Java Servlet Filter Example Tutorial

APRIL 4, 2018 BY PANKAJ — 64 COMMENTS

Java Servlet Filter is used to intercept request and do some pre-processing and can be used to intercept response and do post-processing before sending to client in web application. This is the fourth article in the series of Web Applications Tutorial, you might want to check out earlier articles too.

1. Java Web Application
2. Java Servlet Tutorial
3. Servlet Session Management

Servlet Filter

In this article, we will lean about the Servlet Filter in Java. We will look into various usage of servlet filter, how can we create filter and learn it's usage with a simple web application.

1. Why do we have Servlet Filter?

In the last article, we learned how we can manage session in web application and if we want to make sure that a resource is accessible only when user session is valid, we can achieve this using servlet session attributes. The approach is simple but if we have a lot of servlets and jsps, then it will become hard to maintain because of redundant code. If we want to change the attribute name in future, we will have to change all the places where we have session authentication.

That's why we have servlet filter. Servlet Filters are **pluggable** java components that we can use to intercept and process requests *before* they are sent to servlets and response *after* servlet code is finished and before container sends the response back to the client.

Some common tasks that we can do with servlet filters are:

- Logging request parameters to log files.
- Authentication and autherization of request for resources.
- Formatting of request body or header before sending it to servlet.
- Compressing the response data sent to the client.
- Alter response by adding some cookies, header information etc.

As I mentioned earlier, **servlet filters are pluggable** and configured in deployment descriptor (web.xml) file. Servlets and filters both are unaware of each other and we can add or remove a servlet filter just by editing web.xml.

We can have multiple filters for a single resource and we can create a chain of filters for a single resource in web.xml. We can create a Servlet Filter by implementing `javax.servlet.Filter` interface.

2. Servlet Filter interface

Servlet Filter interface is similar to Servlet interface and we need to implement it to create our own servlet filter. Servlet Filter interface contains lifecycle methods of a Filter and it's managed by servlet container.

Servlet Filter interface lifecycle methods are:

1. **void init(FilterConfig paramFilterConfig)** – When container initializes the Filter, this is the method that gets invoked. This method is called only once in the lifecycle of filter and we should initialize any resources in this method. **FilterConfig** is used by container to provide init parameters and servlet context object to the Filter. We can throw ServletException in this method.
2. **doFilter(ServletRequest paramServletRequest, ServletResponse paramServletResponse, FilterChain paramFilterChain)** – This is the method invoked every time by container when it has to apply filter to a resource. Container provides request and response object references to filter as argument. **FilterChain** is used to invoke the next filter in the chain. This is a great example of **Chain of Responsibility Pattern**.
3. **void destroy()** – When container offloads the Filter instance, it invokes the destroy() method. This is the method where we can close any resources opened by filter. This method is called only once in the lifetime of filter.

3. Servlet WebFilter annotation

`javax.servlet.annotation.WebFilter` was introduced in Servlet 3.0 and we can use this annotation to declare a servlet filter. We can use this annotation to define init parameters, filter name and description, servlets, url patterns and dispatcher types to apply the filter. If you make frequent changes to the filter configurations, its better to use web.xml because that will not require you to recompile the filter class.

Read: **Java Annotations Tutorial**

4. Servlet Filter configuration in web.xml

We can declare a servlet filter in web.xml like below.

We can map a Filter to servlet classes or url-patterns like below.

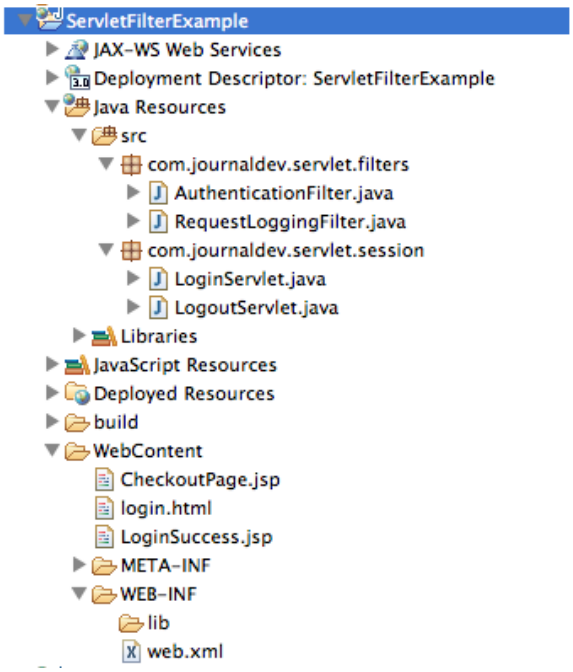
Note: While creating the filter chain for a servlet, container first processes the url-patterns and then servlet-names, so if you have to make sure that filters are getting executed in a particular order, give extra attention while defining the filter mapping.

Servlet Filters are generally used for client requests but sometimes we want to apply filters with **RequestDispatcher** also, we can use dispatcher element in this case, the possible values are REQUEST, FORWARD, INCLUDE, ERROR and ASYNC. If no dispatcher is defined then it's applied only to client requests.

5. Servlet Filter Example for Logging and session validation

In our **servlet filter example**, we will create filters to log request cookies and parameters and validate session to all the resources except static HTMLs and LoginServlet because it will not have a session.

We will create a dynamic web project **ServletFilterExample** whose project structure will look like below image.



login.html is the entry point of our application where user will provide login id and password for authentication.

login.html code:

LoginServlet is used to authenticate the request from client for login.

When client is authenticated, it's forwarded to LoginSuccess.jsp

LoginSuccess.jsp code:

Notice that there is no session validation logic in the above JSP. It contains link to another JSP page, CheckoutPage.jsp.

CheckoutPage.jsp code: