프로젝트로 배우는

자바 웹 프로그래밍

Servlet, JSP, JDBC

황희정 지음

Chapter 13. 리스너와 필터

한빛아카데미 Hanbit Academy, Inc.

목차

학습목표

- 고급 웹 프로그래밍 기술인 리스너와 필터의 개념을 이해하고 웹 애플리케이션 초기화 매개변수와의 연계 방안을 배운다.
- 리스너 종류를 배우고 리스너 프로그래밍 기법을 익힌다.
- 필터의 활용 방안을 알아보고 프로그래밍 기법을 익힌다.

01. 웹 애플리케이션 초기화 매개변수 관리

1. 초기화 매개변수의 개요

톰캣

- 웹 애플리케이션 초기화 매개변수는 웹 애플리케이션이 컨테이너에 의해 구동될 때 로딩되는 정보로서, 웹 애플리케이션 전반에 걸쳐 공통적으로 참조하는 값을 설정하는 용도로 사용한다.
- 일반적으로 프로그램을 동작시킬 때 필요한 기본 정보는 소스코드 내에 하드코딩하지 않고 별도의 환경설정 파일을 통해 제공한다.
- 예를 들어, 데이터베이스 접속 주소나 환경설정 파일의 위치 같은 정보를 프로그램 내에 둘 경우, 해당 내용을 필요로 하는 소스코드마다 동일한 내용을 넣어야 한다.
- 따라서 파일이 분산될 경우, 변경에 따른 관리가 어렵고, 변경이 발생했을 경우 해당 클래스들을 모두 수정한 후 다시 컴파일해야 하는 등의 문제가 발생한다.
- 초기화 매개변수는 웹애플리케이션 전체 혹은 특정 서블릿에 대한 값을 설정 할 수 있다.

01. 웹 애플리케이션 초기화 매개변수 관리

원리를 알면 IT가 맛있다
IT COOKBOOK

- 서블릿 초기화 매개변수는 ServletConfig를 통해 접근할 수 있다.
- 웹 애플리케이션 초기화 매개변수는 ServletContext를 통해 접근 할 수 있다.
- 초기 값의 설정은 web.xml에서 한다.

[표 13-1] ServletConfig와 ServletContext의 비교

구분	적용 범위	web.xml 설정
ServletConfig A:1:1	해당 서블릿에서만 사용할 수 있다.	<servlet> <init-param> <param-name> </param-name> <param-value> </param-value> </init-param> </servlet>
ServletContext 전역	동일 웹 애플리케이션 내 모든 서블릿(혹은 JSP)에서 사용할 수 있다.	<web-app> <context-param> <param-name> </param-name> <param-value> </param-value> </context-param> </web-app>

01. 웹 애플리케이션 초기화 매개변수 관리

ServletConfig를 사용하는 경우

- 서블릿 단위로 설정하기 때문에 해당 서블릿에서만 참조할 수 있다. 서블릿 동작에 필요한 정보를 제공하기 위한 목적으로 사용한다.
- 서블릿 코드 내에서는 다음과 같이 값을 참조 할 수 있다.

```
getInitParameter("param-name에서의 설정 이름")
```

- 서블릿 3.0 이상에서는 web.xml 이외에도 서블릿 클래스에 애너테이션 기반으로 초기값을 설정할 수 있다.

```
@WebServlet(name="test",
    urlPatterns={"/test"},
    initParams={@InitParam(name="n1", value="v1"), @InitParam(name="n2", value="v2")})
```

01. 웹 애플리케이션 초기화 매개변수 관리

ServletContext를 사용하는 경우

- 웹 애플리케이션 내 모든 JSP 혹은 서블릿에서의 참조가 가능하다.
- 서블릿 코드 내에서는 다음과 같이 값을 참조할 수 있다.

```
getServletContext().getInitParameter("param-name에서의 설정 이름")
```

- 웹 애플리케이션 초기화 매개변수는 web.xml에서 <context-param> 태그를 이용해 설정할 수 있다.

```
<web-app>
    <context-param>
        <param-name> </param-name>
        <param-value> </param-value>
    </context-param>
</web-app>
```

02. [기본실습]초기화 매개변수 관리

원리를 알면 IT가 맛있다
IT COOKBOOK

1. 실습 개요

- 이 실습은 web.xml 파일에 초기화 매개변수를 설정해두고 서블릿 및 JSP에서 이를 참조하는 방법을 살펴보는 것이다.
- 실습은 ServletConfig 및 ServletContext 초기화 매개변수를 설정하고, 서블릿과 JSP에서 해당 속성을 참조해 출력하도록 구성되어 있다.

[표 13-2] 프로그램 소스 목록

디렉터리	파일 이름	설명
src₩jspbook₩ch13	PropertyServlet.java	서블릿 초기화 매개변수를 실습하기 위한 서블릿 코드다.
WebContent₩ch13	property.jsp	서블릿 초기화 매개변수를 확인하기 위한 jsp 파일이다. 코드 수정을 통해 ServletContext 초기화 매개변수를 확인하는 데도 사용할 수 있다.

02. [기본실습]초기화 매개변수 관리

원리를 알면 IT가 맛있다
IT COOKBOOK

PropertyServlet 만들기

- **PropertyServlet.java** – 교재 p.548 ~ 549 참고
- 애너테이션을 이용해 서블릿 초기화 매개변수를 설정함.
- 애너테이션은 프로그램 소스에 들어가지만 클래스 파일 외부에서 접근이 가능함.
- 서블릿 접근을 위한 urlPatterns와 초기화 매개변수인 initParams를 설정함.
- 설정값은 name, value 쌍으로 구성

```
07 @WebServlet(
        urlPatterns = { "/PropertyServlet" },
        initParams = { @WebInitParam(name = "name1", value = "user1"),
                       @WebInitParam(name = "name2", value = "user2")})
    public class PropertyServlet extends javax.servlet.http.HttpServlet
```

- 초기화 매개변수는 getInitParameter 메서드를 통해 가져올 수 있음.

```
33      out.println("name1 : "+ getInitParameter("name1")+"<BR>");
34      out.println("name2 : "+ getInitParameter("name2"));
```

02. [기본실습]초기화 매개변수 관리

원리를 알면 IT가 맛있다
IT COOKBOOK

서블릿 3.0 이하에서 초기화 매개변수 설정하기

ServletContext 초기화 매개변수 설정하기

- **서블릿 3.0 이하에서의 ServletContext 초기화 매개변수 설정** – 교재 p.550 참고
- 서블릿 3.0 이하에서는 web.xml 파일에 서블릿 초기화 매개변수를 설정 함.

```
05    <servlet-class> jspbook.ch13.PropertyServlet</servlet-class>
06        <init-param>
07            <param-name>name1</param-name>
08            <param-value>user1</param-value>
09        </init-param>
10        <init-param>
11            <param-name>name2</param-name>
12            <param-value>user2</param-value>
13        </init-param>
```

02. [기본실습]초기화 매개변수 관리

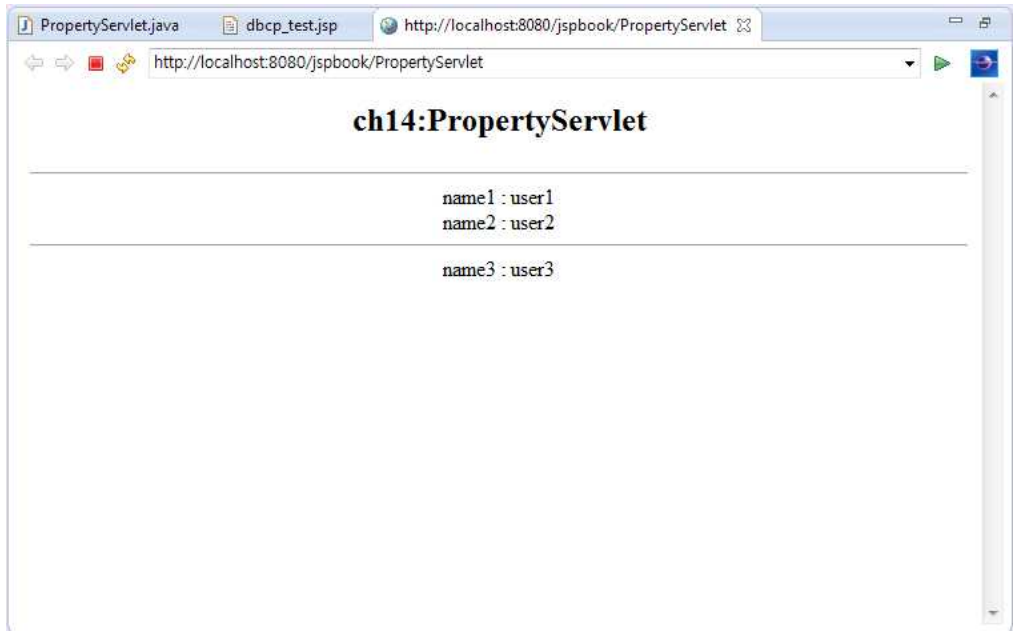
원리를 알면 IT가 맛있다
IT COOKBOOK

- **Servlet 초기화 매개변수 설정하기**
  - **서블릿 3.0 이하에서의 서블릿 초기화 매개변수 설정** – 교재 p.551 참고
  - 웹 애플리케이션 초기화 매개변수 역시 web.xml 에 다음과 같이 설정

```
01 <context-param>
02     <param-name>workspace</param-name>
03     <param-value>c:\dev\workspace</param-value>
04 </context-param>
05 <context-param>
06     <param-name>name3</param-name>
07     <param-value>user3</param-value>
08 </context-param>
```

02. [기본실습]초기화 매개변수 관리

[그림 13-1] 실행 결과

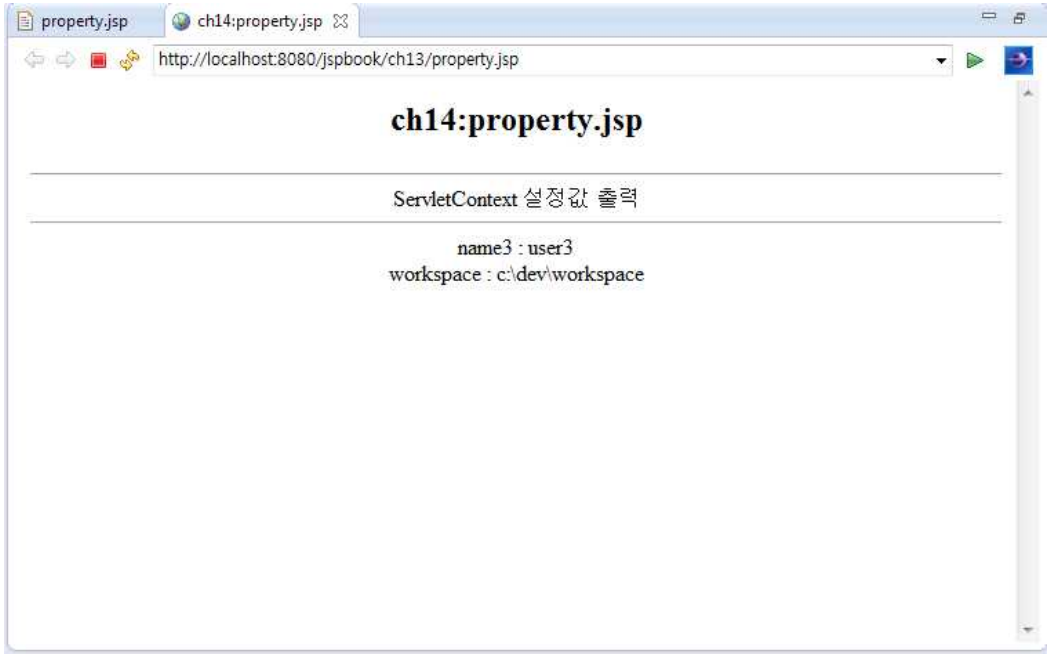


[그림 13-2] PropertyServlet.java 실행 결과

02. [기본실습]초기화 매개변수 관리

Property.jsp 작성하기

- **property.jsp** - 교재 p.552 참고



[그림 13-3] property.jsp 실행 결과

03. 리스너

1. 리스너란

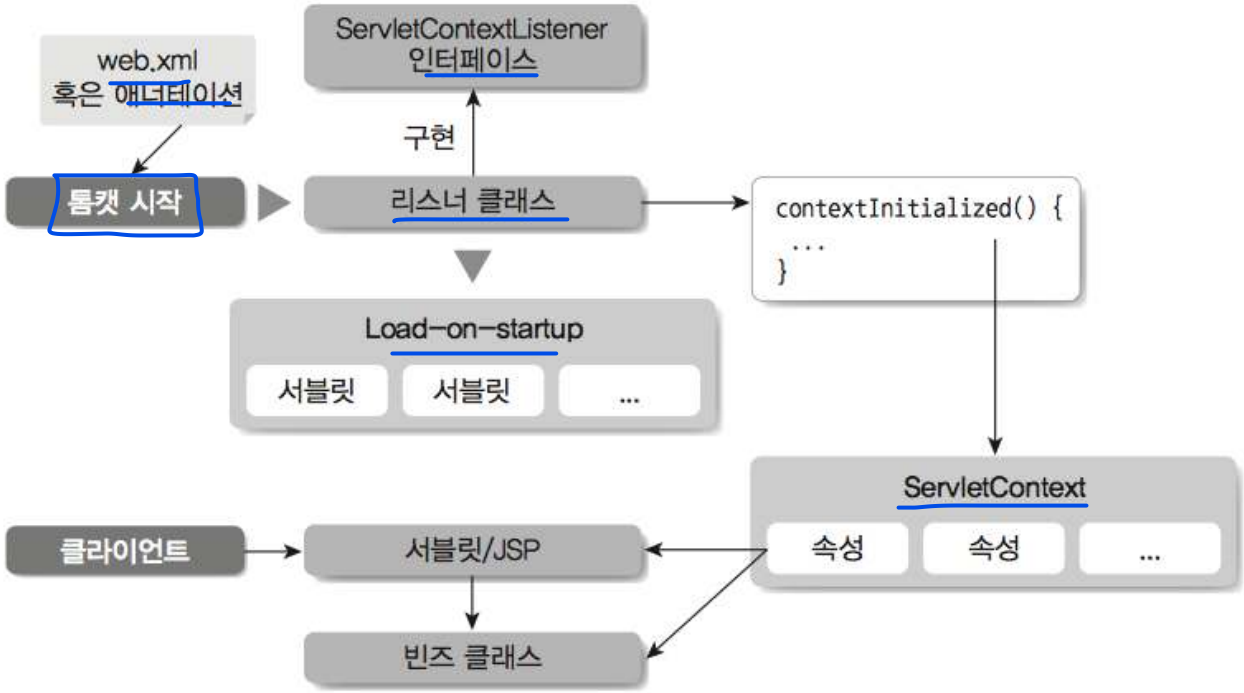
- 리스너는 컨테이너에서 발생하는 특정 이벤트 상황을 모니터링하다가 실행되는 특수한 형태의 서블릿이다.
- 웹 애플리케이션에 초기 데이터를 공급하거나 특정 상황에 따라 자동으로 동작하는 프로그램을 구현할 때 사용한다.
- 리스너는 일반적인 형태의 서블릿이 아니라, 특정 이벤트에 따라 동작하는 인터페이스를 구현한 클래스라고 이해하면 쉽다.
- 서블릿과 마찬가지로 애너테이션 기반으로 코드로 작성할 수 있다. 리스너 활용의 대표적인 유형은 다음과 같다.

❶ 초기화 매개변수와 연동 : 톰캣이 시작될 때 초기화 매개변수를 읽어 그에 따라 특정 객체를 초기화한 후 서블릿이나 JSP에 제공하는 형태다.
❷ 예제 프로그램 등을 배포할 때 샘플 데이터 제공 : 프로그램을 실행할 때 데이터베이스가 필요한 경우 데이터베이스의 연결, 테이블 생성 및 샘플 데이터 로딩 등의 작업을 사전에 수행해서 추가적인 작업 없이 프로그램을 실행할 수 있게 한다.
❸ 복잡한 환경 설정 제공 : 프로그램 실행에 필요한 여러 정보를 파일로부터 읽어와 JSP 및 서블릿 등에 제공한다.
❹ 특정 이벤트에 동작하는 기능 구현 : 웹 애플리케이션을 실행할 때 함께 동작해야 하는 외부 프로그램이나 서비스의 동작 유무를 확인하고 자동으로 실행하게 한다.

03. 리스너

2. 리스너 구조와 종류

- 리스너 동작 구조 : ServletContextListener 기준



[그림 13-4] 리스너 개발 절차 및 동작 구조

- 톰캣 시작시 web.xml 혹은 애너테이션 정보를 참조해 ServletContextLister 인터페이스를 구현한 리스너 클래스를 시작한다. contextInitialized() 메서드가 호출되고 여기서 애플리케이션에서 공유할 객체들을 초기화할 수 있다.

03. 리스너

- 리스너는 크게 ServletContext, Session, Request 의 상태나 속성의 변화를 모니터링하고 동작한다.
- 대표적인 리스너는 다음과 같다.

[표 13-3] 리스너 종류

유형	설명	리스너 인터페이스
ServletContext 생명주기 변화	대표적인 리스너 이벤트로, ServletContext의 생성과 소멸 시점에 동작한다. 보통 톰캣의 시작과 종료와 일치한다.	javax.servlet.**ServletContextListener**
ServletContext 속성 변화	ServletContext 즉, JSP 내장객체에 있는 application scope의 속성이 추가되거나 변경되는 상황에 동작한다.	javax.servlet. **ServletContextAttributeListener**
Session 생명주기 변화	session의 생성과 소멸 등 변화에 따라 동작한다.	javax.servlet.http.**HttpSessionListener**
Session 속성 변화	JSP 내장객체에 있는 session scope의 속성이 추가되거나 변경되는 상황에 동작한다.	javax.servlet.http. **HttpSessionAttributeListener**
Request 생명주기 변화	HttpServletRequest 즉, request 내장객체의 생성과 소멸 등 변화에 따라 동작한다.	javax.servlet.**HttpServletRequestListener**
Request 속성 변화	HttpServletRequest 객체의 속성이 추가되거나 변경되는 상황에 동작한다.	javax.servlet. **HttpServletRequestAttributeListener**

04. [기본실습]리스너 개발 : ServletContextListener 구현

1. 실습 개요

- 이번 예제는 ServletContextListener를 구현하고 톰캣이 시작될 때 Book 클래스의 인스턴스를 만든 뒤 ServletContext(application 내장객체) 범위에 속성으로 저장하고, 별도의 JSP에서 표현 언어를 이용해서 Book 클래스 객체를 참조하는 프로그램이다.
- 실습을 통핸 리스너 개발과정과 동작원리를 이해한다.

[표 13-4] 소스 디렉터리 구조

디렉터리	파일 이름
src₩jspbook₩ch13	TestContextListener.java
WebContent₩ch13	ListenerTest.jsp

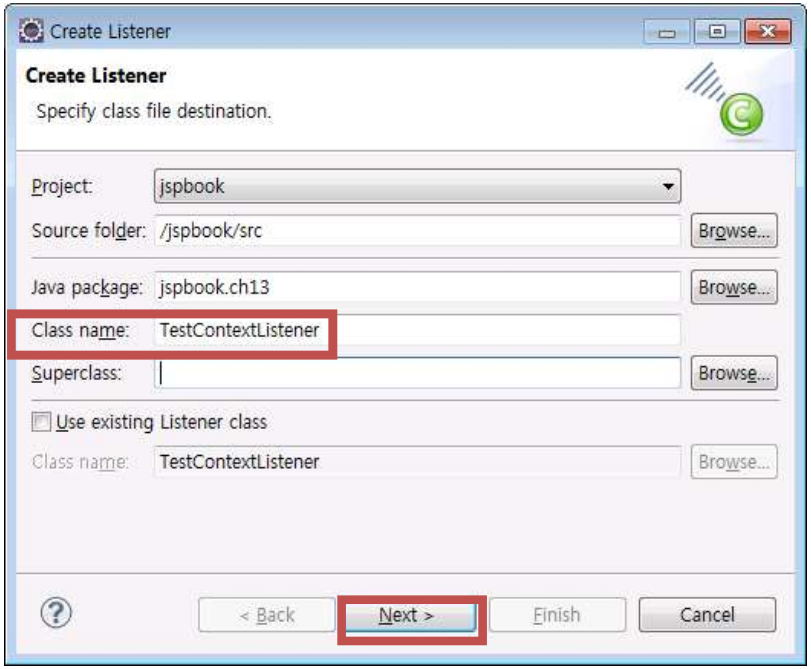
04. [기본실습]리스너 개발 : ServletContextListener 구현

2. 리스너 클래스 구현

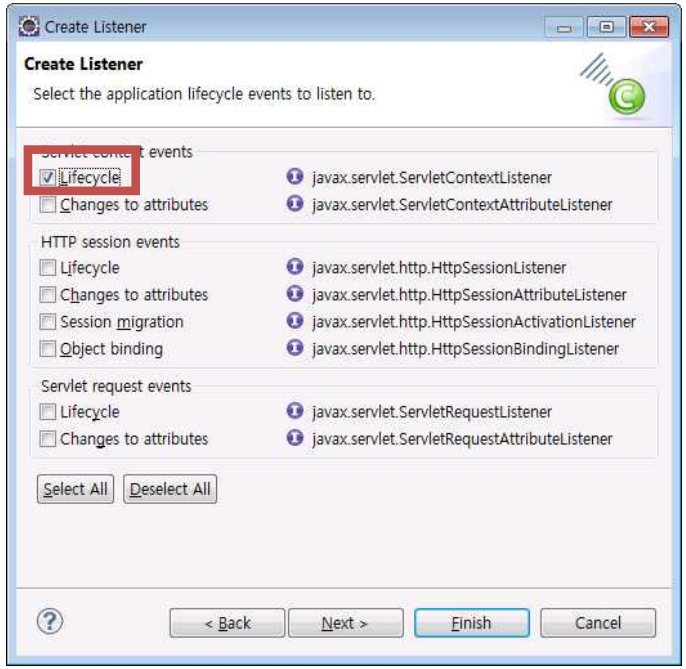
- 리스너 클래스 구현은 이클립스에서 제공하는 템플릿을 통해 쉽게 생성 할 수 있다.
- 위저드 형식으로 단계별로 필요한 정보를 입력하거나 선택해 서블릿 클래스를 생성한다.

❶ jspbook.ch13 패키지를 선택한 다음 <마우스 오른쪽> 버튼을 눌러 [New] → [listener]를 선택한다.

❷



[그림 13-5] 인터페이스 구현 추가 ❶



[그림 13-5] 인터페이스 구현 추가 ❷

❸ <Finish> 버튼을 누른다.

❹ 리스너 클래스(TestContextListener.java) – 교재 p.558 ~ 559 참고

04. [기본실습]리스너 개발 : ServletContextListener 구현

원리를 알면 IT가 맛있다
IT COOKBOOK

주요 소스코드 분석(TestContextListener.java)

- 자동생성된 코드를 기반으로 필요한 메서드의 내용을 구현하면 된다.
- 여기서는 웹 애플리케이션이 시작될때 호출되는 contextInitialized() 메서드를 구현한다.
- Book 객체를 생성하고 ServletContext 즉 JSP 관점에서는 application scope에 속성으로 저장한다.

```
09    public void contextInitialized(ServletContextEvent arg0) {
10        ServletContext ctx = arg0.getServletContext();
11        Book mybook = my Book("자바 웹 프로그래밍","황희정",20000,"한빛미디어");
12        ctx.setAttribute("book", mybook);
13        System.out.println("TestContextListener 초기화되었습니다.");
14    }
```

04. [기본실습]리스너 개발 : ServletContextListener 구현

3. Book 클래스 구현

- **Book 클래스(Book.java)** – 교재 p.559 ~ 560 참고

4. ListenerTest.jsp 구현

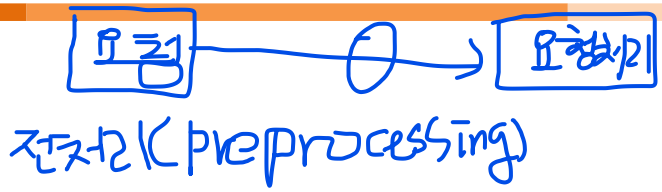
- **ListenerTest.jsp** – 교재 p.561 참고
- TestContextListener에서 application scope에 Book 객체를 넣어두었기 때문에 JSP에서는 별도의 객체 참조를 위한 선언 없이 표현언어를 사용해 출력할 수 있다.

```
12 도서명: ${book.title} <BR>
13 저자명: ${book.author} <BR>
14 가격: ${book.price} <BR>
15 출판사: ${book.publisher} <BR>
```

5. 실행 및 결과 확인

[그림 13-7] ListenerTest.jsp 실행 결과 확인

05. 필터

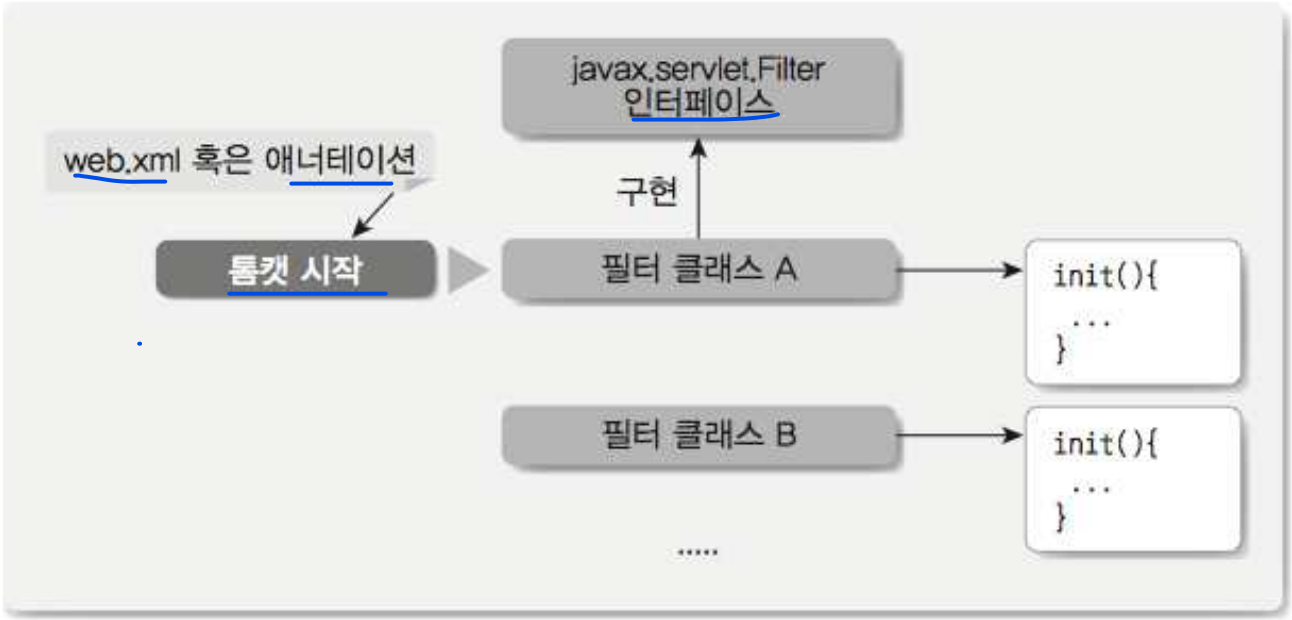


1. 필터란

- 필터란 특정 요청에 대해서만 동작하는 특수한 형태의 웹 프로그램을 말한다.
- 여러 개가 정해진 순서에 따라 배치될 수 있으며 사용자 요청 처리 이전에 먼저 실행된다.
- 리스너와 마찬가지로 단순히 기능만 구현하는 웹 프로그램의 경우에는 필터를 꼭 만들지 않아도 되지만, 애플리케이션 설계 관점에서 좀더 유연하고 효과적인 애플리케이션 구현 및 운영이 필요하다면 필터에 대해 잘 알아둘 필요가 있다.
- 필터는 여러 분야에 활용할 수 있지만 널리 활용 되는 유형은 다음과 같다.
  - 인증 (Authentication)
  - 로깅 / 감사 (Logging and Auditing)
  - 이미지 변환 (Image Conversion), 데이터 압축(Data Compression)
  - 국제화 (Localization)
  - XML 변환(XSL/T Transformations of XML Content)

2. 필터의 구조와 동작 과정

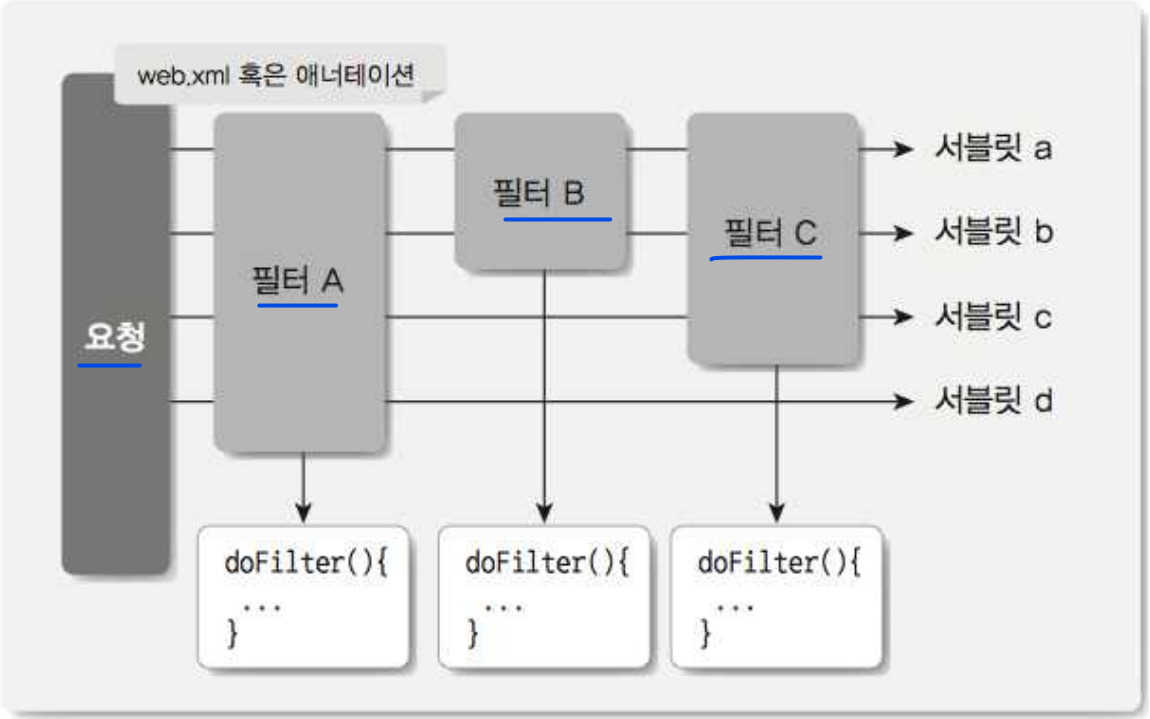
- 필터의 구조는 리스너와 유사하다. 다만 리스너는 특정 이벤트에 따라 실행되는 것이고 필터의 경우에는 사용자가 요청하는 리소스 패턴을 지정해 동작한다는 점이 차이가 있다.
- 필터는 여러개가 존재할 수 있으며 각각의 필터는 init() 메서드를 통해 초기화 작업을 수행할 수 있다.
- 필터의 실행은 톰캣 시작시 web.xml 혹은 애너테이션 정보를 참조해 Filter 인터페이스를 구현한 클래스가 실행되는 구조이다.



[그림 13-8] 필터의 구조

05. 필터

- 실제 필터의 기능 동작은 doFilter() 메서드에서 이루어진다.
- 만일 여러개의 필터가 있다면 하나의 필터가 수행된 다음 필터 체인에 의해 다음 필터가 실행된다.
- 다음은 여러 필터가 서로 다른 요청에 반응해 동작하는 과정이다.



[그림 13-9] 필터의 동작 과정

06. [기본실습]필터 개발 : 한글 처리 필터 구현

원리를 알면 IT가 맛있다
IT COOKBOOK

1. 실습 개요

- 여기서는 HTML 폼을 통해 submit되는 항목의 한글이 깨지는 문제를 해결하기 위해 필터를 개발한다.
- 캐릭터셋 변환 필터는 스프링 프레임워크에는 기본적으로 포함되어 있을 정도로 웹 애플리케이션 구현에서 꼭 필요한 기능이다.
- 지금까지의 예제에서는 HTML form 데이터의 한글 처리를 위해 jsp 에서 request.setCharacterEncoding("UTF-8")을 사용했지만 데이터 처리가 필요한 모든 jsp에서 해당 코드를 넣어줘야 하고 캐릭터셋 변경시 유연하게 대응하기 어려운 문제가 있었다.

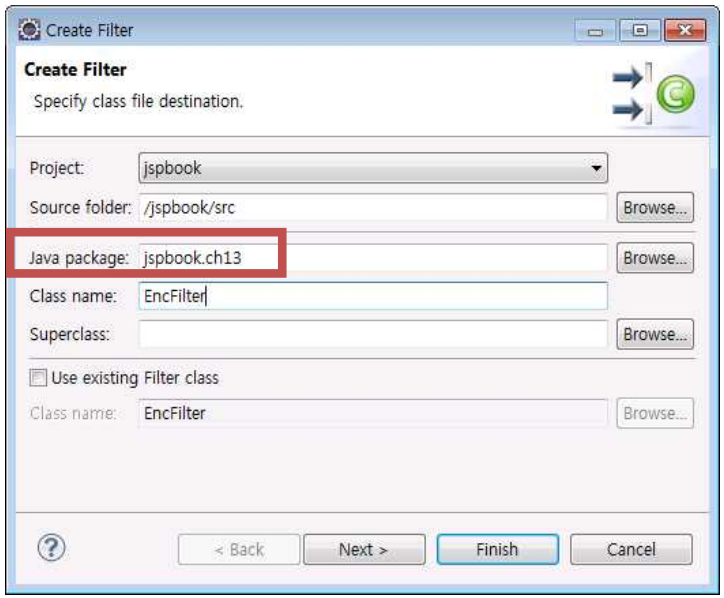
[표 13-6] 예제 프로그램 구조

디렉터리	파일 이름	설명
src₩jspbook₩ch13	EncFilter.java	캐릭터셋 변환을 수행할 필터 클래스다.
WebContent₩ch13	EncForm.jsp	폼을 통해 입력된 한글이 제대로 처리되는지 확인하기 위한 jsp 파일들이다.
	result.jsp	

06. [기본실습]필터 개발 : 한글 처리 필터 구현

2. Encfilter 클래스 구현

❶ 리스너와 마찬가지로 jspbook.ch13을 선택한 후 [New] → [filter]를 통해 생성한다.



[그림 13-10] 필터 동작 과정

❷ Class name에 EncFilter라고 입력한 다음 <Next> 버튼을 누른다. 다음 필터를 매핑할 URL과 초기화 매개변수를 지정하는 화면에서는 그냥 <Finish> 버튼을 누른다.

- **EncFilter.java** – 교재 p.567 ~ 568 참고

06. [기본실습]필터 개발 : 한글 처리 필터 구현

원리를 알면 IT가 맛있다
IT COOKBOOK

주요 소스코드 분석(EncFilter.jsp)

- 필터 역시 리스너와 마찬가지로 이클립스의 템플릿을 이용해 비교적 쉽게 기본 코드를 생성할 수 있다.
- 자동으로 생성된 코드에 init(), doFilter() 등의 메서드를 구현한다.
- Init() 메서드에서는 초기화 매개변수를 읽어와 인코딩 캐릭터를 설정한다.(web.xml 별도 설정 필요)

```
22    public void init(FilterConfig fConfig) throws ServletException {
23
24        this.encoding =
          fConfig.getServletContext().getInitParameter("encoding");
25    }
```

- doFilter()에서는 캐릭터인코딩 설정이 되어 있지 않은 경우 인코딩을 적용하도록 구현하였다.
- 필터 처리 후에는 chain.doFilter() 메서드를 통해 다음 필터를 실행한다.

```
15    public void doFilter(ServletRequest request, ServletResponse
      response, FilterChain chain) throws IOException,ServletException {
16        if(arg0.getCharacterEncoding() == null) {
17            request.setCharacterEncoding(encoding);
18            chain.doFilter(request, response);
```

06. [기본실습]필터 개발 : 한글 처리 필터 구현

3. 필터 테스트 JSP 구현

- **EncForm.html** – 교재 p.569 참고
- **Result.jsp** – 교재 p.570 참고

[그림 13-10] 실행 결과

06. [기본실습]필터 개발 : 한글 처리 필터 구현

4. 필터 매핑 설정 및 실행

- 필터는 url 매핑에 따라 동작한다. 즉 사용자가 서버의 어떤 자원을 요청하는지 패턴을 정해 놓고 해당 형태의 요청이 들어왔을 때 해당 필터가 동작하는 구조이다.
- 필터 매핑은 필터 애너테이션 설정에서 변경할 수 있다.
- 여기서는 모든 jsp에 동작하도록 다음과 같이 애너테이션을 수정한다.

```
@WebFilter("*.jsp")
```

[그림 13-12] 실행 결과

07. [응용실습]리스너 및 필터 : 애플리케이션 설정 관리 구현

1. 실습 개요

- 이번에는 실제 프로젝트에서 더욱 유용하게 사용할 수 있는 고급 응용 예제를 구현해 볼 것이다.
- 앞서 초기화 매개변수를 통해 변경 가능한 정보들을 애플리케이션에 제공하는 예를 살펴보았는데, 실제 프로젝트에서는 더욱 많은 정보를 설정하고 참조한다.
- 이 경우에는 단순히 초기화 매개변수 이상의 기능이 요구된다.

[표 13-7] 프로그램 소스 목록

디렉터리	파일 이름	디렉터리	파일 이름
src₩jspbook₩ch13	PropertyListener.java	WebContent₩ch13₩admin	PropTest.jsp
src₩jspbook₩ch13	AdminFilter.java	WebContent₩ch13	PropTest.jsp

❶ **PropertyListener.java** : ServletContextLister로 톰캣을 시작할 때 web.xml에서 초기화 매개변수를 읽어들이고 해당 경로의 매개변수 파일로부터 자바 매개변수 객체를 로딩해 application scope에 저장한다.
❷ **AdminFilter.java** : 특정 디렉터리의 jsp 요청이 있을 경우 동작하는 필터 클래스로서, 해당 요청에 대해서만 추가적인 정보를 제공한다.
❸ **PropTest** : 각각의 동작을 테스트하려는 jsp 파일이다.

2. 프로퍼티 파일 생성 및 초기화 매개변수 등록

- 자바 프로퍼티 파일 구조로 만들어진 설정 파일 이다. key-value 쌍으로 원하는 정보를 등록 한다.
- 일반 텍스트 파일이므로 메모장 등을 이용해 작성 한다.
- **my.conf** – 교재 p.573 참고

```
01 version=1.0
02 url=jdbc:mysql://127.0.0.1/jspdb
03 user=jspbook
04 passwd=1234
```

- 한글은 처리가 안되므로 반드시 영문값만 사용하도록 한다. 주석은 #으로 처리한다.
- 만들어진 프로퍼티 파일은 컴퓨터에 적당한 디렉토리에 저장한다.
- 파일의 위치는 web.xml에 웹 애플리케이션 초기화 매개변수로 등록한다.

```
<context-param>
    <param-name> profile</param-name>
    <param-value>c:/tmp/my.conf</param-value>
</context-param>
```

3. PropertyListener 작성

- 프로퍼티 파일 정보를 읽어와 application scope에 저장하기 위한 리스너 클래스이다.
- **ProlertyListener.java** – 교재 p.574 ~ 575 참고

```
21  public void contextInitialized(ServletContextEvent arg0) {
22      ServletContext ctx = arg0.getServletContext();
23      String file = ctx.getInitParameter("propfile");
24
25          Properties p = new Properties();
26
27          try {
28              p.load(new FileInputStream(file));
29          } catch (FileNotFoundException e) {
30              e.printStackTrace();
31          } catch(IOException e) {
32              e.printStackTrace();
33          }
34          ctx.setAttribute("prop", p);
35  }
```

4. PropTest.jsp 작성 및 실행

- my.conf에 저장된 정보를 web.xml의 초기화 매개변수를 통해 리스너에서 저장한 프로퍼티 객체를 통해 표현어로 출력한다.
- application scope에 저장된 객체이므로 별도의 선언 없이 표현언어에서 접근이 가능하다.
- Property 객체의 get 메서드를 사용해 필요한 정보를 출력하면 된다.
- **PropTest.jsp** – 교재 p.575 ~ 576 참고

```
12 버전 : ${prop.get('version')} <BR>
13 url : ${prop.get('url')} <BR>
14 user : ${prop.get('user')} <BR>
15 password : ${prop.get('passwd')}
```

[그림 13-13] 실행 결과

5. AdminFilter.jsp 작성 및 실행

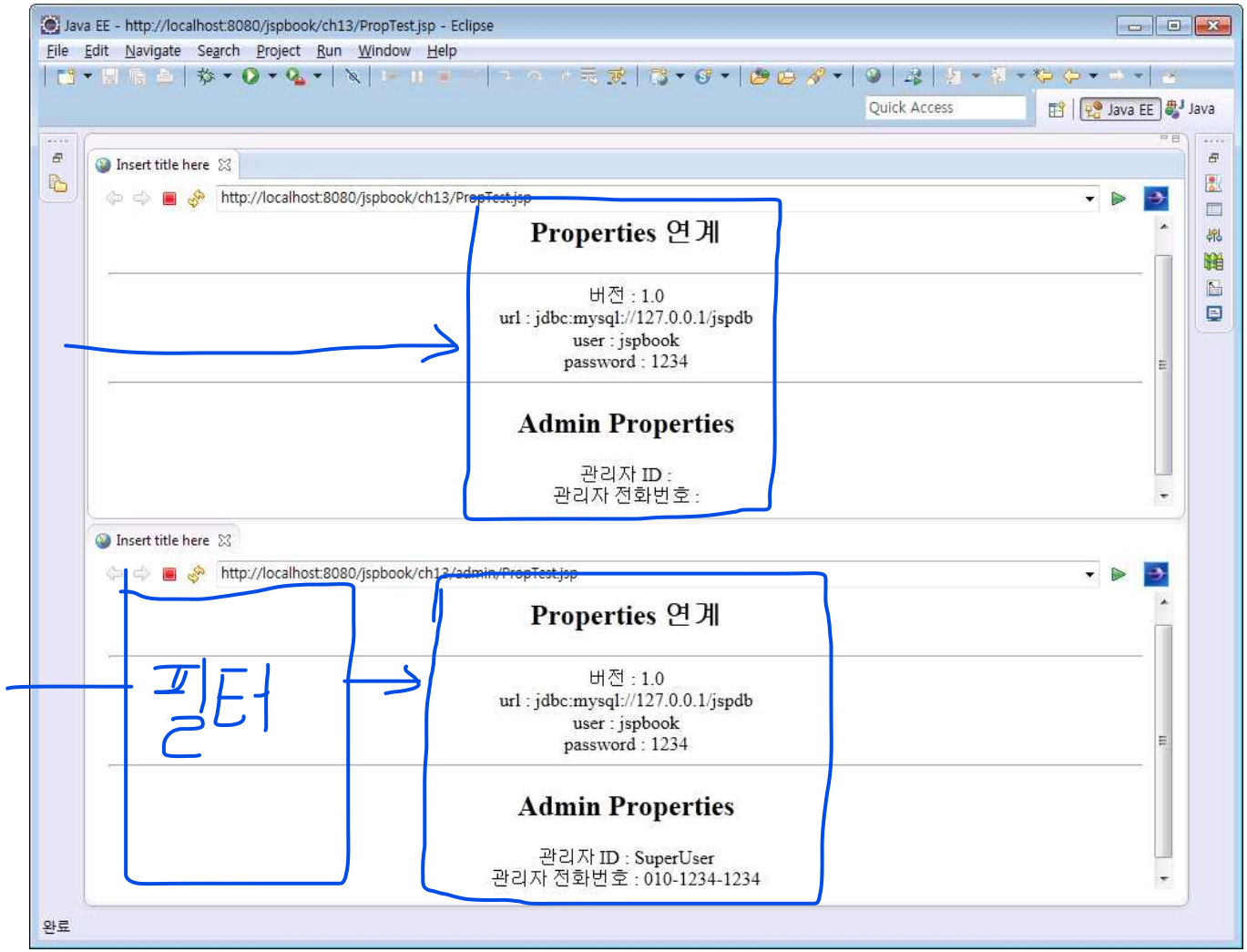
- **AdminFilter.java** – 교재 p.577 ~ 578 참고
- my.conf 에 설정된 정보 이외의 추가적인 정보를 제공하는 필터 클래스이다.
- ch13/admin 폴더 아래에 있는 자원에 접속하는 경우에만 제공한다. 즉 ch13/admin 이외의 폴더에서는 필터가 실행되지 않으므로 추가적인 정보가 출력되지 않는다.
- 필터 매핑 설정

```
14 @WebFilter("/ch13/admin/*")
15 public class AdminFilter implements Filter {
16     Properties p;
17     public AdminFilter() {
18
19     }
```

- doFilter() 메서드에서 프로퍼티 객체를 가져와 새로운 정보를 추가한다.

```
26     p = (Properties) request.getServletContext().getAttribute("prop");
27     p.put("adminId", "SuperUser");
28
29     request.setAttribute("tel", "010-1234-1234");
30     chain.doFilter(request, response);
```

07. [응용실습]리스너 및 필터 : 애플리케이션 설정 관리 구현



[그림 13-14] 실행 결과

프로젝트로 배우는

자바 웹 프로그래밍

Servlet, JSP, JDBC

황희정 지음