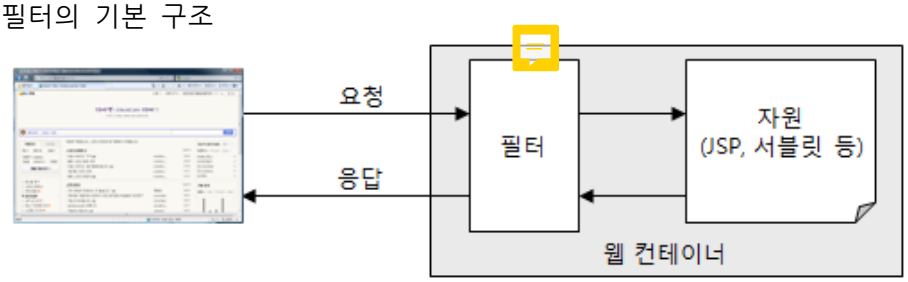
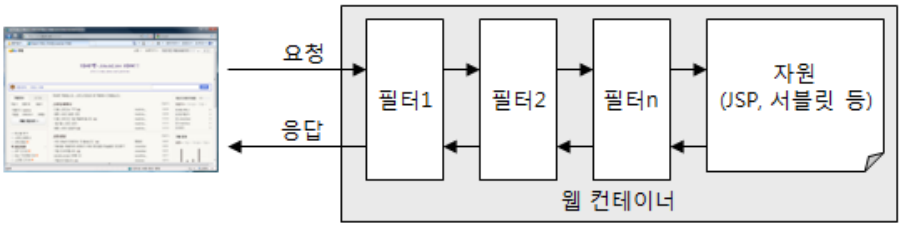
19장 필터

19.1 필터란 무엇인가?

- HTTP 요청과 응답을 변경할 수 있는 재사용 가능한 코드
- 필터의 기본 구조



- 요청과 응답의 내용을 변경 가능
- 1개 이상의 필터 연동 가능



19.2 필터의 구현

- 필터를 구현하는데 있어 핵심은 다음의 3개 타입이다.
  - javax.servlet.Filter 인터페이스: 클라이언트와 최종 자원 사이에 위치하는 필터를 나타내는 객체가 구현해야 하는 인터페이스이다.
  - javax.servlet.ServletRequestWrapper 클래스: 필터가 요청을 변경한 결과를 저장하는 레퍼이다.
  - javax.servlet.ServletResponseWrapper 클래스: 필터가 응답을 변경하기 위해 사용하는 래퍼이다.

19.2.1 Filter 인터페이스

- Filter 인터페이스의 메서드
  - public void init(FilterConfig filterConfig) throws ServletException :
    필터를 초기화할 때 호출된다.
  - public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws java.io.IOException, ServletException :
    체인을 따라 다음에 존재하는 필터로 이동한다. 체인의 가장 마지막에는 클라이언트가 요청한 최종 자원이 위치한다.
  - public void destroy() :
    필터가 웹 컨테이너에서 삭제될 때 호출된다

```
public class FirstFilter implements Filter {
   public void init(FilterConfig filterConfig) throws ServletException {
      // 필터 초기화 작업
   }

   public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
                throws IOException, ServletException {
      // 1. request 파라미터를 이용하여 요청의 필터 작업 수행
      ...
      // 2. 체인의 다음 필터 처리
      chain.doFilter(request, response);

      // 3. response를 이용하여 응답의 필터링 작업 수행
      ...
   }

   public void destroy() {
      // 주로 필터가 사용한 자원을 반납
   }
}
```

19.2.2 필터 설정하기: web.xml 이용

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns="http://xmlns.jcp.org/xml/ns/javaee"
          xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                    http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
          version="3.1">

          <filter>
                    <filter-name>NullParameter</filter-name>
                    <filter-class>filter.NullParameterFilter</filter-class>
                    <init-param>
                              <param-name>parameterNames</param-name>
                              <param-value>id,name</param-value>
                    </init-param>
          </filter>

          <filter-mapping>
                    <filter-name>NullParameter</filter-name>
                    <url-pattern>*.jsp</url-pattern>
          </filter-mapping>

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

          <filter-mapping>
                    <filter-name>LoginCheck</filter-name>
                    <url-pattern>/board/*</url-pattern>
          </filter-mapping>

</web-app>
```

19.2.3 필터 설정하기: @WebFilter 애노테이션 이용

■ web.xml 파일에 지정하지 않더라도 Filter 클래스가 @WebFilter 애노테이션을 가지면 자동으로 필터로 등록된다.

```
import javax.servlet.annotation.WebFilter;

@WebFilter(filterName = "xsltFilter".urlPatterns = {"/xml/*"})
public class XSLTFilter implements Filter {
   ... 필터 구현
}
```

19.2.4 요청 및 응답 래터 클래스

■ 래퍼 클래스를 이용하면 다음을 할 수 있다.
- 요청 정보를 변경하여 최종 자원인 서블릿/JSP/HTML/기타 자원에 전달한다.
- 최종 자원으로부터의 응답을 변경하여 새로운 응답 정보를 클라이언트에 보낸다.

■ NullParameterRequestWrapper 클래스는 요청 래퍼 클래스다. checkNull() 메서드를 통해서 빈 문자열을 기본값으로 지정할 파라미터의 목록을 전달받아 처리한다.

[chap19/src/filter/NullParameterRequestWrapper.java]

```
package filter;

import java.util.Collections;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletRequestWrapper;

public class NullParameterRequestWrapper extends HttpServletRequestWrapper {

    private Map<String, String[]> parameterMap = null;

    public NullParameterRequestWrapper(HttpServletRequest request) {
        super(request);
        parameterMap =
                new HashMap<String, String[]>(request.getParameterMap());
    }

    public void checkNull(String[] parameterNames) {
        for (int i = 0; i < parameterNames.length; i++) {
            if (!parameterMap.containsKey(parameterNames[i])) {
                String[] values = new String[] { "" };
                parameterMap.put(parameterNames[i], values);
            }
        }
    }

    @Override
    public String getParameter(String name) {
        String[] values = getParameterValues(name);
        if (values != null && values.length > 0)
            return values[0];
        return null;
    }

    @Override
    public Map<String, String[]> getParameterMap() {
```

```
        return parameterMap;
    }

    @Override
    public Enumeration<String> getParameterNames() {
        return Collections.enumeration(parameterMap.keySet());
    }

    @Override
    public String[] getParameterValues(String name) {
        return (String[]) parameterMap.get(name);
    }

}
```

다음은 NullParameterRequestWrapper를 사용해서 요청 파라미터를 처리하는 필터 클래스이다.

[chap19/src/filter/NullParameterFilter.java]

```
package filter;

import java.io.IOException;
import java.util.StringTokenizer;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;

public class NullParameterFilter implements Filter {

    private String[] parameterNames = null;

    @Override
    public void init(FilterConfig config) throws ServletException {
        String names = config.getInitParameter("parameterNames");
        StringTokenizer st = new StringTokenizer(names, ", ");
        parameterNames = new String[st.countTokens()];

        for (int i = 0; st.hasMoreTokens(); i++) {
            parameterNames[i] = st.nextToken();
        }
    }

    @Override
    public void doFilter(ServletRequest request, ServletResponse response,
            FilterChain chain) throws IOException, ServletException {
        NullParameterRequestWrapper requestWrapper =
            new NullParameterRequestWrapper((HttpServletRequest) request);
        requestWrapper.checkNull(parameterNames);

        chain.doFilter(requestWrapper, response);
    }

    @Override
    public void destroy() {
    }
}
```

[chap19/WebContent/WEB-INF/web.xml]

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns="http://xmlns.jcp.org/xml/ns/javaee"
          xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                    http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
          version="3.1">

    <filter>
        <filter-name>NullParameter</filter-name>
        <filter-class>filter.NullParameterFilter</filter-class>
        <init-param>
            <param-name>parameterNames</param-name>
            <param-value>id,name</param-value>
        </init-param>
    </filter>

          <filter-mapping>
                    <filter-name>NullParameter</filter-name>
                    <url-pattern>*.jsp</url-pattern>
          </filter-mapping>

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

          <filter-mapping>
                    <filter-name>LoginCheck</filter-name>
                    <url-pattern>/board/*</url-pattern>
          </filter-mapping>

</web-app>
```

[chap19/WebContent/nullParam.jsp]

```
<%@ page contentType = "text/html; charset=utf-8" %>
<html>
<head><title>NullParameterFilter 테스트</title></head>
<body>
id 파라미터 : <%= request.getParameter("id") %> <br>
name 파라미터 : <%= request.getParameter("name") %> <br>
member 파라미터 : <%= request.getParameter("member") %> <br>
</body>
</html>
```

19.3 필터의 응용

■ 필터를 사용하는 방법에는 제한이 없으며, 필터의 특징을 잘 활용하느냐에 따라서 필터의 응용 범위가 달라질 수 있다. 보통 다음과 같은 기능에 필터를 적용한다.

- 사용자 인증
- 캐싱 필터
- 자원 접근에 대한 로깅
- 응답 데이터 변환(HTML 변환, 응답 헤더 변환, 데이터 암호화 등)
- 공통 기능 실행

19.3.1 로그인 검사 필터

■ 간단하게 Session에 "MEMBER" 속성이 존재하면 로그인한 것으로 판단하는 필터인 LoginCheckFilter 클래스를 작성한다.

[chap19/src/filter/LoginCheckFilter.java]

```
01  package filter;
02
03  import java.io.IOException;
04
05  import javax.servlet.Filter;
06  import javax.servlet.FilterChain;
07  import javax.servlet.FilterConfig;
08  import javax.servlet.RequestDispatcher;
09  import javax.servlet.ServletException;
10  import javax.servlet.ServletRequest;
11  import javax.servlet.ServletResponse;
12  import javax.servlet.http.HttpServletRequest;
13  import javax.servlet.http.HttpSession;
14
15  public class LoginCheckFilter implements Filter {
16      @Override
17      public void init(FilterConfig config) throws ServletException {
18      }
19
20      @Override
21      public void doFilter(ServletRequest request, ServletResponse response,
22              FilterChain chain) throws IOException, ServletException {
23          HttpServletRequest httpRequest = (HttpServletRequest) request;
24          HttpSession session = httpRequest.getSession(false);
25
26          boolean login = false;
27          if (session != null) {
28              if (session.getAttribute("MEMBER") != null) {
29                  login = true;
30              }
31          }
32          if (login) {
33              chain.doFilter(request, response);
34          } else {
35              RequestDispatcher dispatcher = request
36                      .getRequestDispatcher("/loginForm.jsp");
37              dispatcher.forward(request, response);
38          }
39      }
40
41      @Override
42      public void destroy() {
43      }
44  }
```

[chap19/WebContent/WEB-INF/web.xml]

```
01  <?xml version="1.0" encoding="UTF-8"?>
02  <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
03           xmlns="http://xmlns.jcp.org/xml/ns/javaee"
04           xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
05                    http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
06           version="3.1">
07
08           <filter>
09                    <filter-name>NullParameter</filter-name>
```

```
10                      <filter-class>filter.NullParameterFilter</filter-class>
11                      <init-param>
12                              <param-name>parameterNames</param-name>
13                              <param-value>id,name</param-value>
14                      </init-param>
15              </filter>
16
17              <filter-mapping>
18                      <filter-name>NullParameter</filter-name>
19                      <url-pattern>*.jsp</url-pattern>
20              </filter-mapping>
21
22              <filter>
23                      <filter-name>LoginCheck</filter-name>
24                      <filter-class>filter.LoginCheckFilter</filter-class>
25              </filter>
26
27              <filter-mapping>
28                      <filter-name>LoginCheck</filter-name>
29                      <url-pattern>/board/*</url-pattern>
30              </filter-mapping>
31
32      </web-app>
```

~~19.3.2 XSLT 필터~~

(내용 없음)

19.3.3 캐릭터 인코딩 필터

■ 요청 파라미터의 글자를 올바르게 처리하기 위해 캐릭터 인코딩을 설정한다. 따라서 요청 파라미터를 사용하는 모든 JSP 코드마다 캐릭터 인코딩을 설정해야 한다. 그러나 이런 동일한 코드가 여러 곳에 중복해서 출현하는 것은 좋은 방법이 아니다.

```
<% request.setCharacterEncoding("utf-8"); %>
```

[chap19/src/util/CharacterEncodingFilter.java]

```
01    package util;
02
03    import java.io.IOException;
04
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 CharacterEncodingFilter implements Filter {
13
14              private String encoding;
15
16              @Override
17              public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain)
18                              throws IOException, ServletException {
```

```
                        req.setCharacterEncoding(encoding);
                        chain.doFilter(req, res);
                }

                @Override
                public void init(FilterConfig config) throws ServletException {
                        encoding = config.getInitParameter("encoding");
                        if (encoding == null) {
                                encoding = "UTF-8";
                        }
                }

                @Override
                public void destroy() {
                }

    }
```

```
// web.xml에 다음과 같은 설정을 추가한다.
          <filter>
                    <filter-name>encodingFilter</filter-name>
                    <filter-class>util.CharacterEncodingFilter</filter-class>it
                    <init-param>
                       <param-name>encoding</param-name>
                       <param-value>utf-8</param-value>
                    </init-param>
          </filter>

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