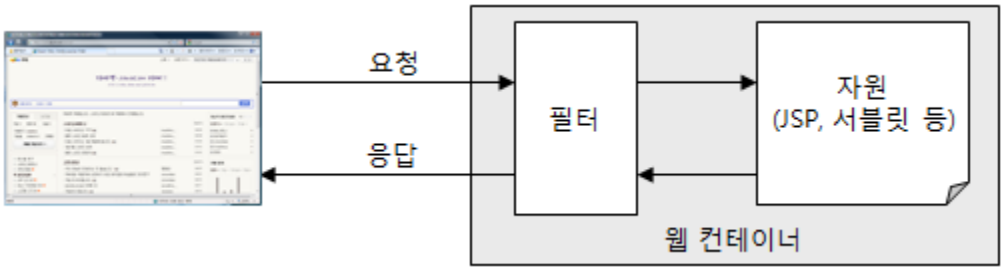
TOC

- 필터 활용
  - 필터란
  - 필터의 구현
  - 필터 응용

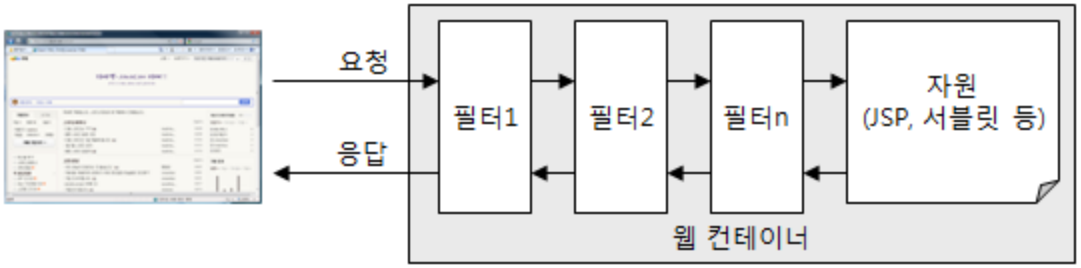
- 웹 애플리케이션 이벤트 처리
  - ServletContextListener
  - 리스너 실행 순서
  - 예외 처리

필터

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



- 요청의 내용을 변경하거나 응답의 내용을 변경 가능
- 1개 이상의 필터 연동 가능



필터 구현

- Filter 인터페이스 사용
- 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.

필터 구현

- doFilter() 메서드에서 필터 기능 구현

```
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
    ...
}
```

필터 설정

- web.xml에 URL 별 매핑 설정 추가

```
<web-app ...>

    <filter>
        <filter-name>FilterName</filter-name>
        <filter-class>javacan.filter.FileClass</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>
```

필터 설정

- <dispatcher> 를 통한 필터 적용 대상 지정

```
<filter-mapping>
    <filter-name>AuthCheckFilter</filter-name>
    <servlet-name>FileDownload</servlet-name>
    <dispatcher>INCLUDE</dispatcher>
</filter-mapping>
```

  - <dispatcher>의 값
    - REQUEST - 클라이언트의 요청인 경우에 필터를 사용 (기본값)
    - FORWARD - forward()를 통해서 제어를 이동하는 경우에 필터를 사용
    - INCLUDE - include()를 통해서 포함하는 경우에 필터를 사용

요청 및 응답 래퍼

- 요청이나 응답을 변경할 때 사용
  - 요청 래퍼: HttpServletRequestWrapper
  - 응답 래퍼: HttpServletResponseWrapper

- doFilter()에서 chain.doFilter()를 호출할 때, 래퍼 객체를 전달해 줌으로써 다음 작업 가능
  - 요청 변경: 파라미터 변경, 헤더 변경, 전송 데이터 변경 등
  - 응답 변경: 응답 데이터 변경, 압축 등

필터의 응용

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

ServletContextListener

- 웹 어플리케이션의 시작 이벤트나 종료 이벤트를 처리
- 웹 컨테이너는 ServletContextListener의 특정 메서드를 호출함
- ServletContextListener 인터페이스의 이벤트 처리 메서드
  - public void contextInitialized(ServletContextEvent sce)
    웹 어플리케이션이 초기화될 때 호출된다.
  - public void contextDestroyed(ServletContextEvent sce)
    웹 어플리케이션이 종료될 때 호출된다.

web.xml 파일에 리스너 등록

- <listener> 태그 이용

```
<web-app ...>
    <listener>
        <listener-class>kame.jdbc.loader.DBCPInitListener</listener-class>
    </listener>

    <listener>
        <listener-class>kame.chap22.CodeInitListener</listener-class>
    </listener>
    ...
</web-app>
```

ServletContextEvent로부터 필요 정보 조회

- contextInitialized()와 contextDestroyed() 메서드에 전달되는 객체
- ServletContext를 구해주는 getServletContext() 메서드 제공
  - ServletContext.getInitParameter() 메서드를 이용해서 web.xml 파일에 등록한 초기화 파라미터 값 조회

```
<web-app ...>
    <context-param>
        <param-name>jdbcdriver</param-name>
        <param-value>com.mysql.jdbc.Driver</param-value>
    </context-param>
</web-app>
```

```
public class DBCPInitListener implements ServletContextListener {
    public void contextInitialized(ServletContextEvent sce) {
        try {
            ServletContext context = sce.getServletContext();
            String drivers = context.getInitParameter("jdbcdriver");
        ...
    }
}
```

실행 순서 & 예외 처리

- 한 개 이상의 리스너 등록 가능
  - contextInitialized() 메서드는 등록된 순서대로 실행
  - contextDestroyed() 메서드는 등록된 반대 순서대로 실행

- 리스너의 메서드에 try - catch로 예외를 잡은 뒤, RuntimeException을 발생시키도록 함
  - 리스너가 예외를 발생할 경우 웹 어플리케이션 시작에 실패함

```
public void contextInitialized(ServletContextEvent sce) {
    try {
        …
        …
    } catch (Exception ex) {
        throw new RuntimeException(ex);
    }
}
```