

제17장 필터(filter)



1. 필터의 개요

❖ 필터(filter)

- 클라이언트와 서버 사이에서 request와 response 객체를 먼저 받아 사전/사후 작업 등 공통적으로 필요한 부분을 처리하는 것
- 필터는 클라이언트의 요청이 웹 서버의 서블릿, JSP, HTML 페이지 같은 정적 리소스에 도달하기 전과, 반대로 정적 리소스에서 클라이언트로 응답하기 전에 필요한 **전처리를 가능**하게 함.
- 필터는 HTTP 요청과 응답을 변경할 수 있는 코드로 재사용이 가능합니다. 한편 클라이언트와 정적 리소스 사이에 여러 개의 필터로 이루어진 필터 체인을 제공하기도 함

개념적으로 필터는 이름에서도 알 수 있듯 정수기 필터, 에어컨 필터, 담배의 필터와 같이 무언가를 걸러내는 필터를 말한다. JSP/Servlet 에서도 필터의 개념은 같지만 걸러내는 대상만 다를 뿐이다.

Jsp/Servlet에서의 필터는 서블릿 2.3 부터 추가된 기능으로 클라이언트(브라우저)가 서버로 요청을 보내올 때 요청이 서블릿으로 전달되기 전, 후에 필터링하기 위한 기술을 의미한다.

Jsp/Servlet 스펙에서 필터는 사용하기 쉽도록 javax.servlet.Filter 인터페이스로 제공하고 있으며 이것을 구현하고 web.xml에 등록하기만 하면 간단하게 사용할 수 있다.

인터페이스의 구현 메서드는 서블릿의 service() 메서드나 doGet() doPost() 메서드와 유사하기때문에 매우 익숙하게 사용할 수 있다.

1. 필터의 개요

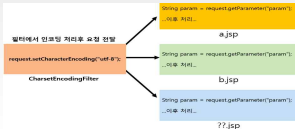


필터의 필요성

- 필터가 필요한 이유는 웹 컴포넌트(Servlet, Jsp)에서 공통적으로 처리해야 할 일들을 필터를 통해 처리할 수 있기 때문이다.
- 예를 요청 데이터의 본문을 UTF-8로 인코딩 하는 부분은 거의 대부분의 웹 컴포넌트에서 처리하는 로직이다.

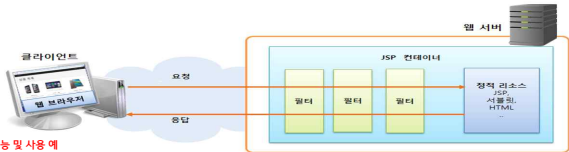
```
request.setCharacterEncoding("utf-8");  
String param = request.getParameter("param");
```

그러나 웹 어플리케이션에 수십 수백개의 웹 컴포넌트를 개발해야 한다면 이런 한글 인코딩 처리 코드는 수백여 개의 중복이 발생할 것이다. 매번 웹 컴포넌트를 개발할 때마다 인코딩 처리 로직을 넣어주는 것도 문제지만, 만약 웹 어플리케이션을 수출하거나 정책이 바뀌어 인코딩 처리를 다른 것으로 바꿔주어야 한다면 중복되는 모든 코드를 각각 바꿔주어야 하므로 유지보수 측면에서도 좋지 못하다. 하여, 필터는 이런 상황에서 인코딩 처리 부분을 분리하여 개발을 빠르고 유지보수 측면에서 나중에 코드를 변경할 때도 쉽게 수정이 용이하게 해준다. 인코딩 정책이 바뀐다 하더라도 필터부분만 수정해주면 되니깐.



1. 필터의 개요

❖ 필터(filter)의 기능과 예



필터의 기능 및 사용 예
필터의 기능은 다음과 같다.

1. 서블릿이 호출되기 전에 요청(request)을 가로채어 조작한다.
2. 서블릿이 호출되기 전에 요청을 가로채어 점검할 수 있다.
3. 서블릿이 호출되기 전에 HTTP 요청의 헤더를 조작할 수 있다.
4. 서블릿이 호출된 이후 응답(response)을 출력하기 전에 가로채어 조작한다.
5. 서블릿이 호출된 이후 응답 헤더를 조작할 수 있다.

그에 따른 사용 예는 다음과 같다.

1. 로그인여부나 권한 검사와 같은 인증 기능
2. 요청이나 응답에 대한 로그(기록) 기능
3. 오류 처리 기능
4. 데이터 압축이나 변환 기능
5. 인코딩 처리 기능 등..

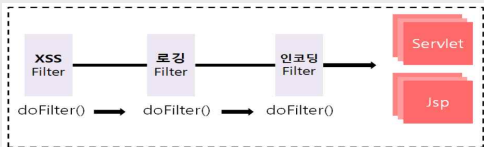
필터는 객체의 형태로 존재하며 클라이언트로부터 오는 요청(request)과 최종 자원(서블릿/JSP/기타 문서) 사이에 위치하여 클라이언트의 요청 정보를 알맞게 변경할 수 있으며, 또한 필터는 최종 자원과 클라이언트로 가는 응답(response) 사이에 위치하여 최종 자원의 요청 결과를 알맞게 변경할 수 있다.

1. 필터의 개요



필터 체인

- 여러 개의 필터가 모여 하나의 체인(chain: 또는 사슬)을 형성되어 있는 것을 칭한다.



1. 여러 개의 필터가 모여서 하나의 체인을 형성할 때 첫 번째 필터가 변경하는 요청 정보는 클라이언트의 요청 정보가 된다.
2. 체인의 두번째 필터가 변경하는 요청 정보는 첫 번째 필터를 통해서 변경된 요청 정보가 된다.
3. 응답 정보의 경우도 요청 정보와 비슷한 과정을 거치며 차이점이 있다면 필터의 적용 순서가 요청 때와는 반대라는 것이다.
4. 필터의 이러한 기능은 사용자 인증이나 권한 체크와 같은 곳에서 사용할 수 있다.

2. Filter 인터페이스의 구현 클래스

❖ Filter 인터페이스

- 필터 기능을 구현하는 데 핵심적인 역할을 합니다. 클라이언트와 서버의 리소스 사이에 위치한 필터의 기능을 제공하기 위해 자바 클래스로 구현해야 함

```
import javax.servlet.Filter;

public class 클래스 이름 implements Filter
{
    ... / 구현 ...
}
```

Filter 인터페이스 메소드의 종류

메소드	설명
init(...)	필터 인스턴스의 초기화 메소드입니다.
doFilter(...)	필터 기능을 작성하는 메소드입니다. 실질적인 필터 역할을 하는 메서드
destroy()	필터 인스턴스의 종료 전에 호출되는 메소드입니다.

2. Filter 인터페이스의 구현 클래스

❖ 2.1 init() 메소드

- JSP 컨테이너가 필터를 초기화할 때 호출되는 메소드

```
public void init(FilterConfig filterConfig) throws ServletException
```

- **init() 메소드는 JSP 컨테이너 내에서 초기화 작업을 수행할 필터 인스턴스를 생성한 후 한 번만 호출**
- **init() 메소드는 JSP 컨테이너에 의해 호출되어 필터의 서비스가 시작되고 있음을 나타냄**

2. Filter 인터페이스의 구현 클래스

FilterConfig 인터페이스 메소드의 종류

메소드	반환 유형	설명
getFilterName()	String	web.xml 파일에 설정된 필터 이름을 반환합니다.
getInitParameter(String name)	String	web.xml 파일에 설정된 매개변수에 대한 매개변수 값을 반환합니다. 초기화 매개변수가 존재하지 않으면 null을 반환합니다.
getInitParameterNames()	Enumeration<String>	web.xml 파일에 설정된 모든 매개변수 이름을 포함하는 Enumeration 객체 타입을 반환합니다. 초기화 매개변수가 존재하지 않으면 비어 있는 Enumeration을 반환합니다.
getServletContext()	ServletContext	ServletContext 객체를 반환합니다.

[init() 메소드 사용 예]

```
@Override
public void init(FilterConfig filterConfig) throws ServletException{
    System.out.println("필터 초기화...");
}
```


2. Filter 인터페이스의 구현 클래스

❖ 2.2 doFilter() 메소드

- JSP 컨테이너가 필터를 리소스에 적용할 때마다 호출되는 메소드
- init() 메소드 후에 호출되며, 필터가 어떤 기능을 수행할 필요가 있을 때마다 호출

```
public void doFilter(ServletRequest request,
                    ServletResponse response,
                    FilterChain chain)
    throws java.io.IOException, ServletException
```

- 첫 번째 매개변수 ServletRequest 객체는 체인을 따라 전달하는 요청이고,
- 두 번째 매개변수 ServletResponse 객체는 체인을 따라 전달할 응답
- 세 번째 매개변수 FilterChain 객체는 체인에서 다음 필터를 호출하는 데 사용
 - 만약 호출 필터가 체인의 마지막 필터이면 체인의 끝에서 리소스를 호출

request파라미터를 이용하여 요청처리의 필터 작업을 수행하고 체인이 있다면 다음 체인의 필터를 처리한다.

2. Filter 인터페이스의 구현 클래스

FilterChain 인터페이스 메소드의 종류

메소드	반환 유형	설명
doFilter(ServletRequest request, ServletResponse response)	void	체인인 다음 필터 또는 리소스로 제어를 전달합니다.

[doFilter() 메소드 사용 예]

```
@Override
public void doFilter(ServletRequest request, ServletResponse response,
    FilterChain filterChain) throws IOException, ServletException{
    System.out.println("JSP 처리 전 필터 수행...");
    filterChain.doFilter(request, response);
    System.out.println("JSP 처리 후 필터 수행...");
}
```

reponse객체를 이용하여 클라이언트에게 응답의 필터링 작업을 수행

2. Filter 인터페이스의 구현 클래스

❖ 2.3 destroy() 메소드

- 필터 인스턴스를 종료하기 전에 호출하는 메소드

```
public void destroy()
```

- JSP 컨테이너가 필터 인스턴스를 삭제하기 전에 청소 작업을 수행하는 데 사용되며, 이는 필터로 열린 리소스를 모두 닫을 수 있는 방법
- destroy() 메소드는 필터의 수명 동안 한 번만 호출

[destroy() 메소드 사용 예]

```
@Override
public void destroy() {
    System.out.println("필터 해제...");
}
```



3. web.xml 파일의 필터 구성

❖ web.xml 파일에 필터를 설정

- 필터를 사용하려면 어떤 필터가 어떤 리소스에 대해 적용되는지 JSP 컨테이너에 알려주어야 함.
- <filter>와 <filter-mapping> 요소를 사용
- web.xml 파일에 여러 개의 필터가 설정되어 있으면 선언된 순서대로 실행

```
<filter>
  <filter-name>...</filter-name>
  <filter-class>...</filter-class>
  [<init-param>
    <param-name>...</param-name>
    <param-value>...</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>...</filter-name>
  <url-pattern>...</url-pattern>
</filter-mapping>
```

<filter>를 구성하는 하위 요소

요소	설명
<filter-name>	필터 이름을 설정합니다.
<filter-class>	자바 클래스 이름을 설정합니다.
<init-param>	매개변수와 값을 설정합니다.

<filter-mapping>을 구성하는 하위 요소

요소	설명
<filter-name>	필터 이름을 설정합니다.
<url-pattern>	URL 패턴을 설정합니다.



3. web.xml 파일의 필터 구성

❖ <filter> 요소

- <filter> 요소는 웹 애플리케이션에서 자바 필터와 매개변수를 설정하는 데 사용

```
<filter>
  <filter-name>필터 이름</filter-name>
  <filter-class>클래스 이름</filter-class>
  [<init-param>
    <param-name>매개변수 이름</param-name>
    <param-value>매개변수 값</param-value>
  </init-param>
</filter>
```

3. web.xml 파일의 필터 구성

❖ <init-param> 요소

- 설정된 매개변수와 값을 자바 또는 JSP 코드에서 접근

```
String value = getServletConfig().getInitParameter("매개변수 이름");
```

[<filter> 요소 사용 예: 필터 이름 myFilter와 클래스 이름 LoggingFilter 설정]

```
<filter>  
  <filter-name>myFilter</filter-name>  
  <filter-class>ch12.com.filter.LoggingFilter</filter-class>  
</filter>
```

3. web.xml 파일의 필터 구성

[<filter> 요소 사용 예: 매개변수 param과 값 admin 설정]

```
<filter>
  <filter-name>MyFilter</filter-name>
  <filter-class>ch12.com.filter.LoggingFilter</filter-class>
  <init-param>
    <param-name>param</param-name>
    <param-value>admin</param-value>
  </init-param>
</filter>
```

- 위의 예에서 <init-param> 요소에 설정된 매개변수와 값을 자바 클래스에서 접근하려면 다음과 같이 작성합니다.

```
String value = getServletConfig().getInitParameter("param");
```



3. web.xml 파일의 필터 구성

❖ <filter-mapping> 요소

- 특정 리소스에 대해 어떤 필터를 사용할지 설정하는 데 사용

```
<filter-mapping>
  <filter-name>필터 이름</filter-name>
  <url-pattern>요청 URL 패턴</url-pattern>
</filter-mapping>
```

[<filter-mapping> 요소 사용 예: URL 패턴을 /*로 설정]

```
<filter-mapping>
  <filter-name>MyFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

“.”으로 시작하는 url-pattern은 확장자에 대한 매핑을 할 때 사용된다.

때로는 <url-pattern> 태그를 사용하지 않고 대신 <servlet-name> 태그를 사용함으로써 특정 서블릿에 대한 요청에 대해서 필터를 적용할 수도 있다

[<filter-mapping> 요소 사용 예: URL 패턴을 /ch12/filter.jsp로 설정]

```
<filter-mapping>
  <filter-name>MyFilter</filter-name>
  <url-pattern>/ch12/filter.jsp</url-pattern>
</filter-mapping>
```

```
<filter-mapping>
  <filter-name>AuthCheckFilter</filter-name>
  <servlet-name>FileDownload</servlet-name>
</filter-mapping>
```


3. web.xml 파일의 필터 구성

폼 페이지에서 전송된 요청 파라미터를 필터로 처리하기

AuthenFilter.java

```
01 package ch12.com.filter;
02
03 import java.io.IOException;
04 import java.io.PrintWriter;
05 import javax.servlet.Filter;
06 import javax.servlet.FilterChain;
07 import javax.servlet.FilterConfig;
08 import javax.servlet.ServletException;
09 import javax.servlet.ServletRequest;
10 import javax.servlet.ServletResponse;
11
12 public class AuthenFilter implements Filter {
13     @Override
14     public void init(FilterConfig filterConfig) throws ServletException {
15         System.out.println("Filter01 초기화...");
16     }
17 }
```

3. web.xml 파일의 필터 구성

```
18     @Override
19     public void doFilter(ServletRequest request, ServletResponse response,
20         FilterChain filterChain) throws IOException, ServletException {
21         System.out.println("Filter01.jsp 수행...");
22         String name = request.getParameter("name");
23
24         if (name == null || name.equals("")) {
25             response.setCharacterEncoding("UTF-8");
26             response.setContentType("text/html; charset=UTF-8");
27             PrintWriter writer = response.getWriter();
28             String message = "입력된 name 값은 null입니다.";
29             writer.println(message);
30             return;
31         }
32         filterChain.doFilter(request, response);
33     }
34
35     @Override
36     public void destroy() {
37         System.out.println("Filter01 해제...");
38     }
```

3. web.xml 파일의 필터 구성

web.xml

```
01 <web-app>
02     ...{생략}...
03     <filter>
04         <filter-name>Filter01</filter-name>
05         <filter-class>ch12.com.filter.AuthenFilter</filter-class>
06     </filter>
07     <filter-mapping>
08         <filter-name>Filter01</filter-name>
09         <url-pattern>/ch12/filter01_process.jsp</url-pattern>
10     </filter-mapping>
11 </web-app>
```

필터태그에서는 필터를 등록할 수 있다.
<filter-name>에는 필터의 이름을 지정한다.
<filter-class>에서는 구현한 필터 클래스를 패키지명을 포함하여 지정해준다.
웹 컨테이너는 서버를 띄울 때 web.xml 설정의 <filter> 태그를 읽어들이어 필터 객체를 생성하고 초기화한다.

등록한 필터의 동작할 범위(대상)를 설정할 수 있다.
<filter-name>은 앞서 등록한 필터명을 지정한다.
<url-pattern>에는 URL 패턴을 입력하는데, 서블릿을 설정할 때와 규칙이 같다.
여기서는 특정 서블릿만 등록을 하고 있다.



3. web.xml 파일의 필터 구성

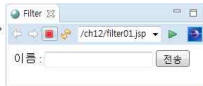
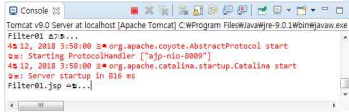
filter01.jsp

```
01 <% page contentType="text/html; charset=utf-8"%>
02 <html>
03 <head>
04 <title>Filter</title>
05 </head>
06 <body>
07     <form method="post" action="filter01_process.jsp">
08         <p> 이름 : <input type="text" name="name">
09             <input type="submit" value="전송">
10     </form>
11 </body>
12 </html>
```

3. web.xml 파일의 필터 구성

filter01_process.jsp

```
01 <% page contentType="text/html; charset=utf-8"%>
02 <html>
03 <head>
04 <title>Filter</title>
05 </head>
06 <body>
07     <%
08         String name = request.getParameter("name");
09     %>
10     <p> 입력된 name 값 :<%=name%>
11 </body>
12 </html>
```





3. web.xml 파일의 필터 구성

필터 처리로 매개변수와 값을 전달받아 로그인 인증 처리하기

InitParamFilter.java

```
01 package ch12.com.filter;
02
03 import java.io.IOException;
04 import java.io.PrintWriter;
05 import javax.servlet.Filter;
06 import javax.servlet.FilterChain;
07 import javax.servlet.FilterConfig;
08 import javax.servlet.ServletException;
09 import javax.servlet.ServletRequest;
10 import javax.servlet.ServletResponse;
11
12 public class InitParamFilter implements Filter {
13     private FilterConfig filterConfig = null;
14
15     @Override
16     public void init(FilterConfig filterConfig) throws ServletException {
17         System.out.println("Filter02 초기화...");
18         this.filterConfig = filterConfig;
19     }
```

3. web.xml 파일의 필터 구성



```
20
21     @Override
22     public void doFilter(ServletRequest request, ServletResponse response,
23         FilterChain filterChain) throws IOException, ServletException {
24
25         System.out.println("Filter02 수행...");
26
27         String id = request.getParameter("id");
28         String passwd = request.getParameter("passwd");
29
30         String param1 = filterConfig.getInitParameter("param1");
31         String param2 = filterConfig.getInitParameter("param2");
32
33         String message;
34
35         response.setCharacterEncoding("UTF-8");
36         response.setContentType("text/html; charset=UTF-8");
37         PrintWriter writer = response.getWriter();
```

3. web.xml 파일의 필터 구성

```
36
37     if (id.equals(param1) && passwd.equals(param2))
38         message = "로그인 성공했습니다.";
39     else
40         message = "로그인 실패했습니다.";
41
42     writer.println(message);
43
44     filterChain.doFilter(request, response);
45 }
46
47 @Override
48 public void destroy() {
49     System.out.println("Filter02 해제..");
50 }
51 }
```


3. web.xml 파일의 필터 구성

WEB-INF/web.xml

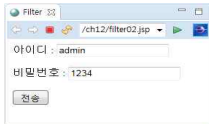
```
01 <web-app>
02   ... (생략) ...
03   <filter>
04     <filter-name>Filter02</filter-name>
05     <filter-class>ch12.com.filter.InitParamFilter</filter-class>
06     <init-param>
07       <param-name>param1</param-name>
08       <param-value>admin</param-value>
09     </init-param>
10     <init-param>
11       <param-name>param2</param-name>
12       <param-value>1234</param-value>
13     </init-param>
14   </filter>
15   <filter-mapping>
16     <filter-name>Filter02</filter-name>
17     <url-pattern>/ch12/filter02_process.jsp</url-pattern>
18   </filter-mapping>
19 </web-app>
```

<param-name>으로 파라미터영역을 <param-value>로 파라미터 값을 지정한다. 이 값들은 모두 문자열(String)타입이다.
<init-param> - 초기화 파라미터 등록 시 사용(여러 번 사용 가능)
<param-name> - 파라미터명 지정
<param-value> - 파라미터 값 지정(String타입으로 등록됨)

3. web.xml 파일의 필터 구성

filter02.jsp

```
01 <% page contentType="text/html; charset=utf-8"%>
02 <html>
03 <head>
04 <title>Filter</title>
05 </head>
06 <body>
07     <form method="post" action="filter02_process.jsp">
08         <p> 아이디 : <input type="text" name="id">
09         <p> 비밀번호 : <input type="text" name="passwd">
10         <p> <input type="submit" value="전송">
11     </form>
12 </body>
13 </html>
```

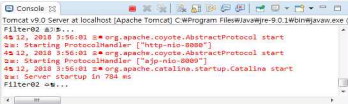
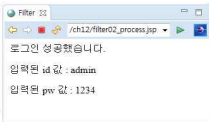


The screenshot shows a web browser window with the title 'Filter'. The address bar displays '/ch12/filter02.jsp'. The form contains two text input fields: '아이디 : admin' and '비밀번호 : 1234'. Below the inputs is a submit button labeled '전송'.

3. web.xml 파일의 필터 구성

filter02_process.jsp

```
01 <% page contentType="text/html; charset=utf-8"%>
02 <html>
03 <head>
04 <title>Filter</title>
05 </head>
06 <body>
07     <%
08         String id = request.getParameter("id");
09         String passwd = request.getParameter("passwd");
10     %>
11     <p> 입력된 id 값 : <%=id%>
12     <p> 입력된 pw 값 : <%=passwd%>
13 </html>
```



3. web.xml 파일의 필터 구성

웹 페이지를 이용하여 필터로 로그 기록하기

LogFileFilter.java

```
01 package ch12.com.filter;
02
03 import javax.servlet.*;
04 import java.util.*;
05 import java.text.DateFormat;
06 import java.text.SimpleDateFormat;
07 import java.io.*;
08 import java.io.PrintWriter;
09 import java.io.IOException;
10
11 public class LogFileFilter implements Filter {
12
13     PrintWriter writer;
14
15     public void init(FilterConfig filterConfig) throws ServletException {
16         String filename = filterConfig.getInitParameter("filename");
17         if(filename==null) throw new ServletException("로그 파일의 이름을 찾을 수 없습니다.");
18         try {
19             writer = new PrintWriter(new FileWriter(filename, true), true);
20         } catch (IOException e) {
21             throw new ServletException("로그 파일을 열 수 없습니다.");
22         }
23     }
```

3. web.xml 파일의 필터 구성

```
24
25     public void doFilter(ServletRequest request, ServletResponse response,
26     FilterChain filterChain) throws IOException, ServletException {
27         writer.printf("현재 일시 : %s %n", getCurrentTime());
28         String clientAddr = request.getRemoteAddr();
29         writer.printf("클라이언트 주소 : %s %n", clientAddr);
30
31         filterChain.doFilter(request, response);
32
33         String contentType = response.getContentType();
34         writer.printf("문서의 콘텐츠 유형 : %s %n", contentType);
35         writer.println("_____");
36     }
37
38     public void destroy() {
39         writer.close();
40     }
41
42     private String getCurrentTime() {
43         DateFormat formatter = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
44         Calendar calendar = Calendar.getInstance();
45         calendar.setTimeInMillis(System.currentTimeMillis());
46         return formatter.format(calendar.getTime());
47     }
```

3. web.xml 파일의 필터 구성

```
01 <web-app>
02 ... (생략) ...
03 <filter>
04     <filter-name>Filter02_2</filter-name>
05     <filter-class>ch12.com.filter.LogFileFilter</filter-class>
06     <init-param>
07         <param-name>filename</param-name>
08         <param-value>C:\\logs\\monitor.log</param-value>
09     </init-param>
10 </filter>
11 <filter-mapping>
12     <filter-name>Filter02_2</filter-name>
13     <url-pattern>ch12/filter02_process.jsp</url-pattern>
14 </filter-mapping>
15 </web-app>
```

monitor.log파일은 반드시 미리 생성하고 실행을 해야 문제가 발생하지 않는다.



감사합니다.

