# 필터(filter) - 기초

## 필터란

JSP / Servlet 등을 실행하기 이전에 요청이 올바른지 또는 자원에 접근할 수 있는 권한을 가지고 있는 지의 여부를 미리 처리

HTTP 요청과 응답을 변경할 수 있는 재사용 가능한 클래스



클라이언트에서 오는 요청(request)과 최종 자원(JSP, 서블릿, 기타 자원) **사이에 위치**하여 클라이언트의 요청 정보를 알맞게 변경.

최종 자원과 클라이언트로가는 응답(response) **사이에 위치**하여 최종 자원의 요청 결과를 알맞게 변경.

## 필터 체인



여러 개의 필터가 모여 하나의 필터체인(chain)을 형성.

필터체인의 경우 요청 정보가 변경에 변경을 거듭.

응답 정보의 필터 적용 순서가 요청 때와는 반대로 적용.

필터는 그 설정에 따라 사용자의 요청과 다른 결과를 전송할 수도 있음.

사용자 인증이나 권한 검사와 같은 기능을 구현할 때 용이하게 사용할 수 있음.

## 필터 구현

javax.servlet.Filter 인터페이스를 사용하여 새로운 필터 정의

[필요시 래퍼 클래스 작성]

javax.serlvet.ServletRequestWrapper 클래스: 필터가 요청을 변경한 결과를 저장하는 래퍼 클래스

javax.servlet.ServletResponseWrapper 클래스: 필터가 응답을 변경하기 위해 사용하는 래퍼 클래스

## 필터 클래스 구현

public void init() :필터를 초기화할 때 호출

public void doFilter() : 필터 기능을 수행. chain을 이용해서 체인의 다음 필터로 처리를 전달할 수 있다.

public void destroy() : 필터가 웹컨테이너에서 삭제될 때 호출



예) filter.NullParameterFilter 클래스 작성

|  |
| --- |
| package filter;  import java.io.IOException;…  public class NullParameterFilter implements Filter {  private String[] parameterNames = null; //필터 처리할 파라미터 저장  @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() {  }  } |

## ServletRequestWrapper 클래스 구현 – new > Filter 로 생성

filter.NullParameterRequestWrapper 클래스

|  |
| --- |
| package filter;  import java.util.Collections;…  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) {//필터 기능 구현  for (int i = 0; i < parameterNames.length; i++) {  if (!parameterMap.containsKey(parameterNames[i])) {  String[] values = new String[] { "없음" };  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);**  **}**  } |

ServletRequestWrapper 클래스의 getParameter, getParameterMap, getParameterNames, getParameterValues를 재정의(Source > Override/Implement Methods)

## 필터 등록

web.xml에 등록

|  |
| --- |
| <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> |

## 화면 처리

nullParam.jsp

|  |
| --- |
| <%@ page contentType = *"text/html; charset=utf-8"* %>  <html>  <head><title>NullParameterFilter 테스트</title></head>  <body>  id 파라미터 : ${param.id}<br>  name 파라미터 : ${param.name}<br>  </body>  </html> |