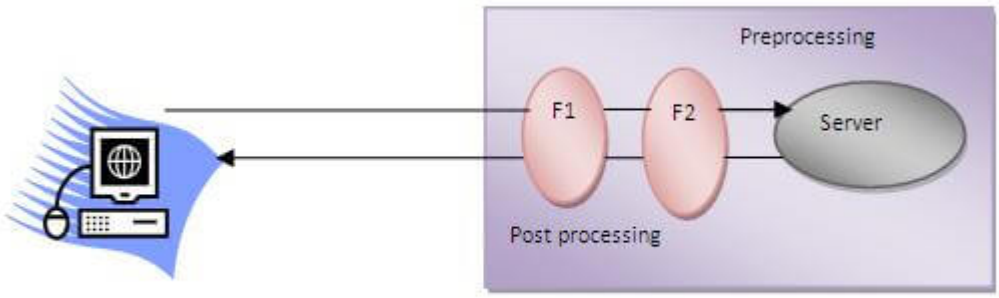
Servlet Filter

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.

```
<web-app>

<filter>
<filter-name>...</filter-name>
<filter-class>...</filter-class>
</filter>

<filter-mapping>
<filter-name>...</filter-name>
<url-pattern>...</url-pattern>
</filter-mapping>

</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

```
<a href="servlet1">click here</a>
```

MyFilter.java

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

import javax.servlet.*;
```

```
public class MyFilter implements Filter{

public void init(FilterConfig arg0) throws ServletException {}

public void doFilter(ServletRequest req, ServletResponse resp,
    FilterChain chain) throws IOException, ServletException {

    PrintWriter out=resp.getWriter();
    out.print("filter is invoked before");

    chain.doFilter(req, resp);//sends request to next resource

    out.print("filter is invoked after");
    }
    public void destroy() {}
}
```

HelloServlet.java

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

import javax.servlet.ServletException;
import javax.servlet.http.*;

public class HelloServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        out.print("<br>welcome to servlet<br>");

    }
```

```
}
```

web.xml

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

```
<web-app>

<servlet>
<servlet-name>s1</servlet-name>
<servlet-class>HelloServlet</servlet-class>
</servlet>

<servlet-mapping>
<servlet-name>s1</servlet-name>
<url-pattern>/servlet1</url-pattern>
</servlet-mapping>

<filter>
<filter-name>f1</filter-name>
<filter-class>MyFilter</filter-class>
</filter>

<filter-mapping>
<filter-name>f1</filter-name>
<url-pattern>/servlet1</url-pattern>
</filter-mapping>



</web-app>
```

download this example (developed using Myeclipse IDE)
download this example (developed using Eclipse IDE)
download this example (developed using Netbeans IDE)

⇧

<<prev next>>

Please Share

Learn Latest Tutorials

DS

PDFBox

Ant

JMeter

jBPM