8253jang@daum.net
장희정

Filter란?

Filter는 클라이언트 요청이 Servlet이나 JSP와 같은 웹 리소스에 도달하기 전에 처리되거나, 응답이 클라이언트에게 전달되기 전에 처리되는 로직을 추가할 수 있는 기능으로 주로 **요청 및 응답을 가로채어 사전, 사후 처리를 담당한다**.

☞ Filter의 주요 역할:

1) 요청/응답 전처리 및 후처리:
필터는 클라이언트 요청을 가로채어 필요한 전처리를 수행할 수 있다. 예를 들어, 사용자가 인증되지 않은 상태라면 요청을 거부 하고 인증페이지로 이동 시킬 수 있다.
응답이 클라이언트에게 전달되기 전에 후처리를 수행할 수도 있다. 예를 들어, 응답 데이터의 내용이나 헤더를 수정

2)로깅 및 모니터링:
요청 및 응답에 대한 로깅을 통해 시스템의 상태를 모니터링 할 수 있다. 필터를 사용하여 응답 시간, 요청 URL 등을 기록하는 로직을 추가.

3) 인코딩 설정:
요청 데이터의 인코딩 방식을 설정하거나, 응답 데이터의 인코딩을 변경할 수 있다.

4)보안:
필터는 사용자의 인증 및 권한을 확인하는 데 많이 사용된다. 특정 URL 패턴에 대해 사용자가 접근할 수 있는지 여부를 검증하고, 필요할 경우 인증 페이지로 리다이렉트할 수 있다. Spring Security는 Filter 기반으로 만들어져 있다.

☞ Filter의 작동 방식

필터 체인(Filter Chain) : 여러 개의 필터가 존재할 수 있으며, 이 필터들은 체인처럼 연결되어 작동한다. 즉, 하나의 필터가 실행된 후 다음 필터로 제어를 넘기거나, 필터 체인을 중단할 수 있다.

Filter 인터페이스

javax.servlet.Filter 인터페이스를 구현하여 필터를 정의한다. 이 인터페이스는 세 가지 메서드를 가지고 있다.

1) init() : 필터 초기화 로직을 정희
2) doFilter() : 필터의 핵심 작업이 이루어지는 메서드로, 요청/응답을 처리하거나 필터 체인에서 다음 필터로 제어를 넘긴다.
3) destroy() : 필터가 종료될 때 호출된다.

8253jang@daum.net
장희정

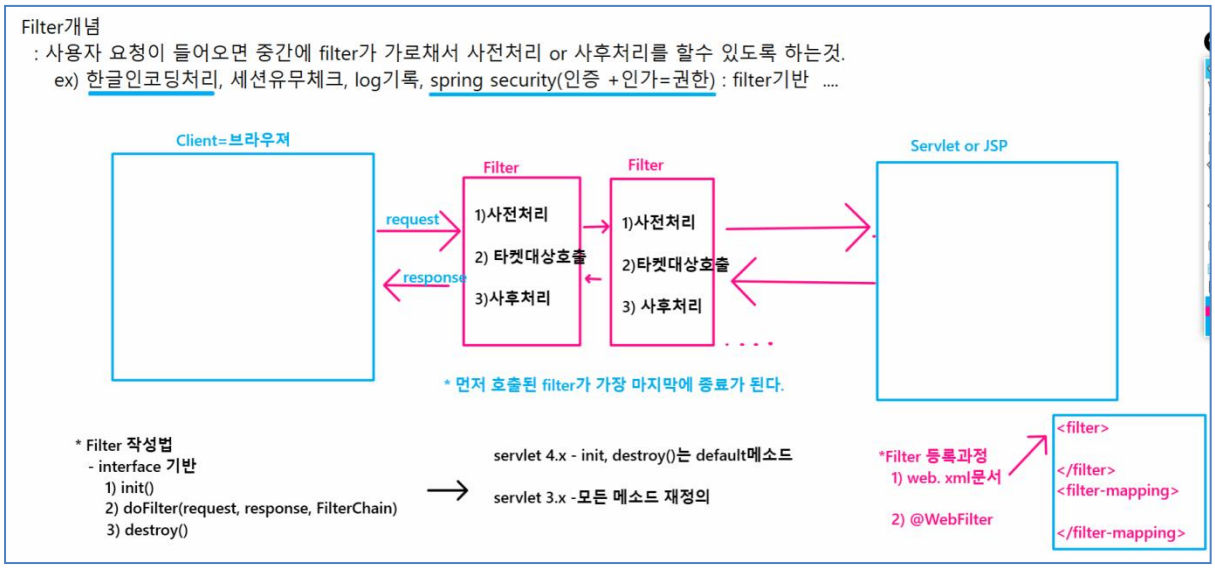
```
public class LoggingFilter implements Filter {

    @Override
    public void init(FilterConfig filterConfig) throws ServletException {
        // 필터 초기화
    }

    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
            throws IOException, ServletException {

        //사전처리

        // 다음 필터로 요청을 넘기거나 서블릿으로 전달
        chain.doFilter(request, response);

        //사후처리
    }

    @Override
    public void destroy() {
        // 필터 종료 시 자원 해제
    }
}
```

☞ Filter 를 사용하는 이유

공통 기능 분리 : 여러 서블릿이나 JSP 에서 반복적으로 사용되는 **보안, 로깅, 인코딩 등**의 기능을 필터로 추출하여 코드 중복을 줄일 수 있다

유연한 처리 : 요청을 처리하는 서블릿 전에 사전 처리를 할 수 있어, 모든 요청에 대해 일관된 처리가 가능하다

8253jang@daum.net
장희정



[그림 11-5] 연속해서 호출되는 필터들로 이루어지는 필터 체인

☞ Filter 등록방법

1) web.xml

```
<web-app>
    <filter>
            ← 필터를 등록하는 엘리먼트
    </filter>
    <filter-mapping>
            ← 필터를 적용할 웹 컴포넌트를 지정하는 엘리먼트
    </filter-mapping>
</web-app>
```

Ex)

8253jang@daum.net
장희정

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

같은 웹 애플리케이션 디렉터리에 있는
모든 웹 컴포넌트를 가리키는 URL 패턴

- <url-pattern> 엘리먼트 안에 계층적인 URL 경로명을 쓸 수도 있다. 이 경로명은 웹 애플리케이션 디렉터리의 루트 디렉터리를 의미하는 슬래시(/) 문자로 시작해야 한다.

```
<filter-mapping>
    <filter-name>simple-filter</filter-name>
    <url-pattern>/sub1/*</url-pattern>
</filter-mapping>
```

/sub1/이라는 URL 경로명으로 시작하는
모든 웹 컴포넌트를 가리키는 URL 패턴

- <url-pattern> 엘리먼트 안에 계층적인 URL 경로명을 쓸 때는 와일드카드(*)문자와 파일 확장자를 함께 쓰면 안 된다.

```
<filter-mapping>
    <filter-name>simple-filter</filter-name>
    <url-pattern>/sub1/*.jsp</url-pattern>
</filter-mapping>
```

잘못된 URL 패턴

- <filter-mapping> 엘리먼트 안에 여러 개의 <url-pattern> 서브엘리먼트를 쓸 수도 있다.

```
<filter-mapping>
    <filter-name>simple-filter</filter-name>
    <url-pattern>/sub1/*</url-pattern>
    <url-pattern>/sub2/*</url-pattern>
</filter-mapping>
```

URL 경로명이 /sub1/ 또는 /sub2/로 시작하는
모든 웹 컴포넌트에 필터를 적용합니다

- <filter-mapping> 엘리먼트 안에 여러 개의 <servlet-name> 엘리먼트를 쓸 수도 있고, <servlet-name>과 <url-pattern> 엘리먼트를 혼용해서 쓸 수도 있다.

```
<filter-mapping>
    <filter-name>simple-filter</filter-name>
    <url-pattern>/sub1/*</url-pattern>
    <url-pattern>/sub2/*</url-pattern>
    <servlet-name>hello-servlet</servlet-name>
</filter-mapping>
```

URL 경로명이 /sub1/ 또는 /sub2/로 시작하는 웹 컴포넌트와
이름이 hello-servlet인 서블릿에 필터를 적용합니다

8253jang@daum.net
장희정

2) @annotation

```
@WebFilter(urlPatterns = "/*")
public class LogFilter implements Filter {

}
```