CHAP 23.
필터

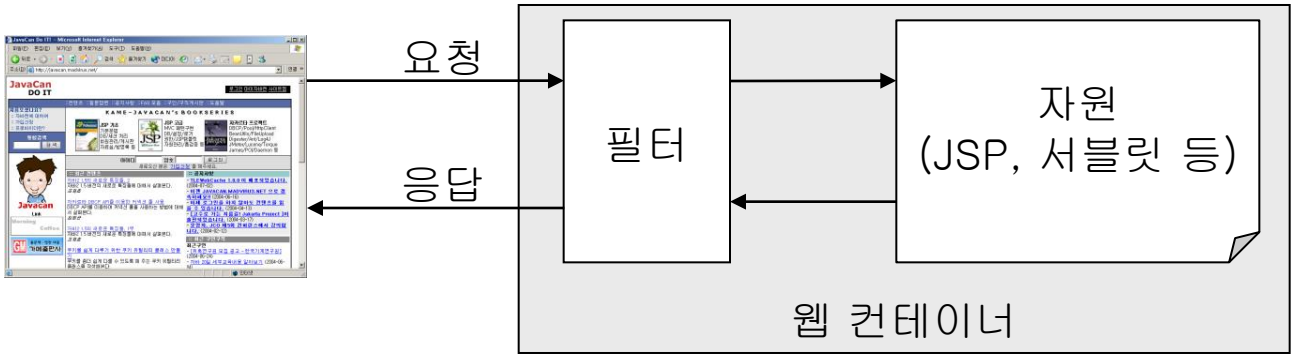
CONTENTS

- 필터란 무엇인가?
- Filter 인터페이스
- 요청/응답 래퍼 클래스 작성
- 필터의 응용

필터란 무엇인가?

"필터란 HTTP 요청과 응답을 변경할 수 있는 재사용 가능한 코드"

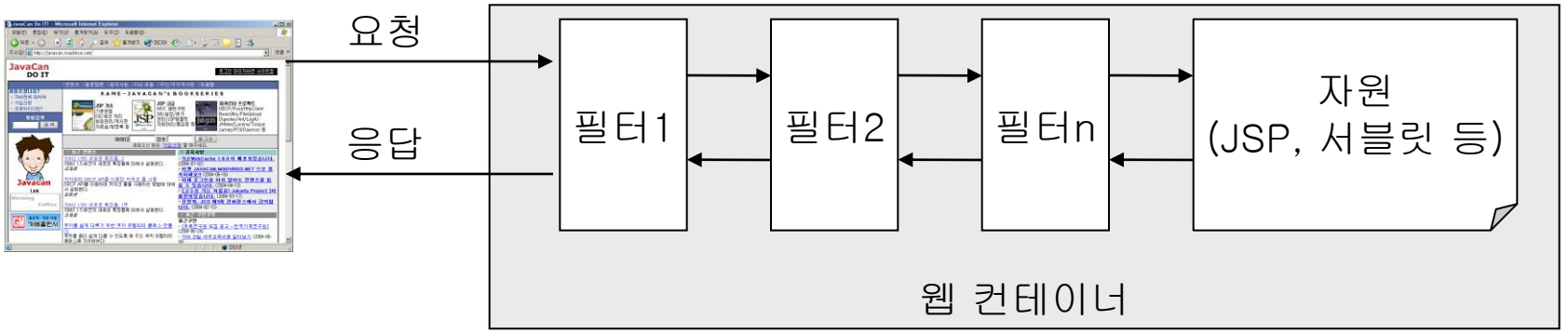
필터의 기본 구성



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

필터 체인

필터가 연결된 필터 체인(Chain)



- 여러 개의 필터가 모여 필터 체인을 이룬다.
- 클라이언트의 요청은 차례대로 필터를 통과한 뒤 최종 자원에 도달한다.
- 자원의 응답은 요청시의 역순으로 필터를 통과한 뒤 클라이언트에 도달한다.

필터의 구현: Filter 인터페이스

필터로 사용될 클래스는 Filter 인터페이스를 implements해야 한다.

Filter 인터페이스의 주요 메소드

- public void init(FilterConfig filterConfig) throws ServletException
  필터를 웹 콘테이너내에 생성한 후 초기화할 때 호출한다.

- 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 콘테이너에서 삭제될 때 호출된다.

필터의 구현

Code

```
public class FirstFilter implements javax.servlet.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 파리미터를 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
    }
}
```

1. request 파리미터를 이용하여 클라이언트의 요청 필터링
   1 단계에서는 RequestWrapper 클래스를 사용하여 클라이언트의 요청을 변경한다.
2. chain.doFilter() 메소드 호출
   2 단계에서는 요청의 필터링 결과를 다음 필터에 전달한다.
3. response 파리미터를 사용하여 클라이트로 가는 응답 필터링
   3 단계에서는 체인을 통해서 전달된 응답 데이터를 변경하여 그 결과를 클라이언트에 전송한다.

필터 적용

Code

```
<? xml version="1.0" encoding="euc-kr"?>

<web-app ...>

    <filter>
        <filter-name>FilterName</filter-name>
        <filter-class>javacan.filter.FilterClass</filter-class>
        <init-param>
            <param-name>paramName</param-name>
            <param-value>value</param-value>
        </init-param>
    </filter>

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

    ...
</web-app>
```

web.xml 파일에 사용할 필터 및 필터를 적용할 URL 패턴을 명시

요청 응답 래퍼 클래스

- *클라이언트의 요청을 변경하기 위해서는 요청 래퍼 클래스(HttpServletRequestWrapper)를 사용한다.*

- *JSP/서블릿 등의 응답 결과를 변경하기 위해서는 응답 래퍼 클래스(HttpServletResponseWrapper)를 사용하면 된다*

요청 래퍼 클래스 작성

Code

```
public class SomeRequestWrapper extends HttpServletRequestWrapper {

    // HttpServletRequest 중에서 변경할 메소드를 직접 구현

}
```

Example

```
public class NullParameterRequestWrapper extends HttpServletRequestWrapper {
    public NullParameterRequestWrapper(HttpServletRequest request) {
        super(request);
        parameterMap = new java.util.HashMap(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);
            }
        }
    }
    public String getParameter(String name) {
        String[] values = getParameterValues(name);
        if (values != null && values.length > 0) return values[0];
        return null;
    }
    ...
}
```

필터에서 요청 래퍼 클래스 사용

Example

```
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);
}
```

요청 래퍼 클래스 생성

요청 래퍼 객체를 필터체인에 전달

요청 필터를 통해서 전달되는 요청 래퍼 객체는 최종적으로 JSP의 request 기본객체로 사용된다.

응답 래퍼 클래스 작성

Code

```
public class SomeResponseWrapper extends HttpServletResponseWrapper {

    // HttpServletResponse 중에서 변경할 메소드를 직접 구현
    // 주로 응답을 변경하기 위해 응답 결과를 저장하는 코드가 온다

}
```

필터의 응용

- 데이터 변환(파일 압축/데이터 암호화/이미지 변환 등)
- XSL/T를 이용한 XML 문서 변경
- 사용자 인증
- 캐싱 필터
- 자원 접근에 대한 로깅