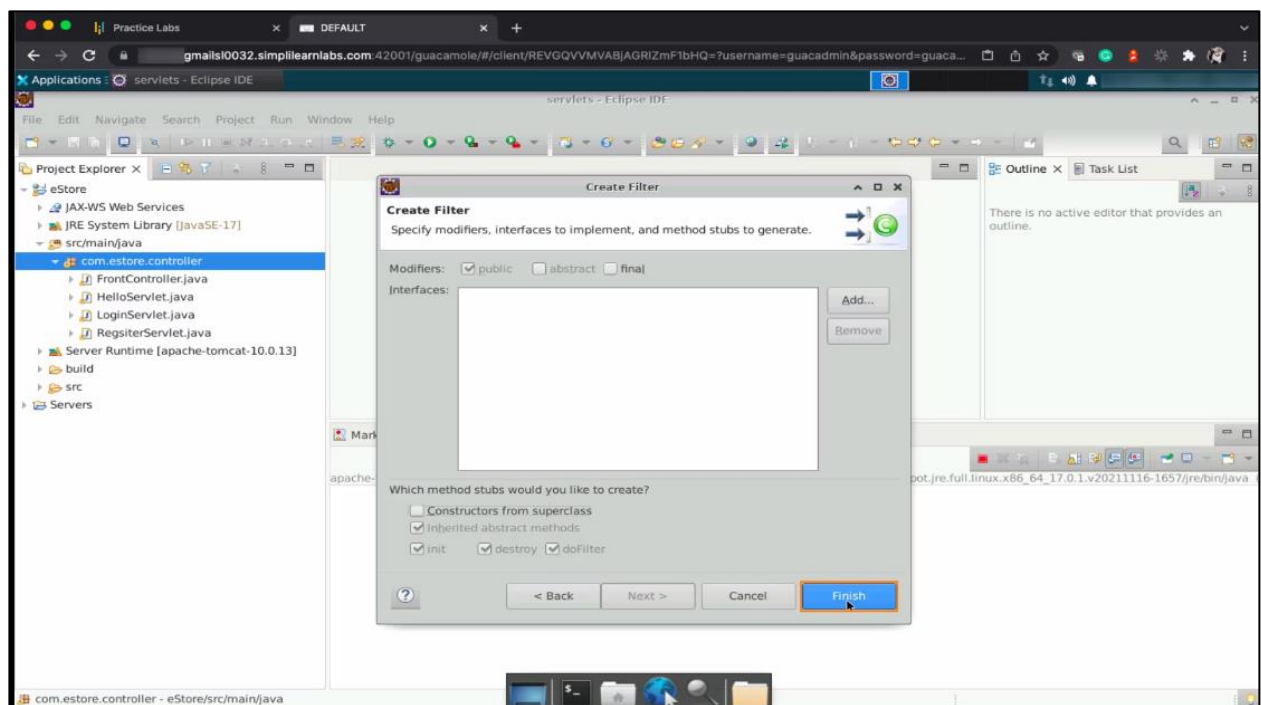
## 1.6 Specify the URL patterns by selecting the **/LoginFilter** as the URL-pattern and clicking on **Next**



## 1.7 Unselect the checkbox for **Constructor from superclass**



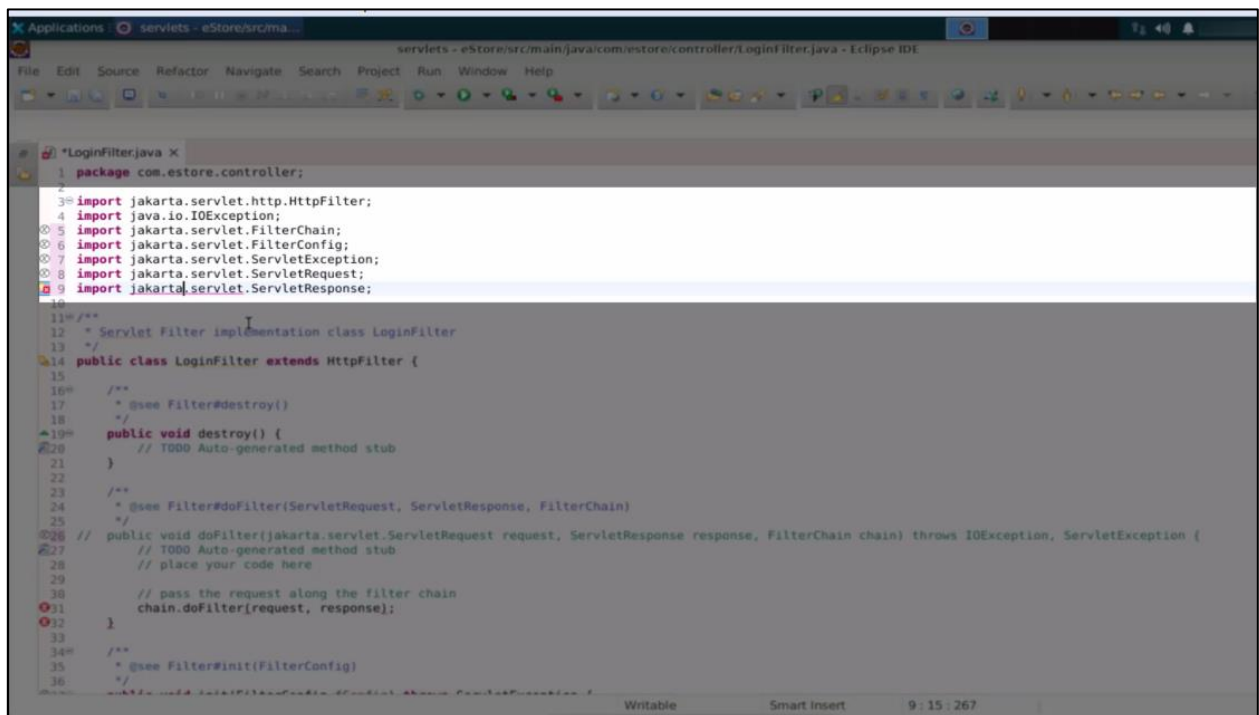
## 1.8 Click on **Finish**



## Step 2: Work with mandatory methods

2.1 Navigate to the **LoginFilter.java** file and import the following packages:

```
import jakarta.servlet.FilterChain;  
import jakarta.servlet.FilterConfig;  
import jakarta.servlet.ServletException;  
import jakarta.servlet.ServletRequest;  
import jakarta.servlet.ServletResponse;
```



The screenshot shows the Eclipse IDE interface with the `LoginFilter.java` file open. The code includes the following imports and class definition:

```
1 package com.estore.controller;  
2  
3 import jakarta.servlet.http.HttpServlet;  
4 import java.io.IOException;  
5 import jakarta.servlet.FilterChain;  
6 import jakarta.servlet.FilterConfig;  
7 import jakarta.servlet.ServletException;  
8 import jakarta.servlet.ServletRequest;  
9 import jakarta.servlet.ServletResponse;  
10  
11 /**  
12  * Servlet Filter implementation class LoginFilter  
13  */  
14 public class LoginFilter extends HttpServlet {  
15  
16     /**  
17      * @see Filter#destroy()  
18      */  
19     public void destroy() {  
20         // TODO Auto-generated method stub  
21     }  
22  
23     /**  
24      * @see Filter#doFilter(ServletRequest, ServletResponse, FilterChain)  
25      */  
26     // public void doFilter(jakarta.servlet.ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {  
27         // TODO Auto-generated method stub  
28         // place your code here  
29  
30         // pass the request along the filter chain  
31         chain.doFilter(request, response);  
32     }  
33  
34     /**  
35      * @see Filter#init(FilterConfig)  
36      */  
37 }
```

2.2 Specify the **LoginFilter** and include the execution of the destroy and init methods, you can modify the code as follows:

**System.out.println ("[LoginFilter] – destroy executed");**

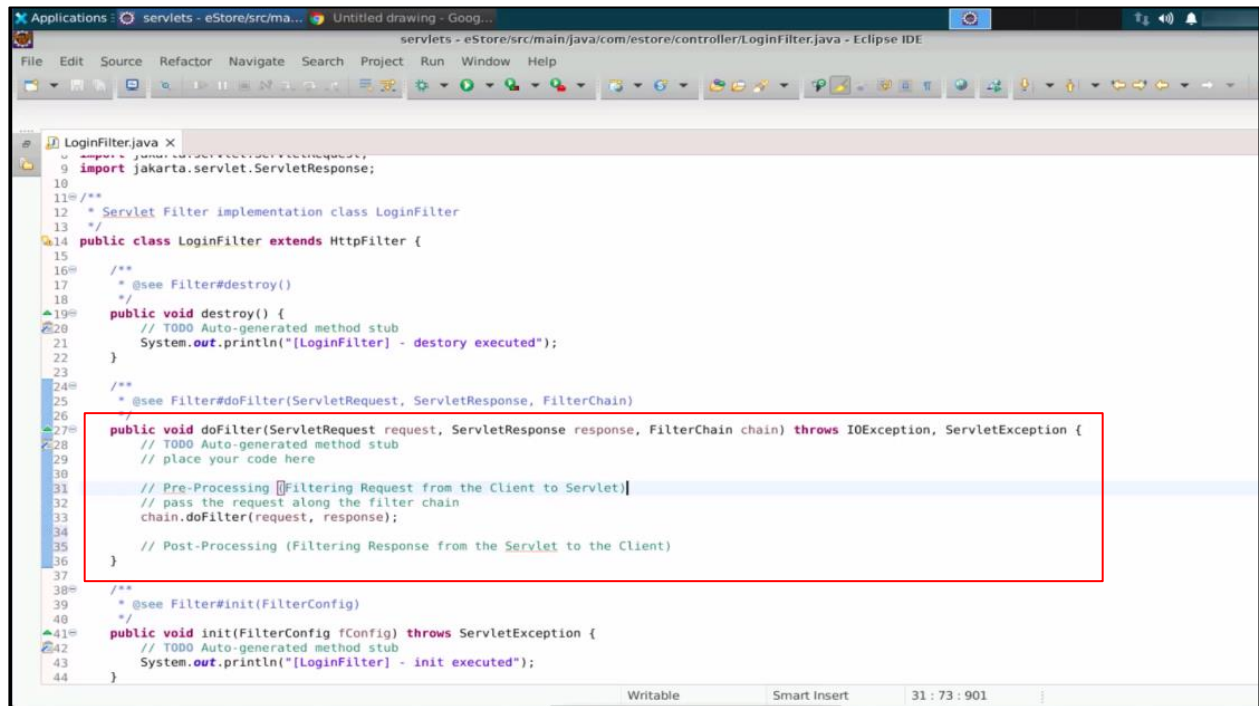
**System.out.println ("[LoginFilter] – init executed");**

```
1 LoginFilter.java X
2 import jakarta.servlet.ServletException;
3 import jakarta.servlet.http.HttpServletRequest;
4 import jakarta.servlet.http.HttpServletResponse;
5
6 /**
7  * Servlet Filter implementation class LoginFilter
8  */
9 public class LoginFilter extends HttpFilter {
10
11     /**
12     * @see Filter#destroy()
13     */
14     public void destroy() {
15         // TODO Auto-generated method stub
16         System.out.println("[LoginFilter] - destroy executed");
17     }
18
19     /**
20     * @see Filter#doFilter(ServletRequest, ServletResponse, FilterChain)
21     */
22     public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {
23         // TODO Auto-generated method stub
24         // place your code here
25
26         // pass the request along the filter chain
27         chain.doFilter(request, response);
28     }
29
30     /**
31     * @see Filter#init(FilterConfig)
32     */
33     public void init(FilterConfig fConfig) throws ServletException {
34         // TODO Auto-generated method stub
35         System.out.println("[LoginFilter] - init executed");
36     }
37 }
38
39
40
41
42
43
44
```



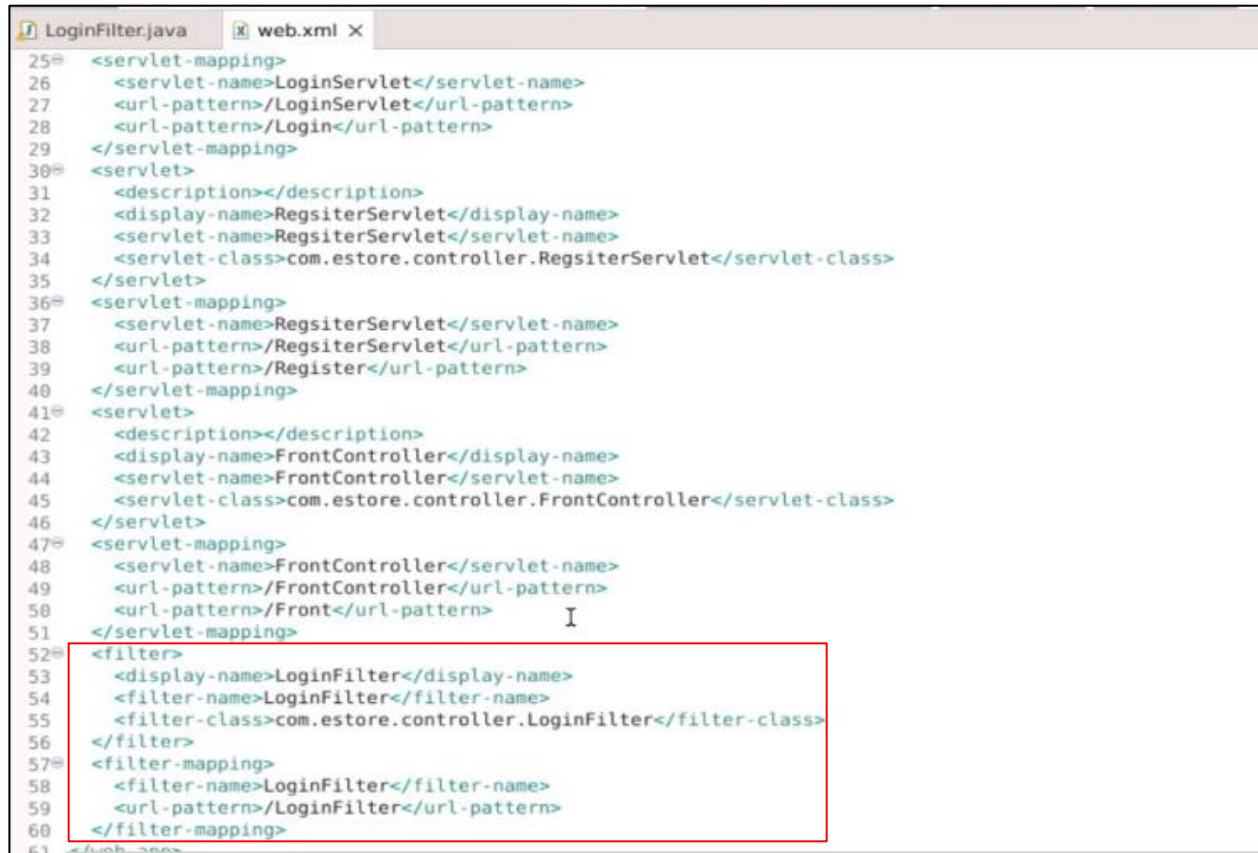
## Step 3: Work with a filter mechanism

### 3.1 Return to the **LoginServlet.java** file and specify the pre-processing and post-processing of data



```
1  LoginFilter.java X
2  package com.estore.controller;
3  import jakarta.servlet.ServletException;
4  import jakarta.servlet.http.HttpServletRequest;
5  import jakarta.servlet.http.HttpServletResponse;
6  import java.io.IOException;
7
8  /**
9   * Servlet Filter implementation class LoginFilter
10   */
11  public class LoginFilter extends HttpFilter {
12
13      /**
14       * @see Filter#destroy()
15       */
16      public void destroy() {
17          // TODO Auto-generated method stub
18          System.out.println("[LoginFilter] - destroy executed");
19      }
20
21      /**
22       * @see Filter#doFilter(ServletRequest, ServletResponse, FilterChain)
23       */
24      public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {
25          // TODO Auto-generated method stub
26          // place your code here
27
28          // Pre-Processing (Filtering Request from the Client to Servlet)
29          // pass the request along the filter chain
30          chain.doFilter(request, response);
31
32          // Post-Processing (Filtering Response from the Servlet to the Client)
33      }
34
35      /**
36       * @see Filter#init(FilterConfig)
37       */
38      public void init(FilterConfig fConfig) throws ServletException {
39          // TODO Auto-generated method stub
40          System.out.println("[LoginFilter] - init executed");
41      }
42  }
```

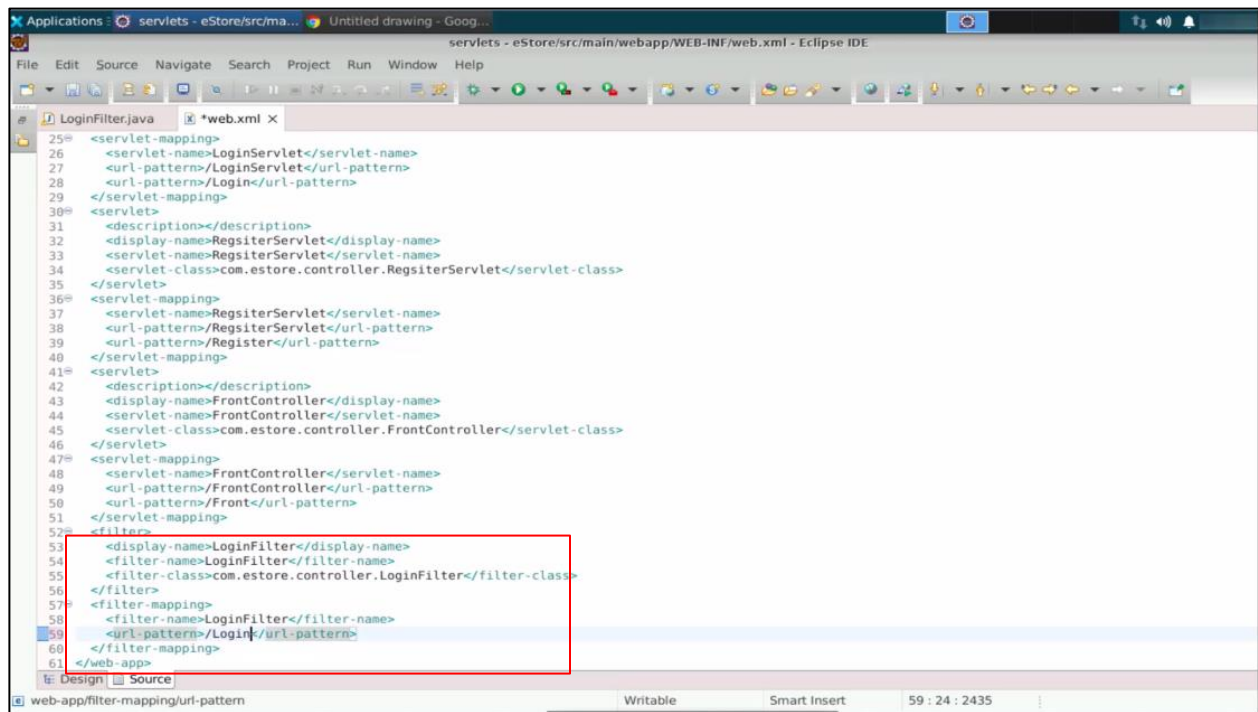
3.2 Open the **web.xml** file where you will find the login filter and filter mapping that has been created with the name **LoginFilter**



```
25 <servlet-mapping>
26   <servlet-name>LoginServlet</servlet-name>
27   <url-pattern>/LoginServlet</url-pattern>
28   <url-pattern>/Login</url-pattern>
29 </servlet-mapping>
30 <servlet>
31   <description></description>
32   <display-name>RegsiterServlet</display-name>
33   <servlet-name>RegsiterServlet</servlet-name>
34   <servlet-class>com.estore.controller.RegisiterServlet</servlet-class>
35 </servlet>
36 <servlet-mapping>
37   <servlet-name>RegsiterServlet</servlet-name>
38   <url-pattern>/RegsiterServlet</url-pattern>
39   <url-pattern>/Register</url-pattern>
40 </servlet-mapping>
41 <servlet>
42   <description></description>
43   <display-name>FrontController</display-name>
44   <servlet-name>FrontController</servlet-name>
45   <servlet-class>com.estore.controller.FrontController</servlet-class>
46 </servlet>
47 <servlet-mapping>
48   <servlet-name>FrontController</servlet-name>
49   <url-pattern>/FrontController</url-pattern>
50   <url-pattern>/Front</url-pattern>
51 </servlet-mapping>
52 <filter>
53   <display-name>LoginFilter</display-name>
54   <filter-name>LoginFilter</filter-name>
55   <filter-class>com.estore.controller.LoginFilter</filter-class>
56 </filter>
57 <filter-mapping>
58   <filter-name>LoginFilter</filter-name>
59   <url-pattern>/LoginFilter</url-pattern>
60 </filter-mapping>
61 </web-app>
```

## Step 4: Work on the login filter's response

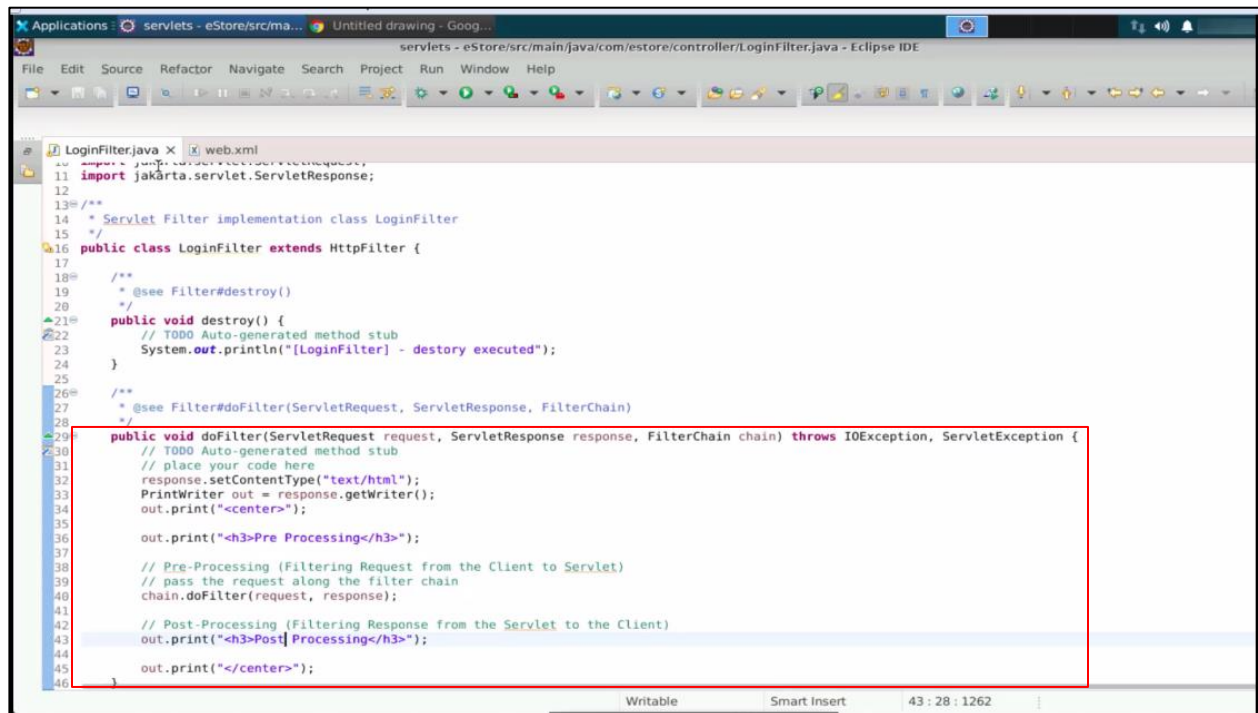
### 4.1 Change the value of the url-pattern from /LoginFilter to /Login



```
25<= <servlet-mapping>
26   <servlet-name>LoginServlet</servlet-name>
27   <url-pattern>/LoginServlet</url-pattern>
28 </servlet-mapping>
29 </servlet-mapping>
30<= <servlet>
31   <description></description>
32   <display-name>RegsiterServlet</display-name>
33   <servlet-name>RegsiterServlet</servlet-name>
34   <servlet-class>com.estore.controller.RegisiterServlet</servlet-class>
35 </servlet>
36<= <servlet-mapping>
37   <servlet-name>RegsiterServlet</servlet-name>
38   <url-pattern>/RegsiterServlet</url-pattern>
39   <url-pattern>/Register</url-pattern>
40 </servlet-mapping>
41<= <servlet>
42   <description></description>
43   <display-name>FrontController</display-name>
44   <servlet-name>FrontController</servlet-name>
45   <servlet-class>com.estore.controller.FrontController</servlet-class>
46 </servlet>
47<= <servlet-mapping>
48   <servlet-name>FrontController</servlet-name>
49   <url-pattern>/FrontController</url-pattern>
50   <url-pattern>/Front</url-pattern>
51 </servlet-mapping>
52<= <filter>
53   <display-name>LoginFilter</display-name>
54   <filter-name>LoginFilter</filter-name>
55   <filter-class>com.estore.controller.LoginFilter</filter-class>
56 </filter>
57<= <filter-mapping>
58   <filter-name>LoginFilter</filter-name>
59   <url-pattern>/Login</url-pattern>
60 </filter-mapping>
61 </web-app>
```

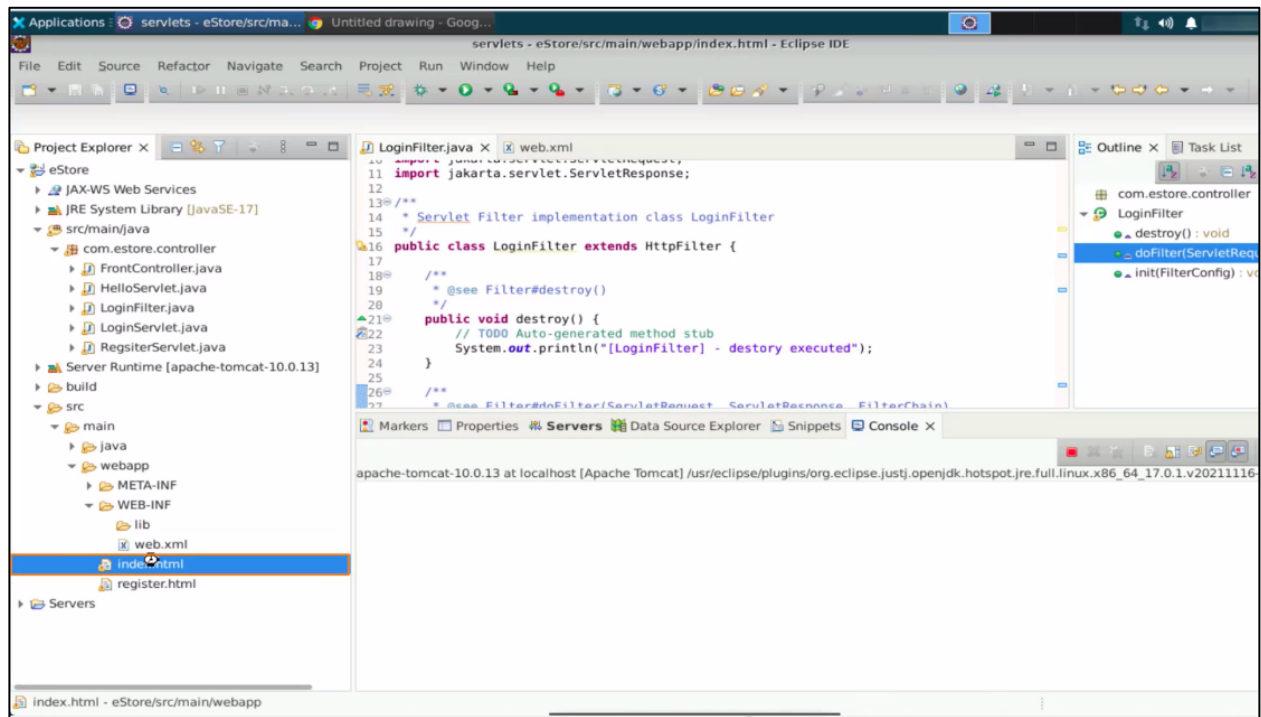
This is because the URL pattern specified for the filter will match that of the servlet. As a result, the request will be intercepted by the filter before it reaches the servlet directly.

4.2 In **LoginFilter.java**, modify the response to include pre-processing and post-processing messages using the **setContentType()** method and **<h3>** tags

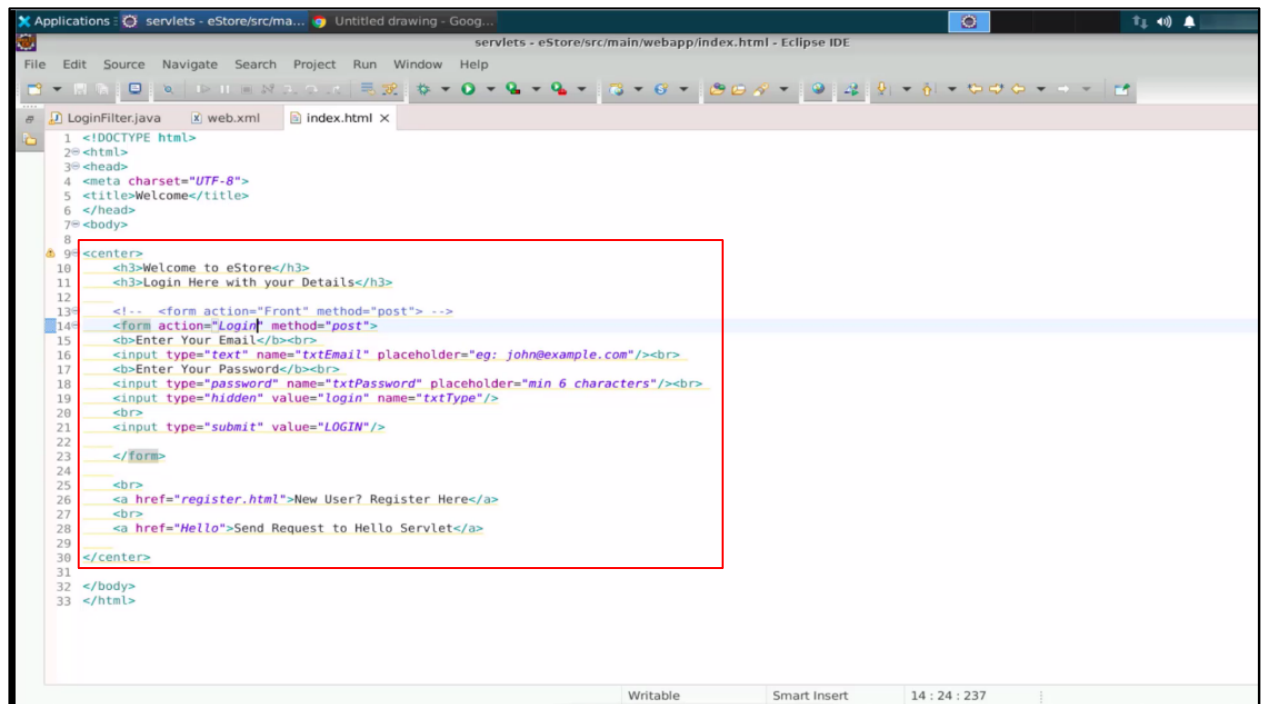


```
10 import javax.servlet.ServletException;
11 import jakarta.servlet.ServletResponse;
12
13 /**
14  * Servlet Filter implementation class LoginFilter
15  */
16 public class LoginFilter extends HttpFilter {
17
18     /**
19      * @see Filter#destroy()
20      */
21     public void destroy() {
22         // TODO Auto-generated method stub
23         System.out.println("[LoginFilter] - destroy executed");
24     }
25
26     /**
27      * @see Filter#doFilter(ServletRequest, ServletResponse, FilterChain)
28      */
29     public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {
30         // TODO Auto-generated method stub
31         // place your code here
32         response.setContentType("text/html");
33         PrintWriter out = response.getWriter();
34         out.print("<center>");
35
36         out.print("<h3>Pre Processing</h3>");
37
38         // Pre-Processing (Filtering Request from the Client to Servlet)
39         // pass the request along the filter chain
40         chain.doFilter(request, response);
41
42         // Post-Processing (Filtering Response from the Servlet to the Client)
43         out.print("<h3>Post Processing</h3>");
44
45         out.print("</center>");
46     }
47 }
```

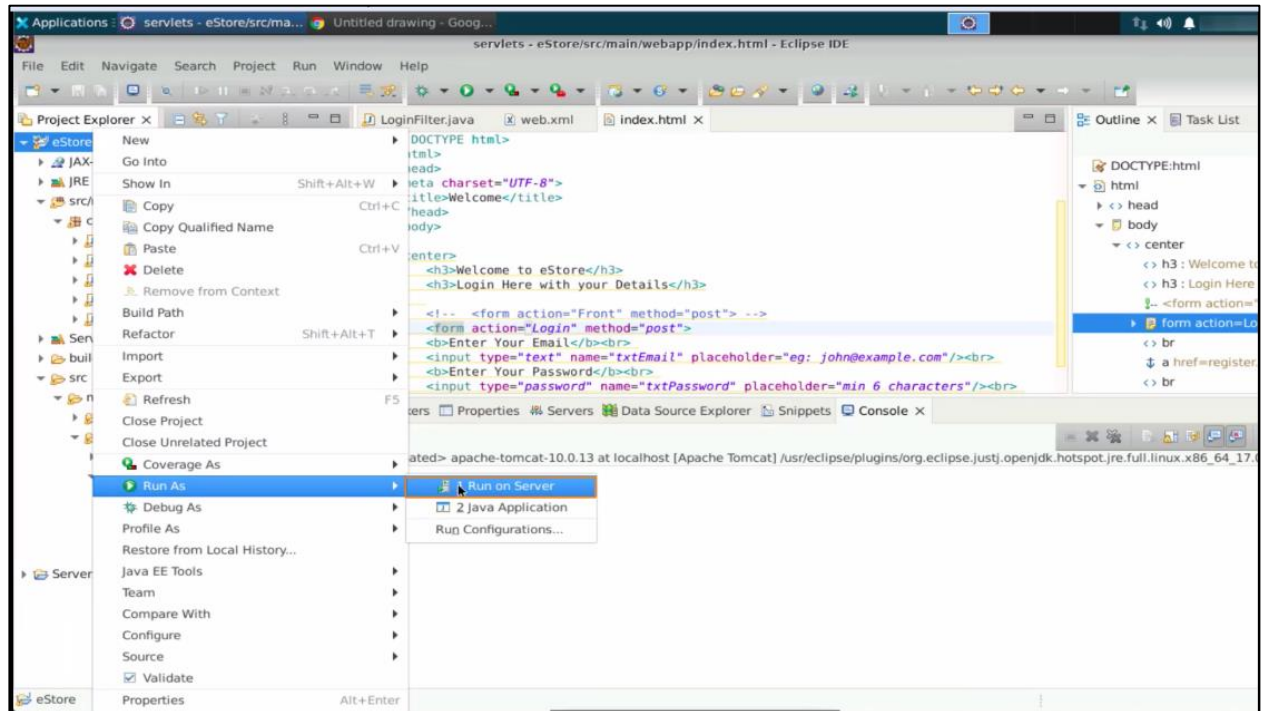
#### 4.3 Open the **index.html** file



#### 4.4 In the **index.html** file, change the action to **Login** and the method to **post** (13 and 14)

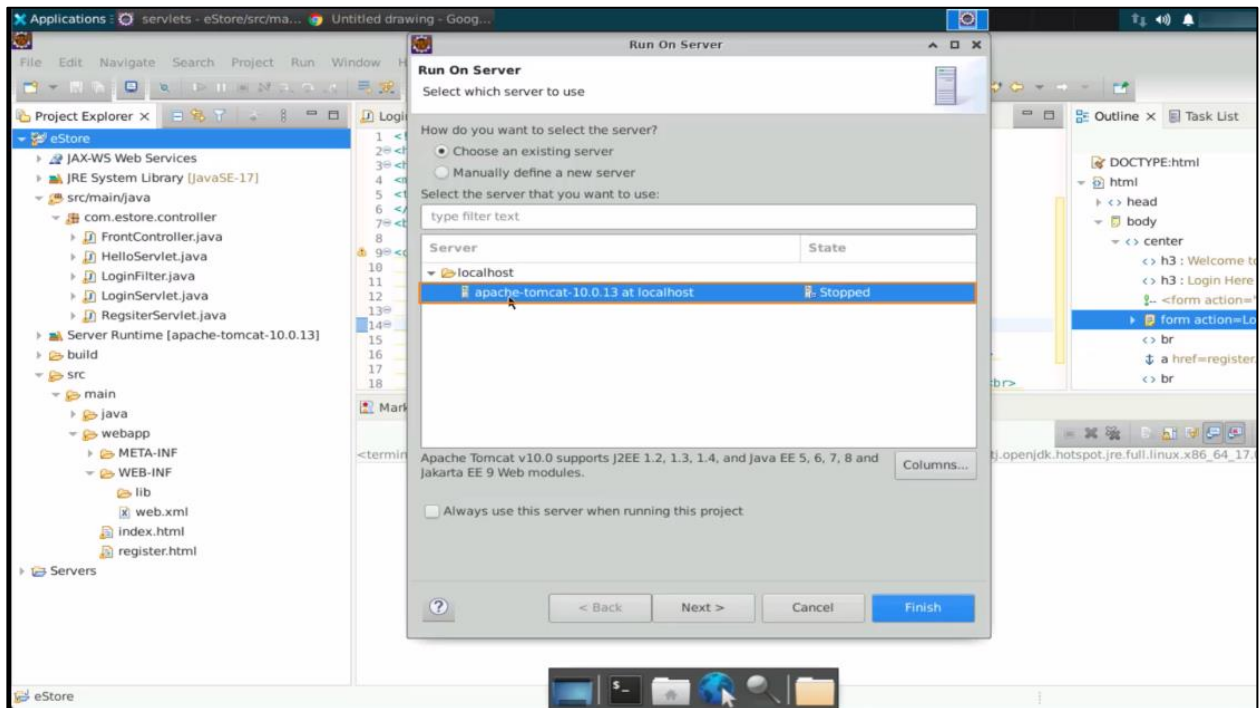


#### 4.5 Navigate to the **estore** project, right-click on it, select **Run As**, and click on **Run on Server**

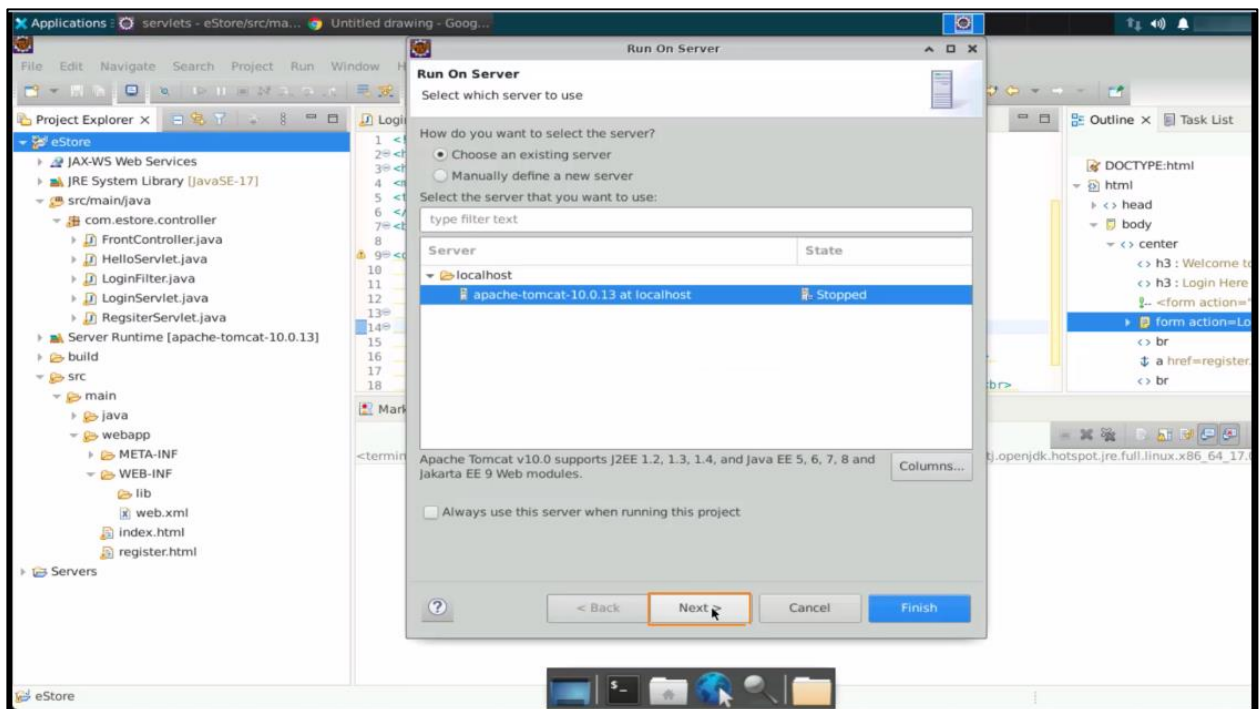




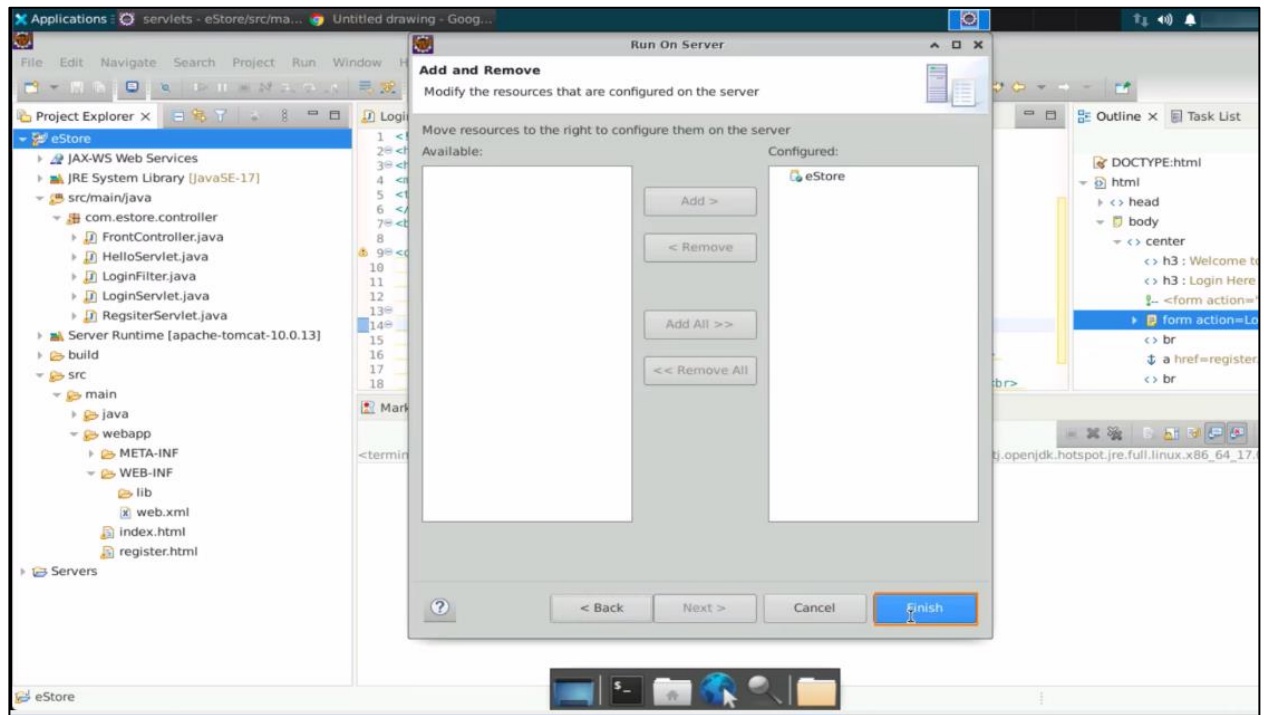
#### 4.6 Select the **apache-tomcat-10.0.13** as the server



#### 4.7 Click on **Next**

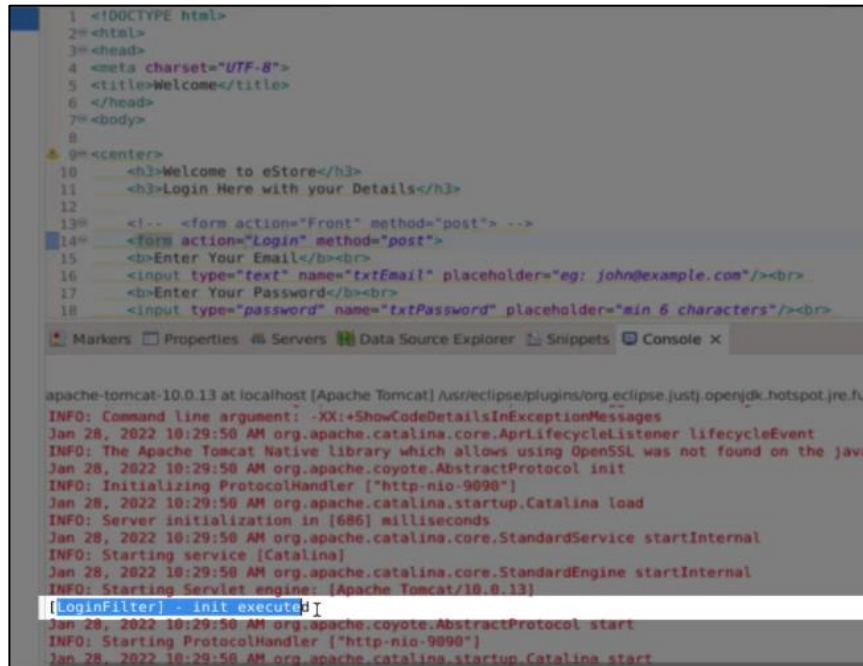


## 4.8 Click on **Finish**





4.9 Navigate to the Eclipse Console tab where the message **[LoginFilter] – init executed** indicates the successful initialization of the **LoginFilter**



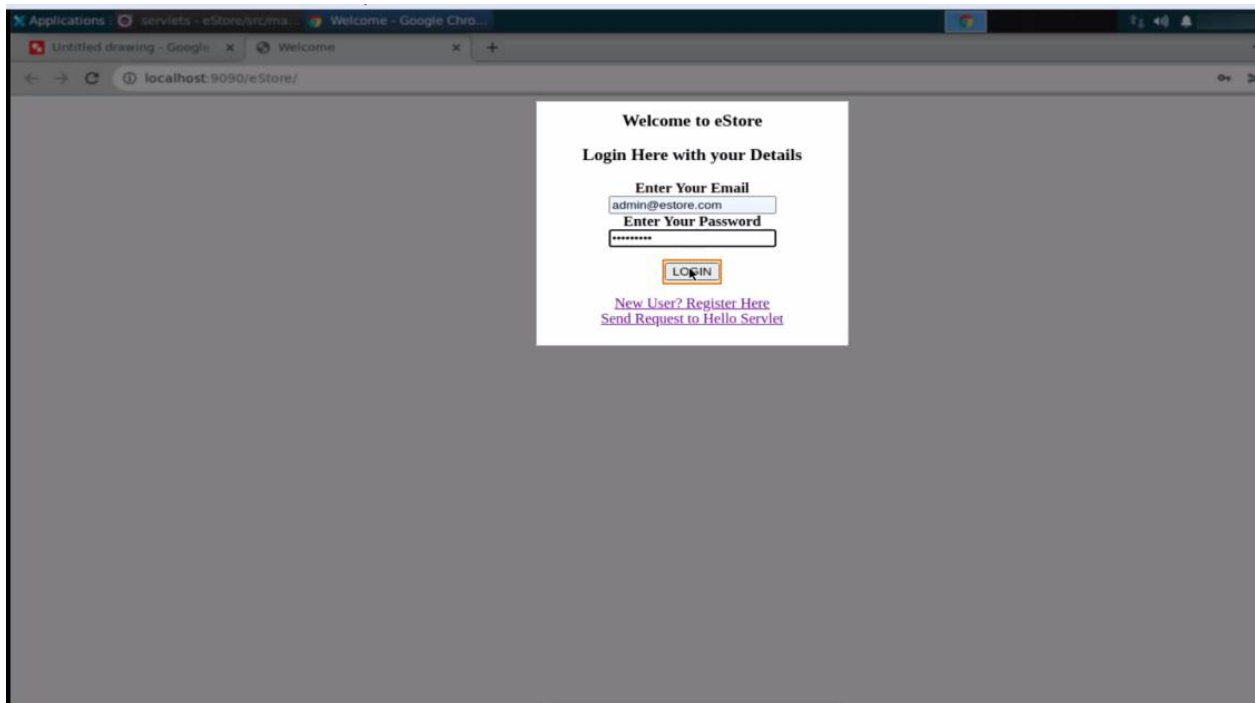
The screenshot shows the Eclipse IDE with two tabs: 'Markers' and 'Console'. The 'Console' tab is active, displaying the following log messages:

```

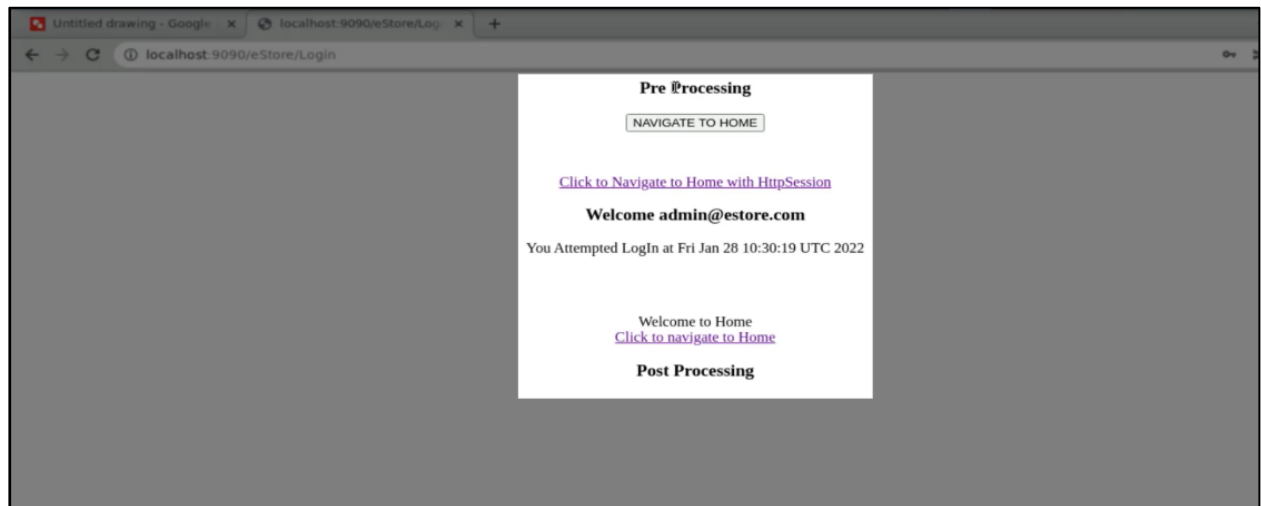
apache-tomcat-10.0.13 at localhost [Apache Tomcat] /usr/eclipse/plugins/org.eclipse.justi.openjdk.hotspot.jre.full
INFO: Command line argument: -XX:+ShowCodeDetailsInExceptionMessages
Jan 28, 2022 10:29:58 AM org.apache.catalina.core.AprLifecycleListener lifecycleEvent
INFO: The Apache Tomcat Native library which allows using OpenSSL was not found on the java
Jan 28, 2022 10:29:58 AM org.apache.coyote.AbstractProtocol init
INFO: Initializing ProtocolHandler ["http-nio-9090"]
Jan 28, 2022 10:29:58 AM org.apache.catalina.startup.Catalina load
INFO: Server initialization in [686] milliseconds
Jan 28, 2022 10:29:58 AM org.apache.catalina.core.StandardService startInternal
INFO: Starting service [Catalina]
Jan 28, 2022 10:29:58 AM org.apache.catalina.core.StandardEngine startInternal
INFO: Starting Servlet engine: [Apache Tomcat/10.0.13]
[LoginFilter] -- init executed
Jan 28, 2022 10:29:58 AM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-nio-9090"]
Jan 28, 2022 10:29:58 AM org.apache.catalina.startup.Catalina start
  
```

The HTML code in the background shows a login form with fields for email and password, and a 'LOGIN' button.

4.10 Enter the login details and click on **LOGIN**



You can see the output as **Pre-processing** and **Post-processing**.



This means that the request is sent from the filter to the Servlet, and the response is sent from the Servlet to the filter.

### Step 5: Make the filter work for login parameters

5.1 Navigate to **LoginFilter.java** and extract the email and the password from the request using the **request.getParameter()** method (lines 38 and 39)

```

22 // TODO Auto-generated method stub
23 System.out.println("[LoginFilter] - destroy executed");
24 }
25
26 /**
27  * @see Filter#doFilter(ServletRequest, ServletResponse, FilterChain)
28  */
29 public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {
30     // TODO Auto-generated method stub
31     // place your code here
32     response.setContentType("text/html");
33     PrintWriter out = response.getWriter();
34     out.print("<center>");
35
36     out.print("<h3>Pre Processing</h3>");
37
38     String email = request.getParameter("txtEmail");
39     String password = request.getParameter("txtPassword");
40
41     // Pre-Processing (Filtering Request from the Client to Servlet)
42     // pass the request along the filter chain
43     chain.doFilter(request, response);
44
45     // Post-Processing (Filtering Response from the Servlet to the Client)
46     out.print("<h3>Post Processing</h3>");
47
48     out.print("</center>");
49 }
50
51 /**
52  * @see Filter#init(FilterConfig)
53  */
54 public void init(FilterConfig fConfig) throws ServletException {
55     // TODO Auto-generated method stub
56     System.out.println("[LoginFilter] - init executed");
57 }

```

5.2 Navigate to the console logs where you will see that the filter has been destroyed

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Welcome</title>
6 </head>
7 <body>
8
9 <center>
10 <h3>Welcome to eStore</h3>
11 <h3>Login Here with your Details</h3>
12
13 <!-- <form action="Front" method="post"> -->
14 <form action="Login" method="post">
15 <b>Enter Your Email</b><br>
16 <input type="text" name="txtEmail" placeholder="eg: johnd@example.com"/><br>
17 <b>Enter Your Password</b><br>
18 <input type="password" name="txtPassword" placeholder="min 6 characters"/><br>

```

```

apache-tomcat-10.0.13 at localhost [Apache Tomcat] Aus/eclipse/plugins/org.eclipse.justi.openjdk.hotspot.jre.full.linux.x86_64_17.0.1.v20211111
INFO: Starting service [Catalina]
Jan 28, 2022 10:29:50 AM org.apache.catalina.core.StandardEngine startInternal
INFO: Starting Servlet engine: [Apache Tomcat/10.0.13]
[LoginFilter] - init executed
Jan 28, 2022 10:29:50 AM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-nio-9090"]
Jan 28, 2022 10:29:50 AM org.apache.catalina.startup.Catalina start
INFO: Server startup in [374] milliseconds
Jan 28, 2022 10:31:30 AM org.apache.catalina.core.StandardContext reload
INFO: Reloading Context with name [/eStore] has started
[LoginFilter] - destroy executed
[LoginFilter] - init executed
Jan 28, 2022 10:31:30 AM org.apache.catalina.core.StandardContext reload
INFO: Reloading Context with name [/eStore] is completed

```

5.3 For general pre-processing, check if the email or password is empty by using the **isEmpty()** function and use the **doFilter()** method to perform the filtering in pre-processing

```

22 // TODO Auto-generated method stub
23 System.out.println("[LoginFilter] - destroy executed");
24 }
25
26 /**
27  * @see Filter#doFilter(ServletRequest, ServletResponse, FilterChain)
28  */
29 public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {
30 // TODO Auto-generated method stub
31 // place your code here
32 response.setContentType("text/html");
33 PrintWriter out = response.getWriter();
34 out.print("<center>");
35
36 // Pre-Processing (Filtering Request from the Client to Servlet)
37 out.print("<h3>Pre Processing</h3>");
38
39 String email = request.getParameter("txtEmail");
40 String password = request.getParameter("txtPassword");
41
42 if(email.isEmpty() || password.isEmpty()) {
43 // pass the data to LoginServlet
44 chain.doFilter(request, response);
45 } else {
46 out.print("<h3>Sorry!! Email and Password Cannot be Blank</h3>");
47 }
48
49
50
51 // Post-Processing (Filtering Response from the Servlet to the Client)
52 out.print("<h3>Post Processing</h3>");
53
54 out.print("</center>");
55 }
56

```

5.4 Next, rerun the code by right-clicking on the **eStore** project, selecting **Run As**, and clicking on **Run on Server**

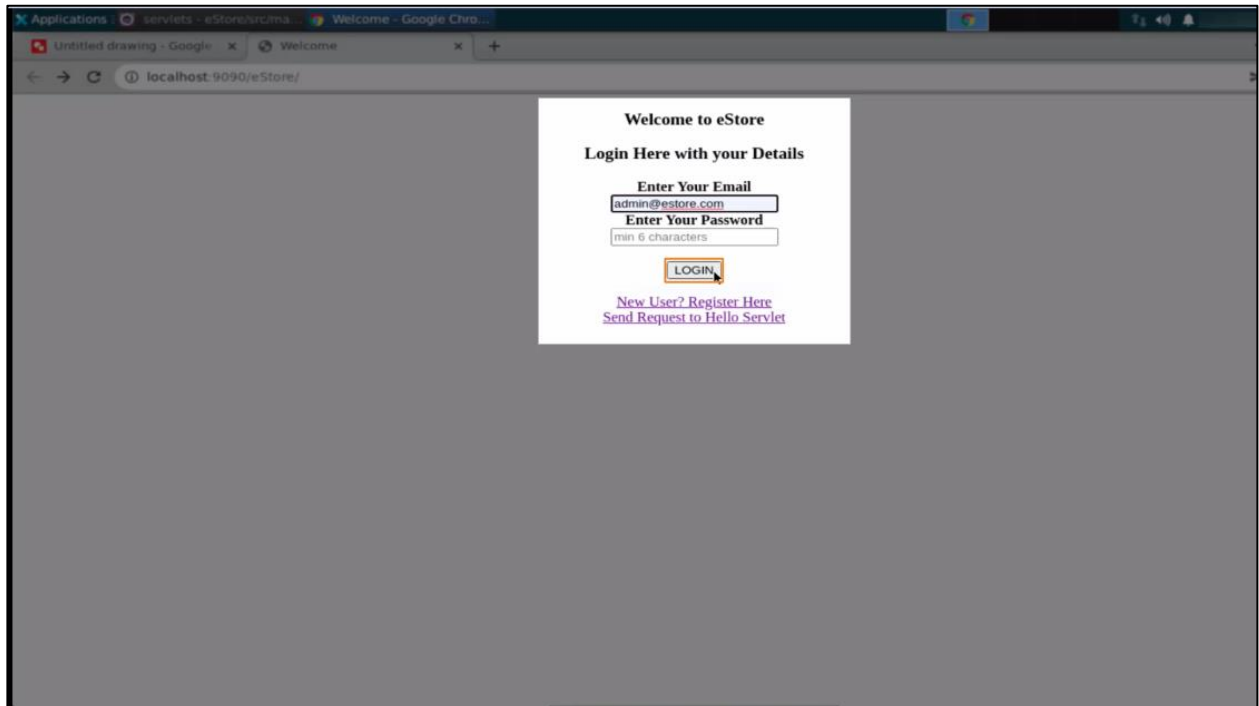
```

1 Run on Server
2 Java Application
Run Configurations...

[er] - init executed
122 10:33:30 AM org.apache.catalina.core.StandardContext reload
loading Context with name [/eStore] is completed
122 10:34:00 AM org.apache.catalina.core.StandardContext reload
loading Context with name [/eStore] has started
[er] - destroy executed
[er] - init executed
122 10:34:00 AM org.apache.catalina.core.StandardContext reload
loading Context with name [/eStore] is completed

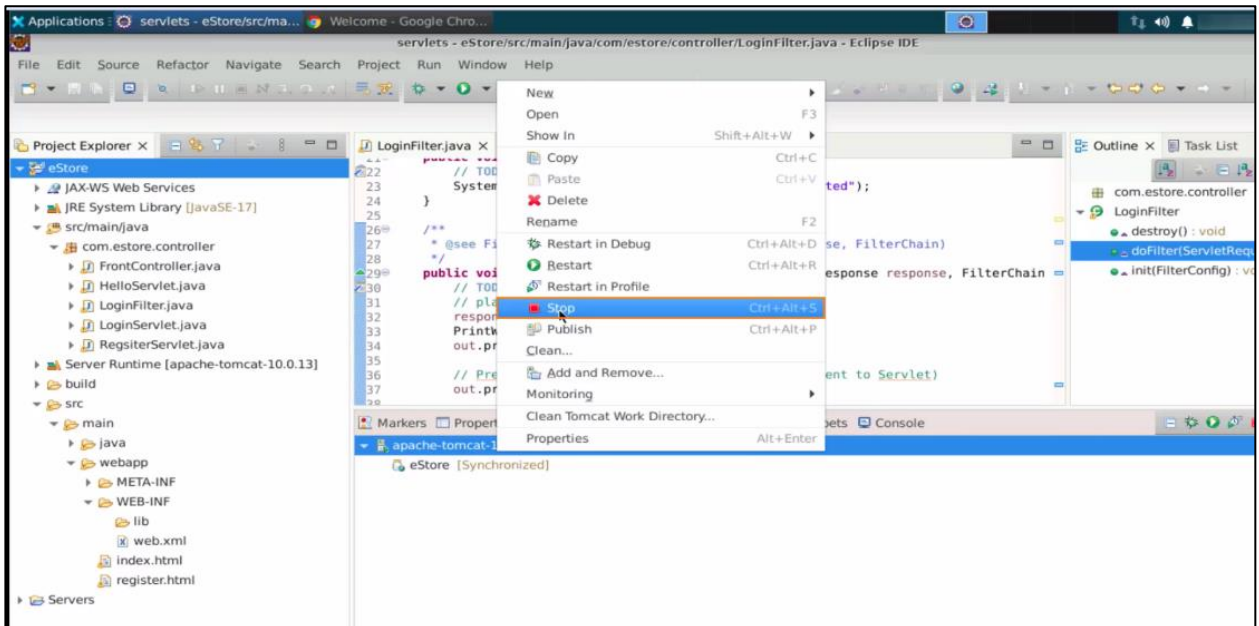
```

## 5.5 Refresh the page and enter only the email, and click on **LOGIN**



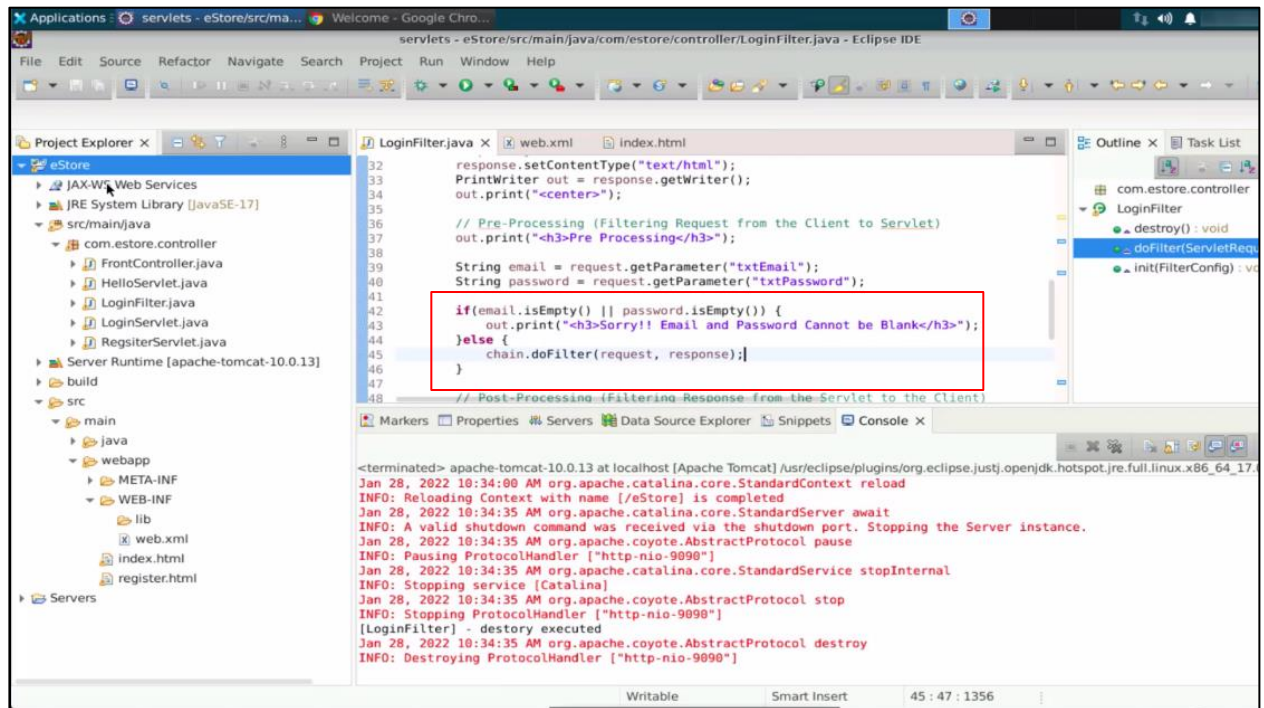
You can see that the server is not yet initialized.

## 5.6 Go back and stop the server by clicking on **Stop**

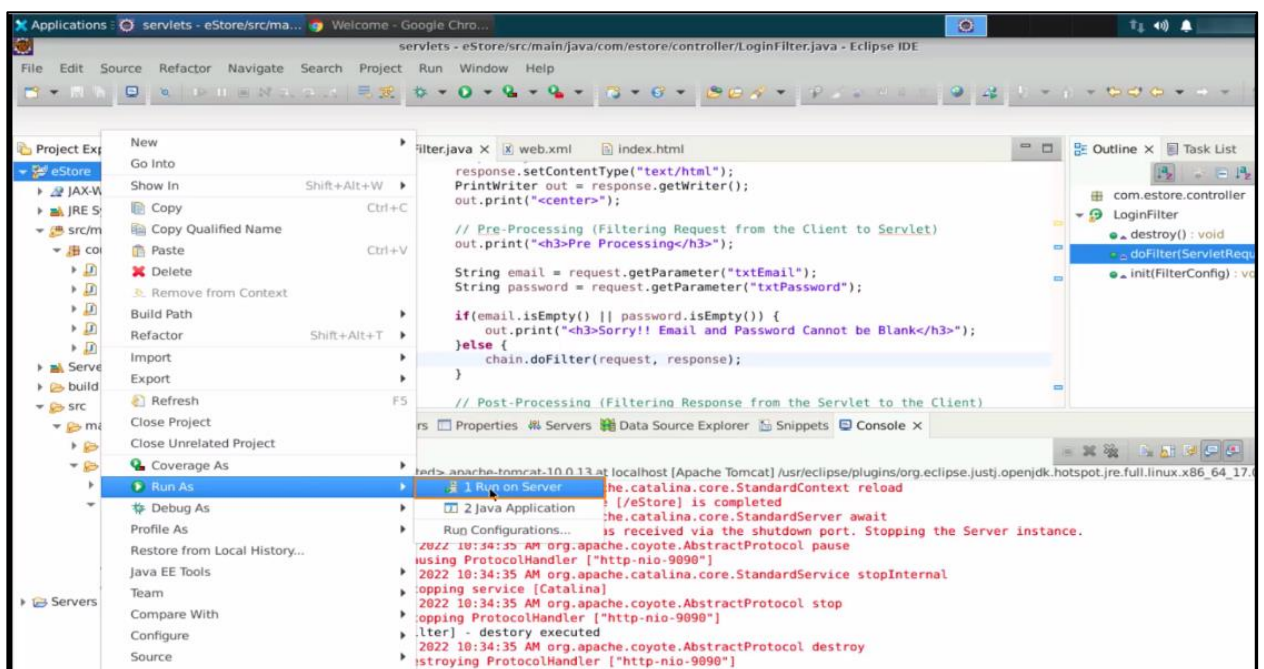




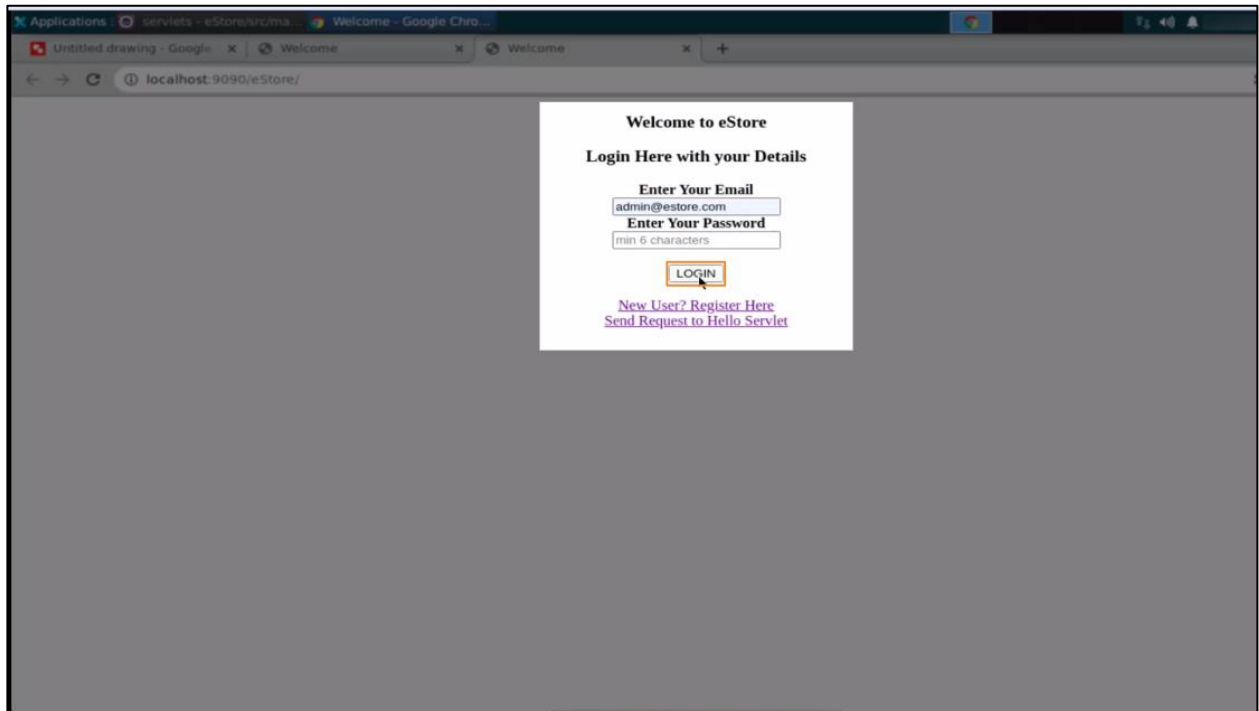
## 5.7 If the email or password is blank, filter the request and the response using the `chain.doFilter()` method



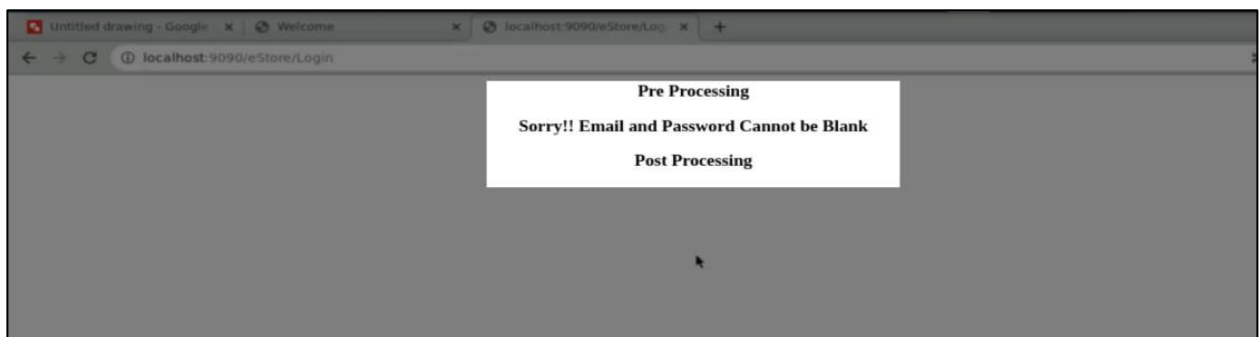
## 5.8 Right-click on eStore, select Run As, and Run on Server



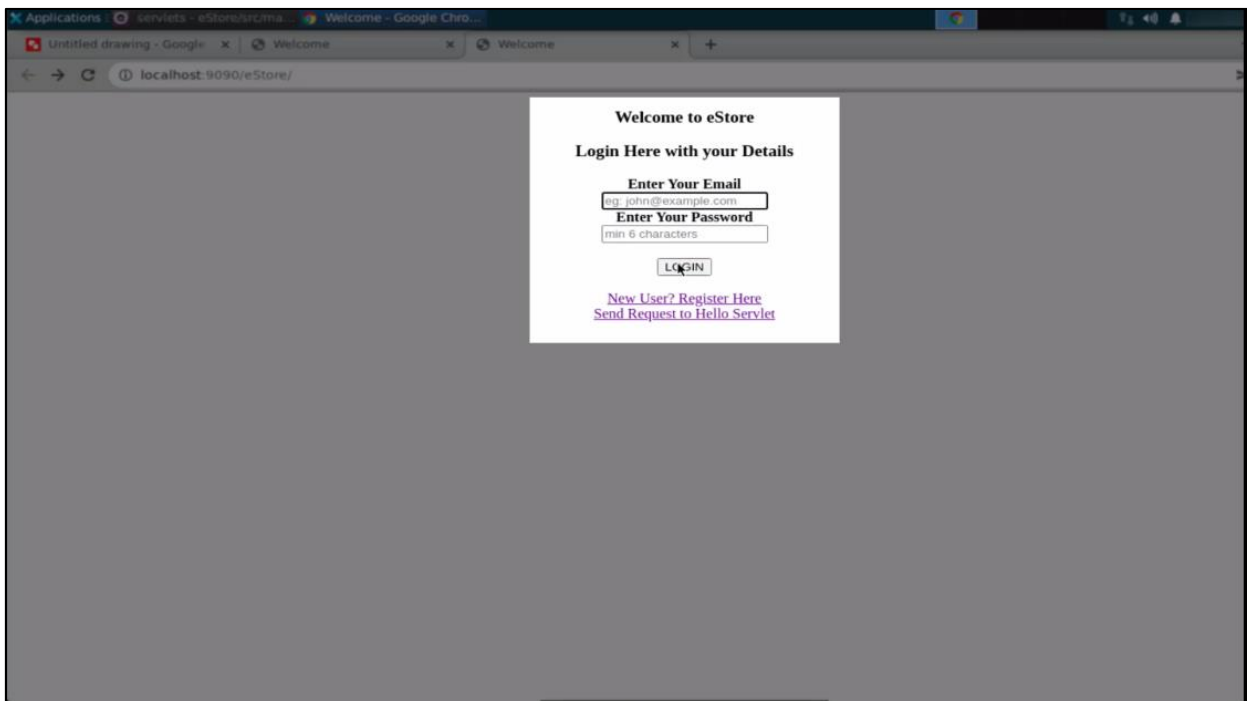
## 5.9 Enter the email and click on **LOGIN**



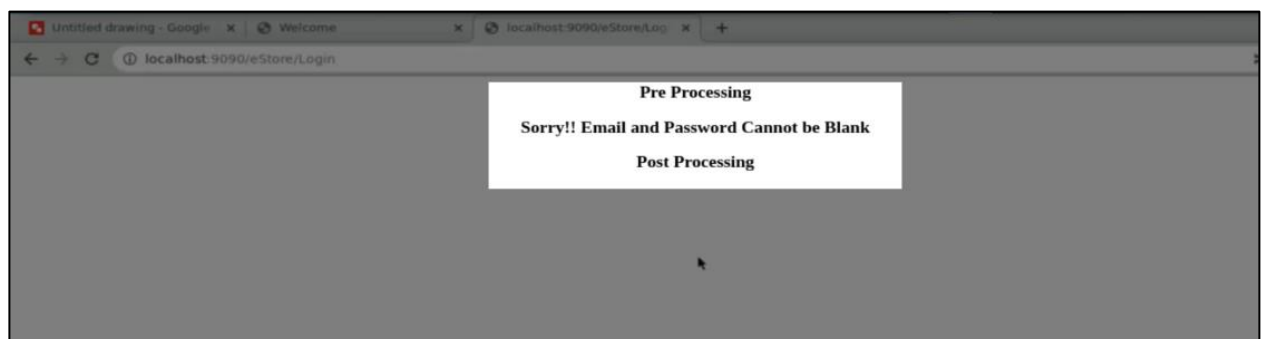
You will see the output: **Sorry!! Email and Password Cannot be Blank**. This means that the filter is working.



## 5.10 Keep both the text boxes empty, and click on **LOGIN**

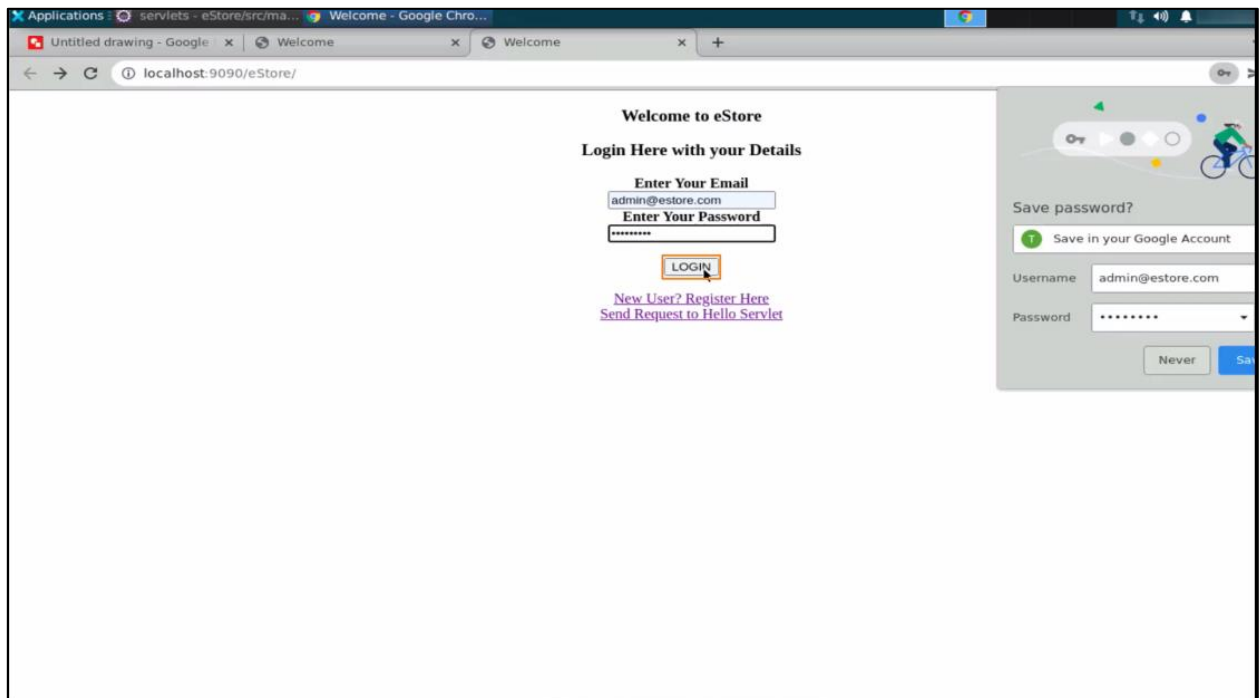


The filter will check for both the username and password, and if either of them is empty, the same output will be displayed.

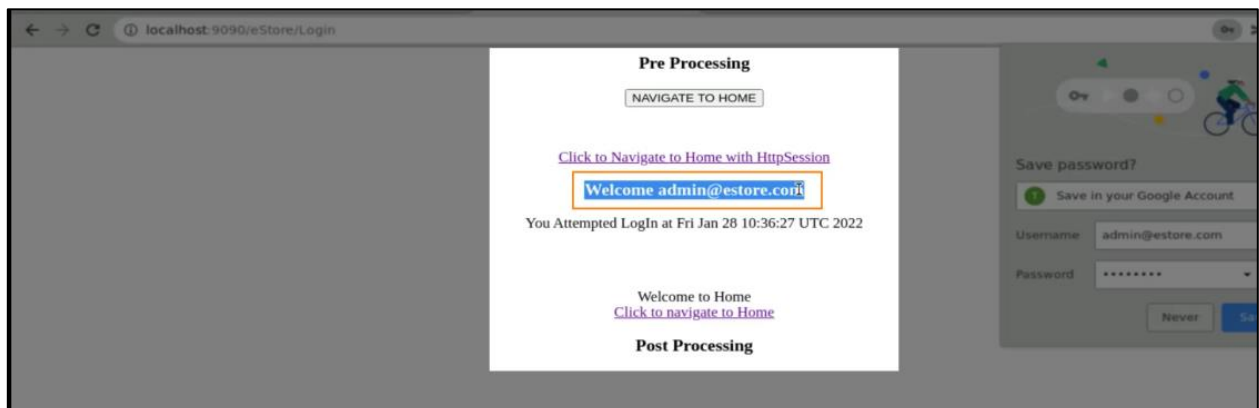


5.11 Enter the username and password, and click on **LOGIN**





**Observation:** You can observe the output as **Welcome admin@estore.com**, indicating that the filter has successfully forwarded the request to the Servlet and returned the response to the client.



Following these steps, you have successfully created a new filter and observed the login filter's response, pre-processing, and post-processing.