

12 필터

1

필터의 개요

2

Filter 인터페이스의 구현 클래스

3

web.xml 파일의 필터 구성

4

[웹 쇼핑몰] 로그 기록하기

학습목표

- 필터의 개념을 이해합니다.
- Filter 인터페이스의 구현 클래스 작성 방법을 익힙니다.
- web.xml 파일에 필터를 구성하는 방법을 익힙니다.
- 웹 쇼핑몰의 로그 기록을 만듭니다.

1. 필터의 개요

❖ 필터(filter)

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

1. 필터의 개요

❖ 필터(filter)

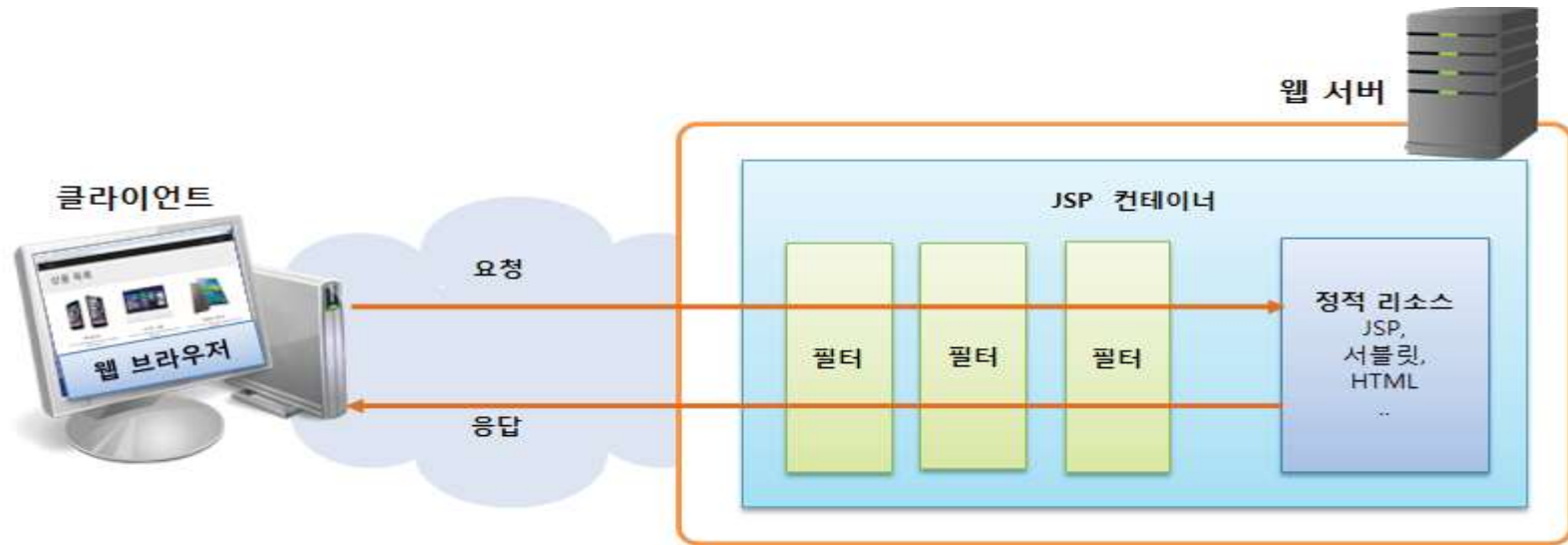


표 12-1 필터의 기능

필터	기능
Request 필터	인증(사용자 인증) 요청 정보를 로그 파일로 작성 암호화 인코딩 작업
Response 필터	응답 결과 데이터 압축 응답 결과에 내용 추가/수정 총 서비스 시간 측정

2. Filter 인터페이스의 구현 클래스

❖ Filter 인터페이스

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

```
import javax.servlet.Filter;

public class 클래스 이름 implements Filter
{
    ... (생략) ...
}
```

표 12-2 Filter 인터페이스 메소드의 종류

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

2. Filter 인터페이스의 구현 클래스

❖ 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

표 12-3 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

❖ 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

2. Filter 인터페이스의 구현 클래스

표 12-4 FilterChain 인터페이스 메소드의 종류

메소드	반환 유형	설명
doFilter(ServletRequest request, ServletResponse response)	void	체인의 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...");
}
```

2. Filter 인터페이스의 구현 클래스

❖ 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>
```

표 12-5 <filter>를 구성하는 하위 요소

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

표 12-6 <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 작성

```
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>
```

[<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>
```

3. web.xml 파일의 필터 구성

예제 12-1 폼 페이지에서 전송된 요청 파라미터를 필터로 처리하기

JSPBook/src/ch12/com/filter/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     }
39 }
```

3. web.xml 파일의 필터 구성

JSPBook/WebContent/WEB-INF/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>
```

3. web.xml 파일의 필터 구성

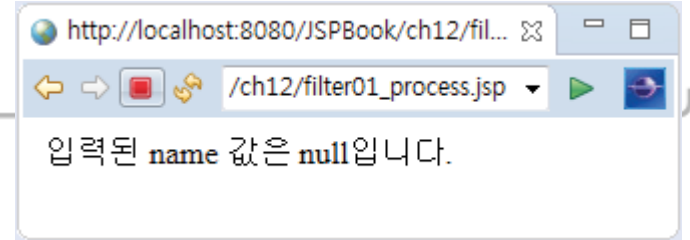
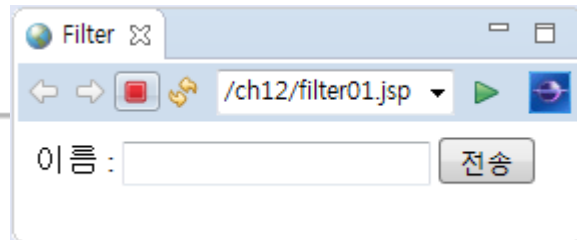
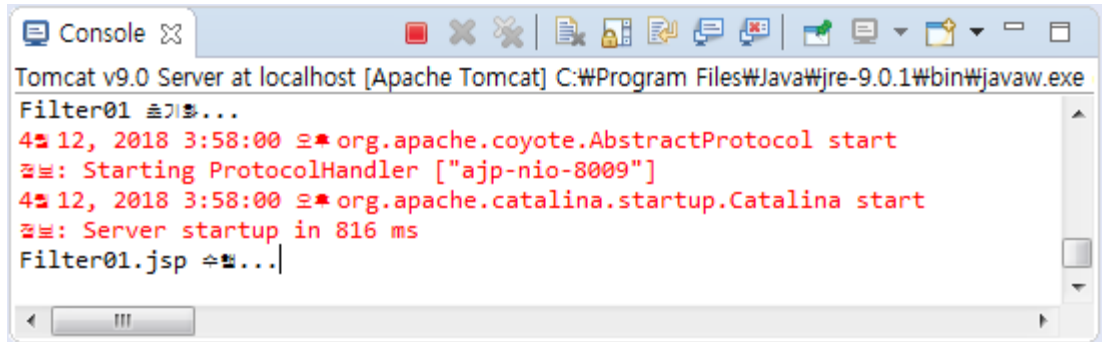
JSPBook/WebContent/ch12/filter01.jsp

```
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

JSPBook/WebContent/ch12/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

예제 12-2 필터 처리로 매개변수와 값을 전달받아 로그인 인증 처리하기

JSPBook/src/ch12/com/filter/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

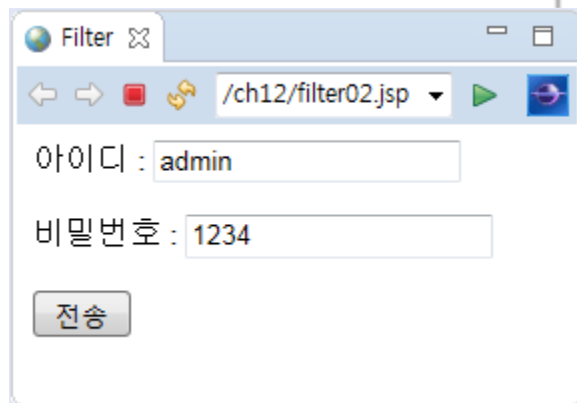
JSPBook/WebContent/WEB-INF/web.xml

```
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>
```


3. web.xml 파일의 필터 구성

JSPBook/WebContent/ch12/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>
```



Filter

/ch12/filter02.jsp

아이디 : admin

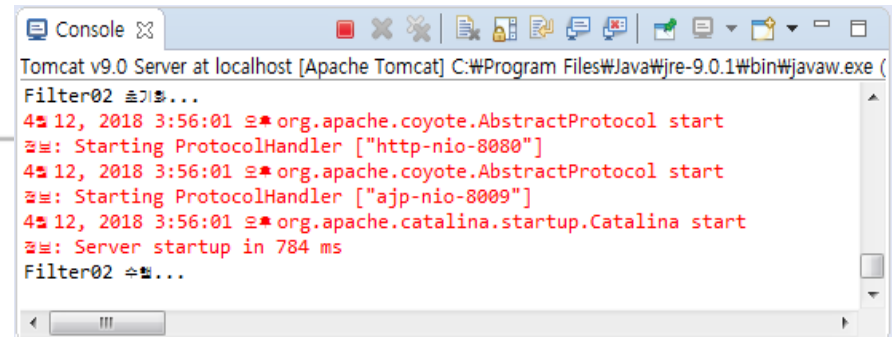
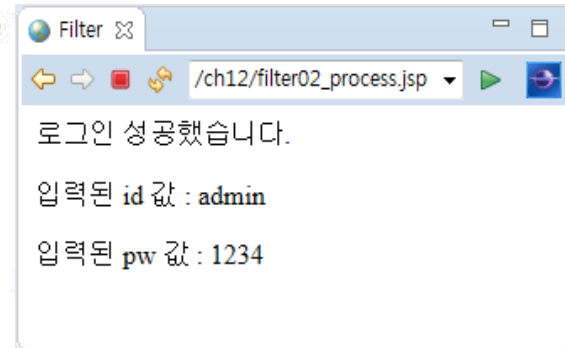
비밀번호 : 1234

전송

3. web.xml 파일의 필터 구성

JSPBook/WebContent/ch12/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 </body>
14 </html>
```



3. web.xml 파일의 필터 구성

예제 12-3 [예제 12-2]의 웹 페이지를 이용하여 필터로 로그 기록하기

JSPBook/src/ch12/com/filter/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.FileWriter;
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

JSPBook/WebContent/WEB-INF/web.xml

```
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>
```

Filter

/ch12/filter02.jsp

아이디 :

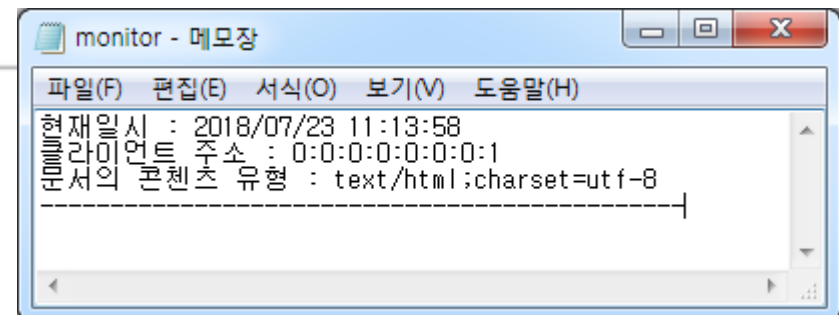
비밀번호 :

Filter

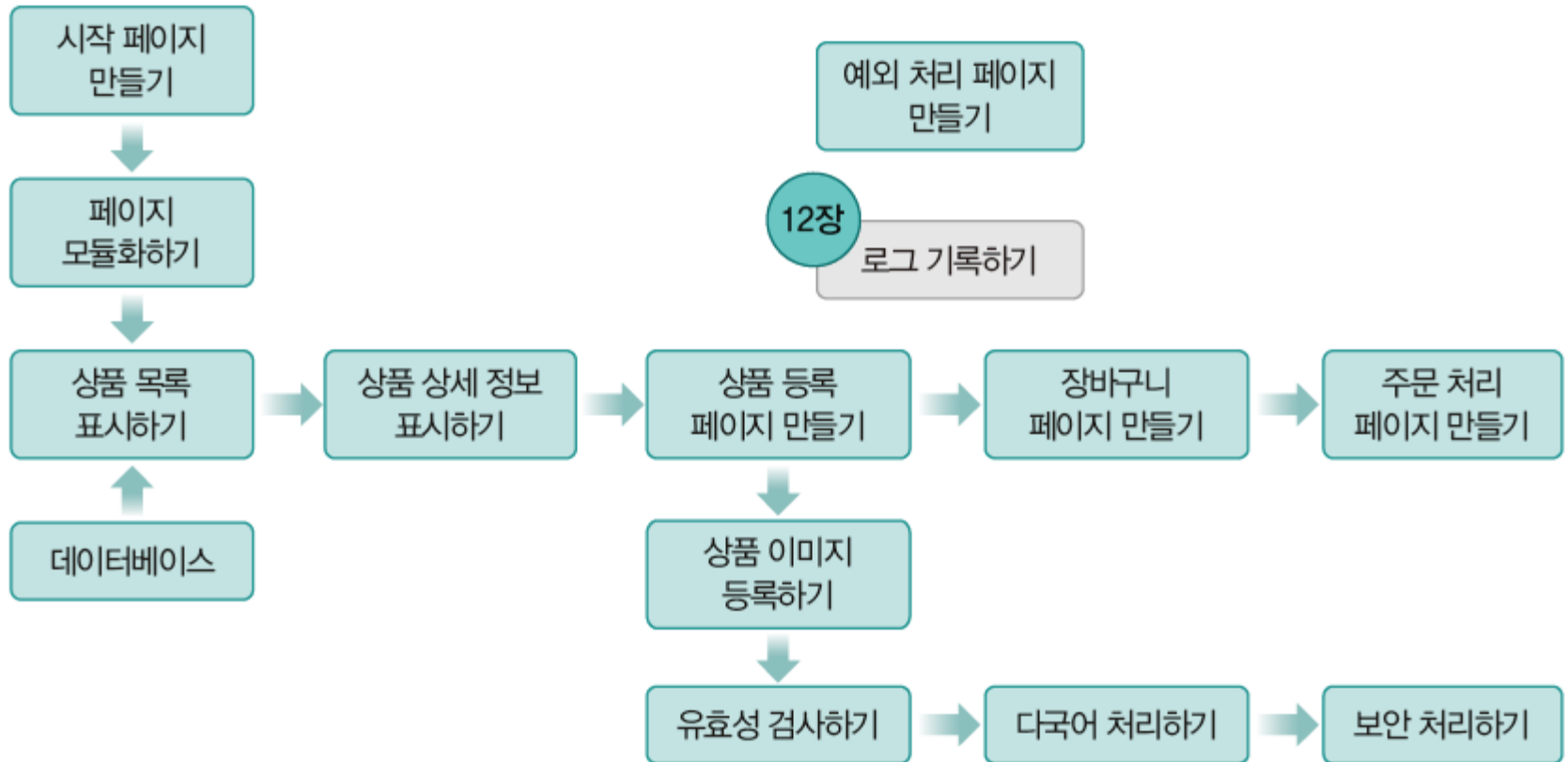
/ch12/filter02_process.jsp

입력된 id 값 : admin

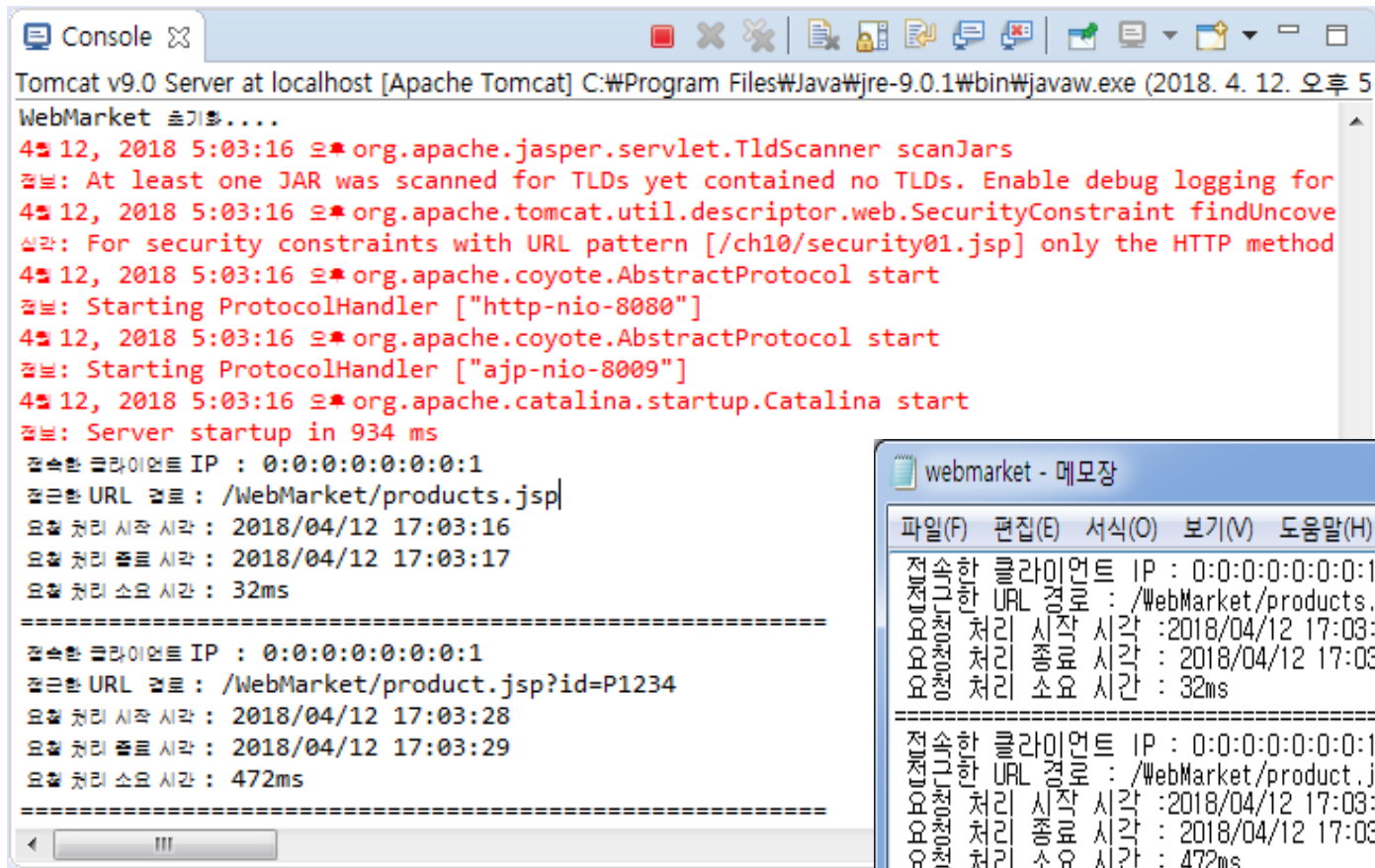
입력된 pw 값 : 1234



4. [웹 쇼핑몰] 로그 기록하기



4.[웹 쇼핑몰] 로그 기록하기



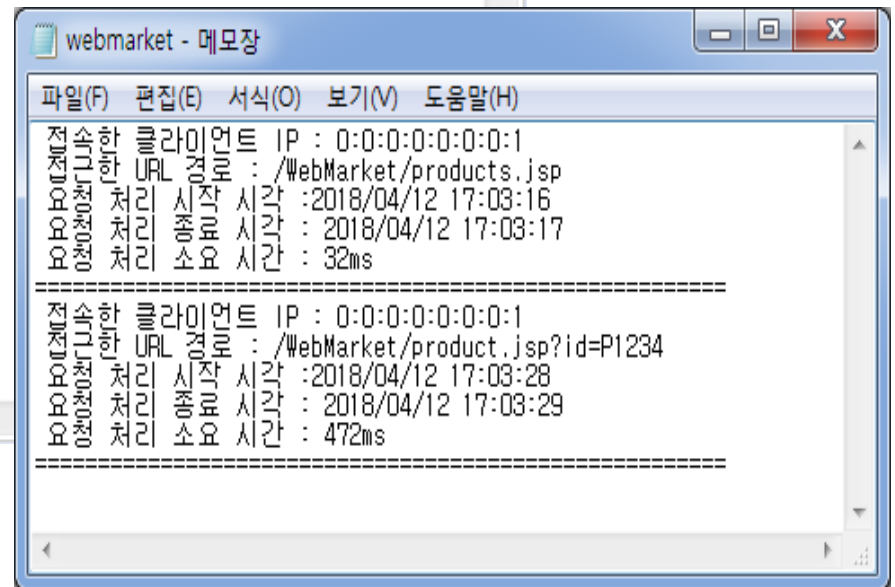
```
Console
Tomcat v9.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre-9.0.1\bin\javaw.exe (2018. 4. 12. 오후 5
WebMarket 초기화....
4월 12, 2018 5:03:16 오후 org.apache.jasper.servlet.TldScanner scanJars
정보: At least one JAR was scanned for TLDs yet contained no TLDs. Enable debug logging for
4월 12, 2018 5:03:16 오후 org.apache.tomcat.util.descriptor.web.SecurityConstraint findUncove
실라: For security constraints with URL pattern [/ch10/security01.jsp] only the HTTP method
4월 12, 2018 5:03:16 오후 org.apache.coyote.AbstractProtocol start
정보: Starting ProtocolHandler ["http-nio-8080"]
4월 12, 2018 5:03:16 오후 org.apache.coyote.AbstractProtocol start
정보: Starting ProtocolHandler ["ajp-nio-8009"]
4월 12, 2018 5:03:16 오후 org.apache.catalina.startup.Catalina start
정보: Server startup in 934 ms

접속한 클라이언트 IP : 0:0:0:0:0:0:0:1
접근한 URL 경로 : /WebMarket/products.jsp
요청 처리 시작 시각 : 2018/04/12 17:03:16
요청 처리 종료 시각 : 2018/04/12 17:03:17
요청 처리 소요 시간 : 32ms

=====

접속한 클라이언트 IP : 0:0:0:0:0:0:0:1
접근한 URL 경로 : /WebMarket/product.jsp?id=P1234
요청 처리 시작 시각 : 2018/04/12 17:03:28
요청 처리 종료 시각 : 2018/04/12 17:03:29
요청 처리 소요 시간 : 472ms

=====
```



```
webmarket - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
접속한 클라이언트 IP : 0:0:0:0:0:0:0:1
접근한 URL 경로 : /WebMarket/products.jsp
요청 처리 시작 시각 : 2018/04/12 17:03:16
요청 처리 종료 시각 : 2018/04/12 17:03:17
요청 처리 소요 시간 : 32ms

=====

접속한 클라이언트 IP : 0:0:0:0:0:0:0:1
접근한 URL 경로 : /WebMarket/product.jsp?id=P1234
요청 처리 시작 시각 : 2018/04/12 17:03:28
요청 처리 종료 시각 : 2018/04/12 17:03:29
요청 처리 소요 시간 : 472ms

=====
```

4.[웹 쇼핑몰] 로그 기록하기

예제 12-4 필터 처리로 로그 기록하기

WebMarket/src/filter/LogFilter.java

```
01 package filter;
02
03 import javax.servlet.*;
04 import javax.servlet.http.*;
05 import java.util.*;
06 import java.text.DateFormat;
07 import java.text.SimpleDateFormat;
08
09 public class LogFilter implements Filter {
10
11     public void init(FilterConfig config) throws ServletException{
12         System.out.println("WebMarket 초기화....");
13     }
14
15     public void doFilter(ServletRequest request, ServletResponse response,
16         FilterChain chain) throws java.io.IOException, ServletException {
17         System.out.println(" 접속한 클라이언트 IP : " + request.getRemoteAddr());
18         long start = System.currentTimeMillis();
19         System.out.println(" 접근한 URL 경로 : " + getURLPath(request));
20         System.out.println(" 요청 처리 시작 시각 : " + getCurrentTime());
21         chain.doFilter(request, response);
22
23         long end = System.currentTimeMillis();
24         System.out.println(" 요청 처리 종료 시각 : " + getCurrentTime());
25         System.out.println(" 요청 처리 소요 시간 : " + (end-start)+ "ms ");
26         System.out.println("=====
        =====");
    }
}
```

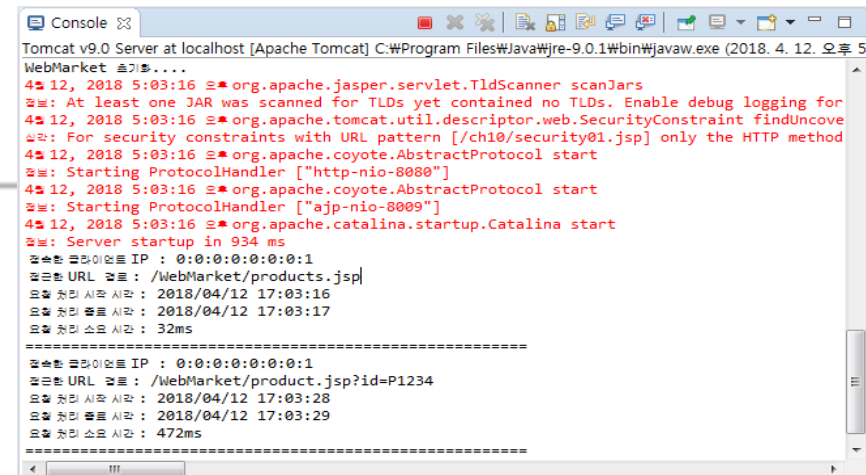

4.[웹 쇼핑몰] 로그 기록하기

```
27
28     public void destroy( ){
29
30     }
31
32     private String getURLPath(ServletRequest request) {
33         HttpServletRequest req;
34         String currentPath="";
35         String queryString="";
36         if(request instanceof HttpServletRequest){
37             req = (HttpServletRequest)request;
38             currentPath = req.getRequestURI();
39             queryString = req.getQueryString();
40             queryString = queryString == null ? "" : "?" + queryString;
41         }
42         return currentPath+queryString;
43     }
44
45     private String getCurrentTime() {
46         DateFormat formatter = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
47         Calendar calendar = Calendar.getInstance();
48         calendar.setTimeInMillis(System.currentTimeMillis());
49         return formatter.format(calendar.getTime());
50     }
51 }
```

4. [웹 쇼핑몰] 로그 기록하기

WebMarket/WebContent/WEB-INF/web.xml

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <web-app>
03     ... (생략) ...
04     <filter>
05         <filter-name>LogFilter</filter-name>
06         <filter-class>filter.LogFilter</filter-class>
07     </filter>
08     <filter-mapping>
09         <filter-name>LogFilter</filter-name>
10         <url-pattern>/*</url-pattern>
11     </filter-mapping>
12 </web-app>
```



4.[웹 쇼핑몰] 로그 기록하기

예제 12-5 필터 처리로 로그 기록 파일 만들기

- 로그 기록 파일의 저장 폴더 만들기: C 드라이브에 log 폴더를 생성
- Filter 인터페이스의 구현 클래스 작성하기

WebMarket/src/filter/LogFileFilter.java

```
01 package filter;
02
03 import javax.servlet.*;
04 import javax.servlet.http.*;
05 import java.util.*;
06 import java.text.DateFormat;
07 import java.text.SimpleDateFormat;
08 import java.io.FileWriter;
09 import java.io.PrintWriter;
10 import java.io.IOException;
11
12 public class LogFileFilter implements Filter {
13
14     PrintWriter writer;
```

4.[웹 쇼핑몰] 로그 기록하기

```
15
16     public void init(FilterConfig config) throws ServletException {
17         String filename = config.getInitParameter("filename");
18
19         if (filename == null)
20             throw new ServletException("로그 파일의 이름을 찾을 수 없습니다.");
21
22         try {
23             writer = new PrintWriter(new FileWriter(filename, true), true);
24         } catch (IOException e) {
25             throw new ServletException("로그 파일을 열 수 없습니다.");
26         }
27     }
28
29     public void doFilter(ServletRequest request, ServletResponse response,
30                         FilterChain chain) throws java.io.IOException, ServletException {
31
32         writer.println(" 접속한 클라이언트 IP : " + request.getRemoteAddr());
33         long start = System.currentTimeMillis();
34         writer.println(" 접근한 URL 경로 : " + getURLPath(request));
35         writer.println(" 요청 처리 시작 시각 : " + getCurrentTime());
36
37         chain.doFilter(request, response);
38
39         long end = System.currentTimeMillis();
40         writer.println(" 요청 처리 종료 시각 : " + getCurrentTime());
41         writer.println(" 요청 처리 소요 시간 : " + (end - start) + "ms ");
42         writer.println("=====");
43     }
44
45     public void destroy() {
46         writer.close();
47     }
```

4.[웹 쇼핑몰] 로그 기록하기

```
42
43     public void destroy() {
44         writer.close();
45     }
46
47     private String getURLPath(ServletRequest request) {
48         HttpServletRequest req;
49         String currentPath = "";
50         String queryString = "";
51         if (request instanceof HttpServletRequest) {
52             req = (HttpServletRequest) request;
53             currentPath = req.getRequestURI();
54             queryString = req.getQueryString();
55             queryString = queryString == null ? "" : "?" + queryString;
56         }
57         return currentPath + queryString;
58     }
59
60     private String getCurrentTime() {
61         DateFormat formatter = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
62         Calendar calendar = Calendar.getInstance();
63         calendar.setTimeInMillis(System.currentTimeMillis());
64         return formatter.format(calendar.getTime());
65     }
66 }
```

4.[웹 쇼핑몰] 로그 기록하기

JSPBook/WebContent/WEB-INF/web.xml

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <web-app>
03     ... (생략) ...
04     <filter>
05         <filter-name>LogFileFilter</filter-name>
06         <filter-class>filter.LogFileFilter</filter-class>
07         <init-param>
08             <param-name>filename</param-name>
09             <param-value>C:\\logs\\webmarket.log</param-value>
10         </init-param>
11     </filter>
12     <filter-mapping>
13         <filter-name>LogFileFilter</filter-name>
14         <url-pattern>/*</url-pattern>
15     </filter-mapping>
16 </web-app>
```

