Servlet Filter

1. [Filter](https://www.javatpoint.com/servlet-filter)
2. [Usage of Filter](https://www.javatpoint.com/servlet-filter#filterusage)
3. [Advantage of Filter](https://www.javatpoint.com/servlet-filter#filteradvantage)
4. [Filter API](https://www.javatpoint.com/servlet-filter#filterapi)
   1. [Filter interface](https://www.javatpoint.com/servlet-filter#filterinterface)
   2. [FilterChain interface](https://www.javatpoint.com/servlet-filter#filterchain)
   3. [FilterConfig interface](https://www.javatpoint.com/servlet-filter#filterconfig)
5. [Simple Example of Filter](https://www.javatpoint.com/servlet-filter#filterex)

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 API

Like servlet filter have its own API. The javax.servlet package contains the three interfaces of Filter API.

1. Filter
2. FilterChain
3. FilterConfig

1) Filter interface

For creating any filter, you must implement the Filter interface. Filter interface provides the life cycle methods for a filter.

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void init(FilterConfig config) | init() method is invoked only once. It is used to initialize the filter. |
| public void doFilter(HttpServletRequest request,HttpServletResponse response, FilterChain chain) | doFilter() method is invoked every time when user request to any resource, to which the filter is mapped.It is used to perform filtering tasks. |
| public void destroy() | This is invoked only once when filter is taken out of the service. |

2) FilterChain interface

The object of FilterChain is responsible to invoke the next filter or resource in the chain.This object is passed in the doFilter method of Filter interface.The FilterChain interface contains only one method:

1. **public void doFilter(HttpServletRequest request, HttpServletResponse response):** it passes the control to the next filter or resource.

How to define Filter

We can define filter same as servlet. Let's see the elements of filter and filter-mapping.

1. <web-app>
3. <filter>
4. <filter-name>...</filter-name>
5. <filter-**class**>...</filter-**class**>
6. </filter>
8. <filter-mapping>
9. <filter-name>...</filter-name>
10. <url-pattern>...</url-pattern>
11. </filter-mapping>
13. </web-app>

For mapping filter we can use, either url-pattern or servlet-name. The url-pattern elements has an advantage over servlet-name element i.e. it can be applied on servlet, JSP or HTML.

Simple Example of Filter

In this example, we are simply displaying information that filter is invoked automatically after the post processing of the request.

index.html

1. <a href="servlet1">click here</a>

MyFilter.java

1. **import** java.io.IOException;
2. **import** java.io.PrintWriter;
4. **import** javax.servlet.\*;
6. **public** **class** MyFilter **implements** Filter{
8. **public** **void** init(FilterConfig arg0) **throws** ServletException {}
10. **public** **void** doFilter(ServletRequest req, ServletResponse resp,
11. FilterChain chain) **throws** IOException, ServletException {
13. PrintWriter out=resp.getWriter();
14. out.print("filter is invoked before");
16. chain.doFilter(req, resp);//sends request to next resource
18. out.print("filter is invoked after");
19. }
20. **public** **void** destroy() {}
21. }

HelloServlet.java

1. **import** java.io.IOException;
2. **import** java.io.PrintWriter;
4. **import** javax.servlet.ServletException;
5. **import** javax.servlet.http.\*;
7. **public** **class** HelloServlet **extends** HttpServlet {
8. **public** **void** doGet(HttpServletRequest request, HttpServletResponse response)
9. **throws** ServletException, IOException {
11. response.setContentType("text/html");
12. PrintWriter out = response.getWriter();
14. out.print("<br>welcome to servlet<br>");
16. }
18. }

**web.xml**

|  |
| --- |
| For defining the filter, filter element of web-app must be defined just like servlet. |

1. <web-app>
3. <servlet>
4. <servlet-name>s1</servlet-name>
5. <servlet-**class**>HelloServlet</servlet-**class**>
6. </servlet>
8. <servlet-mapping>
9. <servlet-name>s1</servlet-name>
10. <url-pattern>/servlet1</url-pattern>
11. </servlet-mapping>
13. <filter>
14. <filter-name>f1</filter-name>
15. <filter-**class**>MyFilter</filter-**class**>
16. </filter>
18. <filter-mapping>
19. <filter-name>f1</filter-name>
20. <url-pattern>/servlet1</url-pattern>
21. </filter-mapping>

24. </web-app>

Authentication Filter

We can perform authentication in filter. Here, we are going to check to password given by the user in filter class, if given password is admin, it will forward the request to the WelcomeAdmin servlet otherwise it will display error message.

Example of authenticating user using filter

Let's see the simple example of authenticating user using filter.

Here, we have created 4 files:

* index.html
* MyFilter.java
* AdminServlet.java
* web.xml

**index.html**

1. <form action="servlet1">
2. Name:<input type="text" name="name"/><br/>
3. Password:<input type="password" name="password"/><br/>
5. <input type="submit" value="login">
7. </form>

**MyFilter.java**

1. **import** java.io.IOException;
2. **import** java.io.PrintWriter;
3. **import** javax.servlet.\*;
5. **public** **class** MyFilter **implements** Filter{
7. **public** **void** init(FilterConfig arg0) **throws** ServletException {}
9. **public** **void** doFilter(ServletRequest req, ServletResponse resp,
10. FilterChain chain) **throws** IOException, ServletException {
12. PrintWriter out=resp.getWriter();
14. String password=req.getParameter("password");
15. **if**(password.equals("admin")){
16. chain.doFilter(req, resp);//sends request to next resource
17. }
18. **else**{
19. out.print("username or password error!");
20. RequestDispatcher rd=req.getRequestDispatcher("index.html");
21. rd.include(req, resp);
22. }
24. }
25. **public** **void** destroy() {}
27. }

**AdminServlet.java**

1. **import** java.io.IOException;
2. **import** java.io.PrintWriter;
4. **import** javax.servlet.ServletException;
5. **import** javax.servlet.http.\*;
7. **public** **class** AdminServlet **extends** HttpServlet {
8. **public** **void** doGet(HttpServletRequest request, HttpServletResponse response)
9. **throws** ServletException, IOException {
11. response.setContentType("text/html");
12. PrintWriter out = response.getWriter();
14. out.print("welcome ADMIN");
15. out.close();
16. }
17. }

**web.xml**

1. <web-app>
2. <servlet>
3. <servlet-name>AdminServlet</servlet-name>
4. <servlet-**class**>AdminServlet</servlet-**class**>
5. </servlet>
7. <servlet-mapping>
8. <servlet-name>AdminServlet</servlet-name>
9. <url-pattern>/servlet1</url-pattern>
10. </servlet-mapping>
12. <filter>
13. <filter-name>f1</filter-name>
14. <filter-**class**>MyFilter</filter-**class**>
15. </filter>
16. <filter-mapping>
17. <filter-name>f1</filter-name>
18. <url-pattern>/servlet1</url-pattern>
19. </filter-mapping>
21. </web-app>

==

# FilterConfig

An object of FilterConfig is created by the web container. This object can be used to get the configuration information from the web.xml file.

## Methods of FilterConfig interface

There are following 4 methods in the FilterConfig interface.

1. **public void init(FilterConfig config):** init() method is invoked only once it is used to initialize the filter.
2. **public String getInitParameter(String parameterName):** Returns the parameter value for the specified parameter name.
3. **public java.util.Enumeration getInitParameterNames():** Returns an enumeration containing all the parameter names.
4. **public ServletContext getServletContext():** Returns the ServletContext object.

### Example of FilterConfig

In this example, if you change the param-value to no, request will be forwarded to the servlet otherwise filter will create the response with the message: this page is underprocessing. Let's see the simple example of FilterConfig. Here, we have created 4 files:

* index.html
* MyFilter.java
* HelloServlet.java
* web.xml

**index.html**

1. <a href="servlet1">click here</a>

**MyFilter.java**

1. **import** java.io.IOException;
2. **import** java.io.PrintWriter;
4. **import** javax.servlet.\*;
6. **public** **class** MyFilter **implements** Filter{
7. FilterConfig config;
9. **public** **void** init(FilterConfig config) **throws** ServletException {
10. **this**.config=config;
11. }
13. **public** **void** doFilter(ServletRequest req, ServletResponse resp,
14. FilterChain chain) **throws** IOException, ServletException {
16. PrintWriter out=resp.getWriter();
18. String s=config.getInitParameter("construction");
20. **if**(s.equals("yes")){
21. out.print("This page is under construction");
22. }
23. **else**{
24. chain.doFilter(req, resp);//sends request to next resource
25. }
27. }
28. **public** **void** destroy() {}
29. }

**HelloServlet.java**

1. **import** java.io.IOException;
2. **import** java.io.PrintWriter;
4. **import** javax.servlet.ServletException;
5. **import** javax.servlet.http.\*;
7. **public** **class** HelloServlet **extends** HttpServlet {
8. **public** **void** doGet(HttpServletRequest request, HttpServletResponse response)
9. **throws** ServletException, IOException {
11. response.setContentType("text/html");
12. PrintWriter out = response.getWriter();
14. out.print("<br>welcome to servlet<br>");
16. }
18. }

**web.xml**

1. <web-app>
3. <servlet>
4. <servlet-name>HelloServlet</servlet-name>
5. <servlet-**class**>HelloServlet</servlet-**class**>
6. </servlet>
8. <servlet-mapping>
9. <servlet-name>HelloServlet</servlet-name>
10. <url-pattern>/servlet1</url-pattern>
11. </servlet-mapping>
13. <filter>
14. <filter-name>f1</filter-name>
15. <filter-**class**>MyFilter</filter-**class**>
16. <init-param>
17. <param-name>construction</param-name>
18. <param-value>no</param-value>
19. </init-param>
20. </filter>
21. <filter-mapping>
22. <filter-name>f1</filter-name>
23. <url-pattern>/servlet1</url-pattern>
24. </filter-mapping>

27. </web-app>