Servlet Filters

Step 1: Creating a dynamic web project

- Open Eclipse
- Go the File menu. Choose New->Dynamic Web Project
- Enter the project name as FilterDemo. Click on Next
- Enter nothing in the next screen and click on Next
- Check the checkbox Generate web.xml deployment descriptor and click on Finish
- This will create the project files in the Project Explorer

Step 2: Creating an HTML page

- In the Project Explorer, expand the project FilterDemo
- Expand WebContent. Right click on WebContent. Choose New-> HTML File
- Enter the filename as index.html and click on Finish
- Enter the following code:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Filter Demo</title>
</head>
<body>
	<a href="dashboard?userid=johndoe">Account Dashboard (allow filter)</a><br>
	<a href="profile?userid=johndoe">Account Profile (allow filter)</a><br>
	<a href="dashboard">Account Dashboard (block filter)</a><br>
	<a href="info">Info Page </a><br>
</body>
</html>
```

- Click on the Save icon

Step 3: Creating an AccountProfile servlet

- In the Project Explorer, expand FilterDemo->Java Resources
- Right click on src and choose New->Servlet
- In Class Name, enter AccountProfile and click on Finish
- Enter the following code:

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
* Servlet implementation class AccountProfile
*/
@WebServlet("/AccountProfile")
public class AccountProfile extends HttpServlet {
	private static final long serialVersionUID = 1L;

    /**
* @see HttpServlet#HttpServlet()
*/
    public AccountProfile() {
        super();
        // TODO Auto-generated constructor stub
    }

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
		// TODO Auto-generated method stub
		response.getWriter().write("I am in Account Profile after passing through
LoginFilter");
	}

	/**
	 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
```

```
	 */
	protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
		// TODO Auto-generated method stub
		doGet(request, response);
	}

}
```

Step 4: Creating an AccountDashboard servlet

- In the Project Explorer, expand FilterDemo->Java Resources
- Right click on src and choose New->Servlet
- In Class Name, enter AccountDashboard and click on Finish
- Enter the following code:

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
* Servlet implementation class AccountDashboard
*/
@WebServlet("/AccountDashboard")
public class AccountDashboard extends HttpServlet {
	private static final long serialVersionUID = 1L;

    /**
* @see HttpServlet#HttpServlet()
*/
    public AccountDashboard() {
        super();
        // TODO Auto-generated constructor stub
    }

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse response)
```

```
throws ServletException, IOException {
		// TODO Auto-generated method stub

		response.getWriter().write("I am in Account Dashboard after passing through
LoginFilter");
	}

	/**
	 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
	 */
	protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
		// TODO Auto-generated method stub
		doGet(request, response);
	}

}
```

Step 5: Creating an InfoPage servlet

- In the Project Explorer, expand FilterDemo->Java Resources
- Right click on src and choose New->Servlet
- In Class Name, enter InfoPage and click on Finish
- Enter the following code:

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
* Servlet implementation class InfoPage
*/
@WebServlet("/InfoPage")
public class InfoPage extends HttpServlet {
	private static final long serialVersionUID = 1L;

    /**
```

```
* @see HttpServlet#HttpServlet()
*/
    public InfoPage() {
        super();
        // TODO Auto-generated constructor stub
    }

        /**
         * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
response)
         */
        protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
                // TODO Auto-generated method stub
                response.getWriter().write("I am in InfoPage");
        }

        /**
         * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
         */
        protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
                // TODO Auto-generated method stub
                doGet(request, response);
        }

}
```

Step 6: Creating a LoginFilter filter

- In the Project Explorer, expand FilterDemo->Java Resources
- Right click on src and choose New->Filter
- In Class Name, enter LoginFilter and click on Finish
- Enter the following code:

```
import java.io.IOException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
```

```
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebFilter;

/**
* Servlet Filter implementation class LoginFilter
*/
@WebFilter("/LoginFilter")
public class LoginFilter implements Filter {

    /**
* Default constructor.
*/
    public LoginFilter() {
        // TODO Auto-generated constructor stub
    }

        /**
         * @see Filter#destroy()
         */
        public void destroy() {
                // TODO Auto-generated method stub
        }

        /**
         * @see Filter#doFilter(ServletRequest, ServletResponse, FilterChain)
         */
        public void doFilter(ServletRequest request, ServletResponse response,
FilterChain chain) throws IOException, ServletException {
                // TODO Auto-generated method stub
                // place your code here

                String userId = request.getParameter("userid");

                if( userId != null){
                        chain.doFilter(request, response);
           }


        }

        /**
         * @see Filter#init(FilterConfig)
         */
        public void init(FilterConfig fConfig) throws ServletException {
                // TODO Auto-generated method stub
        }
```

}

Step 7: Configuring web.xml

- In the Project Explorer, expand FilterDemo->WebContent->WEB-INF
- Double click on web.xml to open it in the editor
- Enter the following script:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd" id="WebApp_ID" version="4.0">
  <display-name>FilterDemo</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>


  <filter>
    <filter-name>loginFilter</filter-name>
    <filter-class>LoginFilter</filter-class>
  </filter>
  <filter-mapping>
    <filter-name>loginFilter</filter-name>
    <url-pattern>/dashboard</url-pattern>
    <url-pattern>/profile</url-pattern>
  </filter-mapping>


  <servlet>
    <servlet-name>AccountDashboard</servlet-name>
    <servlet-class>AccountDashboard</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>AccountProfile</servlet-name>
    <servlet-class>AccountProfile</servlet-class>
```

```
 </servlet>
 <servlet>
   <servlet-name>InfoPage</servlet-name>
   <servlet-class>InfoPage</servlet-class>
 </servlet>
 <servlet-mapping>
   <servlet-name>AccountDashboard</servlet-name>
   <url-pattern>/dashboard</url-pattern>
 </servlet-mapping>
  <servlet-mapping>
   <servlet-name>AccountProfile</servlet-name>
   <url-pattern>/profile</url-pattern>
 </servlet-mapping>
  <servlet-mapping>
   <servlet-name>InfoPage</servlet-name>
   <url-pattern>/info</url-pattern>
 </servlet-mapping>
</web-app>
```

Step 8: Checking for servlet-api.jar

- Before building the project, we need toadd servlet-api.jar to the project

- Servlet-api.jar file is already present in your practice lab. (Refer FSD: Lab Guide - Phase 2)

- To add it to the project, follow the below mentioned steps:

  - In the Project Explorer, right click on FilterDemo and choose Properties
  - Select Java Build Path from the options on the left
  - Click on Libraries tab on the right
  - Under ClassPath, expand the node that says Apache Tomcat
  - If there is an existing entry for the servlet-api.jar, then click on

Cancel and exit the window

- If it is not there, then click on Classpath entry and click on Add External JARs button on the right
- From the file list, select the servlet-api.jar file and click on Ok
- Click on Apply and Close

Step 9: Building the project

- From the Project menu at the top, click on Build
- If any compile errors are shown, fix them as required

Step 10: Publishing and starting the project

- If you do not see the Servers tab near the bottom of the IDE, go to Window menu and click on Show View->Servers
- Right click on the Server entry and choose Add and Remove
- Click the Add button to move FilterDemo from the Available list to the Configured list
- Click on Finish
- Right click on the Server entry and click on Publish
- Right click on the Server entry and click on Start
- This will start the server

Step 11: Running the project

- To run the project, open a web browser and type: http://localhost:8080/FilterDemo

Step 12: Pushing the code to your GitHub repositories

Open your command prompt and navigate to the folder where you have created your files

cd <folder path>

Initialize your repository using the following command:

git init

Add all the files to your git repository using the following command:

git add .

Commit the changes using the following command:

git commit . -m “Changes have been committed.”

Push the files to the folder you initially created using the following command:

git push -u origin master