FILTER API’S

Exercise-VII

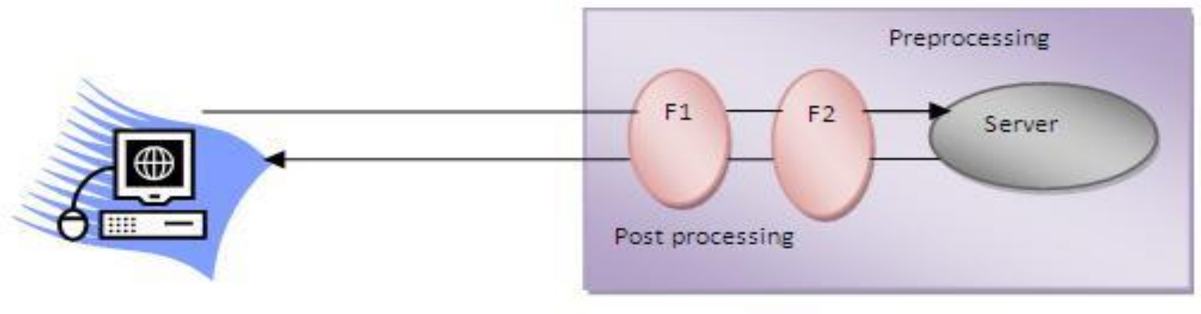
Table of Contents

A **filter** is an object that is invoked at the preprocessing and postprocessing of a request.

It is mainly used to perform filtering tasks such as conversion, logging, compression, encryption and decryption, input validation etc.

The **servlet filter is pluggable**, i.e. its entry is defined in the web.xml file, if we remove the entry of filter from the web.xml file, filter will be removed automatically and we don't need to change the servlet.

So maintenance cost will be less.



Note: Unlike Servlet, One filter doesn't have dependency on another filter.

Usage of Filter

- recording all incoming requests
- logs the IP addresses of the computers from which the requests originate
- conversion
- data compression
- encryption and decryption
- input validation etc.

Advantage of Filter

1. Filter is pluggable.
2. One filter don't have dependency onto another resource.
3. Less Maintenance

Filter, FilterConfig and FilterChain

- ✓ The filter API is defined by the Filter, FilterChain, and FilterConfig interfaces in the **javax.servlet** package. -
- ✓ A filter chain, passed to a filter by the container, provides a mechanism for invoking a series of filters. A filter config contains initialization data.

Filter Interface

- ✓ **Filter** is an interface that defines three methods these are called lifecycle methods of Filter. You can define a filter by implementing the Filter interface.

```
package javax.servlet;

import java.io.IOException;

public interface Filter {

   public void init(FilterConfig filterConfig) throws ServletException;
   public void doFilter(ServletRequest request, ServletResponse response,
                  FilterChain chain)
         throws IOException, ServletException;

   public void destroy();
}
```

The most important method in the Filter interface is the **doFilter** method, which is the heart of the filter. This method usually performs some of the following actions:

- ✓ Examines the request headers.
- ✓ Customizes the request object if it wishes to modify request headers or data or block the request entirely.
- ✓ Customizes the response object if it wishes to modify response headers or data.
- ✓ It invokes the next entity in the filter chain by calling the doFilter method on the chain object
- ✓ Examines response headers after it has invoked the next filter in the chain.

FilterConfig Interface

- ✓ A filter configuration object used by a servlet container to pass information to a filter during initialization.

```
package javax.servlet;
import java.util.Enumeration;
public interface FilterConfig {
   public String getFilterName();
   public ServletContext getServletContext();
   public String getInitParameter(String name);
   public Enumeration<String> getInitParameterNames();


 }
```

FilterChain Interface

```
package javax.servlet;

import java.io.IOException;

public interface FilterChain {

   public void doFilter ( ServletRequest request, ServletResponse response )
      throws IOException, ServletException;

}
```

Servlet Filter Application

Required Files

index.html
ValidationFilter.java → Create Filter for this.
WelcomeServlet.java

Index.html

```
<form action="WelcomeServlet" method="post">
		<table>
			<tr>
				<td>UserName:</td>
				<td><input type="text" name="userName"></td>
			</tr>
			<tr>
				<td>Password</td>
				<td><input type="password" name="password"></td>
			</tr>
			<tr>
				<td><input type="submit" value="Login"></td>
			</tr>
		</table>
	</form>
```

ValidationFilter.java

```
package com.kalibermind.filter;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class ValidationFilter implements Filter {

        @Override
       public void destroy() {

          }
     @Override
      public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
                  throws IOException, ServletException
      {
               response.setContentType("text/html");
               PrintWriter out = response.getWriter();
               String password = request.getParameter("password");

              if("admin".equals(password))
              {
                                  chain.doFilter(request, response);
              }
               else
                {
                          out.println("Invalid UserName and Password");
                           RequestDispatcher dispatcher = request.getRequestDispatcher("index.html");
                           dispatcher.include(request, response);
                  }
             }
            @Override
             public void init(FilterConfig arg0) throws ServletException            { }

  }
```

WelcomeServlet.java

```
package com.kalibermind.servlet;

import java.io.IOException;
import java.io.PrintWriter;

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

        protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
         IOException {
                        response.setContentType("text/html");
                        PrintWriter out = response.getWriter();

                        String userName= request.getParameter("userName");

                        out.println("UserName : " +userName);
           }
  }
```

Web.xml

```
<web-app>
  <display-name>servlet-filter-application</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>

  <filter>
          <filter-name>ValidatorFilter</filter-name>
          <filter-class>com.kalibermind.filter.ValidationFilter</filter-class>
  </filter>
  <filter-mapping>
          <filter-name>ValidatorFilter</filter-name>
          <url-pattern>/WelcomeServlet</url-pattern>
  </filter-mapping>
</web-app>
```

Filter-Request-Response Application-2

In this application we do some changes, add another **ValidationFilter1.java** file and change inside the web.xml file .and all remaining files are same

Required Files:

- index.html
- ValidationFilter1.java
- ValidationFilter2.java
- WelcomeServlet.java
- Web.xml

ValidationFilter1.java

```
package com.sgs.filter;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class ValidationFilter1 implements Filter {

         @Override
         public void destroy() {
                }
        @Override
        public void doFilter(ServletRequest request, ServletResponse response, FilterChain
                chain) throws IOException, ServletException {

                   response.setContentType("text/html");
                   PrintWriter out = response.getWriter();
                  String password = request.getParameter("password");
                  System.out.println("Processing is done by Filter-1");

                  if(password.length() > 3)

                  {
                     Chain.doFilter(request, response);

                  }
```

```
            else{
                out.println("Password should be Greater 3.");
                RequestDispatcher dispatcher = request.getRequestDispatcher("index.html");
                dispatcher.include(request, response);
            }
                System.out.println("Processing is done by Filter1");
        }

        @Override
         public void init(FilterConfig arg0) throws ServletException {
                                    }
 }
```

Here **ValidationFilter2.java** is same file as it is **in ValidationFilter.java** in the previous Application

Web.xml

```
<web-app>
  <display-name>servlet-filter-application</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>

  <filter>
      <filter-name>ValidatorFilter1</filter-name>
      <filter-class>com.kalibermind.filter.ValidationFilter1</filter-class>
  </filter>
  <filter-mapping>
      <filter-name>ValidatorFilter1</filter-name>
      <url-pattern>/WelcomeServlet</url-pattern>
  </filter-mapping>

  <filter>
      <filter-name>ValidatorFilter2</filter-name>
      <filter-class>com.kalibermind.filter.ValidationFilter2</filter-class>
  </filter>
  <filter-mapping>
      <filter-name>ValidatorFilter2</filter-name>
      <url-pattern>/WelcomeServlet</url-pattern>
  </filter-mapping>

</web-app>
```