```
1
   Filter & Wrapper
 2
 3
   1. What is Filter?
 4
      1)HTTP Request와 Response을 변경할 수 있는 재 사용 가능한 Code
 5
      2)Request를 가로채서 요청된 리소스에 전달된 ServletRequest나 ServletResponse를 처리하는 객체
 6
      3)객체의 형태로 존재
 7
      4)Client로 부터 들어오는 Request나 최종 Resource(Servlet / JSP / 기타등등) 사이에 위치하여 Client의
      Request정보를 알맞게 변경할 수 있게 한다.
 8
      5)최종 Resource과 Client로 가는 Response사이에 위치하여 최종 Resource의 Request결과를 알맞게 변경할 수 있게
      하다.
 9
10
   Client's Browser -----(request)----> Filter -----> servlet/Jsp
11
12
   Client's Browser <---(response) --- Filter <-----
13
14
15 2. 기능
16
      1)Servlet / JSP등을 실행하기 이전에 Request가 올바른지 또는 Resource에 접근할 수 있는 권한을 가졌는 지의 여부를
      미리 처리할 수 있다.
17
      2)Resource가 받게 되는 Request정보는 Client와 Resource사이에 존재하는 Filter에 의해 변경된 Request정보이다.
18
      3)Servlet / JSP이 생성한 Response데이터를 변경하거나 취소할 수 있는 기능을 구현할 수 있도록 해 준다.
19
      4)Client가 보게 되는 Response 정보는 Client와 Resource 사이에 존재하는 Filter에 의해 변경된 Response 정보가
      된다.
20
      5)Filter는 Client와 Resource 사이에 한 개만 존재할 수 있는 것은 아니며, 여러 개의 Filter가 모여 하나의 Filter
      Chain을 형성할 수도 있다.
21
      6)만일 여러 개의 Filter가 모여서 하나의 체인을 형성할 때 첫 번째 Filter가 변경하는 Request 정보는 Client의
      Request 정보가 되지만, Chain의 두 번째 Filter가 변경하는 Request 정보는 첫 번째 Filter를 통해서 변경된 Request
      정보가 된다.
22
      7)즉, Request 정보는 변경에 변경을 거듭하게 되는 것이다.
23
      8)Filter는 변경된 정보를 변경하는 역할 뿐만 아니라 흐름을 변경하는 역할도 할 수 있다.
24
      9)즉, Filter는 Client의 요청을 Filter Chain의 다음 단계(결과적으로 Client 가 요청한 자원)에 보내는 것이 아니라 다른
      자원의 결과를 Client에 전송할 수 있다.
25
      10)Filter 는 이러한 기능을 이용해서 사용자 인증이나 권한 체크와 같은 곳에서 사용할 수 있다.
26
      11)로그인 여부의 확인이나 요금 부과 처리와 같은 일은 Filter를 구현한 클래스 안에 구현하면 되지만, Web Browser와
      Web Component 사이에 오가는 데이터를 변형하는 일은 할 수 없다.
27
      12)오고 가는 메시지를 암호화 하거나, 메시지에 포함된 데이터의 일부를 걸러서 전달되지 못하도록 막는 일은 할 수 없는데,
      그런 일을 하기 위해서는 Filter 뿐만 아니라 Wrapper도 필요하다.
28
      13)Wrapper는 Web Browser와 Web Component 사이를 오가는 Request Message와 Response Message를
      포장한다.
29
30
31
        [HelloServlet.java]
32
        package com.example.servlets;
33
34
        public class HelloServlet extends HttpServlet{
35
           public void doGet(HttpServletRequest req, HttpServletResponse res)
             throws IOException, ServletException{
36
37
             res.setContentType("text/html;charset=utf-8");
38
             PrintWriter out = res.getWriter();
39
             out.println("<html><body bgcolor='red'>");
40
             out.println("<font size='7' color='white'><b>안녕 Servlet</b></font>");
             out.println("</body></html>");
41
42
             out.close();
43
           }
44
        }
45
46
        [web.xml]
47
        <servlet>
48
           <servlet-name>HelloServlet</servlet-name>
49
           <servlet-class>com.example.servlets.HelloServlet</servlet-class>
50
        </servlet>
51
        <servlet-mapping>
           <servlet-name>HelloServlet</servlet-name>
52
53
           <url-pattern>/servlets/HelloServlet</url-pattern>
54
         </servlet-mapping>
55
```

14)http://localhost:8080/MyContext/servlets/HelloServlet 를 실행하여 HelloServlet 을 실행할 것이다.

56

```
57
        15)여기서 아래와 같이 web.xml에 Filter를 추가하자.
 58
 59
 60
           [web.xml]
           <filter>
 61
              <filter-name>MyFilter</filter-name>
 62
 63
              <filter-class>com.example.filter.MyFilter</filter-class>
 64
 65
           <filter-mapping>
              <filter-name>MyFilter</filter-name>
 66
 67
              <!--Servlet Class 를 사용하고자 할 때 -->
              <servlet-name>HelloServlet/servlet-name>
 68
 69
              <!--JSP file 을 사용하고자 할 때 -->
 70
               <url-pattern>*.jsp</url-pattern>
 71
           </filter-mapping>
 72
 73
        16)HelloServlet 을 요청하면 MyFilter가 호출되고, MyFiler는 MyFilter.java 객체를 호출한다.
 74
 75
           [MyFilter.java]
 76
           package com.example.filter;
 77
 78
           import javax.servlet.Filter;
 79
           import javax.servlet.FilterChain;
 80
           import javax.servlet.FilterConfig;
 81
           import javax.servlet.ServletException;
 82
           import javax.servlet.ServletRequest;
 83
           import javax.servlet.ServletResponse;
 84
 85
           public class MyFilter implements Filter{
 86
             @Override
             public void init(FilterConfig filterConfig) throws ServletException{}
 87
 88
             @Override
 89
             public void destroy(){}
 90
             @Override
 91
             public void doFilter(ServletRequest request, ServletResponse response,
 92
                   FilterChain chain) throws java.io.IOException, ServletException{
 93
                System.out.println("Filter is Starting...");
 94
                chain.doFilter(request, response);
 95
                System.out.println("Filter is Ending...");
 96
             }
           }
 97
 98
 99
           [HelloServlet.java]
           package com.example.servlets;
100
101
102
           import javax.servlet.http.HttpServlet;
103
           import javax.servlet.http.HttpServletResponse;
104
           import javax.servlet.http.HttpServletRequest;
105
           import javax.servlet.ServletException;
106
           import java.io.IOException;
107
           import java.io.PrintWriter;
108
109
           public class HelloServlet extends HttpServlet {
110
             public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException,
             IOException{
                res.setContentType("text/html;charset=utf-8");
111
112
                PrintWriter out = res.getWriter();
113
                out.println("<html><body bgcolor='yellow>");
114
                out.println("<font size='7' color='red'>안녕 Servlet</font>");
115
                out.println("</body></html>");
116
                out.close();
117
             }
           }
118
119
120
           [test.jsp]
           <body bgcolor="yellow">
121
122
              <h1>Message Filter 적용 중</h1>
```

```
123
          </body>
124
125
126
     3. Filter 의 구현
127
        1)Filter를 구현하려면 핵심적인 역할을 하는 javax.servlet.Filter interface,
       javax.servlet.ServletReguestWrapper class, javax.servlet.ServletResponseWrapper classo
128
       2)Filter interface는 Client와 최종 Resource 사이에 위치하는 Filter를 나타내는 객체가 구현해야 하는 interface이다.
129
       3)ServletRequestWrapper class 와 ServletResponseWrapper class는 Filter가 요청한 변경한 결과 또는
       응답을 변경한 결과를 저장할 Wrapper class를 나타낸다.
130
       4)Methods
131
          a. void init(FilterConfig filterConfig) throws ServletException
132
             -Filter를 Web Container내에 생성한 후 초기화할 때 호출
133
134
          b. void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws
          java.io.IOException, ServletException
135
             -Chain에 따라 다음에 존재하는 Filter로 이동한다.
136
             -Chain의 가장 마지막에는 Client가 요청한 최종 Resource이 위치한다.
137
138
          c. void destroy()
139
            -Filter가 Web Container에서 삭제될 때 호출된다.
140
141
142
     4. Filter 설정하기
143
        1)web.xml에 <filter> tag와 <filter-mapping> tag를 설정한다.
144
       2)<filter>
145
          -Web Application에서 사용될 Filter를 지정하는 역할
146
147
       3)<filter-mapping>
148
          -특정 자원에 대해 어떤 Filter를 사용할지를 결정
149
150
       4)<url-pattern>
151
          -Client가 요청한 특정 URI에 대해서 Filtering할 때 사용
          -'/' 로 시작하고, '/*' 로 끝나는 경로 매핑
152
153
          -'*.' 로 시작하는 확장자에 대한 매핑
          -오직 '/'만 포함하는 경우 어플리케이션의 기본 서블릿으로 매핑
154
155
          -주의할 점은 계층적인 URL 경로명을 지정할 대는 와일드카드 문자와 파일 확장자를 함께 쓰면 안된다.
156
          -<url-pattern>/sub1/*.jsp</url-pattern>을 잘못된 것이다.
157
158
       5) 사용예
159
          a. <filter-mapping>
160
               <filter-name>Simple-filter</filter-name>
161
               <url-pattern>/*</url-pattern>
162
             <filter-mapping>
163
164
          b. <filter-mapping>
165
               <filter-name>Simple-filter</filter-name>
166
               <url-pattern>*.jsp</url-pattern>
167
             <filter-mapping>
168
          c. <filter-mapping>
169
170
               <filter-name>Simple-filter</filter-name>
171
               <url-pattern>/sub1/*</url-pattern>
172
             <filter-mapping>
173
          d. <filter-mapping>
174
175
               <filter-name>Simple-filter</filter-name>
176
               <url-pattern>/sub1/*</url-pattern>
177
               <url-pattern>/sub2/*</url-pattern>
178
             <filter-mapping>
179
180
          e. <filter-mapping>
181
               <filter-name>Simple-filter</filter-name>
182
               <servlet-name>hello-servlet</servlet-name>
183
             <filter-mapping>
184
185
          f. <filter-mapping>
186
               <filter-name>Simple-filter</filter-name>
```

```
187
               <url-pattern>/sub1/*</url-pattern>
188
               <url-pattern>/sub2/*</url-pattern>
189
               <servlet-name>hello-servlet</servlet-name>
190
             <filter-mapping>
191
192
193
     5. <dispatcher> tag
194
       1)이 tag는 다음과 같이 <filter-mapping>태그의 자식 태그로 사용된다.
195
196
          <filter-mapping>
197
             <filter-name>Simple-filter</filter-name>
198
             <servlet-name>hello-servlet</servlet-name>
199
             <dispatcher>INCLUDE</dispatcher>
200
          <filter-mapping>
201
202
       2)이 태그는 실행되는 자원을 Client가 요청한 것인지, 아니면 RequestDispatcher의 forward()를 통해서 이동한 것인지
       아니면, include()통해서 포함된 것인지에 따라서 필터를 적용하도록 지정가능하다.
203
       3)다음의 값을 가질 수 있다.
204
          a. REQUEST: Client 의 요청인 경우에 필터를 사용(기본값)
205
          b. FORWARD: forward() 를 통해서 제어를 이동하는 경우에 필터를 사용한다.
206
          c. INCLUDE: include() 를 통해서 포함하는 경우에 필터를 사용한다.
207
          d. ERROR: exception 이 발생했을 때 웹 컨테이너가 자동으로 호출하는 방법
208
209
210 6. FilterChain의 방향 바꾸기
211
       1)/member/welcome.jsp를 요청한다.
212
       2)session이 없기 때문에 Filter가 확인한 후, LoginForm.html로 이동시킨다.
213
       3)로그인이 성공하면 session 이 생성됐기 때문에, 다시 welcome.jsp로 요청하면 바로 들어간다.
214
215
216
          [web.xml]
217
          <filter>
218
             <filter-name>LoginCheckFilter</filter-name>
219
             <filter-class>com.example.filter.LoginCheckFilter</filter-class>
220
          </filter>
221
222
          <filter-mapping>
223
             <filter-name>LoginCheckFilter</filter-name>
224
             <url-pattern>/member/*</url-pattern>
225
          </filter-mapping>
226
227
          [member/welcome.jsp]
228
          <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
229
          <body>
230
             <h1>회원전용페이지</h1>
231
          </body>
232
233
          [LoginCheckFilter.java]
234
          package com.example.filter;
235
236
          public class LoginCheckFilter implements Filter{
237
            @Override
238
            public void doFilter(ServletRequest request, ServletResponse response,
239
                 FilterChain chain) throws java.io.IOException, ServletException{
               HttpServletResponse res = (HttpServletResponse)response;
240
241
               HttpServletRequest req = (HttpServletRequest)request;
242
               HttpSession session = req.getSession();
243
               if(session == null)
244
                 res.sendRedirect("/FilterDemo/LoginForm.html");
245
               String userid = (String)session.getAttribute("userid");
246
               if(userid == null)
247
                 res.sendRedirect("/FilterDemo/LoginForm.html");
248
               chain.doFilter(request, response);
249
250
            @Override
251
            public void destroy(){}
252
            @Override
```

```
253
            public void init(FilterConfig filterConfig) throws ServletException {}
254
          }
255
256
          [LoginForm.html]
257
          <body>
258
            <div align="center">
               <form action="login.jsp" method="post">
259
                 260
261
                    262
                      ID : 
263
                      <input type="text" name="userid" />
264
                    265
                    266
                      PWD: 
267
                      <input type="password" name="userpwd" />
268
                    269
                    270
                      271
                      <input type="submit" value="Login" />
272
                    273
                 274
               </form>
275
             </div>
276
          </body>
277
278
          [login.jsp]
279
          <%
280
            String userid = request.getParameter("userid");
281
            String passwd = request.getParameter("userpwd");
282
            if(userid.equals("javasoft") && passwd.equals("123456")){
283
               session.setAttribute("userid", userid);
284
285
          %>
286
287
288
     7. FilterConfig 사용하는 예제
289
290
       [web.xml]
291
       <filter>
292
          <filter-name>LogFilter</filter-name>
293
          <filter-class>com.example.filter.LogFilter</filter-class>
294
          <init-param>
295
             <param-name>FILE</param-name>
296
             <param-value>D:\\temp\\logfilter.txt</param-value>
297
          </init-param>
298
       </filter>
299
       <filter-mapping>
300
          <filter-name>LogFilter</filter-name>
301
           <servlet-name>HelloServlet</servlet-name>
302
       </filter-mapping>
303
304
       [LogFilter.java]
305
       package com.example.filter;
306
307
       public class LogFilter implements Filter{
308
          private PrintWriter writer;
309
310
          public void init(FilterConfig filterConfig) throws ServletException{
311
            String filename = filterConfig.getInitParameter("FILE"); //D:\\temp\\logfilter.txt
312
313
               writer = new PrintWriter(new FileWriter(filename, true), true);
314
            }catch(IOException ex){
315
               throw new ServletException("Log file Error...");
316
            }
317
318
          @Override
319
          public void destroy(){
```

```
320
             writer.close();
321
           }
322
          @Override
323
          public void doFilter(ServletRequest request, ServletResponse response,
324
                FilterChain chain) throws java.io.IOException, ServletException{
325
             writer.println("Filter is Staring...");
326
             writer.flush();
             chain.doFilter(request, response);
327
             writer.println("Filter is Ending...");
328
329
             writer.flush();
330
          }
331
        }
332
333
334
        [web.xml]
335
        <filter-mapping>
336
           <filter-name>LogMessageFilter</filter-name>
337
           <url-pattern>*.jsp</url-pattern>
338
        </filter-mapping>
339
        <filter>
340
           <filter-name>LogMessageFilter</filter-name>
           <filter-class>com.example.filter.LogMessageFilter</filter-class>
341
342
           <init-param>
343
             <param-name>FILE1</param-name>
344
             <param-value>D:\\temp\\logmessage_filter.txt</param-value>
345
           </init-param>
346
        </filter>
347
348
        [LogMessageFilter.java]
349
        package com.example.filter;
350
351
        public class LogMessageFilter implements Filter {
352
          private PrintWriter writer;
353
           @Override
354
          public void destroy() {
355
             writer.close();
356
          }
357
358
          @Override
359
          public void doFilter(ServletRequest req, ServletResponse res,
360
                FilterChain chain) throws IOException, ServletException {
361
             Date now = new Date();
362
             this.writer.printf("방문일시: %1$tF %1$tT %n", now);
             HttpServletRequest request = (HttpServletRequest)req;
363
             this.writer.printf("고객 IP 정보: %s %n", request.getRemoteAddr());
364
365
             this.writer.flush();
             chain.doFilter(req, res);
366
367
             HttpServletResponse response = (HttpServletResponse)res;
368
             this.writer.printf("Content Type: %s %n", response.getContentType());
             this.writer.println("-----");
369
370
             this.writer.flush();
371
          }
372
373
           @Override
          public void init(FilterConfig config) throws ServletException {
374
375
             String filename = config.getInitParameter("FILE1");
376
377
                this.writer = new PrintWriter(new FileWriter(filename, true), true);
378
             }catch(IOException ex){
379
                throw new ServletException("Error");
380
             }
381
          }
382
        }
383
384
        [test.jsp]
        <body bgcolor="yellow">
385
386
           <h1>Message Filter 적용 중</h1>
```

```
</body>
388
389
390
     8. Wrapper class의 작성, 설치, 사용
391
        1)Web Browser와 Web Component사이를 오가는 데이터에 변형을 가하려면 Filter를 구현한 클래스와 더불어
       Wrapper class를 작성해야 한다.
392
       2)요청 객체를 포장하는 요청 래퍼 클래스와 응답 객체를 포장하는 응답 래퍼 클래스 두가지
393
       3)javax.servlet.http.HttpServletRequestWrapper와 javax.servlet.http.HttpServletResponseWrapper
        클래스를 상속받아야 한다.
394
395
396
     9. RequestWrapper Demo
397
398
       [ParamUpperCaseRequestWrapper.java]
399
       import javax.servlet.http.*;
400
       import java.io.*;
401
       import java.util.*;
402
       public class ParamUpperCaseRequestWrapper extends HttpServletRequestWrapper {
403
          HttpServletRequest request;
404
          public ParamUpperCaseRequestWrapper(HttpServletRequest request) {
405
             super(request);
406
             this.request = request;
407
          }
408
          public String getParameter(String name) {
409
             String str = request.getParameter(name);
410
             if (str == null) return null;
411
             return str.toUpperCase();
412
          }
413
          public String[] getParameterValues(String name) {
414
             String str[] = request.getParameterValues(name);
415
             if (str == null) return null;
416
             for (int cnt = 0; cnt < str.length; cnt++)
417
               str[cnt] = str[cnt].toUpperCase();
418
             return str;
419
          }
420
       }
421
422
       [ParamUpperCaseFilter.java]
423
       import javax.servlet.http.*;
424
       import javax.servlet.*;
       import java.io.*;
425
426
       public class ParamUpperCaseFilter implements Filter {
427
          public void init(FilterConfig config) throws ServletException {}
428
          public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
429
                          throws IOException, ServletException {
430
             ParamUpperCaseRequestWrapper requestWrapper
431
                      = new ParamUpperCaseRequestWrapper((HttpServletRequest) request);
432
             chain.doFilter(requestWrapper, response);
433
434
          public void destroy() {}
435
       }
436
437
       [welcome.jsp?NAME=Jessica]
438
        <%@page contentType="text/html; charset=euc-kr"%>
439
        <% String name = request.getParameter("NAME"); %>
440
        <HTML>
          <HEAD><TITLE>환영 인사</TITLE></HEAD>
441
442
          <BODY>
443
             안녕하세요, <%= name %>님.
444
          </BODY>
445
        </HTML>
446
447
       [colors.jsp?COLOR=White&COLOR=Red&COLOR=Blue]
448
        <%@page contentType="text/html; charset=euc-kr"%>
449
        <% String color[] = request.getParameterValues("COLOR"); %>
450
        <HTML>
451
          <HEAD><TITLE>색 고르기</TITLE></HEAD>
```

387

```
452
          <BODY>
453
             <H4>선택하신 색은 다음과 같습니다.</H4>
454
             <%
455
               if (color != null) {
456
                  for (int cnt = 0; cnt < color.length; cnt++)
457
                     out.println(color[cnt] + "<BR>");
458
               }
459
             %>
460
           </BODY>
461
        </HTML>
462
463
464
     10. ResponseWrapper Demo
465
466
        [CookieLowerCaseResponseWrapper.java]
467
        import javax.servlet.http.*;
468
        import javax.servlet.*;
469
        import java.io.*;
470
        public class CookieLowerCaseResponseWrapper extends HttpServletResponseWrapper {
471
          private HttpServletResponse response;
472
          public CookieLowerCaseResponseWrapper(HttpServletResponse response) {
473
             super(response);
474
             this.response = response;
475
476
          public void addCookie(Cookie cookie) {
477
             String value = cookie.getValue();
478
             String newValue = value.toLowerCase();
479
             cookie.setValue(newValue);
480
             response.addCookie(cookie);
481
          }
        }
482
483
484
        [CookieLowerCaseFiler.java]
485
        import javax.servlet.http.*;
486
        import javax.servlet.*;
487
        import java.io.*;
488
        public class CookieLowerCaseFilter implements Filter {
489
          public void init(FilterConfig config) throws ServletException {
490
491
          public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
492
                          throws IOException, ServletException {
493
             CookieLowerCaseResponseWrapper responseWrapper
494
                          = new CookieLowerCaseResponseWrapper((HttpServletResponse) response);
495
             chain.doFilter(request, responseWrapper);
496
          }
497
          public void destroy() {
498
          }
499
        }
500
501
        [cookieSaver.jsp]
502
        <%@page contentType="text/html; charset=euc-kr"%>
503
        <%
          Cookie cookie = new Cookie("CART", "Lemon, Apple, Chocolate");
504
505
          response.addCookie(cookie);
506
507
        <HTML>
          <HEAD><TITLE>쿠키 저장하기</TITLE></HEAD>
508
509
          <BODY>
510
             쿠키가 저장되었습니다.
511
          </BODY>
512
        </HTML>
513
514
        [cookieRetriever.jsp]
515
        <%@page contentType="text/html; charset=euc-kr"%>
516
        <HTML>
517
          <HEAD><TITLE>쿠키 가져오기</TITLE></HEAD>
          <BODY>
518
```

```
519
            CART = ${cookie.CART.value}
520
          </BODY>
521
        </HTML>
522
523
524
     11. WebFilter annotation을 이용한 Filter 설정
525
        1)@WebFilter를 사용할 경우에는 배포 기술서를 수정할 필요 없이 Filter class에 annotation만 붙여주면 되기 때문에
       편리하다.
526
       2)하지만, 설정 정보를 바꿀 때마다 다시 컴파일해야 한다.
527
       3)반면 배포 기술서를 사용할 경우에는 컴파일을 할 필요 없이 텍스트 파일만 수정하면 된다.
528
       4)@WebFilter 속성
529
          a. asyncSupported: 비동기 동작 모드 사용 여부
          b. description
530
531
          c. dispatcherTypes
532
          d. displayName
533
          e. filterName
534
          f. initParams
535
          g. largeIcon
536
          h. servletNames
          i. smallIcon
537
538
          i. urlPatterns
539
          k. value
540
541
       5)@WebFiler(filterName="DataCompressionFilter", urlPattern={"/*"})
542
          <filter-mapping>
543
             <filter-name>DataCompressionFilter</filter-name>
544
             <url-pattern>/*</url-pattern>
545
          </filter-mapping>
546
       6)@WebFiler(filterName="Security Filter", urlPattern={"/*"},
547
548
                   initParams = {
549
                      @WebInitParam(name="frequency", value="1909"),
550
                       @WebInitParam(name="resolution", value="1024")
551
                 }
552
          )
          <filter>
553
554
             <filter-name>Security Filter</filter-name>
555
             <filter-class>filterClass</filter-class>
556
             <init-param>
557
               <param-name>frequency</param-name>
558
               <param-value>1909</param-value>
559
             </init-param>
560
             <init-param>
561
               <param-name>resolution</param-name>
562
               <param-value>1024</param-value>
563
            </init-param>
564
          </filter>
565
          <filter-mapping>
566
             <filter-name>Security Filter</filter-name>
567
             <url-pattern>/*</url-pattern>
568
          </filter-mapping>
569
570
571
     12. @WebFilter Demo
572
573
       [test.jsp]
        <%@ page contentType="text/html;charset=utf-8" %>
574
575
        <!DOCTYPE html>
576
       <html lang="ko">
577
       <head>
578
          <meta charset="utf-8">
579
          <title>필터</title>
580
       </head>
581
       <body>
582
          필터 테스트
583
        </body>
```

584

</html>

```
585
586
        [LoggingFilter.java]
587
        package com.example.filter;
588
        import java.io.File;
589
        import java.io.FileNotFoundException;
590
        import java.io.IOException;
591
        import java.io.PrintWriter;
592
        import java.util.Date;
593
594
        import javax.servlet.Filter;
595
        import javax.servlet.FilterChain;
596
        import javax.servlet.FilterConfig;
597
        import javax.servlet.ServletException;
598
        import javax.servlet.ServletRequest;
599
        import javax.servlet.ServletResponse;
600
        import javax.servlet.annotation.WebFilter;
601
        import javax.servlet.annotation.WebInitParam;
602
        import javax.servlet.http.HttpServletRequest;
603
604
        @WebFilter(filterName = "LoggingFilter", urlPatterns = { "/*" },
605
             initParams = {
606
                   @WebInitParam(name = "logFileName", value = "log.txt"),
                   @WebInitParam(name = "prefix", value = "URI: ") })
607
608
        public class LoggingFilter implements Filter {
609
           private PrintWriter logger;
610
           private String prefix;
611
           @Override
612
           public void init(FilterConfig filterConfig) throws ServletException {
613
              prefix = filterConfig.getInitParameter("prefix");
              String logFileName = filterConfig.getInitParameter("logFileName");
614
615
              String appPath = filterConfig.getServletContext().getRealPath("/");
              // logFileName으로 경로가 전달되지 않았을 땐,
616
617
              // $TOMCAT HOME/bin 에 로그파일을 생성함
618
              System.out.println("logFileName:" + logFileName);
619
              try {
620
                logger = new PrintWriter(new File(appPath,
621
                      logFileName));
622
              } catch (FileNotFoundException e) {
623
                e.printStackTrace();
624
                throw new ServletException(e.getMessage());
625
              }
           }
626
627
628
           @Override
629
           public void destroy() {
              System.out.println("필터 제거 중");
630
631
              if (logger != null) {
632
                logger.close();
633
              }
           }
634
635
636
           @Override
637
           public void doFilter(ServletRequest request, ServletResponse response, FilterChain filterChain)
638
                throws IOException, ServletException {
639
              System.out.println("LoggingFilter.doFilter");
640
              HttpServletRequest httpServletRequest = (HttpServletRequest) request;
              logger.println(new Date() + " " + prefix + httpServletRequest.getRequestURI());
641
642
              logger.flush();
643
              filterChain.doFilter(request, response);
644
        }
645
```