스킨 제작 매뉴얼

저작권

이 문서는 정보 제공의 목적으로만 제공됩니다. NHN㈜는 이 문서에 수록된 정보의 완전성과 정확성을 검증하기 위해 노력하였으나, 발생할 수 있는 내용상의 오류나 누락에 대해서는 책임지지 않습니다. 따라서 이 문서의 사용이나 사용 결과에 따른 책임은 전적으로 사용자에게 있으며, NHN㈜는 이에 대해 명시적 혹은 묵시적으로 어떠한 보증도 하지 않습니다.
관련 URL 정보를 포함하여 이 문서에서 언급한 특정 소프트웨어 상품이나 제품은 해당 소유자가 속한 현지 및 국내외 관련법을 따르며, 해당 법률을 준수하지 않음으로 인해 발생하는 모든 결과에 대한 책임은 전적으로 사용자 자신에게 있습니다.

NHN㈜는 이 문서의 내용을 예고 없이 변경할 수 있습니다.

오픈 소스 라이선스 관련 고지

XE는 여러 종류의 오픈 소스 라이선스 중 LGPL(GNU Lesser General Public License) v2를 따르고 있습니다. LGPL v2와 v3에는 약간의 차이가 있으므로 버전까지 기억해야 합니다. LGPL은 기본적으로 GPL과 동일하나 적용 범위가 더 제한적입니다. LGPL도 GPL처럼 해당 라이선스를 가진 소프트웨어를 포함한 소프트웨어도 같은 라이선스를 갖도록 강제하는 효력이 있습니다. 그러나 GPL이 GPL 라이선스를 가진 소프트웨어를 포함한 모든 소프트웨어에 무조건적인 소스 공개를 강제하는데 비해, LGPL 라이선스를 가진 프로그램은 특정 조건 내에서 사용할 경우에는 소스 공개 의무가 없습니다. 따라서 LGPL 라이선스를 가진 소프트웨어는 독점 소프트웨어를 개발하는 데에도 사용할 수 있습니다. 자세한 내용은 아래 사이트를 참조하시기 바랍니다.
LGPL 라이선스: http://www.gnu.org/copyleft/lesser.html
GPL 라이선스: http://www.gnu.org/licenses/gpl.html

문서 정보

문서 개요

이 문서는 XE 스킨을 제작하는 방법을 설명합니다. 이 문서의 내용은 XE core 1.4.4.2 버전을 기준으로 합니다.

독자

이 문서의 독자는 다양한 XE 스킨을 직접 제작해서 자신만의 웹 사이트를 만들고자 하는 사용자입니다. XE 스킨은 HTML과 CSS, 자바스크립트, jQuery를 사용해서 작성하므로 관련 지식이 있으면 더 쉽게 스킨 제작 방법을 이해할 수 있습니다.

XE를 설치하고 사용하는 방법은 "XE 사용자 매뉴얼"을 참조하기 바랍니다.

문의처

이 문서의 내용에 오류가 있거나 내용과 관련한 의문 사항이 있으면 아래의 연락처로 문의하십시오.

연락처: developers@xpressengine.com

문서 버전 및 이력

버전	일자	이력사항
1.0	2010.12.31	1.0 배포
1.1	2011.12.06	XE core 1.5 기준 개정

표기 규칙

참고 표기

> **참고**
>
> 독자가 참고해야 할 내용을 기술합니다.

주의 표기

> **주의**
>
> 독자가 반드시 알아야 할 사항, 시스템 에러를 유발할 수 있는 사항, 수행하지 않았을 때 재산상의 피해를 줄 수 있는 사항을 기술합니다.

윈도(창) 이름/사이트 이름/메뉴 이름/필드 이름/선택 값 및 기호 표기

이 문서에서 윈도(창) 이름, 사이트 이름, 메뉴 이름, 입력 필드 이름, 선택 값은 다음과 같이 표기합니다.

- 윈도(창) 이름: **윈도 이름** 창(단, 소스 코드에 사용된 기호는 이 표기 규칙에 해당하지 않음)
- 사이트 이름: '네이버 데스크톱 다운로드' 사이트
- 메뉴 이름: **메뉴 > 하위메뉴**
- 입력값: *home page*를 입력합니다.

소스 코드 표기

이 문서에서 소스 코드는 회색 바탕에 검정색 글씨로 표기합니다.

```
COPYDATASTRUCT st;
st.dwData = PURPLE_OUTBOUND_ENDING;
st.cbData = sizeof(pp);
st.lpData = &pp;
::SendMes(GetTargetHwnd(), WM_COPYDATA, (WPARAM)this->m_hWnd, (LPARAM)&st);
```

목차

표 및 그림 목록

표 목록

그림 목록

1. XE 스킨 제작 개요

이 장에서는 XE 스킨의 정의와 스킨 제작에 필요한 요소를 설명합니다.

1.1 XE 스킨이란

스킨은 XE로 생성한 데이터를 사용자 화면에 표현하는 양식입니다. 모듈, 레이아웃, 위젯, 위젯 스타일에는 스킨이 하나 이상 있습니다. XE 사용자는 XE 공식 웹 사이트(http://www.xpressengine.com)에서 제공하는 다양한 스킨을 다운로드해서 사용하거나, 이 문서를 참조해서 자신만의 스킨을 직접 제작할 수 있습니다.

아래 그림은 하나의 웹 사이트(텍스타일)에 여러 종류의 스킨을 번갈아 가며 적용한 예입니다.

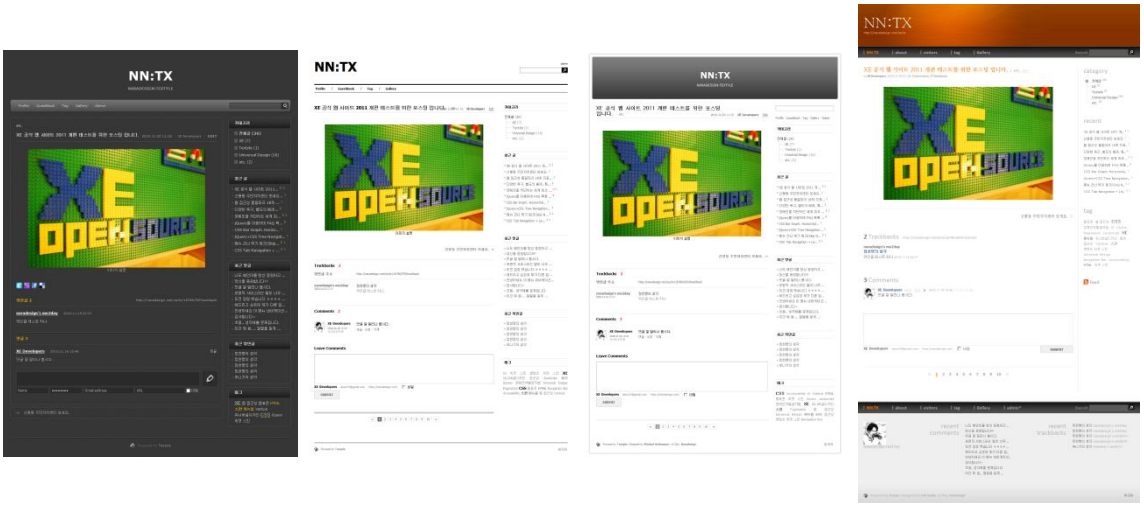


그림 1-1 다양한 스킨 적용 화면

모듈과 위젯의 스킨은 해당 모듈이나 위젯이 설치된 디렉터리의 하위에 있는 'skins' 디렉터리에 저장됩니다. 레이아웃과 위젯 스타일의 스킨은 해당 레이아웃과 위젯 스타일이 설치된 디렉터리에 저장됩니다.

1.2 XE 스킨 제작에 필요한 요소

XE 스킨을 제작하려면 HTML과 CSS, 자바스크립트, XE 템플릿 문법을 알아야 합니다.

HTML(Hyper Text Markup Language)

웹 페이지의 구조를 만드는 언어입니다. 표현하려는 콘텐츠의 제목, 문단, 목록 등을 HTML 태그로 작성하면 웹 브라우저가 이 태그를 해석해서 웹 페이지로 출력합니다. HTML 문서를 표준 문법에 따라 의미 있게 작성하면 검색 엔진이 웹 문서를 수집하기 좋고 다양한 접근 환경에서 사용할 수 있습니다.

CSS(Cascading Style Sheet)

HTML이 웹 페이지의 구조를 담당하는 언어라면 CSS는 표현을 담당하는 언어입니다. 웹 브라우저에는 HTML 요소에 대한 기본적인 표현 양식이 있는데, CSS를 사용하여 이 표현 양식을 더 보기 좋게 바꿀 수 있습니다. 특정 HTML 요소를 선택하고 해당 요소에 CSS 속성을 부여하여 화면에서 배치, 색, 선, 모양, 배경, 이미지 등을 자유롭게 표현할 수 있습니다.

자바스크립트와 jQuery

HTML, CSS가 정적인 표현을 담당하는 언어라면 자바스크립트는 XE 문서의 동적인 표현을 다룹니다. 자바스크립트를 사용하면 페이지를 이동하지 않고도 사용자의 실행 결과를 즉시 화면에 출력할 수 있습니다. 미리 로딩된 콘텐츠를 상황에 알맞게 화면에 보여주거나 감추고, 사용자의 입력 데이터에 오류가 없는지 즉시 확인해서 오류에 대한 추가 설명을 제공하기도 합니다. XE는 자바스크립트를 쉽게 사용할 수 있도록 jQuery라는 자바스크립트 라이브러리를 사용하고 있습니다.

자바스크립트가 모든 환경에서 지원되는 것은 아니므로, HTML만으로 구현할 수 있는 기능을 전적으로 자바스크립트에 의존하여 구현하면 상호 운용성이 떨어집니다. 자바스크립트를 사용할 때는 항상 이를 온전하게 지원하지 못하는 장애 상황도 고려해야 합니다.

XE 템플릿 문법

XE 템플릿 문법은 서버에서 PHP 문법으로 해석되며 DB로부터 원하는 정보를 뽑아내어 사용자 화면에 출력합니다. XE 템플릿 문법을 사용하면 모듈 또는 위젯의 기능을 허용된 범위 안에서 확장하거나 축소할 수 있습니다.

2. XE 스킨 제작의 기초

XE 스킨 제작에 필요한 HTML과 CSS, 자바스크립트/jQuery, XE 템플릿 문법을 설명합니다.

2.1 HTML 의 이해

HTML은 데이터에 '의미'를 부여해서 무질서한 데이터를 유용한 '정보'로 바꾸는 힘이 있습니다. HTML 문서를 배치하고 꾸미려면 CSS를 사용해야 합니다. 데이터의 '의미'와 '표현'을 잘 구별해서 적합한 언어를 사용하여 HTML을 작성하면 다른 사람이 쉽게 이해하고 편집할 수 있습니다. 문서의 '표현'을 CSS로 분리한다면 디자인을 바꾸기 위해 HTML을 편집해야 하는 수고를 덜 수 있습니다.

2.1.1 요소와 속성, 값

HTML은 요소와 속성, 값으로 구성되어 있습니다.

요소(Element)

```
<h1 title="Xpress Engine">XE</h1>
```

<h1>로 시작해서 </h1>로 끝나는 전체가 '요소'입니다. 이 요소는 '<h1> 요소'라고 지칭하며, <h1>과 </h1>은 각각 '시작 태그'와 '종료 태그'라고 부릅니다. 그 사이에 있는 'XE'는 이 요소의 '내용(contents)'이라고 합니다.

'h'는 제목을 의미하는 'heading'의 약어로서 많은 태그가 이렇게 약어를 사용합니다. 숫자는 제목의 단계를 표시하고 '1'이면 최상위 수준의 제목입니다. 대부분의 웹 브라우저는 제목을 크고 굵게 표현하는데, 웹 제작자는 단순히 글꼴을 크고 굵게 표현하기 위하여 이 요소를 사용해서는 안 됩니다. 데이터에 아무런 의미를 부여하지 않고 크고 굵게 만드는 것은 '표현'에 해당하기 때문에 CSS로 처리해야 합니다.

속성(Attribute)

```
<h1 title="Xpress Engine">XE</h1>
```

이 코드에서 'title'은 속성입니다. '속성'은 현재의 요소가 다른 요소와 어떤 차이가 있는지 나타냅니다. 'title' 속성은 현재 요소의 의미를 부연 설명하는 '참고 제목' 역할을 합니다. HTML 요소마다 사용할 수 있는 속성이 다르게 규정되어 있습니다.

값(Value)

```
<h1 title="Xpress Engine">XE</h1>
```

속성에는 반드시 '값'이 있습니다. 'Xpress Engine'은 'title' 속성의 값입니다. 값은 현재의 '요소'가 어떤 '속성'을 지녔는지 구체적으로 정의하며, 화면 표시에 직접 영향을 주기도 합니다. 속성에 따라 잘못된 값을 입력하면 화면이 제대로 표시되지 않을 수도 있습니다. 값은 속성에 따라서 이미 정해져 있기도 하지만, 어떤 속성의 값은 사용자가 정의해야 합니다. 'Xpress Engine'은 사용자 정의 값에 해당합니다.

2.1.2 HTML 의 시작과 끝

모든 HTML 요소에는 데이터의 범위를 명시하기 위하여 시작과 끝이 있습니다.

```
<h1 title="Xpress Engine">XE</h1>
```

- 이 요소의 시작은 <h1>입니다. 여기서부터 첫 번째 제목이 시작됩니다.
- 이 요소의 끝은 </h1>입니다. 첫 번째 제목은 여기서 끝납니다.

텍스트 콘텐츠가 아닌 경우 HTML 요소 자체가 데이터가 되는 경우도 있는데 이런 경우는 시작과 동시에 끝납니다. 다음과 같은 경우 이미지가 다른 콘텐츠를 포함하지 않으므로 시작과 동시에 닫습니다.

```
<img />
```

2.1.3 부모 요소와 자식 요소

HTML에서 요소는 다른 요소를 포함할 수 있습니다.

예를 들어, 제목 'XE'를 클릭하면 초기 화면으로 이동하는 콘텐츠를 작성해 보겠습니다.

```
<h1 title="Xpress Engine">
   <a href="index.html">XE</a>
</h1>
```

<h1> 요소 안에 <a> 요소가 포함되어 있습니다. 이와 같이 하나의 데이터는 여러 개의 요소로 중첩할 수 있습니다. 이때 <h1> 요소와 같이 바깥쪽에 있는 요소를 '부모 요소'라 부르고, <a> 요소와 같이 안쪽에 있는 요소를 '자식 요소'라 부릅니다.

<a> 요소는 데이터가 다른 자원을 참조하도록 연결합니다. href 속성은 <a> 요소 안에서 참조 대상을 지정합니다. 즉, 위의 예제 코드는 "XE라는 데이터는 최상위 수준 제목이며 index.html을 참조하고 있다"라는 것을 정의합니다.

단, 요소를 중첩해서 사용할 때는 시작 태그와 종료 태그의 사용 순서를 지켜야 합니다. 다음과 같이 선언하는 것은 중첩 규칙을 어긴 잘못된 예입니다.

```
<h1 title="Xpress Engine">
   <a href="index.html">XE</h1>
</a>
```

2.1.4 인라인 요소와 블록 요소

블록 요소는 '하나의 독립된 덩어리'로 취급하며 줄을 바꿔서 표현합니다. , <h>, <p> 요소 등이 블록 요소에 해당합니다. 인라인 요소는 '행 안의 일부'로서 줄 바꿈 없이 연속으로 표현합니다. , <img>, <a> 요소 등이 대표적인 인라인 요소입니다.

블록 요소는 인라인 요소를 포함할 수 있지만 인라인 요소는 블록 요소를 포함할 수 없습니다. 어떤 요소가 블록이고 어떤 요소가 인라인인지는 HTML 규격을 보면 알 수 있습니다.

다음은 블록 요소가 인라인 요소를 감싸고 있는 올바른 마크업입니다. <h1> 요소는 블록 요소이고 <a> 요소는 인라인 요소입니다.

```
<h1><a>XE</a></h1>     <!--올바른 마크업-->
```

다음은 인라인 요소가 블록 요소를 감싸고 있는 잘못된 마크업입니다.

```
<a><h1>XE</h1></a>    <!--잘못된 마크업-->
```

참고

W3C HTML 4.01 규격: http://trio.co.kr/webrefer/html/cover.html
W3C HTML 4.01 요소 색인: http://trio.co.kr/webrefer/html/index/elements.html
W3C HTML 4.01 속성 색인: http://trio.co.kr/webrefer/html/index/attributes.html
W3C XHTML 1.0 규격: http://trio.co.kr/webrefer/xhtml/overview.html
W3C 마크업 유효성 검사: http://validator.w3.org/
W3 Schools 온라인 자습서: http://www.w3schools.com/

2.2 CSS 의 이해

HTML이 웹 문서에 '의미'를 부여해서 데이터를 '정보'로 바꾼다면 CSS는 문서의 '배치와 표현'에 관여하여 콘텐츠를 이해하기 쉽고 보기 좋게 만듭니다. XE 스킨에서 대부분의 경우 CSS 파일은 HTML과 분리되어 있지만 HTML 문서가 CSS 파일을 참조하고 있기 때문에 사용자는 HTML과 CSS가 모두 반영된 최종 화면을 보게 됩니다. CSS 문법을 익히면 문서를 보기 좋게 꾸밀 수 있을 뿐만 아니라 HTML을 더 의미 있고 적법하게 사용할 수 있습니다.

2.2.1 선택자와 속성, 값

CSS는 '선택자'와 '속성'과 '값'으로 구성되어 있습니다. '선택자'는 HTML의 특정 '요소'를 선택하고 '속성'과 '값'은 선택된 HTML 요소가 화면에 어떻게 출력될 것인지를 정의합니다.

선택자(selector)

```
h1 { font-size:24px; }
```

'h1'은 '선택자'입니다. <h1> 요소를 가리킵니다.

속성(property)

```
h1 { font-size:24px; }
```

'font-size'는 '속성'입니다. <h1> 요소의 글꼴 크기를 제어하는 선언입니다.

값(value)

```
h1 { font-size:24px; }
```

'24px'은 '값'입니다. <h1> 요소의 font-size 속성(글꼴 크기) 값을 구체적으로 명시합니다.

2.2.2 CSS 선택자의 종류

CSS 선택자는 특정 HTML 요소를 선택하는 역할을 합니다. 선택자를 사용하면 선택된 요소에 고유한 CSS 양식을 표현할 수 있습니다. CSS는 다양한 유형의 선택자를 지원합니다.

타입 선택자(type selector)

```
h1 { … }
```

HTML 요소 이름을 그대로 선택자 이름으로 사용하는 것을 타입 선택자라고 합니다. HTML 문서에서 요소 이름이 일치하면 모두 선택합니다. 표준 HTML 요소 이름을 선택자 이름으로 사용할 수 있습니다.

아이디 선택자(id selector)

```
#selector { … }
```

아이디 선택자는 선택자 이름 앞에 숫자 기호(#)를 표시합니다. HTML 문서에서 아이디의 값이 일치하는 요소를 선택합니다. 아이디 선택자의 이름은 사용자가 정의할 수 있습니다. 아이디 선택자의 이름을 정의하는 규칙은 다음과 같습니다.

- 모든 자연어를 선택자 이름으로 사용할 수 있지만 영문 대소문자와 숫자의 조합을 권장합니다.
- 선택자 이름으로 특수문자를 사용할 수 없지만 언더스코어(_)와 하이픈(-)은 사용할 수 있습니다.
- 숫자로 시작할 수 없습니다.

HTML 문서는 한 페이지에 2개 이상의 아이디 값을 사용하도록 허용하지 않습니다. 아이디 값은 하나의 HTML 문서 안에서 고유해야 합니다.

클래스 선택자(class selector)

```
.selector { … }
```

클래스 선택자는 선택자 이름 앞에 점(.)을 표시합니다. HTML 문서에서 클래스의 값이 일치하는 요소를 선택합니다. 클래스 선택자의 이름은 사용자가 정의할 수 있습니다. 클래스 선택자의 이름을 정의하는 규칙은 다음과 같습니다.

- 모든 언어를 선택자 이름으로 사용하는 것이 가능하지만 영문 대소문자와 숫자의 조합을 권장합니다.
- 선택자 이름으로 특수문자를 사용할 수 없지만 언더스코어(_)와 하이픈(-)은 사용할 수 있습니다.
- 클래스 선택자 이름은 숫자로 시작할 수 없습니다.

HTML 문서는 한 페이지에 2개 이상의 동일한 클래스 값을 사용할 수 있습니다.

자식 선택자(child selector)

```
h1>a { … }
```

자식 선택자는 선택자 사이에 꺾은 괄호(>)를 적용해서 표시합니다. 선택자의 나열이 HTML 요소의 중첩 순서와 일치하면 선택합니다. 중첩이 2단계 이상 깊어지는 자손(자식의 자식)은 선택하지 않습니다. 오직 한 단계 바로 아래 포함된 자식만 선택합니다.

> **주의**
> IE6 브라우저는 자식 선택자를 지원하지 않습니다.

자손 선택자(descendant selector)

```
h1 a { … }
```

자손 선택자는 선택자 사이에 띄어쓰기를 적용해서 표시합니다. 선택자의 나열이 HTML 요소의 중첩 순서와 일치하면 선택합니다. 즉, 다음과 같이 <h1> 요소 안에 포함된 <a> 요소만을 선택한다는 의미입니다.

```
<h1><a>XE</a></h1>
```

<h1> 요소 밖에 존재하는 <a> 요소는 이 선언의 영향을 받지 않습니다.

공용 선택자(universal selector)

```
* { … }
```

공용 선택자는 별표(*)로 표시합니다. 이 선택자는 모든 HTML 요소를 선택합니다. 웹 브라우저는 보통 HTML 요소마다 기본 CSS 양식을 지정해 두는데 이것을 동일한 값으로 초기화할 수 있습니다.

공용 선택자를 사용하면 페이지 표시 속도가 느려지므로 꼭 필요한 경우가 아니면 사용을 권장하지 않습니다.

가상 선택자(pseudo selector)

```
a:focus { … }
```

가상 선택자는 다른 선택자 뒤에 콜론(:)을 붙여 표시합니다. 요소의 상태를 선택하는 '가상 클래스 선택자'와 HTML 코드 속에 없는 가상의 요소를 선택하는 '가상 요소 선택자'가 있습니다. 다른 선택자들이 정적인 상태의 HTML 요소를 선택하는 것에 반해서 가상 선택자는 HTML 요소의 동적인 변화 상태를 선택하거나 HTML 코드에 없는 가상의 요소를 선택합니다.

가상 클래스 선택자의 종류는 다음과 같습니다.

표 2-1 가상 클래스 선택자

가상 클래스 선택자	설명	예제
:link	아직 방문한 적이 없는 링크를 선택합니다. 'a' 라는 타입 선택자와 결합해서 사용합니다.	a:link
:visited	이미 방문한 링크를 선택합니다. a 라는 타입 선택자와 결합해서 사용합니다.	a:visited
:hover	마우스 포인터가 가리키는 동안의 상태를 선택합니다.	a:hover
:active	마우스가 클릭되는 순간 또는 엔터 키를 누른 순간의 상태를 선택합니다.	a:active
:focus	포커스를 받은 요소의 상태를 선택합니다.	a:focus
:first-child	첫 번째 자식 요소를 선택합니다.	li:first-child
:lang(language)	언어 속성이 일치하면 선택합니다.	p:lang(en)

주의

IE6 브라우저는 ':first-child, :lang()' 가상 클래스 선택자를 지원하지 않습니다. IE6 브라우저는 :hover, :active, :focus 가상 클래스 선택자를 'a' 이외의 요소에 지원하지 않습니다.

'a' 요소에 다양한 상태의 가상 클래스 선택자를 지정할 때는 :link(링크) :visited(방문시) :hover(지나치게) :active(활동하면) :focus(주목받습니다) 순서를 유지하는 것이 좋습니다. 나중에 선언된 클래스의 우선순위가 높기 때문에 먼저 선언된 것들을 모두 덮어쓰게 됩니다. 예를 들어, :hover 뒤에 a:visited를 쓰면 이미 방문한 링크 위에 마우스를 올려도 :hover가 적용되지 않습니다. 이미

방문한 링크 위에 마우스를 올렸을 때 반응하게 하려면 :visited 다음에 :hover 클래스가 오도록 작성해야 합니다.

가상 요소 선택자의 종류는 다음과 같습니다.

표 2-2 가상 요소 선택자

가상 요소 선택자	설명	예제
:first-line	첫 번째 행을 선택합니다. 블록 요소에만 적용할 수 있습니다.	p:first-line
:first-letter	첫 번째 문자를 선택합니다. 블록 요소에만 적용할 수 있습니다.	p:first-letter
:before	요소 시작 지점 안쪽의 가상 요소를 선택합니다.	div:before{ content:"..." } (div 요소 시작 지점 안쪽에 가상의 '...' 문자열을 생성하고 선택합니다.)
:after	요소 종료 지점 안쪽의 가상 요소를 선택합니다.	div:after{ content:"..." } (div 요소 종료 지점 안쪽에 가상의 '...' 문자열을 생성하고 선택합니다.)

주의

IE6 브라우저는 '가상 요소 선택자'를 지원하지 않습니다.

선택자 묶음(Selector Grouping)

```
.apple,
.tomato { color:red }
```

선택자 묶음은 쉼표(,)를 사용해서 여러 개의 선택자를 하나로 묶고 속성과 값을 한 번만 선언합니다. 여러 선택자가 같은 속성과 값을 공유할 때 하나의 선택자로 묶을 수 있습니다.

위의 선택자 묶음 선언은 아래의 선언과 같은 의미입니다.

```
.apple { color:red }
.tomato { color:red }
```

인접 형제 선택자(Adjacent Sibling Selector)

```
h1+h2 { color:red }
```

형제 선택자는 선택자 사이에 덧셈 기호(+)를 넣어 표시합니다. 요소가 형제 관계에 있고 나열 순서가 같으면 뒤에 등장하는 요소를 선택합니다.

위의 인접 형제 선택자 선언 예제에 따르면, 다음과 같은 HTML 코드에서 <h1> 요소 다음에 <h2> 요소가 순서대로 등장하므로 <h2> 요소는 붉은색 글꼴로 처리됩니다.

```
<h1>XE</h1>
<h2>Textyle</h2>
```

주의

IE6 브라우저는 '형제 선택자'를 지원하지 않습니다.

속성 선택자(Attribute Selector)

```
input[checked] { ... } /* input 요소에 checked 속성이 사용되면 무조건 선택 */
input[type=text] { ... } /* input 요소의 type 속성이 text인 경우에만 선택 */
img[alt~=dog] { ... }  /* img 요소의 alt 속성 값에 'dog'이라는 단어가 포함되면 선택 */
p[lang|=en] { ... } /* p 요소의 lang 속성이 en-us와 같이 en-으로 시작되면 선택 */
```

HTML 요소에 선언된 속성을 사용하여 해당 요소를 선택합니다. 속성 선택자의 종류는 다음과 같습니다.

표 2-3 속성 선택자

속성 선택자	설명	예제
[attribute]	HTML 요소에 attribute 라는 속성이 사용되면 값의 존재 여부와 무관하게 요소를 선택합니다.	input[checked]
[attribute=value]	HTML 요소에 attribute 라는 속성이 사용되고 오직 하나의 값이 존재하는데 그 값이 정확하게 value 일 때 요소를 선택합니다.	input[type=text]
[attribute~=value]	HTML 요소에 attribute 라는 속성이 사용되고 하나 이상의 값이 공백으로 분리되어 존재하는데 그 중 하나의 값이 value 일 때 요소를 선택합니다.	img[alt~=dog]
[attribute|= value]	HTML 요소에 attribute 라는 속성이 사용되고 값이 value 로 시작되면 요소를 선택합니다.	p[lang|=en]

주의

IE6 브라우저는 '속성 선택자'를 지원하지 않습니다.

참고

CSS 규격 한글 번역문: http://trio.co.kr/webrefer/css2/cover.html
CSS Reference: http://www.w3schools.com/css/css_reference.asp

2.3 자바스크립트와 jQuery 사용

XE는 자바스크립트를 쉽게 사용할 수 있도록 jQuery라는 자바스크립트 라이브러리를 사용하고 있습니다. jQuery를 사용하면 스킨 제작자가 자바스크립트 전문가가 아닌 경우에도 자바스크립트를 쉽게 다룰 수 있습니다.

이 절에서는 XE 스킨을 제작할 때 HTML 문서에 자바스크립트를 포함시키는 방법을 설명합니다.

2.3.1 jQuery 라이브러리 포함

XE core에는 이미 jQuery 라이브러리가 포함되어 있고 모든 페이지에서 참조하도록 선언되기 때문에 별도로 jQuery 라이브러리를 불러오는 선언을 할 필요가 없습니다. XE로 생성된 모든 웹 문서에는 HTML 코드의 <head> 요소 안쪽에 다음과 같이 jQuery 라이브러리 참조 코드가 포함됩니다.

```
<script type="text/javascript" src="./common/js/jquery.js"></script>
```

자바스크립트 참조 코드는 문법적으로 HTML 코드의 <head> 요소 또는 <body> 요소 안쪽의 어느 곳에도 포함시킬 수 있지만 XE에서는 jQuery 라이브러리 참조 코드를 HTML 문서의 <head> 요소 안쪽에 포함시키고 있습니다. 웹 브라우저는 자바스크립트 코드를 선언된 순서대로 해석하는데 자바스크립트 코드의 해석 순서가 화면 표시에 영향을 미치는 경우가 있습니다. HTML 해석과 동시에 즉시 결과를 실행해야 하는 jQuery 의존 코드가 있는 경우 jQuery 라이브러리보다 먼저 해석되면 안 되기 때문에 XE는 jQuery 라이브러리 참조 선언을 HTML 문서의 <head> 요소 안쪽에 포함한 것입니다.

> **참고**
> XE 설치 경로에 따라 jquery.js 파일의 참조 경로는 보기와 다를 수 있습니다.

2.3.2 XE 템플릿 문법으로 자바스크립트 선언

jQuery 코드는 자바스크립트이기 때문에 자바스크립트 문법에 따라 작성해야 합니다. 자바스크립트의 선언 위치는 HTML 문서의 <head> 요소 또는 <body> 요소 안쪽이어야 하며 이 밖의 위치에 선언하면 HTML 문법 오류가 발생합니다. HTML 문법이 허용하지 않는 위치에 자바스크립트 코드를 작성하면 표준에 따라 엄격하게 구현된 브라우저에서는 자바스크립트를 해석하지 못할 수도 있습니다. 자바스크립트 코드를 HTML 문서에 직접 포함하는 방법도 있지만 별도의 *.js 파일로 분리하여 HTML 문서에서 불러오는 방법도 있습니다.

자바스크립트는 용도에 따라 다음과 같이 선언 위치를 구분해서 사용합니다.

<head> 요소에서 JS 파일 참조

웹 브라우저가 화면 표시를 끝내기 전에 자바스크립트로 사용자 화면의 일부 콘텐츠를 보여주거나 감추는 동작을 실행한다면 자바스크립트 코드를 HTML 문서의 <head> 요소 위치에 포함하는 것이

좋습니다. 이런 경우 자바스크립트 코드를 <body> 요소에 포함하면 자바스크립트가 HTML보다 늦게 해석되어 일시적으로 화면이 제대로 보이지 않을 수 있습니다.

<head> 요소에 포함된 자바스크립트는 HTML 문서보다 먼저 해석되지만 HTML 문서의 로딩이 완료된 이후에 실행하도록 코드를 작성해야 합니다.

<body> 요소에서 JS 파일 참조

자바스크립트가 HTML 문서를 로딩하는 시점에 화면 표시를 위한 어떤 동작을 실행하지 않는다면 HTML 문서의 <body> 요소에 포함하되 가장 아래쪽에 선언하는 것이 좋습니다. 웹 브라우저는 HTML 코드와 자바스크립트 코드를 동시에 해석하지 않고 작성된 순서대로 해석하기 때문에 자바스크립트 코드를 나중에 해석하도록 하면 HTML 문서를 화면에 표시하는 속도가 빨라집니다.

XE 템플릿 문법으로 자바스크립트를 선언하는 자세한 방법은 "JS 파일 참조"를 참조하십시오.

2.3.3 표준 문법으로 자바스크립트 선언

별도로 작성된 자바스크립트 파일을 HTML 문서에서 불러오는 경우 XE 템플릿 문법을 사용하면 HTML 문서의 <head> 요소 또는 <body> 요소의 종료 지점 가운데 선언 위치를 선택할 수 있습니다. 그러나 <body> 요소의 종료 지점이 아니라 <body> 요소 내부의 원하는 위치에 정확하게 삽입하려면 삽입을 원하는 위치에 다음과 같이 HTML 표준 문법으로 선언합니다.

```
<body>
    ...
    <script type="text/javascript" src="xxx.js"></script>
    ...
</body>
```

별도의 자바스크립트 파일을 작성하지 않고 HTML 문서에서 직접 자바스크립트 코드를 작성하려면 다음과 같이 선언합니다.

```
<body>
    ...
    <script type="text/javascript">
    // <![CDATA[
        자바스크립트 코드를 이곳에 작성합니다.
    // ]]>
    </script>
    ...
</body>
```

> **참고**
>
> 자바스크립트 코드의 시작과 종료 지점에 <![CDATA[...]]>를 선언하는 것은 자바스크립트 코드에 포함된 문자들이 HTML 코드로 해석되는 것을 방지하기 위한 것입니다. <![CDATA[...]]>를 선언하지 않으면 꺾은 괄호(<, >)와 각종 자바스크립트 연산자가 문자 그대로 해석되지 않고 HTML 태그가 시작되거나 HTML 엔티티가 시작되는 것으로 잘못 해석될 수 있습니다. <![CDATA[...]]> 선언은 자바스크립트 문법이 아니기 때문에 선언된 라인을 한 줄 주석으로 처리합니다.

2.3.4 HTML 해석 이후 jQuery 실행

XE로 생성된 모든 문서는 jQuery 라이브러리를 참조하기 때문에 <script> 요소 안에서 jQuery 문법을 사용할 수 있습니다. 그런데 HTML 문서를 완전히 해석하기 전에 자바스크립트가 실행되면 HTML 구조에 의존하고 있는 자바스크립트 구문에서 오류가 발생할 수 있습니다. 이런 오류를 예방하기 위하여 jQuery 문서의 해석 시점과 실행 시점을 다르게 처리할 수 있습니다.

다음과 같은 방법으로 jQuery 함수를 선언하면 HTML 문서의 로딩이 끝난 이후에 jQuery를 실행할 수 있습니다.

```
<script type="text/javascript">
// <![CDATA[
   jQuery(function($){
      jQuery 문법을 이곳에 작성합니다.
   });
// ]]>
</script>
```

주의

jQuery 문법에서 'jQuery'는 '달러 기호($)'로 치환하여 표기할 수 있습니다. 그러나 XE는 '$' 기호를 사용하는 다른 자바스크립트 라이브러리와의 충돌을 피하기 위하여 jQuery 함수를 시작하는 부분에서만큼은 '$' 기호로 치환하는 것을 허락하지 않습니다. 따라서 XE에서는 jQuery 함수를 처음 작성할 때 jQuery(function($){ ... })라고 작성해야 합니다.

2.3.5 jQuery 의 동작 방식 실습

jQuery 동작 방식을 한 마디로 설명하면 '선택과 실행'이라고 할 수 있습니다. 이 말을 조금 더 친절하게 풀이하면 'HTML 요소를 선택하고 필요한 시점에는 동작을 실행한다'라는 의미입니다.

간단한 예제를 통해서 살펴보겠습니다.

```
<head>
   <style type="text/css">
   #help { display:none; }
   </style>
   <script type="text/javascript" src="jquery.js"></script>
</head>
<body>
   <a href="#help" class="helpBtn">도움말</a>
   <p id="help">이 내용은 CSS 선언에 의해 화면에 표시되지 않습니다.<p>
   <script type="text/javascript">
   // <![CDATA[
   jQuery(function($){
      $('a.helpBtn').click(function(){ // HTML 요소를 선택하고
         $('p#help').css('display','block'); // 동작을 실행합니다
      });
   });
   // ]]>
   </script>
</body>
```

이 코드는 'p#help'가 최초에 display:none 상태로 되어 있지만 사용자가 'a.helpBtn'이라는 요소를 클릭하면 display:block으로 상태가 바뀌는 동작을 구현한 것입니다. jQuery 함수 안에서 HTML 요소를 '선택'하고 사용자의 이벤트를 기다렸다가 선택한 요소의 CSS 변경을 '실행'했습니다.

특정 HTML 요소를 화면에서 감추거나 보여주는 동작은 매우 빈번하게 발생하기 때문에 jQuery는 이런 동작을 좀 더 쉽게 처리할 수 있도록 show(), hide()라는 내장 함수를 제공합니다. 위 코드는 다음과 같이 더 간결하게 작성할 수 있습니다.

```
<head>
   <style type="text/css">
   #help { display:none; }
   </style>
   <script type="text/javascript" src="jquery.js"></script>
</head>
<body>
   <a href="#help" class="helpBtn">도움말</a>
   <p id="help">이 내용은 CSS 선언에 의해 화면에 표시되지 않습니다.<p>
   <script type="text/javascript">
   // <![CDATA[
      jQuery(function($){
         $('a.helpBtn').click(function(){ // HTML 요소를 선택하고
            $('p#help').show(); // show() 함수를 실행합니다
         });
      });
   // ]]>
   </script>
</body>
```

다음 예제는 'a.helpBtn' 링크를 한 번 클릭하면 보여주고 다시 한 번 클릭하면 감추는 토글 함수를 실행하도록 동작을 개선한 것입니다.

```
<head>
   <style type="text/css">
   #help { display:none; }
   </style>
   <script type="text/javascript" src="jquery.js"></script>
</head>
<body>
   <a href="#help" class="helpBtn">도움말</a>
   <p id="help">이 내용은 CSS 선언에 의해 화면에 표시되지 않습니다.<p>
   <script type="text/javascript">
   // <![CDATA[
      jQuery(function($){
         $('a.helpBtn').click(function(){ // HTML 요소를 선택하고
            $('p#help').toggle(); // toggle() 함수를 실행합니다
         });
      });
   // ]]>
   </script>
</body>
```

지금까지 jQuery 사용과 간단한 동작 방식을 알아보았습니다. jQuery 매뉴얼이나 jQuery 관련 서적을 참고하면 보다 다양한 방법으로 HTML 요소를 선택하고 더 많은 동작을 구현할 수 있습니다.

참고

jQuery를 사용하여 HTML을 선택하고 어떤 동작을 실행하는 자세한 방법은 jQuery 공식 웹 사이트 (http://jquery.com/)를 참조하십시오.

2.4 XE 템플릿 문법

XE 템플릿 문법은 서버 측에서 PHP 문법으로 해석되며 DB로부터 원하는 정보를 뽑아내어 사용자 화면에 출력합니다. XE 템플릿 문법을 사용하면 모듈 또는 위젯의 기능이나 내용을 허용된 범위 안에서 확장하거나 축소할 수 있습니다.

XE 템플릿 문법을 적용하여 XE 스킨을 작성하는 방법에는 다음과 같은 네 가지가 있습니다.

- HTML 주석 <!--...-->안에 작성하는 방법. 예) <!--@if(...)-->...<!--@end-->
- 가상의 <block> 요소 안에 작성하는 방법. 예) <block>...</block>
- HTML 요소에 직접 작성하는 방법. 예) ...
- 주석이나 요소에 의존하지 않고 작성하는 방법. 예) {$content} 내용 변수로 데이터를 출력함.

참고

<block> 요소는 HTML 표준 요소가 아니라 XE core 1.4.4 버전에 새로 추가된 가상의 요소입니다. HTML 요소의 형식을 빌려 쓰고는 있지만 제어문을 실행할 뿐 실제로 화면에 요소가 출력되지는 않습니다. cond 속성 또한 XE core 1.4.4 버전에 추가된 가상의 속성으로서 조건문 역할을 합니다.

2.4.1 변수

변수는 프로그램에서 미리 선언한 내용을 출력하거나 사용자들이 입력한 내용을 다시 화면에 출력할 때 필요합니다. 사용자 화면에서 볼 수 있는 대부분의 내용이 모두 변수로 출력된 것이라 해도 과언이 아닙니다.

변수의 사용 형식은 다음과 같습니다.

기본 형식

```
{$string}
```

변수의 기본 형식은 중괄호({ }) 안에 변수 이름을 적고 변수 이름 앞에 달러 기호($)를 붙이는 것입니다. 변수 이름은 '문자열(string)'로 작성해야 하고 미리 선언된 이름만 사용할 수 있습니다.

변수 속의 변수 형식

```
{$string->string}
{$string->string->string}
```

변수는 또 다른 변수를 포함하고 있을 수 있는데 이런 변수를 사용하려면 '하이픈과 꺾은 괄호(->)'를 사용하여 변수와 변수를 연결합니다.

변수 속의 함수 형식

```
{$string->string()}
{$string->string(string | integer)}
```

변수에 함수를 연결하려면 '하이픈과 꺾은 괄호(->)'를 사용합니다. 함수 이름 뒤에는 항상 소괄호(( ))가 붙습니다. 소괄호에는 함수의 내용으로 문자열(string)이나 정수(integer)를 포함할 수 있습니다. 함수 이름으로는 미리 선언된 이름만 사용할 수 있습니다.

2.4.2 XE core 변수

'XE core 변수'란 XE core에서 사용하는 변수를 말합니다. XE core 변수는 여러 모듈에서 두루 사용할 수 있습니다. 특정 모듈에서만 사용할 수 있는 변수들도 있는데 이런 변수들은 그냥 '변수'라고 부릅니다. XE core 변수는 다음과 같습니다.

표 2-4 XE core 변수

변수	설명
`$is_logged`	사용자의 로그인 여부를 확인 `<!--@if($is_logged)-->Welcome!<!--@end-->` `<p cond="$is_logged">Welcome!</p>`
`$current_url`	현재 페이지 URL
`$request_uri`	XE core 설치 URL
`$logged_info`	로그인 사용자에게 자신의 회원정보를 보여 줌
`$logged_info->member_srl`	로그인 사용자 고유번호
`$logged_info->user_id`	로그인 사용자 아이디
`$logged_info->email_address`	로그인 사용자 이메일 주소
`$logged_info->email_id`	로그인 사용자 이메일 아이디
`$logged_info->email_host`	로그인 사용자 이메일 호스트
`$logged_info->user_name`	로그인 사용자 이름
`$logged_info->nick_name`	로그인 사용자 닉네임
`$logged_info->homepage`	로그인 사용자 홈페이지
`$logged_info->blog`	로그인 사용자 블로그
`$logged_info->birthday`	로그인 사용자 생년월일 (YYYYMMDD)
`$logged_info->profile_image`	로그인 사용자 프로필 이미지
`$logged_info->image_name`	로그인 사용자 이름 이미지 경로
`$logged_info->image_mark`	로그인 사용자 그룹 이미지 경로
`$logged_info->signature`	로그인 사용자 서명
`$logged_info->group_list`	로그인 사용자 가입 그룹 목록

변수	설명
$logged_info->is_admin	로그인 사용자가 관리자인지 확인
$logged_info->is_site_admin	로그인 사용자가 가상 사이트 관리자인지 확인
$module_info	모듈 정보로서 현재 모듈의 정보를 보여 줌
$module_info->module	모듈 이름
$module_info->use_mobile	모듈 모바일 스킨 사용 여부
$module_info->mid	모듈 아이디
$module_info->skin	모듈 스킨 이름
$module_info->mskin	모듈 모바일 스킨 이름
$module_info->browser_title	모듈 문서 제목
$module_info->string	모듈에서 만들어진 확장 변수

2.4.3 조건문

주어진 조건에 따라 필요한 내용을 문맥에 알맞게 출력하거나 출력하지 않아야 할 때 조건문이 필요합니다. 조건문은 'if, elseif, else, end'와 '조건식'으로 이루어져 있습니다. if문이 시작되면 반드시 end문으로 닫아서 조건문이 끝났음을 선언해야 합니다. 조건식 내용은 PHP로 해석되기 때문에 PHP에서 사용 가능한 여러 가지 연산자(&&, ||, ==, ! 등)를 사용할 수도 있습니다.

다음은 "Welcome XE!"라는 문장을 표현하는 다양한 조건문 사용법입니다.

표 2-5 조건문 사용 예제

조건문	설명	비고
<!--@if(조건식)--> Welcome XE! <!--@end-->	조건식이 참이면 포함된 내용을 출력	
<block cond="조건식"> Welcome XE! </block>	조건식이 참이면 포함된 내용을 출력	XE core 1.4.4 이상
   Welcome XE! 	조건식이 참이면 요소와 함께 포함된 내용을 출력	XE core 1.4.4 이상
<p attr="value"|cond="조건식"> Welcome XE! 	요소는 무조건 출력하는데 조건식이 참이면 attr="value" 속성과 값을 출력	XE core 1.4.4 이상

> **참고**
>
> <block> 요소는 HTML 표준 요소가 아니라 XE core 1.4.4 버전에 새로 추가된 가상의 요소입니다. HTML 요소의 형식을 빌려 쓰고는 있지만 제어문을 실행할 뿐 실제로 화면에 요소가 출력되지는 않습니다. 'cond' 속성 또한 HTML 표준 속성이 아니며 XE core 1.4.4 버전에 새로 추가된 템플릿 문법의 하나입니다. 'cond'는 조건식을 표현할 때 사용하는 가상의 속성입니다.

if, elseif, else문을 동시에 사용하면 참 또는 거짓뿐만 아니라 여러 가지 조건을 부여하고 조건에 따라 다른 결과를 출력할 수 있습니다.

```
<!--@if(조건A)-->
   <p>조건A를 만족하면 이 문장을 출력합니다.</p>
<!--@elseif(조건B)-->
   <p>조건A를 만족하지 못하고 조건B를 만족하면 이 문장을 출력합니다.</p>
<!--@else-->
   <p>조건A, 조건B를 동시에 만족하지 못하면 이 문장을 출력합니다.</p>
<!--@end-->
```

위와 같은 조건식을 XE core 1.4.4 버전부터 새로 추가된 'cond' 조건식으로 바꾸면 다음과 같이 더 단순하게 표현할 수 있습니다.

```
<p cond="조건A">조건A를 만족하면 이 문장을 출력합니다.</p>
<p cond="조건B">조건B를 만족하면 이 문장을 출력합니다.</p>
<p cond="!조건A && !조건B">조건A, 조건B를 동시에 만족하지 못하면 이 문장을 출력합니다.</p>
```

2.4.4 반복문

주어진 조건에 따라 필요한 내용을 반복해서 출력해야 할 때 반복문이 필요합니다. 반복문은 'foreach, end'와 '조건식'으로 이루어져 있습니다. foreach문이 시작되면 반드시 end문으로 닫아서 반복문이 끝났음을 선언해야 합니다.

표 2-6 반복문 사용 예제

반복문	설명	비고
`<!--@foreach(변수명 as $val)-->` `  <tr>...</tr>` `<!--@end-->`	$key 값 없이 <tr>…</tr> 반복	
`<block loop="변수명=>$val">` `  <tr>...</tr>` `</block>`	$key 값 없이 <tr>…</tr> 반복	XE core 1.4.4 이상
`<tr loop="변수명=>$val">...</tr>`	$key 값 없이 <tr>…</tr> 반복	XE core 1.4.4 이상
`<!--@foreach(변수명 as $key => $val)-->` `  <tr>...</tr>` `<!--@end-->`	$key 값 포함 <tr>…</tr> 반복	
`<block loop="변수명=>$key, $val">` `  <tr>...</tr>` `</block>`	$key 값 포함 <tr>…</tr> 반복	XE core 1.4.4 이상
`<tr loop="변수명=>$key,$val">...</tr>`	$key 값 포함 <tr>…</tr> 반복	XE core 1.4.4 이상

반복문	설명	비고
<!--@foreach($i=0;$i<100;$i++)--> <tr>...</tr> <!--@end-->	초기값 0 부터 시작하여 <tr>…</tr> 100 회 반복	
<block loop="$i=0;$i<100;$i++"> <tr>...</tr> </block>	초기값 0 부터 시작하여 <tr>…</tr> 100 회 반복	XE core 1.4.4 이상
<tr loop="$i=0;$i<100;$i++">...</tr>	초기값 0 부터 시작하여 <tr>…</tr> 100 회 반복	XE core 1.4.4 이상
<!--@while(조건문)--> … <!--@end-->	조건문이 "참"이면 … 를 반복(조건문이 "거짓"이 될 수 없다면 무한 반복 오류가 발생할 수 있음)	

참고

'loop' 속성은 HTML 표준 속성이 아니라 XE core 1.4.4 버전에 새로 추가된 템플릿 문법의 하나입니다. 'loop'는 foreach문의 조건식을 표현할 때 사용하는 가상의 속성입니다.

2.4.5 간단한 PHP 문 사용

XE 스킨 파일에서는 PHP 문법을 사용할 수 없지만, 다음과 같이 중괄호 안에 앳 기호(@)를 포함하면 간단한 PHP 문장을 사용할 수 있습니다.

```
{@$is_logged=Context::get('is_logged')}
```

PHP문을 사용할 때 하나의 문장은 하나의 줄에 작성해야 합니다. 예를 들어 다음과 같은 문장은 PHP에서는 문제가 없지만 XE에서는 치명적인 오류를 내기도 합니다. 여러 개의 문장이 한 줄에 작성되었기 때문입니다.

```
{@$test=364; $test=$test*$test}
```

위 문장은 다음과 같이 한 줄에 한 문장씩 작성해야 합니다.

```
{@
    $test=364;
    $test=$test*$test
}
```

2.4.6 include 문

스킨을 제작할 때 여러 페이지에 걸쳐 반복되는 콘텐츠 블록이 있으면 별도의 파일로 분할하여 관리하는 것이 편리합니다. 하나의 파일만 수정하면 여러 페이지에 한 번에 적용할 수 있기 때문입니다. include는 별도의 파일을 현재 페이지로 불러오는 명령어입니다.

표 2-7 include문 사용 예제

include 문	설명	비고
<!--#include("header.html")-->	header.html 을 포함(include)	

`<include target="header.html" />`	header.html 을 포함(include)	XE core 1.4.4 이상

참고

<include /> 요소는 HTML 표준 요소가 아니라 XE core 1.4.4 버전에 새로 추가된 가상의 요소입니다. HTML 요소의 형식을 빌려 쓰고는 있지만 include문을 실행할 뿐 실제로 화면에 <include /> 요소가 출력되지는 않습니다. 'target' 속성 또한 XE core 1.4.4 버전에 새로 추가된 템플릿 문법의 하나입니다. 'target' 속성에는 포함(include)하려는 파일의 경로를 작성합니다.

2.4.7 CSS 파일 참조

CSS 파일을 HTML 문서에서 참조하는 방법을 설명합니다. XE core 1.4.4 버전부터는 여러 개의 CSS 파일을 하나의 파일로 통합하는 optimizer 기능이 제거되었기 때문에 하나의 모듈을 제작할 때 CSS 파일을 분리하지 않고 되도록 하나의 파일로 작성하는 것을 권장합니다.

CSS 참조

```
<!--%import("xe.css")-->
또는
<load target="xe.css" />
```

HTML 문서의 <head> 요소에서 xe.css 파일을 참조합니다. <load /> 요소는 XE core 1.4.4 버전부터 사용할 수 있습니다. 결과 코드는 다음과 같습니다.

```
<head>
    <link rel="stylesheet" type="text/css" href="xe.css" />
</head>
```

CSS 파일은 HTML 문서의 <head> 요소에서만 참조할 수 있습니다. <body> 요소에서 별도의 CSS 파일을 참조하면 HTML 문법 오류가 발생합니다.

media 지정

```
<load target="xe.css" media="print" />
```

CSS 파일의 대상이 되는 미디어를 선택하여 지정할 수 있습니다. <load /> 요소와 media 속성은 XE core 1.4.4 버전부터 사용할 수 있습니다. 결과 코드는 다음과 같습니다.

```
<head>
    <link rel="stylesheet" type="text/css" href="xe.css" media="print" />
</head>
```

media 속성의 값에 쉼표를 사용하면 여러 개의 값을 지정할 수 있습니다. 기본값은 all입니다. 사용 가능한 값은 다음과 같습니다.

표 2-8 media 속성 값

속성 값	설명
all	media 를 지정하지 않았을 때의 기본값. 모든 출력 장치에 대응
aural	음성 출력 장치(스크린 리더)

속성 값	설명
braille	점자 출력기
embossed	점자 인쇄기
handheld	모바일 장치
print	인쇄기
projection	투사 장치
screen	스크린(모니터)
tty	전신 타자기(Tele Type Writer)
tv	텔레비전

참고

CSS media 속성을 지정할 때에는 의도하는 장치가 CSS 표준을 준수하는지 반드시 확인해야 합니다. 장치가 지원하지 않으면 지정한 media 속성은 적용되지 않습니다.

IE 조건부 주석 사용

```
<load target="xe.css" targetie="IE 6" />
```

targetie 속성을 사용하면 CSS 파일을 IE의 특정 브라우저 버전에서만 해석할 수 있도록 조건부 주석으로 출력합니다. <load /> 요소와 targetie 속성은 XE core 1.4.4 버전부터 사용할 수 있습니다. 결과 코드는 다음과 같습니다.

```
<!--[if IE 6]>
   <link rel="stylesheet" type="text/css" href="xe.css" />
<![endif]-->
```

이 코드는 모든 브라우저에서 주석으로 처리하지만 IE 6 브라우저는 주석으로 처리하지 않고 해석합니다.

CSS 파일 참조 선언 순서 변경

```
<load target="xe.css" index="-1" />
```

index 속성을 사용하면 CSS 파일의 선언 순서를 변경할 수 있습니다. <load /> 요소와 index 속성은 XE core 1.4.4 버전부터 사용할 수 있습니다.

index 속성의 값은 양의 정수 또는 음의 정수를 사용할 수 있습니다. 음의 정수를 사용하면 더 빨리 선언할 수 있고, 양의 정수를 사용하면 더 늦게 선언할 수 있습니다. index="-1" 값을 지정하면 다른 CSS 파일보다 한 줄 빠른 위치에서 선언됩니다.

CSS 파일은 동일한 명령이 충돌하는 경우 나중에 선언된 값이 우선순위를 갖게 되기 때문에 우선순위를 높게 두어야 하는 경우 나중에 선언하는 것이 좋습니다.

2.4.8 JS 파일 참조

JS 파일을 HTML 문서에서 참조하는 방법을 설명합니다. XE core 1.4.4 버전부터는 여러 개의 JS 파일을 하나의 파일로 통합하는 optimizer 기능이 제거되었기 때문에 하나의 모듈을 제작할 때 JS 파일을 분리하지 않고 되도록 하나의 파일로 작성하는 것을 권장합니다.

JS 파일은 HTML 문서의 <head> 요소 또는 <body> 요소에서만 불러올 수 있습니다. <head> 요소 또는 <body> 요소 이외의 요소에서 JS 파일을 참조하면 HTML 문법 오류가 발생하여 JS 파일을 해석하지 못하는 브라우저가 있을 수 있습니다.

<head> 요소에서 JS 파일 참조

다음은 xe.js 파일을 HTML 문서의 <head> 요소에서 불러오는 방법입니다.

```
<!--%import("xe.js")-->
또는
<load target="xe.js" />
```

<load /> 문법은 XE core 1.4.4 버전부터 사용할 수 있습니다. 결과 코드는 다음과 같습니다.

```
<head>
   <script type="text/javascript" src="xe.js"></script>
</head>
```

<body> 요소에서 JS 파일 참조

다음은 xe.js 파일을 HTML 문서의 <body> 요소에서 불러오는 방법입니다.

```
<load target="xe.js" type="body" />
```

<load /> 요소는 대상 파일을 HTML 문서에서 불러옵니다. 속성과 값은 다음과 같습니다.

- media="all | aural | braille | embossed | handheld | print | projection | screen | tty | tv" – CSS 파일의 대상이 되는 미디어를 선택하여 지정할 수 있습니다. 쉼표(,)를 이용하여 여러 개의 값을 지정할 수 있습니다. 기본값은 all으로서 생략하는 경우 media="all"입니다.
- targetie="IE 6 | IE 7 | IE8 | ..." – IE 조건부 주석을 사용하여 IE 브라우저 범위 안에서 CSS/JS 적용 브라우저를 선택할 수 있습니다. 기본값은 빈 값으로서 "적용 안 함"입니다.
- index="integer" – CSS/JS 참조 선언의 위치를 변경할 수 있습니다. 값은 양의 정수와 음의 정수를 사용할 수 있습니다. 양의 정수는 현재 위치보다 나중에 선언되고 음의 정수는 현재 위치보다 먼저 선언됩니다. 기본값은 빈 값으로서 선언 순서를 변경하지 않으며 기본값으로 두는 경우 다른 CSS 파일보다 늦게 선언됩니다.
- type="head | body" – JS 참조 선언 위치를 문서의 <head> 요소로 할 것인지 <body> 요소로 할 것인지 결정할 수 있습니다. 기본값은 head로서 생략하는 경우 JS 파일은 문서 <head> 요소에 포함됩니다.

<load /> 문법은 XE core 1.4.4 버전부터 사용할 수 있습니다. <body> 요소에서 JS 파일을 불러오면 <body> 요소가 끝나기 직전에 JS 파일 참조가 선언됩니다. 결과 코드는 다음과 같습니다.

```
<body>
   ...
   <script type="text/javascript" src="xe.js"></script>
</body>
```

IE 조건부 주석 사용

```
<load target="xe.js" targetie="IE 6" />
```

targetie 속성을 사용하면 JS 파일을 IE의 특정 브라우저 버전에서만 해석할 수 있도록 조건부 주석으로 출력합니다. <load /> 요소와 targetie 속성은 XE core 1.4.4 버전부터 사용할 수 있습니다. 결과 코드는 다음과 같습니다.

```
<!--[if IE 6]>
   <script type="text/javascript" src="xe.js"></script>
<![endif]-->
```

이 코드는 모든 브라우저에서 주석으로 처리하지만 IE 6 브라우저는 주석으로 처리하지 않고 해석합니다.

JS 파일 참조 선언 순서 변경

```
<load target="xe.js" index="-1" />
```

index 속성을 사용하면 JS 파일의 선언 순서를 변경할 수 있습니다. <load /> 요소와 index 속성은 XE core 1.4.4 버전부터 사용할 수 있습니다.

index 속성의 값은 양의 정수 또는 음의 정수를 사용할 수 있습니다. 음의 정수를 사용하면 더 빨리 선언할 수 있고, 양의 정수를 사용하면 더 늦게 선언할 수 있습니다. index="-1" 값을 지정하면 다른 CSS 파일보다 한 줄 빠른 위치에서 선언됩니다.

2.4.9 XML JS 필터 적용

XML JS 필터는 XML 형식으로 입력 항목을 정의해 두면 폼을 전송할 때 유효성 검사를 자동으로 수행하는 기능입니다. 유효성 검사는 XML 기반으로 자동으로 변환된 자바스크립트가 수행하므로, 사용자는 복잡한 자바스크립트를 작성할 필요가 없습니다. XML JS 필터는 유효성 검사를 통과한 폼 데이터를 어떤 모듈의 어떤 명령어로 보낼 것인지 정하는 기능도 합니다.

XML JS 필터를 적용하는 방법은 CSS, JS 파일을 참조하는 문법과 동일합니다.

```
<!--%import("xe.xml")-->
```

이렇게 XML JS 필터를 불러오면 XML 문서는 JS 문서로 컴파일되어 소스 코드로 출력됩니다. XE core 1.4.4 이전 버전은 컴파일된 JS 파일을 무조건 <head> 요소에 포함시킵니다.

출력 결과는 다음과 같습니다.

```
<head>
   <script type="text/javascript" src="xe.js"></script>
</head>
```

XE core 1.4.4 버전부터는 컴파일된 JS 파일을 <body> 요소가 끝나기 직전에 포함시킵니다.

```
<!--%import("xe.xml")-->
또는
<load target="xe.xml" />
```

출력 결과는 다음과 같습니다.

```
<body>
```

```
    ...
    <script type="text/javascript" src="xe.js"></script>
</body>
```

<head> 요소 영역에 출력되지 않고 무조건 <body> 요소가 끝나기 직전에 출력된다는 점이 XE core 1.4.4 미만 버전과 다릅니다.

2.4.10 위젯 삽입하기

위젯은 XE core 또는 모듈에 포함된 정보를 사용자에게 의미 있는 형태로 가공해서 보여주는 인터페이스 역할을 합니다. 웹 사이트 초기 화면에 최근 게시물을 나타내는 위젯과 로그인 양식을 보여주는 위젯이 XE core에 포함되어 있습니다. 위젯이 존재하는 이유는 스킨 제작자가 복잡한 프로그램 로직을 모르는 경우에도 쉽게 원하는 기능을 구현할 수 있도록 지원하기 위해서입니다.

스킨 제작자는 템플릿 코드 한 줄만으로도 원하는 기능을 구현할 수 있습니다. 위젯을 삽입하기 위한 템플릿 코드는 <img /> 요소에 widget이라는 속성을 포함하고 있습니다. <img /> 요소는 HTML 표준이기도 하지만 widget이라는 속성을 포함하게 되면 XE 템플릿 문법으로 해석되어 서버에서 표준 HTML로 변환됩니다.

다음은 로그인 양식을 출력하는 위젯 삽입 코드입니다.

```
<img widget="login_info" skin="xe_official" />
```

위젯을 삽입하기 위한 코드는 XE 관리자 페이지에서 **사이트 설정 > 위젯**을 선택하고 **코드생성** 기능을 사용하면 자동으로 생성됩니다. 자세한 내용은 "XE 사용자 매뉴얼"을 참조하십시오.

2.4.11 XE core 1.4.4 새 템플릿 문법

이 절에서는 XE core 1.4.4부터 추가된 템플릿 문법만 정리했습니다. 이전 버전을 사용하다가 XE core 1.4.4 이상으로 업그레이드한 경우에는 아래 표를 참조하시기 바랍니다.

표 2-9 XE core 1.4.4부터 추가된 템플릿 문법

추가된 템플릿 문법	설명
`<block cond="조건식">` `  <p>Welcome XE!</p>` `</block>`	조건식이 참이면 포함된 내용을 출력
`<p cond="조건식">` `  Welcome XE!` `</p>`	조건식이 참이면 <p> 요소와 함께 포함된 내용을 출력
`<p attr="value"|cond="조건식">` `  Welcome XE!` `</p>`	<p> 요소는 무조건 출력하는데 조건식이 참이면 attr="value" 속성과 값을 함께 출력
`<block loop="변수명=>$val">` `  <tr>...</tr>` `</block>`	$key 값 없이 <tr>…</tr> 반복
`<tr loop="변수명=>$val">...</tr>`	$key 값 없이 <tr>…</tr> 반복

추가된 템플릿 문법	설명
`<block loop="변수명=>$key, $val">` `  <tr>...</tr>` `</block>`	$key 값 포함 <tr>…</tr> 반복
`<tr loop="변수명=>$key,$val">...</tr>`	$key 값 포함 <tr>…</tr> 반복
`<block loop="$i=0;$i<100;$i++">` `  <tr>...</tr>` `</block>`	초기값 0 부터 시작하여 <tr>…</tr> 100 회 반복
`<tr loop="$i=0;$i<100;$i++">...</tr>`	초기값 0 부터 시작하여 <tr>…</tr> 100 회 반복
`<include target="header.html" />`	header.html 을 포함(include)
`<load target="xxx.xxx" />`	CSS/JS/SML JS 필터 파일을 <head>에 포함
`<load target="xxx.xxx" type="body" />`	JS/XML JS 필터 파일을 문서 <body>에 포함
`<unload target="xxx.xxx" />`	경로가 일치하는 CSS/JS/XML JS 필터 파일을 제외

3. 레이아웃 스킨 만들기

이 장에서는 예제 레이아웃 스킨을 사용해서 레이아웃 스킨을 만들고 적용하는 방법을 설명합니다.

3.1 레이아웃 스킨이란

XE는 레이아웃과 콘텐츠가 분리되어 있습니다. 레이아웃은 XE 콘텐츠를 담아내는 구조체 또는 얼개입니다. 일반적인 웹 사이트는 헤더, 글로벌 내비게이션, 로컬 내비게이션, 콘텐츠, 푸터 형식으로 구성되어 있는데 이런 요소의 화면 배치를 결정하는 것이 바로 레이아웃 스킨의 역할입니다. 만약 게시판, 위키와 같은 특정 콘텐츠를 사용자에게 보여줄 때 다른 콘텐츠는 보여줄 필요가 없다면 레이아웃 스킨을 사용하지 않아도 됩니다. 그러나 게시판, 위키와 같은 콘텐츠 이외에 웹 사이트를 일관성 있게 탐색할 수 있도록 헤더, 글로벌 내비게이션, 로컬 내비게이션, 푸터 영역 등을 배치해야 한다면 레이아웃 스킨이 반드시 있어야 합니다.

다음은 레이아웃 스킨을 사용한 일반적인 화면 구성을 나타낸 것입니다.

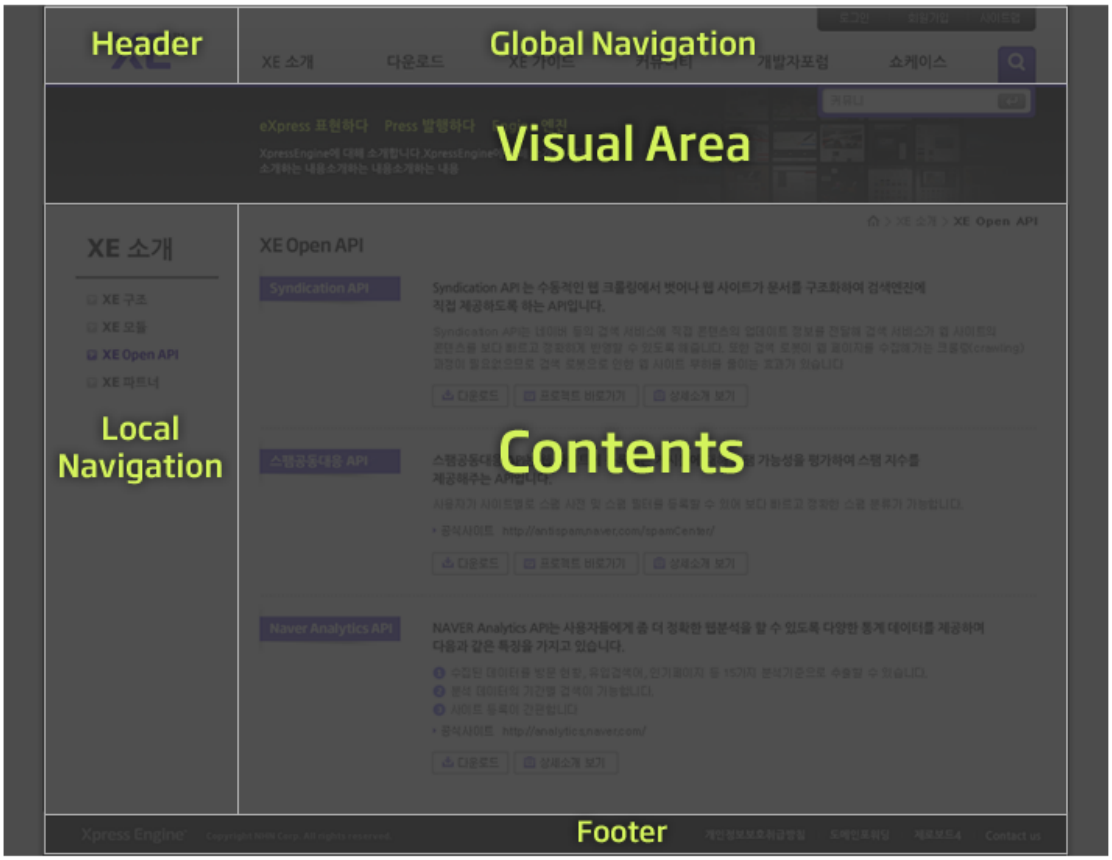


그림 3-1 레이아웃 스킨을 사용한 일반적인 화면 구성

XE core는 하나 이상의 레이아웃 스킨을 포함하고 있습니다. 기본으로 포함되어 있는 레이아웃 스킨은 XE core가 설치된 경로의 /layouts/에 있습니다. 스킨 제작자는 XE core에 포함된 기본 레이아웃 스킨을 사용하거나 새로 만들 수도 있습니다.

3.2 예제 레이아웃 스킨 다운로드

레이아웃 스킨을 새로 만들려면 레이아웃 스킨의 구조에 맞게 디렉터리와 파일을 생성해야 합니다. 이 문서에서는 스킨 제작자의 편의를 위해 예제 레이아웃 스킨을 제공합니다. 예제 레이아웃 스킨은 레이아웃 스킨을 만드는 데 필요한 디렉터리 구조와 파일을 갖추고 있으므로, 원하는 형태의 스킨으로 편리하게 수정할 수 있습니다.

예제 레이아웃 스킨의 다운로드 경로는 다음과 같습니다.

- http://doc.xpressengine.com/manual/user_layout.zip

3.3 레이아웃 스킨의 위치와 디렉터리 구조

3.3.1 레이아웃 스킨의 위치 확인

/xe/라는 사용자 정의 디렉터리에 XE core를 설치했다면 레이아웃 스킨 디렉터리의 위치는 다음과 같습니다.

```
/xe/layouts/
```

다운로드한 예제 레이아웃 스킨(user_layout)의 압축을 해제하고 '/xe/layouts/' 아래에 복사합니다.

```
/xe/layouts/user_layout/
```

> **참고**
> 레이아웃 스킨을 로컬 PC에서 수정할 때는 수정한 내용을 웹 사이트의 레이아웃 스킨 디렉터리에 반영해야 합니다.

3.3.2 레이아웃 스킨 디렉터리 구조

레이아웃 스킨을 제작할 때 반드시 갖추어야 할 디렉터리 구조가 있습니다. 이 구조를 유지하지 않으면 레이아웃 스킨이 될 수 없습니다.

예제 레이아웃 스킨(user_layout)의 구조는 다음과 같습니다.

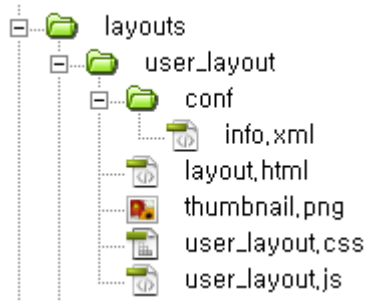


그림 3-2 레이아웃 스킨 디렉터리 구조

info.xml

info.xml은 레이아웃 스킨의 기본 정보들을 포함하며 관리자 화면에 보여줄 설명과 옵션을 제공합니다. info.xml은 반드시 다음 경로에 존재해야 합니다. 'conf' 디렉터리 이름과 'info.xml' 파일 이름은 변경할 수 없습니다.

```
/xe/layouts/user_layout/conf/info.xml
```

layout.html

layout.html은 레이아웃 스킨의 HTML과 CSS, JS 참조 정보를 포함합니다. 'layout.html' 파일 이름은 변경할 수 없습니다.

```
/xe/layouts/user_layout/layout.html
```

thumbnail.png

thumbnail.png는 레이아웃 미리보기용 이미지 파일입니다. 너비 180픽셀, 높이 120픽셀 이상 크기로 제작하여 넣어두면 관리자 화면에서 미리보기 이미지로 출력합니다. 'thumbnail.png'라는 파일 이름은 변경할 수 없습니다.

```
/xe/layouts/user_layout/thumbnail.png
```

CSS, JS, IMG

CSS, JS, IMG 파일은 레이아웃 스킨의 필수 요소가 아니므로 필요에 따라 생성해서 사용합니다. 'conf' 디렉터리를 제외한 어느 곳에 위치해도 무방합니다. 관리해야 할 파일의 수가 많아지면 /user_layout/ 디렉터리 아래에 css, js, img와 같은 이름의 디렉터리를 추가로 생성해서 관리할 수 있습니다.

3.4 레이아웃 스킨 정보 작성

레이아웃 스킨 정보는 info.xml에서 작성합니다. info.xml의 기본 구조는 다음과 같습니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<layout version="0.2">
   <title xml:lang="ko">테스트 레이아웃</title>
   <title xml:lang="en">Test Layout</title>
   <title xml:lang="jp">テストのレイアウト</title>
   <description xml:lang="ko">디자인이 없는 테스트 레이아웃입니다. 새 스킨을 만들 때 이 레이아웃
사본을 사용하면 좋습니다.</description>
   <description xml:lang="en">Is not designed for testing the layout. When you create a
new skin, we recommend you copy the layout.</description>
   <description
xml:lang="jp">デザインがないテストレイアウトです。新しいスキンを作成するときに、レイアウトのコピーを使用
してください。</description>
   <version>1.0</version>
   <date>2010-12-24</date>
   <author email_address="user@user.com" link="http://user-define.com/">
      <name xml:lang="ko">제작자 이름</name>
      <name xml:lang="en">Maker Name</name>
      <name xml:lang="jp">製作者名</name>
   </author>
   <menus>
      <menu name="main_menu" maxdepth="3" default="true">
         <title xml:lang="ko">전역 메뉴</title>
         <title xml:lang="en">Global Menu</title>
         <title xml:lang="jp">グローバルメニュー</title>
      </menu>
   </menus>
</layout>
```

위 코드의 내용은 다음과 같습니다.

코드	설명
`<?xml version="1.0" encoding="UTF-8"?>`	XML 문서 형식 선언
`<layout version="0.2">`	레이아웃 정보 문서임을 선언. version 에는 XE core 에서 지원하는 버전을 표시해야 합니다. XE core 1.4.4.2 를 기준으로 지원하는 레이아웃 버전은 0.2 입니다.
`<title xml:lang="ko">...</title>`	레이아웃 이름
`<description xml:lang="ko">...</description>`	레이아웃 설명
`<version>...</version>`	레이아웃 스킨의 버전
`<date>YYYY-MM-DD</date>`	레이아웃 스킨 제작 날짜. 년-월-일(YYYY-MM-DD) 형식으로 작성합니다.
`<author email_address="..." link="...">` `<name xml:lang="ko">...</name>` `</author>`	레이아웃 스킨 제작자 정보. 이메일 주소, 웹 사이트 주소, 제작자 이름을 작성합니다.
`<menus>`	XE 관리자 페이지의 제어판에서 생성한 메뉴와 연결할 수

코드	설명
<menu name="main_menu" maxdepth="2" default="true"> <title xml:lang="ko">...</title> </menu> </menus>	있습니다. <menus>...</menus> 요소 안에 2 개 이상의 <menu>...</menu>를 작성할 수 있습니다.

이 밖에도 info.xml에는 사용자 정의 가능한 여러 타입의 변수를 <extra_vars> 요소 안에 추가로 포함할 수 있습니다. 다음은 스킨 제작자가 select, image, text, textarea 타입의 데이터를 입력받아서 변수로 사용할 수 있도록 확장한 예제입니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<layout version="0.2">

   ...


   <extra_vars>
      <var name="colorset" type="select">
         <title xml:lang="ko">컬러셋</title>
         <description xml:lang="ko">원하는 컬러셋을 선택해주세요.</description>
         <options value="black">
            <title xml:lang="ko">Black (기본)</title>
         </options>
         <options value="white">
            <title xml:lang="ko">White</title>
         </options>
      </var>
      <var name="logo_image" type="image">
         <title xml:lang="ko">로고 이미지</title>
         <description xml:lang="ko">페이지 상단에 출력할 로고이미지를 입력하세요.</description>
      </var>
      <var name="logo_url" type="text">
         <title xml:lang="ko">로고 이미지 링크</title>
         <description xml:lang="ko">로고 이미지를 클릭할 때 이동할 URL을 입력해
주세요.</description>
      </var>
      <var name="title_description" type="textarea">
         <title xml:lang="ko">본문 상단 내용</title>
         <description xml:lang="ko">본문 상단에 표시되는 영역의 내용을 입력해주세요.(HTML 사용
가능)</description>
      </var>
   </extra vars>

   ...

</layout>
```

위 코드의 내용은 다음과 같습니다.

코드	설명
<extra_vars>...</extra_vars>	확장 변수 컨테이너
<var name="colorset" type="select">...</var>	select 타입의 확장 변수. 스킨 제작자는 {$layout_info->colorset} 형식으로 출력 가능

코드	설명
`<var name="logo_image" type="image">...</var>`	image 타입의 확장 변수. 스킨 제작자는 {$layout_info->image} 형식으로 출력 가능
`<var name="logo_url" type="text">...</var>`	text 타입의 확장 변수. 스킨 제작자는 {$layout_info->logo_url} 형식으로 출력 가능
`<var name="title_description" type="textarea">...</var>`	textarea 타입의 확장 변수. 스킨 제작자는 {$layout_info->title_description} 형식으로 출력 가능

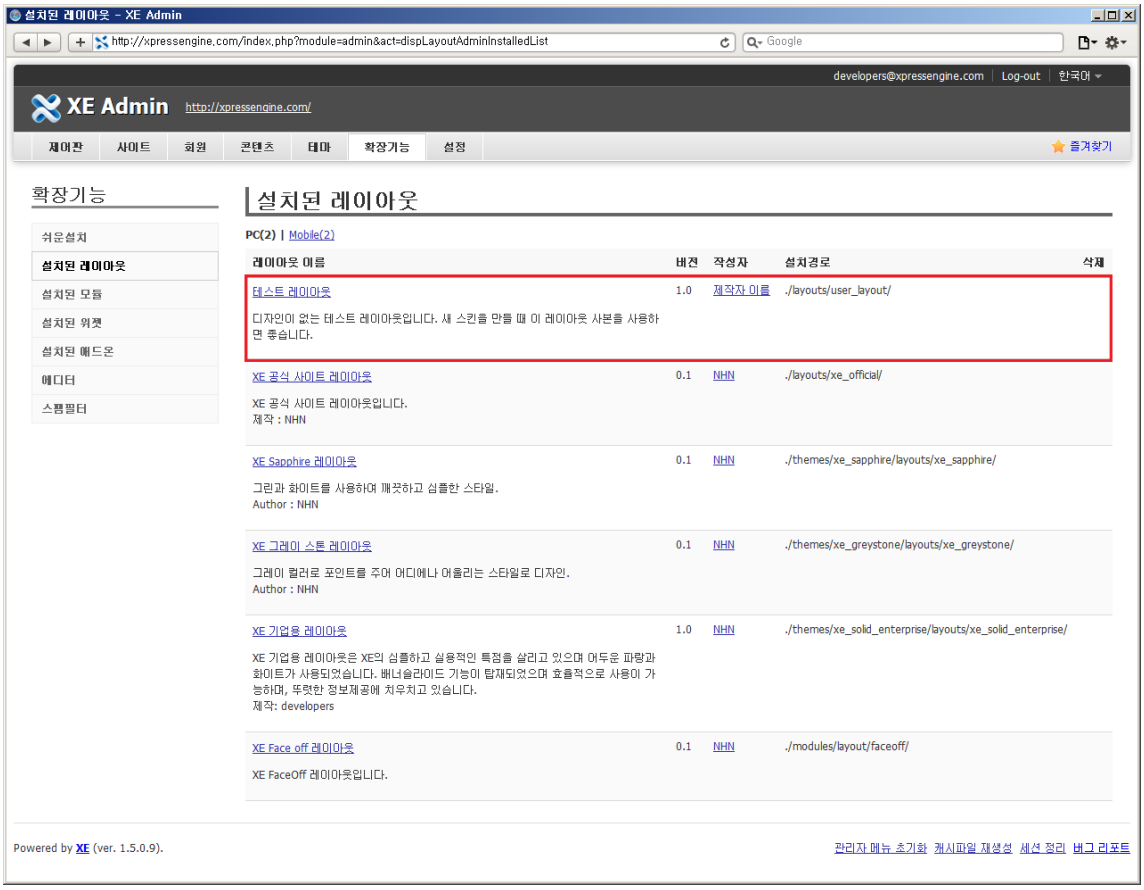
info.xml에서 추가된 확장 변수는 레이아웃이 생성된 이후 **레이아웃 설정** 페이지에 나타납니다. 웹 사이트 관리자가 확장 변수에 어떤 값을 입력하면 그 결과를 스킨에서 출력할 수 있습니다.

3.5 레이아웃 생성

사용자 정의 레이아웃 확인

info.xml을 올바르게 작성했는지 확인하는 방법은 다음과 같습니다.

1. XE 관리자 페이지에서 **확장기능 > 설치된 레이아웃**을 엽니다.
2. **테스트 레이아웃**이 제대로 표시되는지 확인합니다.



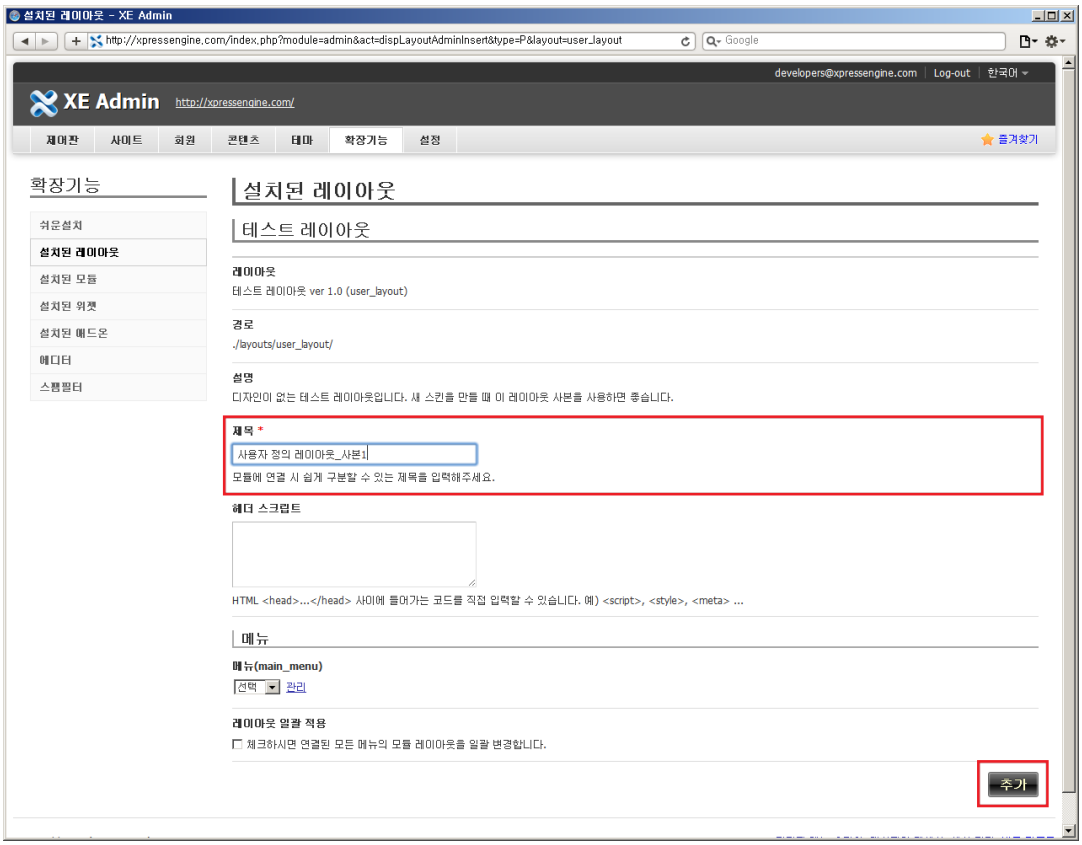
메뉴 수는 info.xml에서 <menu> 요소를 하나만 포함했기 때문에 **1**로 표시됩니다. <menu> 요소를 여러 개 작성하면 **메뉴 수**는 <menu> 요소의 개수만큼 증가합니다.

레이아웃 사본 생성

설치된 레이아웃인 '테스트 레이아웃'에서 레이아웃 '사본'을 생성해야 비로소 사용할 수 있는 상태가 됩니다. 사용자는 'user_layout'이라는 사본을 여러 개 반복해서 생성할 수도 있습니다.

레이아웃 사본을 생성하는 방법은 다음과 같습니다.

1. **테스트 레이아웃** 페이지를 열고 **추가**를 클릭합니다.
2. **제목**에 사본의 이름을 입력하고 **추가**를 클릭합니다. **제목**은 'user_layout'의 또 다른 사본과 구별할 수 있게 입력해야 합니다. 이 예제에서는 **테스트 레이아웃** 사본 이름으로 *사용자 정의 레이아웃_사본1*을 입력했습니다.



3. **테스트 레이아웃** 페이지에서 다음과 같이 **사용자 정의 레이아웃_사본1**이 생성된 것을 확인할 수 있습니다.

이제 이 레이아웃을 사용하고자 하는 모듈의 설정 화면에서 **사용자 정의 레이아웃_사본1**을 선택할 수 있습니다.

> **참고**
>
> layout.html을 생성하지 않은 상태에서 이 레이아웃을 적용한 모듈 페이지에 접근하면 "Err : "./layouts/user_layout/layout.html" template file does not exist." 오류 메시지가 화면에 출력됩니다.

3.6 레이아웃 스킨 작성

레이아웃 스킨의 내용은 layout.html에서 작성합니다.

XE는 'DOCTYPE, html, head, body' 요소를 XE core에서 자동으로 출력한다는 특징이 있습니다. 따라서 스킨 제작자는 이 요소들을 스킨 파일에서 작성할 필요가 없습니다. 스킨 파일에 이 요소들을 포함하면 중첩 문법으로 처리되어 HTML 오류가 발생하거나 화면 깨짐 현상이 발생합니다.

레이아웃 스킨을 적용한 웹 페이지의 기본 구조는 다음과 같습니다.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html lang=".." xml:lang=".." xmlns="http://www.w3.org/1999/xhtml">
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
   <!-- 이곳에 포함할 내용은 layout.html 문서 또는 관리자 모드에서 작성됨 -->
</head>
<body>
   <!-- layout.html 문서의 코드와 내용은 이곳에 출력된다 -->
</body>
</html>
```

'layout.html' 페이지에서 작성하는 HTML 코드와 내용은 모두 <body> 요소 안쪽에 HTML 코드로 출력됩니다. 따라서 스킨 제작자는 <body> 요소 안쪽에서 사용할 수 있는 유효한 HTML 문법으로 'layout.html'를 작성해야 합니다. <head> 요소에 포함해야 할 내용은 XE 템플릿 문법을 사용하여 <body> 요소 내부에서 작성할 수 있습니다.

레이아웃 스킨의 문서 구조

예제 레이아웃 스킨은 header, gnb(global navigation bar), body, lnb(local navigation bar), content, footer로 구성되어 있습니다. 따라서 layout.html의 문서 구조는 다음과 같습니다.

```
<div class="user_layout">
   <div class="header">
      <h1>Site Logo</h1>
      <hr />
      <div class="gnb">.gnb</div>
   </div>
   <hr />
   <div class="body">
      <div class="lnb">.lnb</div>
      <hr />
      <div class="content">.content</div>
   </div>
   <hr />
   <div class="footer">.footer</div>
</div>
```

HTML로 작성된 요소에 클래스 이름을 지정한 이유는 CSS 파일에서 HTML 요소를 선택해서 원하는 배치를 구현할 수 있도록 하기 위해서입니다.

{$content} 변수로 본문 출력

layout.html은 아직 사용자 화면에서 확인할 수 없습니다. 어떤 모듈에서도 아직 이 레이아웃을 선택하지 않았기 때문입니다. 레이아웃 스킨 페이지는 스스로는 존재할 수 없고 특정 모듈에서 사용될 때 해당 모듈 페이지에 접근하면서 확인할 수 있습니다. 레이아웃 스킨은 '모듈'에 의존적입니다.

예제 레이아웃 스킨을 사용자 화면에서 확인하려면 특정 모듈(페이지, 게시판, 위키 등)을 하나 생성하여 이 레이아웃 스킨과 연결해야 합니다. 그 전에 스킨 제작자는 레이아웃 스킨과 연결된 모듈(페이지, 게시판, 위키 등)에서 레이아웃 스킨을 화면에 제대로 표시할 수 있도록 '내용 변수'를 작성해야 합니다.

예제 레이아웃 스킨에서는 다음과 같이 레이아웃의 본문 영역에 {$content} 변수를 사용해서 '내용 변수'를 작성했습니다.

```
<div class="user_layout">
   <div class="header">
      <h1>Site Logo</h1>
      <hr />
      <div class="gnb">.gnb</div>
   </div>
   <hr />
   <div class="body">
      <div class="lnb">.lnb</div>
      <hr />
      <div class="content">
         .content{$content}
      </div>
   </div>
   <hr />
   <div class="footer">.footer</div>
</div>
```

{$content}는 레이아웃 스킨에서 사용할 수 있는 가장 중요한 변수로서, 현재의 레이아웃 스킨이 어떤 종류의 모듈과 연결되더라도 연결된 모듈의 내용을 본문 영역에 출력합니다. 어떤 모듈과 연결되는지에 따라서 정적인 페이지가 될 수도 있고 동적인 게시판, 위키 페이지가 될 수도 있습니다.

XE에서 모듈과 레이아웃의 관계는 '레이아웃 스킨'에 여러 종류의 '모듈'을 탑재하는 것이 아니라 여러 종류의 '모듈'에 '레이아웃 스킨'을 연결하는 개념입니다. XE로 생성한 모듈 페이지에는 고유한 URL이 있지만, 레이아웃 스킨에는 URL이 없고 '모듈'에 연결된 상황에서만 표시된다는 것을 기억해 주시기 바랍니다.

글로벌 메뉴 출력

.gnb 요소 내부에 레이아웃과 연결된 글로벌 메뉴를 출력할 수 있습니다. XE 관리자 페이지에서 레이아웃과 메뉴를 연결하면 레이아웃은 항상 연결된 메뉴를 화면에 표시하는데, 이때 연결된 메뉴를 표시할 수 있도록 레이아웃 스킨에서 미리 코드를 작성해야 합니다.

예제 레이아웃 스킨에서는 다음과 같이 연결된 메뉴를 화면에 출력하는 코드를 작성했습니다. 메뉴를 최대 2단계까지 출력합니다.

```

<div class="gnb">
```

```
    .gnb
    <ul>
        <li loop="$global_menu->list=>$key1,$val1"
class="active"|cond="$val1['selected']"><a href="{$val1['href']}"
target="_blank"|cond="$val1['open_window']=='Y'">{$val1['link']}</a>
            <ul cond="$val1['list']">
                <li loop="$val1['list']=>$key2,$val2"
class="active"|cond="$val2['selected']"><a href="{$val2['href']}"
target="_blank"|cond="$val2['open_window']=='Y'">{$val2['link']}</a></li>
            </ul>
        </li>
    </ul>
</div>
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

템플릿 문법/변수	설명	구분
loop="$global_menu->list=>$key1,$val1"	연결 메뉴 목록이 있으면 출력	반복문
cond="$val1['list']"	연결 메뉴가 하위 목록을 포함하면 출력	조건문
loop="$val1['list']=>$key2,$val2"	연결 메뉴가 하위 목록을 포함하면 출력	반복문
class="active"|cond="$val1['selected'] class="active"\|cond="$val2['selected']"	연결 메뉴 또는 연결 메뉴 하위 목록과 현재 페이지가 일치하면 요소에 class="active" 속성을 추가	조건문
target="_blank"|cond="$val1['open_window']=='Y'" target="_blank"\|cond="$val2['open_window']=='Y'"	연결 메뉴 링크 옵션이 새 창이면 target="_blank" 출력	조건문
{$val1['href']} {$val2['href']}	연결 메뉴 링크 URL	변수
{$val1['link']} {$val2['link']}	연결 메뉴 링크 텍스트	변수

로컬 메뉴 출력

.lnb 요소 내부에 레이아웃과 연결된 로컬 메뉴를 출력할 수 있습니다. 로컬 메뉴를 출력하는 문법은 기본적으로 글로벌 메뉴를 출력하는 방법과 유사합니다.

글로벌 메뉴와 로컬 메뉴 출력의 다른 점은 로컬 메뉴가 1단계 메뉴를 표시하지 않는다는 점입니다. 글로벌 메뉴가 .gnb 영역에 이미 1~2단계를 표시하고 있기 때문에 로컬 메뉴는 현재 페이지를 기준으로 관련된 2~3단계 메뉴를 표시합니다.

로컬 메뉴는 메뉴에 연결된 페이지가 있을 때에만 출력되기 때문에 메뉴를 생성하는 과정에서 실제 연결할 수 있는 페이지의 URL을 입력해야 합니다.

예제 레이아웃 스킨에서는 다음과 같이 로컬 메뉴 출력 코드를 작성했습니다.

```
<div class="lnb">
    .lnb
    <h2 loop="$global menu->list=>$key1,$val1" cond="$val1['selected']"><a
href="{$val1['href']}"
target="_blank"|cond="$val1['open_window']=='Y'">{$val1['link']}</a></h2>
```

```
   <ul loop="$global_menu->list=>$key1,$val1" cond="$val1['selected'] && $val1['list']">
      <li loop="$val1['list']=>$key2,$val2" class="active"|cond="$val2['selected']"><a
href="{$val2['href']}"
target="_blank"|cond="$val2['open_window']=='Y'">{$val2['link']}</a>
         <ul cond="$val2['list']">
            <li loop="$val2['list']=>$key3,$val3"
class="active"|cond="$val3['selected']"><a href="{$val3['href']}"
target="_blank"|cond="$val3['open_window']=='Y'">{$val3['link']}</a>
            </li>
         </ul>
      </li>
   </ul>
</div>
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

템플릿 문법/변수	설명	구분
`loop="$global_menu->list=>$key1,$val1"`	연결 메뉴 목록이 있으면 출력	반복문
`cond="$val1['selected']"`	연결 메뉴와 현재 페이지가 일치하면 내용을 출력	조건문
`cond="$val1['selected'] && $val1['list']"`	연결 메뉴와 현재 페이지가 일치하고 동시에 하위 목록을 포함하면 출력	
`loop="$val1['list']=>$key2,$val2"` `loop="$val2['list']=>$key3,$val3"`	연결 메뉴가 하위 목록을 포함하면 출력	반복문
`cond="$val2['list']"`	연결 메뉴가 하위 목록을 포함하면 출력	조건문
`class="active"|cond="$val2['selected']"` `class="active"\|cond="$val3['selected']"`	연결 메뉴 또는 연결 메뉴 하위 목록과 현재 페이지가 일치하면 요소에 class="active" 속성을 추가	조건문
`target="_blank"|cond="$val1['open_window']=='Y'"` `target="_blank"\|cond="$val2['open_window']=='Y'"` `target="_blank"|cond="$val3['open_window']=='Y'"`	연결 메뉴 링크 옵션이 새 창이면 target="_blank" 출력	조건문
`{$val1['href']}` `{$val2['href']}` `{$val3['href']}`	연결 메뉴 링크 URL	변수
`{$val1['link']}` `{$val2['link']}` `{$val3['link']}`	연결 메뉴 링크 텍스트	변수

통합검색 양식 출력

웹 사이트 통합검색은 특정 모듈만을 검색 대상으로 하지 않고 생성된 모든 모듈을 검색 결과에 포함시키는 기능입니다. 그러나 통합검색 양식을 제공하지 않으면 통합검색 기능도 사용할 수 없습니다. 통합검색 양식은 보통 레이아웃 스킨에서 제공합니다. 통합검색 양식 코드는 XE 관리자 페이지의 **확장기능 > 설치된 모듈 > 통합검색**에서 제공합니다.

예제 레이아웃 스킨에서는 다음과 같이 ... 요소에 통합검색 양식 코드 <form class="search">...</form>을 추가했습니다.

```
<div class="user layout">
   <div class="header">
      <h1>Site Logo</h1>
      <form action="{getUrl()}" method="get" class="search">
         <input type="hidden" name="vid" value="{$vid}" />
         <input type="hidden" name="mid" value="{$mid}" />
         <input type="hidden" name="act" value="IS" />
         <input type="text" name="is_keyword" value="{$is_keyword}" title="{$lang-
>cmd_search}" class="iText" />
         <input type="submit" value="{$lang->cmd_search}" class="btn" />
      </form>
      <hr />
      <div class="gnb">
         .gnb
         ...
      </div>
   </div>
   <hr />
   <div class="body">
      <div class="lnb">
         .lnb
         ...
      </div>
      <hr />
      <div class="content">{$content}</div>
   </div>
   <hr />
   <div class="footer">.footer</div>
</div>
```

위 코드에서 사용된 변수는 다음과 같습니다.

변수	설명
{getUrl()}	사용자의 검색어를 전송할 페이지의 URL
{$vid}	가상 사이트 아이디
{$mid}	모듈 아이디
{$is_keyword}	사용자가 입력한 통합검색 키워드. 검색 결과 페이지에서 사용자가 입력한 키워드를 다시 보여 줍니다.
{$lang->cmd_search}	언어 변수. '검색'이라는 단어가 변수에 할당되어 있습니다.

로그인 양식 출력

운영하려는 웹 사이트가 회원제 웹 사이트면 회원 가입을 받고 회원이 로그인할 수 있어야 합니다. XE core에는 이미 회원 가입 페이지와 로그인 양식이 준비되어 있습니다. 로그인 양식을 레이아웃 스킨에 포함하려면 원하는 위치에 로그인 양식 출력 코드를 추가합니다.

예제 레이아웃 스킨에서는 다음과 같이 ... 내부의 영역 윗부분에 로그인 위젯을 출력하도록 작성했습니다.

```
...
<div class="lnb">
   .lnb
   <div class="account">
      <img widget="login_info" skin="xe_official" />
   </div>
   <h2>...</h2>
   <ul>...</ul>
```

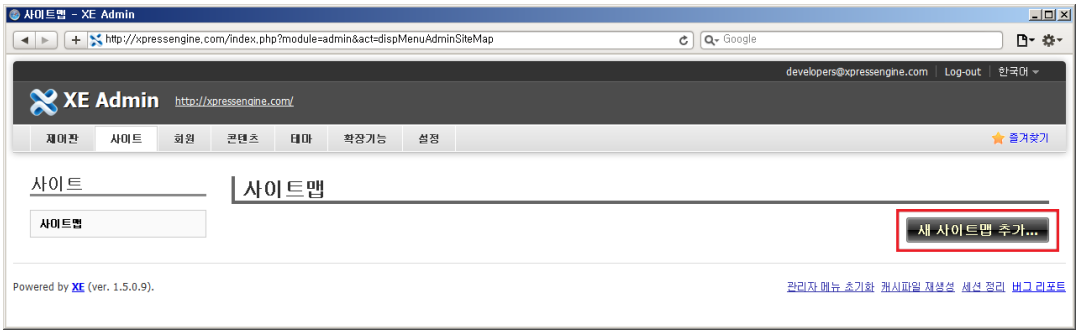
```
</div>
...
```

3.7 사이트맵 생성

사이트맵은 사이트 메뉴를 의미합니다. XE 관리자 페이지에서 사이트맵을 생성하고 원하는 레이아웃 사본과 연결하면 레이아웃이 출력될 때 사이트맵을 메뉴로 출력할 수 있습니다.

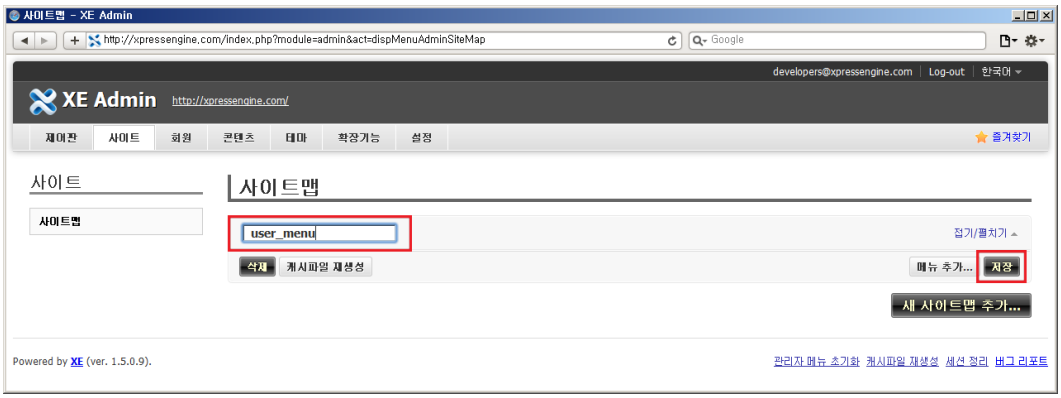
사이트맵을 생성하는 방법은 다음과 같습니다.

1. XE 관리자 페이지에서 **사이트 > 사이트맵**을 선택합니다.
2. 사이트맵 페이지에서 **새 사이트맵 추가**를 클릭합니다.

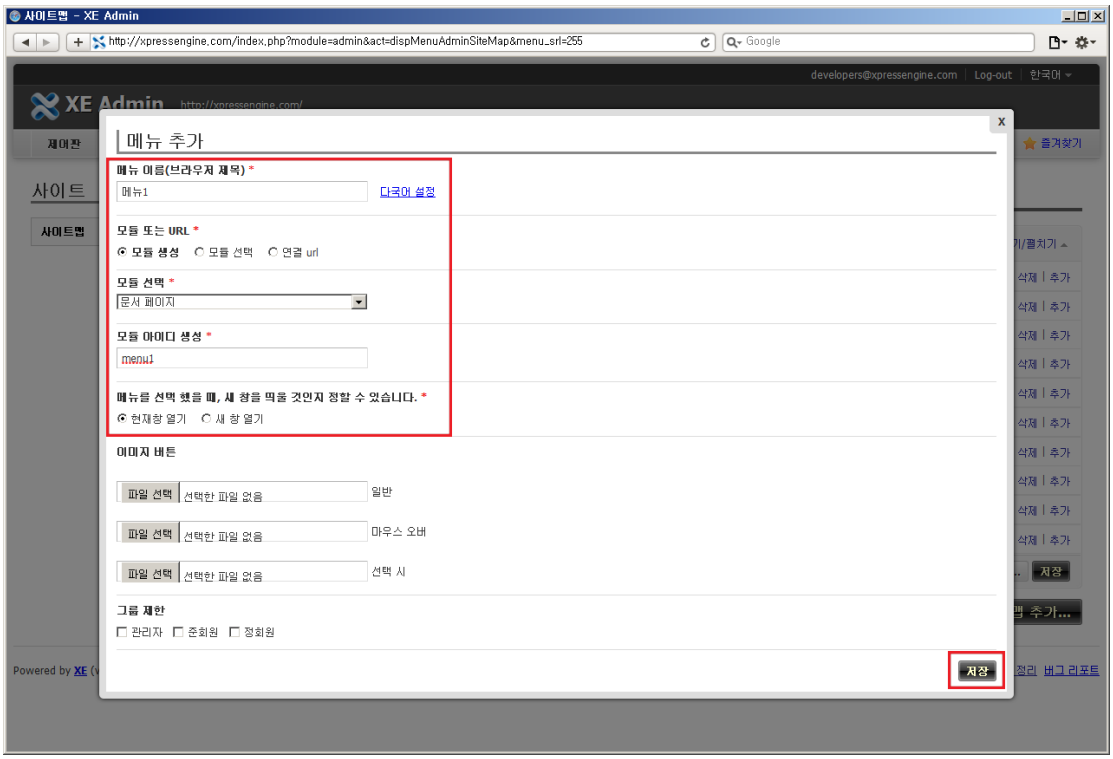


사이트맵은 여러 세트를 추가로 생성할 수 있는데 레이아웃과 연결할 때 다른 사이트맵과 구별할 수 있도록 **제목**을 반드시 입력해야 합니다.

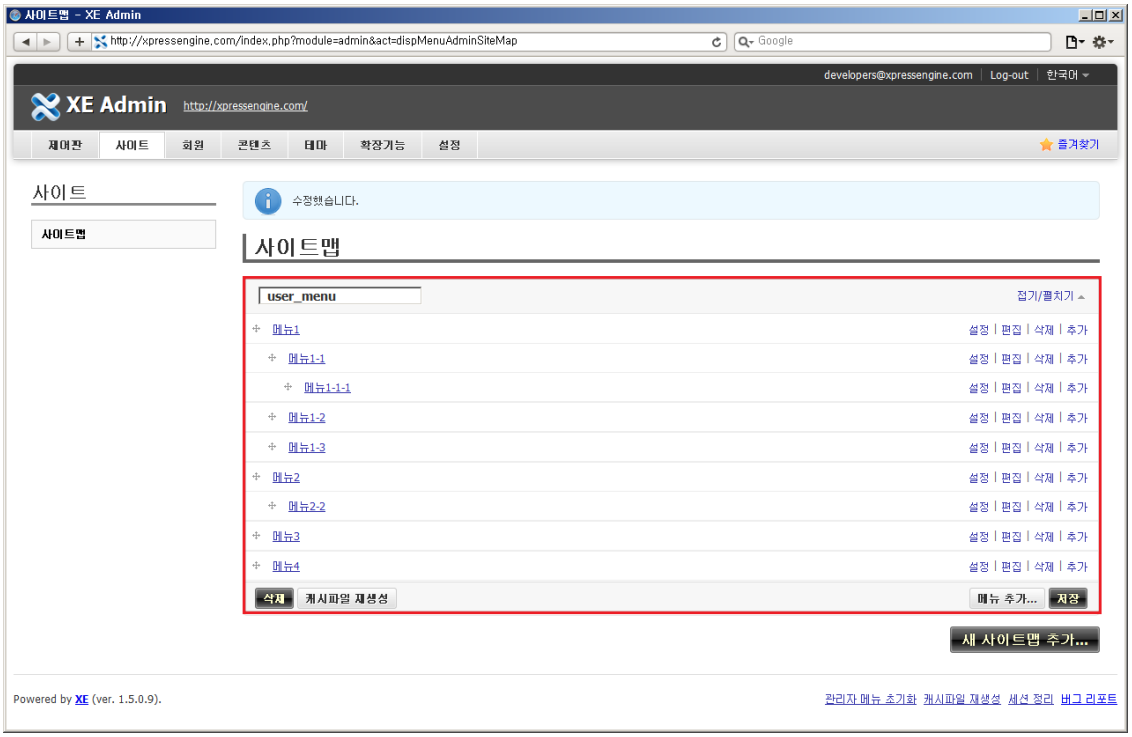
3. **제목**에 *user_menu*라고 입력하고 **저장**을 클릭합니다.



4. **user_menu**에서 **메뉴 추가**를 클릭하고 웹 사이트의 글로벌 메뉴 구조를 완성합니다. 자세한 메뉴 생성 방법은 "XE 사용자 매뉴얼"을 참조하십시오.



5. user_menu라는 이름의 사이트맵이 생성되면 **저장**을 클릭합니다.



아직은 'user_menu'라는 메뉴를 사용자 화면에서 볼 수 없습니다. 레이아웃에서 메뉴를 참조해야만 사용자 화면에 출력됩니다.

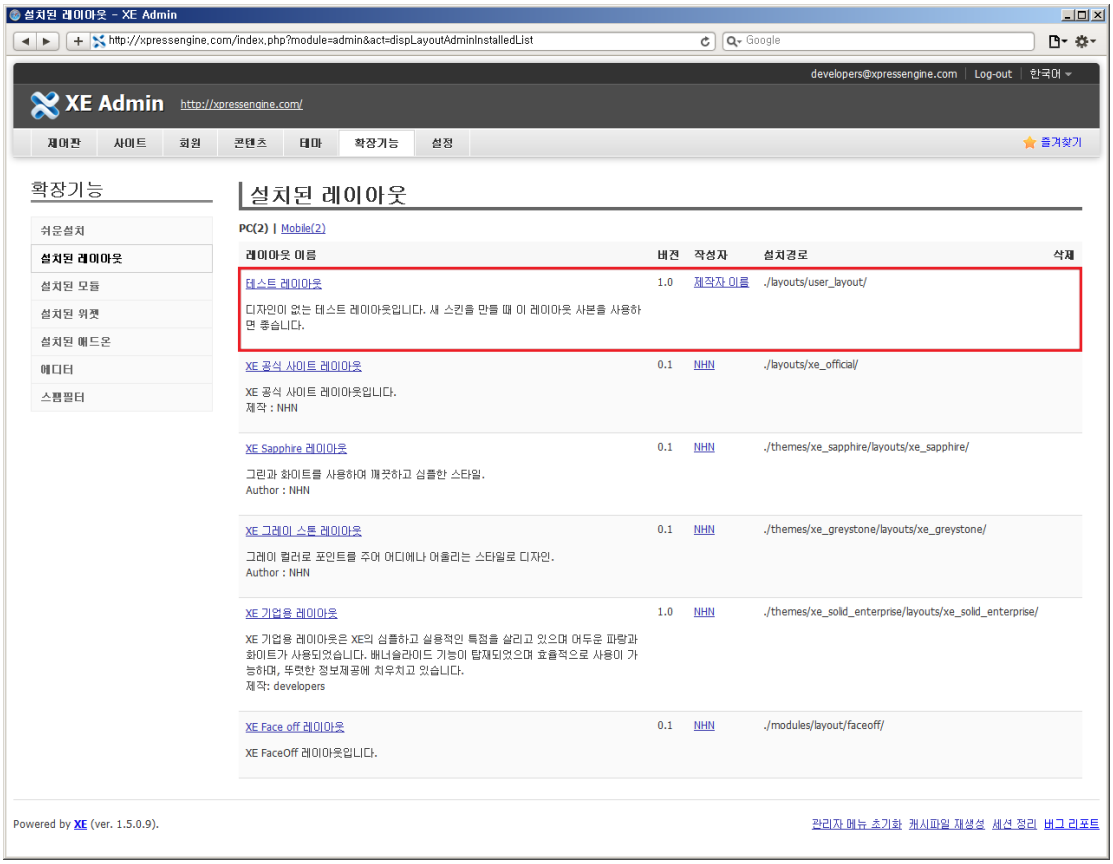
참고

메뉴를 생성하는 과정에서 **모듈 또는 URL** 항목에는 유효한 **모듈 아이디** 또는 **연결 URL**을 지정해야 합니다. **모듈 아이디** 또는 **연결 URL**이 유효하지 않으면 레이아웃에서 메뉴가 정상적으로 표시되지 않을 수 있습니다.

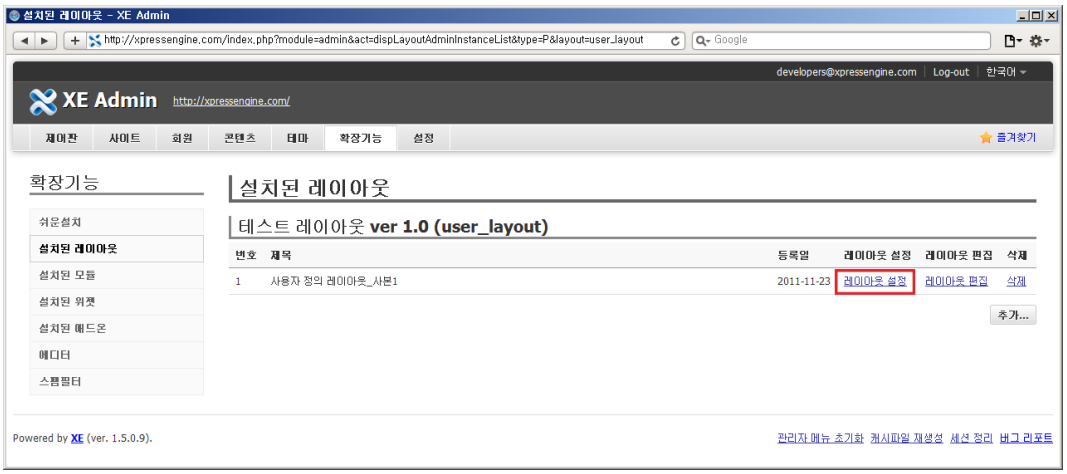
3.8 레이아웃에 사이트맵 연결

'user_menu'라는 사이트맵이 생성되었기 때문에 이제는 레이아웃에서 이 사이트맵을 참조할 수 있습니다. 레이아웃에 'user_menu'를 연결하는 방법은 다음과 같습니다.

1. XE 관리자 페이지에서 **확장기능 > 설치된 레이아웃**을 선택합니다.

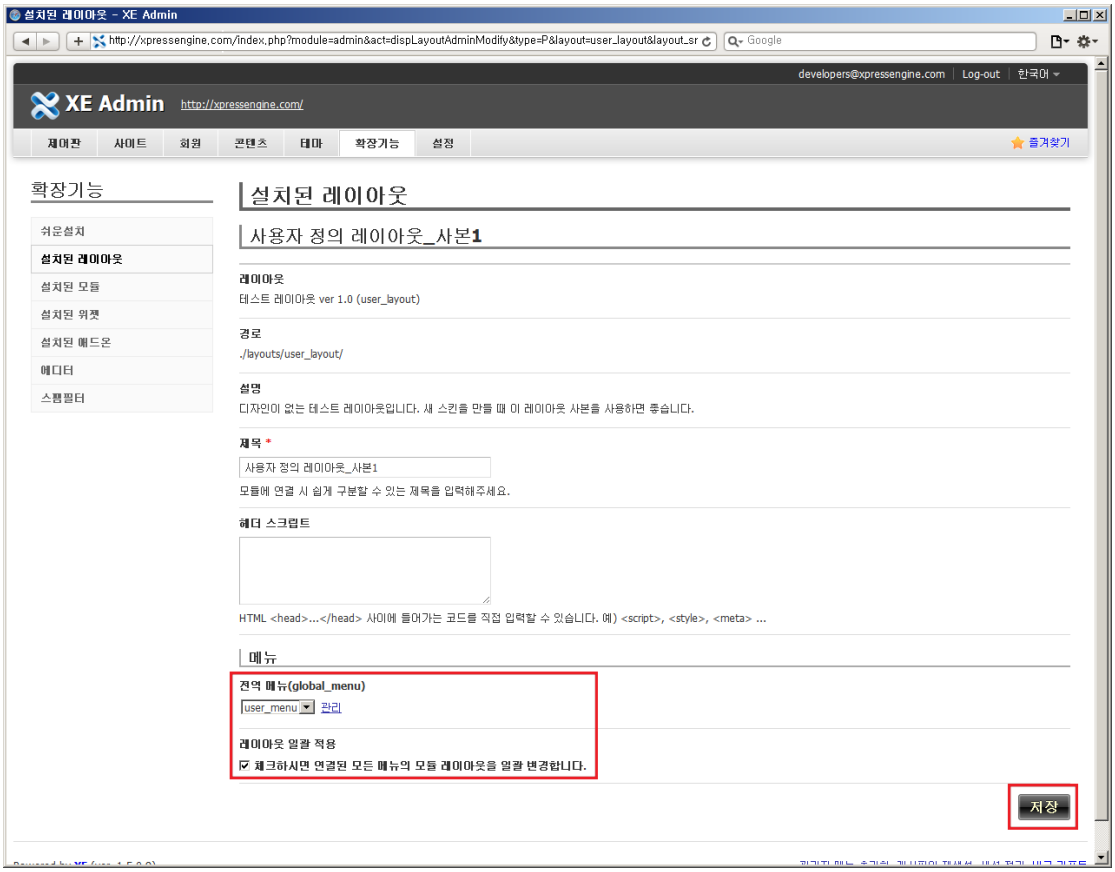


2. **테스트 레이아웃** 항목을 열고 **사용자 정의 레이아웃_사본1**의 **레이아웃 설정**을 클릭합니다.



3. **메뉴 > 전역 메뉴(global_menu)**에서 **user_menu**를 선택하고 **레이아웃 일괄 적용**을 선택한 다음 **저장**을 클릭합니다.

레이아웃 일괄 적용을 선택하면 메뉴를 통해서 연결되어 있는 모든 모듈의 레이아웃이 현재의 레이아웃으로 한 번에 변경됩니다. 이후 **user_menu**는 항상 이 레이아웃과 함께 출력됩니다.

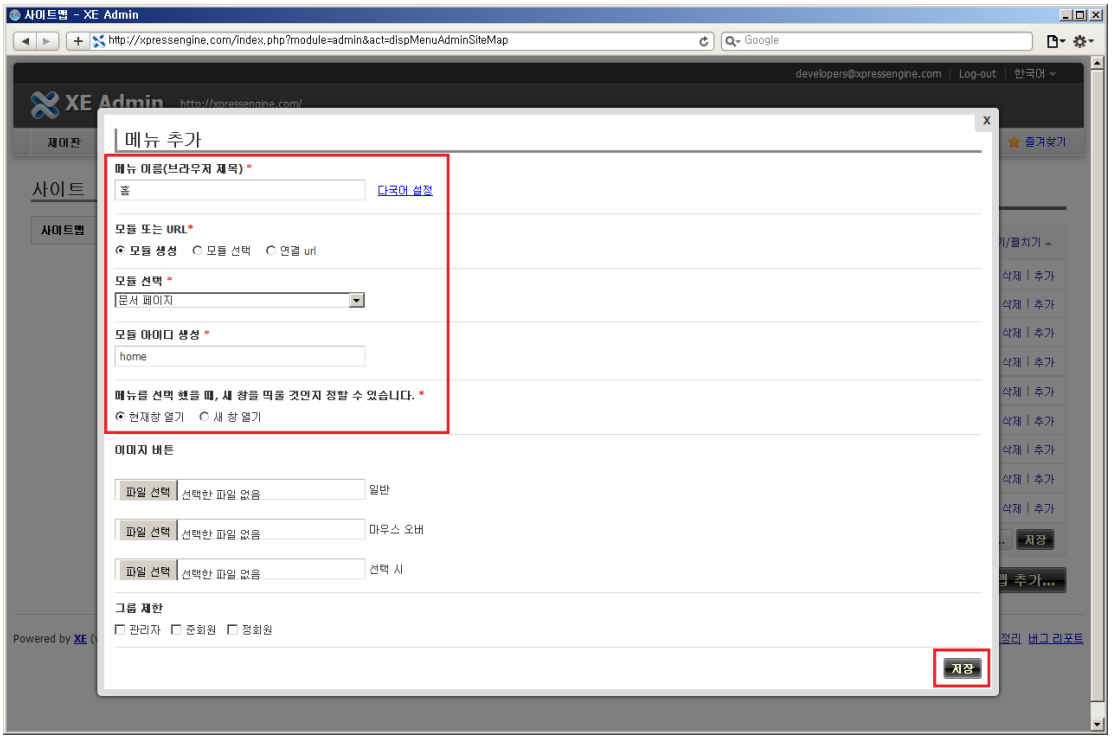


3.9 페이지 모듈에 레이아웃 연결

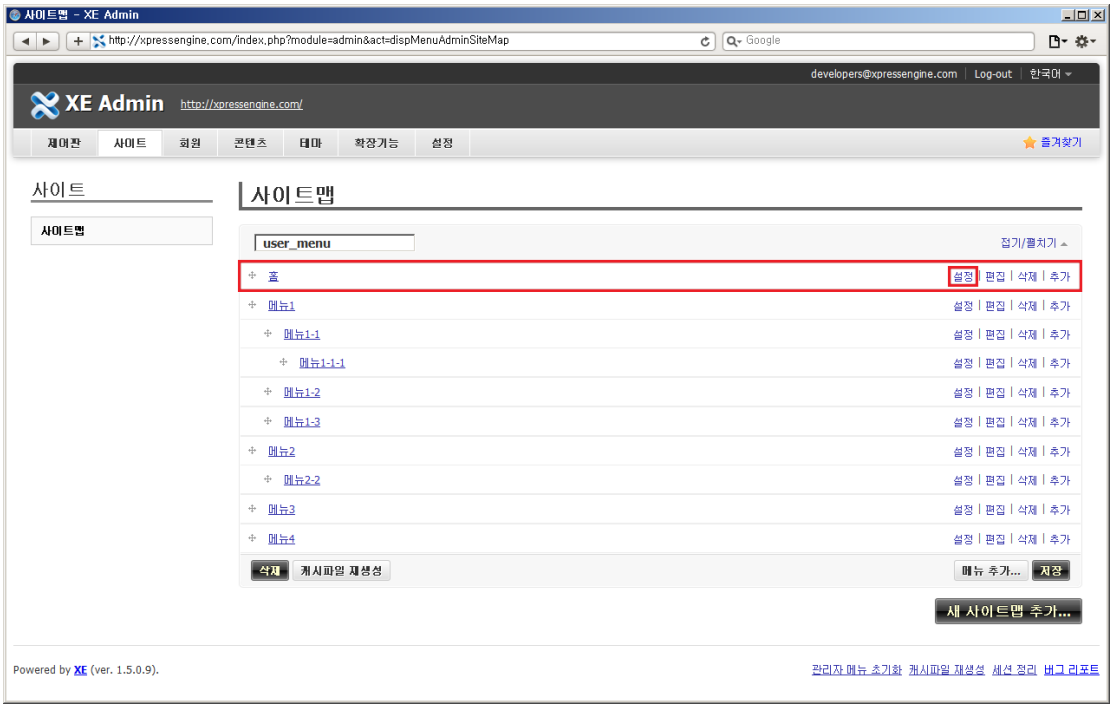
이제 페이지 모듈에 레이아웃 스킨을 적용해서 예제 레이아웃 스킨의 내용이 화면에 바르게 표시되는지 확인해 보겠습니다.

페이지 생성

1. XE 관리자 페이지에서 **사이트 > 사이트맵**을 선택합니다.
2. **user_menu** 아래의 **메뉴 추가를** 클릭합니다.
3. 다음과 같이 **메뉴 추가** 화면이 나타나면, **메뉴 이름(브라우저 제목)**, **모듈 또는 URL**, **모듈 선택**, **모듈 아이디 생성, 현재창 열기**를 입력합니다. 이 예제에서는 다음과 같이 설정하고 **저장**을 클릭합니다.
   - **메뉴 이름**: *홈*
   - **모듈 선택**: **문서 페이지**
   - **모듈 아이디 생성**: *home*



4. **사이트맵 > user_menu**에 **홈**이 생성된 것을 확인한 다음 **홈**의 **설정**을 클릭해서 **페이지 관리** 화면으로 이동합니다.



5. **페이지 관리**의 **레이아웃** 항목에서 **사용자 정의 레이아웃_사본1(user_layout)**을 선택하고 **저장**을 클릭합니다.

레이아웃에 **사용자 정의 레이아웃_사본1(user_layout)** 항목이 보이지 않는다면 'user_layout'의 사본이 아직 생성되지 않은 것입니다. 'user_layout'의 사본을 생성하는 방법은 "레이아웃 사본 생성"을 참조하십시오.

페이지 확인

생성된 페이지를 사용자 화면에서 접근하면 레이아웃 스킨이 적용된 결과를 확인할 수 있습니다. 모듈 이름을 'home'이라고 작성했기 때문에 이 페이지의 접근 경로는 다음과 같습니다. 아래 경로에서 'example.com'은 사용자의 웹 사이트가 설치된 도메인 주소를 의미합니다.

- mode_rewrite를 사용할 경우: http://example.com/home/
- mode_rewrite를 사용하지 않는 경우: http://example.com/?mid=home

home 페이지를 열면 다음과 같은 화면이 나타납니다.

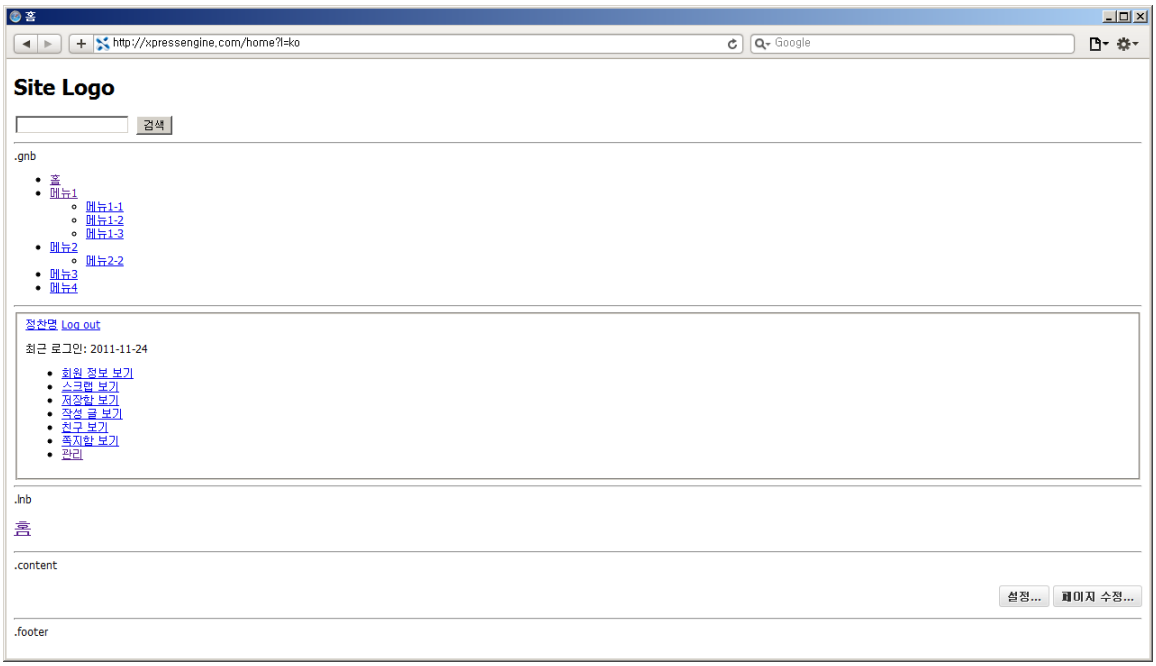


그림 3-3 레이아웃 스킨이 적용된 페이지

페이지 본문에는 아직 아무런 내용도 작성하지 않았기 때문에 표시할 내용이 없고, 본문 이외의 영역에 작성한 레이아웃 콘텐츠만 보입니다. 'Site Logo, gnb, lnb, content, footer'라는 텍스트는 스킨 파일인 'layout.html' 문서에서 작성한 것입니다. 그 밖에 레이아웃 스킨에 포함했던 통합검색 양식, 글로벌 메뉴, 로그인 양식 위젯, 로컬 메뉴가 잘 출력되는 것을 확인할 수 있습니다. content 영역에는 페이지 본문을 편집할 수 있는 버튼이 있는데 이것은 관리자에게만 노출되는 기능입니다. 이 버튼이 보이지 않는다면 관리자 로그아웃 상태이거나 본문 영역에 {$content}라는 변수를 잘못 작성한 것입니다.

> **참고**
> 메뉴를 생성하는 과정에서 **모듈 또는 URL** 항목에 유효한 **모듈 아이디** 또는 **연결 URL**을 지정하지 않으면 로컬 메뉴가 정상적으로 출력되지 않습니다.

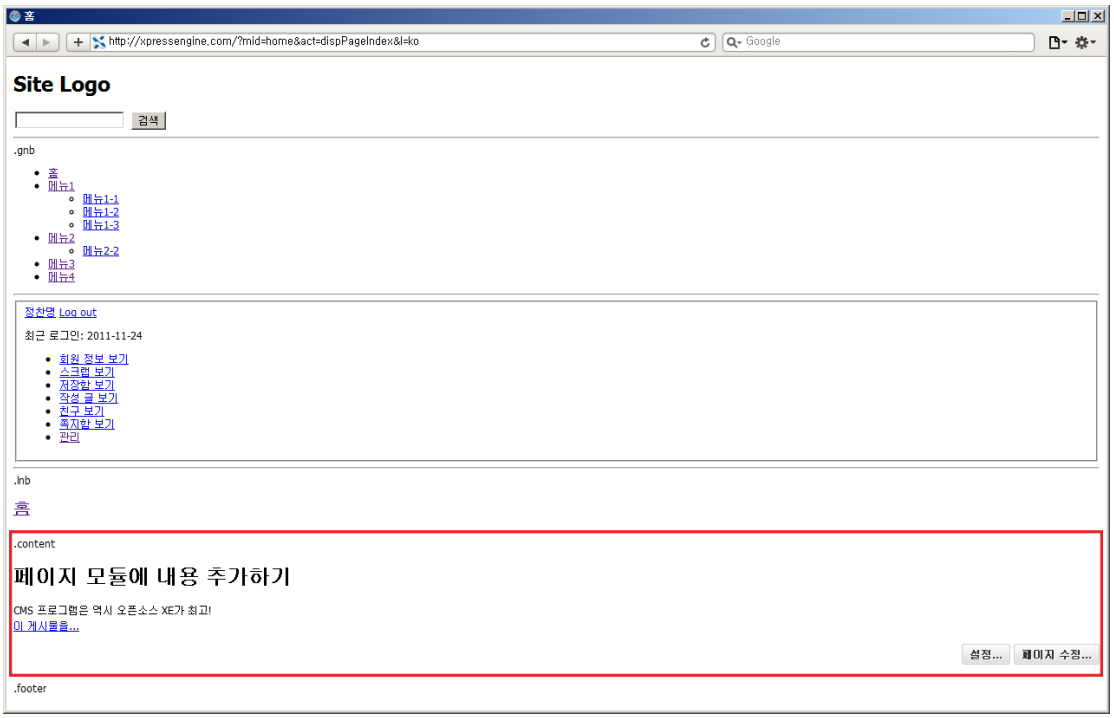
페이지 수정

그림 3-3과 같은 화면에서 페이지에 새로운 내용을 입력할 수 있습니다. 페이지를 수정하는 방법은 다음과 같습니다.

1. 레이아웃 스킨이 적용된 페이지에서 화면 오른쪽 아래에 있는 **페이지 수정**을 클릭합니다.
2. 간단한 제목과 본문 내용을 추가하고 **등록**을 클릭합니다.

추가한 내용이 페이지에 반영됩니다.

3. 사용자 화면에 표시되는 결과 화면은 다음과 같습니다. content 영역 오른쪽 아래에 표시되는 **설정, 페이지 수정**은 현재 페이지 모듈의 편집 권한이 있는 관리자에게만 표시되고 사용자 화면에는 표시되지 않습니다.



아직은 레이아웃이라고 부를 만한 배치가 아닙니다. CSS 코드를 작성하여 적절한 배치를 구현해야 합니다.

3.10 CSS 적용

이 절에서는 레이아웃 스킨에 CSS 코드를 적용하는 방법을 설명합니다. 이 문서에서 CSS 코드를 작성하는 방법은 설명하지 않습니다. 스킨 제작자는 자신의 의도에 맞게 CSS 코드를 수정해서 사용할 수 있습니다.

1. 예제 레이아웃 스킨의 user_layout.css 파일에서는 다음과 같이 .lnb 영역과 .content 영역을 좌우의 칼럼 형식으로 배치하도록 작성했습니다.

```
@charset "utf-8";
/* Layout */
hr{ display:none;}
form, fieldset{ border:0; margin:0; padding:0;}
.user layout{ width:960px; margin:0 auto;}
.header{ zoom:1; background:#ddd;}
.header:after{ content:""; display:block; clear:both;}
.header .search{ float:right;}
.gnb{ float:left;}
.body{ margin:20px 0; zoom:1; background:#eee;}
.body:after{ content:""; display:block; clear:both;}
.lnb{ float:left; width:200px; background:#ddd;}
.account{}
.content{ float:right; width:740px; background:#ddd;}
.footer{ background:#ddd;}
```

2. layout.html에 다음과 같이 user_layout.css를 참조하도록 선언합니다.

```
<load target="user_layout.css" />
```

   자세한 설명은 "CSS 파일 참조"를 참조하십시오.

3. 다음 경로로 접근하여 페이지에 CSS가 제대로 적용되었는지 확인합니다. 아래 경로에서 'example.com'은 사용자의 웹 사이트가 설치된 도메인 주소를 의미합니다.
   - mod_rewrite를 사용할 경우: http://example.com/home/
   - mod_rewrite를 사용하지 않는 경우: http://example.com/?mid=home

다음 그림은 user_layout.css를 정상적으로 참조하여 화면 구성 요소가 올바르게 표시된 모습입니다.

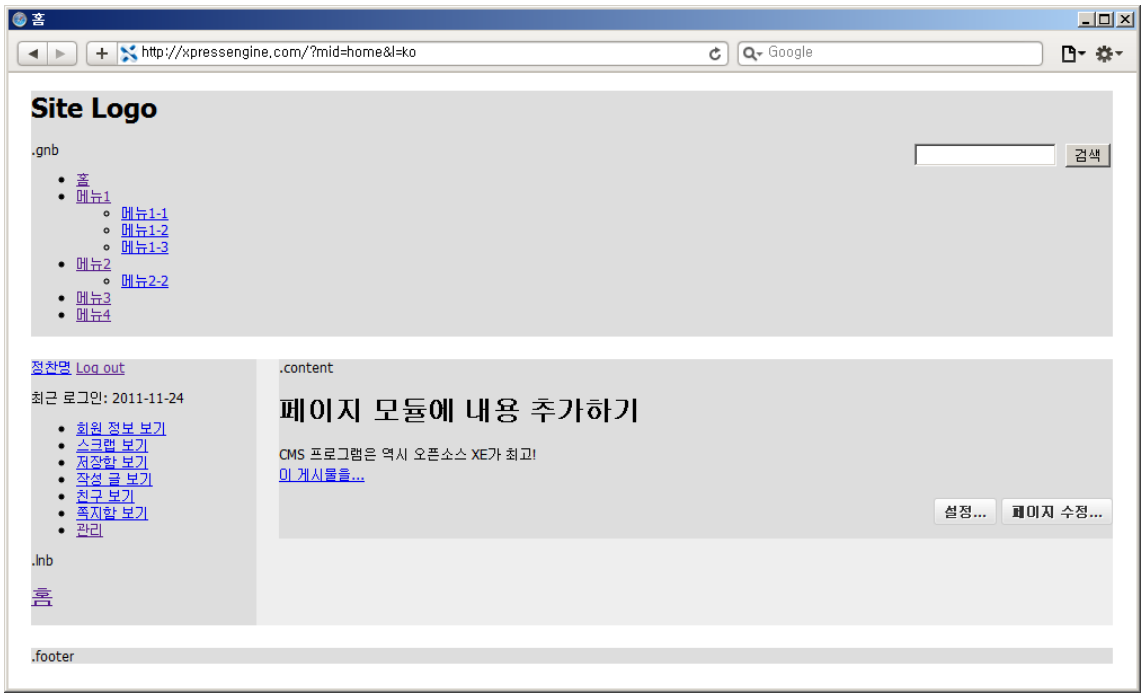


그림 3-4 CSS가 올바르게 적용된 페이지

만약 작성된 코드가 화면에 반영되지 않았다면 CSS를 참조하는 <load /> 템플릿 문법이 올바른지, 또는 CSS 참조 경로(target)가 바르게 지정되었는지 확인하십시오.

3.11 자바스크립트 적용

이 절에서는 레이아웃 스킨에 자바스크립트 코드를 추가하는 방법을 설명합니다. 이 문서에서 jQuery 실행 코드를 작성하는 방법은 설명하지 않습니다. 스킨 제작자는 자신의 의도에 맞는 코드를 작성해서 추가합니다.

1. user_layout.js에 jQuery 문법의 코드를 작성합니다.

```
// <![CDATA[
jQuery(function($){
    // 이곳에 jQuery 코드를 작성합니다.
});
// ]]>
```

2. layout.html에 다음과 같이 user_layout.js를 참조하도록 선언합니다.

```
<load target="user_layout.js" type="body" />
```

   자세한 설명은 "JS 파일 참조"를 참조하십시오.

3. 다음 경로로 접근하여 자바스크립트가 제대로 실행되는지 확인합니다. 아래 경로에서 'example.com'은 사용자의 웹 사이트가 설치된 도메인 주소를 의미합니다.
   - mod_rewrite를 사용할 경우: http://example.com/home/
   - mod_rewrite를 사용하지 않는 경우: http://example.com/?mid=home

만약 작성된 코드가 화면에 반영되지 않았다면 user_layout.js를 참조하는 <load /> 템플릿 문법이 올바른지, 또는 user_layout.js 참조 경로(target)가 바르게 지정되었는지 확인하십시오.

참고

XE에서 jQuery를 사용하는 기본적인 방법은 "자바스크립트와 jQuery 사용"을 참조하십시오.
jQuery를 사용해서 다양한 효과를 구현하려면 http://jquery.com/ 웹 사이트를 참조하십시오.

4. 게시판 스킨 만들기

이 장에서는 예제 게시판 스킨을 사용해서 게시판 스킨을 만들고 적용하는 방법을 설명합니다.

4.1 게시판 스킨이란

게시판 스킨이란 게시판 모듈을 사용자 화면에 보여주는 인터페이스입니다. 게시판 스킨은 사용자 화면을 기준으로 목록, 읽기, 쓰기, 삭제, 댓글 쓰기, 댓글 삭제, 엮인글 삭제, 권한 안내, 비밀번호 입력 페이지로 구성됩니다.

4.2 게시판 모듈 설치

XE에서 게시판을 만들려면 게시판 모듈을 별도로 설치해야 합니다. 게시판 모듈은 XE 관리자 페이지의 쉬운 설치 기능을 사용하거나 게시판 모듈의 소스 파일을 서버에 업로드하여 설치할 수 있습니다.

쉬운 설치

XE 관리자 페이지에서 **확장기능 > 쉬운 설치 > 모듈**을 선택하여 게시판 모듈을 서버에 바로 설치합니다.

쉬운 설치로 게시판 모듈을 설치한 경우 게시판 스킨을 만들려면 FTP를 통해 게시판 모듈이 설치된 디렉터리(/modules/board/)를 로컬 PC로 다운로드해야 합니다.

게시판 모듈의 디렉터리 구조는 다음과 같습니다. 게시판 모듈 디렉터리에 'skins' 디렉터리가 포함되어 있는지 확인합니다. 게시판 스킨은 'skins' 디렉터리에 저장됩니다.

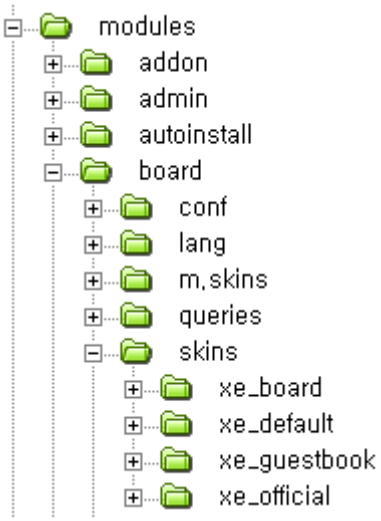


그림 4-1 게시판 모듈의 디렉터리 구조

소스 파일 업로드

XE 공식 웹 사이트에서 게시판 모듈의 소스 파일을 로컬 PC에 다운로드한 후 FTP 프로그램을 통해서 웹 서버의 '/modules/' 경로에 소스 파일을 업로드합니다. 만약 /xe/라는 사용자 정의 디렉터리에 XE core를 설치했다면 '/xe/modules/' 경로에 소스 파일을 업로드해야 합니다.

게시판 모듈이 정상적으로 업로드되었다면 게시판 모듈의 디렉터리 위치는 '/modules/board/' 또는 '/xe/modules/board/'입니다.

4.3 예제 게시판 스킨 다운로드

게시판 스킨을 만들려면 게시판 스킨에 필요한 필수 파일을 생성해야 합니다. 이 문서에서는 스킨 제작자의 편의를 위해 필수 파일을 포함한 예제 게시판 스킨을 제공합니다. 스킨 제작자는 예제 게시판 스킨을 사용하여 편리하게 원하는 형태의 스킨으로 수정할 수 있습니다.

예제 게시판 스킨의 다운로드 경로는 다음과 같습니다.

- http://doc.xpressengine.com/manual/user_board.zip

4.4 게시판 스킨의 위치와 필수 파일

4.4.1 게시판 스킨의 위치 확인

게시판 스킨 디렉터리의 위치는 다음과 같습니다.

```
/modules/board/skins/
```

다운로드한 예제 게시판 스킨(user_board)의 압축을 해제하여 '/modules/board/skins/' 아래에 복사합니다.

```
/modules/board/skins/user_board
```

> **참고**
> 게시판 스킨을 로컬 PC에서 수정할 때는 수정한 내용을 웹 사이트의 게시판 스킨 디렉터리에 반영해야 합니다.

4.4.2 게시판 스킨 필수 파일

게시판 스킨을 만들려면 다음과 같은 필수 파일이 필요합니다.

표 4-1 게시판 스킨 필수 파일

파일	설명	비고
skin.xml	게시판 스킨 정보	파일 이름 변경할 수 없음
list.html	게시물 목록	파일 이름 변경할 수 없음
write_form.html	게시물 쓰기 양식	파일 이름 변경할 수 없음
delete_form.html	게시물 삭제 양식	파일 이름 변경할 수 없음
comment_form.html	댓글 쓰기 양식	파일 이름 변경할 수 없음
delete_comment_form.html	댓글 삭제 양식	파일 이름 변경할 수 없음
delete_trackback_form.html	엮인글 삭제 양식	파일 이름 변경할 수 없음
input_password_form.html	비밀번호 입력 양식	파일 이름 변경할 수 없음
message.html	알림 메시지	파일 이름 변경할 수 없음
_header.html	게시판 헤더 출력	다른 페이지에 포함(include)됨
_footer.html	게시판 푸터 출력	다른 페이지에 포함(include)됨
_read.html	게시물 읽기 페이지 출력	다른 페이지에 포함(include)됨
_comment.html	댓글 출력	다른 페이지에 포함(include)됨
_trackback.html	엮인글 출력	다른 페이지에 포함(include)됨
user_board.css	게시판 스킨 스타일시트	

4.5 게시판 스킨 정보 작성

skin.xml은 게시판 스킨의 기본 정보를 포함하며 관리자 화면에 보여줄 설명과 옵션을 제공합니다.
skin.xml의 이름은 바꿀 수 없습니다.

skin.xml의 기본 구조는 다음과 같습니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<skin version="0.2">
   <title xml:lang="ko">user board</title>
   <title xml:lang="en">user board</title>
   <title xml:lang="jp">user board</title>
   <description xml:lang="ko">게시판 스킨 제작 실습을 위한 user_board입니다.</description>
   <description xml:lang="en">This is user_board for creating a board skin.</description>
   <description xml:lang="jp">掲示板スキン製作実習のためのuser_boardです。</description>
   <version>1.0</version>
   <date>2010-12-24</date>
   <author email_address="user@user.com" link="http://user-define.com/">
      <name xml:lang="ko">제작자 이름</name>
      <name xml:lang="en">Author Name</name>
      <name xml:lang="jp">製作者名</name>
   </author>
   <license>LGPL v2</license>
   <extra vars>
      <var name="title" type="text">
         <title xml:lang="ko">게시판 제목</title>
         <title xml:lang="en">Board Title</title>
         <title xml:lang="jp">掲示板タイトル</title>
                <description xml:lang="ko">작성하면 화면에 표시됨</description>
                <description xml:lang="en">This will be displayed on the screen as you
write.</description>
                <description xml:lang="jp">作成すると画面に表示される</description>
      </var>
      <var name="comment" type="textarea">
         <title xml:lang="ko">게시판 설명</title>
         <title xml:lang="en">Board Details</title>
         <title xml:lang="jp">掲示板詳細説明</title>
                <description xml:lang="ko">작성하면 화면에 표시됨</description>
                <description xml:lang="en">This will be displayed on the screen as you
write.</description>
                <description xml:lang="jp">作成すると画面に表示される</description>
      </var>
   </extra_vars>
</skin>
```

이 코드의 내용은 다음과 같습니다.

코드	설명
`<?xml version="1.0" encoding="UTF-8"?>`	XML 문서 형식 선언
`<skin version="0.2">`	스킨 정보 문서임을 선언. version 에는 XE core 에서 지원하는 버전을 표시해야 합니다. XE core 1.4.4.2 를 기준으로 지원하는 버전은 0.2 입니다.
`<title xml:lang="ko">...</title>`	게시판 스킨의 이름

코드	설명
`<description xml:lang="ko">...</description>`	게시판 스킨의 설명
`<version>...</version>`	게시판 스킨의 버전
`<date>YYYY-MM-DD</date>`	게시판 스킨 제작 날짜. 년-월-일(YYYY-MM-DD) 형식으로 작성해야 합니다.
`<author email_address="..." link="..."><name xml:lang="ko">...</name></author>`	게시판 스킨 제작자 정보. 이메일 주소, 웹 사이트 주소, 제작자 이름을 작성합니다.
`<license>LGPL v2</license>`	게시판 스킨 라이선스 정보. XE core 와 동일한 LGPL v2 라이선스를 권장합니다.
`<extra_vars>...</extra_vars>`	게시판 모듈에서 지원하는 확장 변수를 사용하려면 태그 안쪽에 내용을 추가할 수 있습니다.
`<var name="title" type="text"><title xml:lang="ko">게시판 제목</title><description xml:lang="ko">작성하면 화면에 표시됨</description></var>`	게시판의 제목을 입력받아서 화면에 표시합니다.
`<var name="title" type="textarea"><title xml:lang="ko">게시판 설명</title><description xml:lang="ko">작성하면 화면에 표시됨</description></var>`	게시판의 설명을 입력받아서 화면에 표시합니다.

'user_board' 게시판 스킨의 skin.xml 문서가 제대로 작성되었는지 확인하려면 게시판을 하나 생성한 다음 'user_board' 스킨을 사용하도록 설정합니다.

이 밖에도 skin.xml에는 사용자 정의 가능한 여러 타입의 변수를 <extra_vars> 요소 안에 추가로 포함할 수 있습니다. 다음은 스킨 제작자가 select, image 타입의 데이터를 입력받아서 변수로 사용할 수 있도록 확장한 예제입니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<layout version="0.2">
    ...
    <extra vars>
        <var name="colorset" type="select">
            <title xml:lang="ko">컬러셋</title>
            <description xml:lang="ko">원하는 컬러셋을 선택해주세요.</description>
            <options value="black">
                <title xml:lang="ko">Black (기본)</title>
            </options>
            <options value="white">
                <title xml:lang="ko">White</title>
            </options>
        </var>
        <var name="board_image" type="image">
            <title xml:lang="ko">게시판 헤더 이미지</title>
            <description xml:lang="ko">게시판 상단에 출력할 이미지를 입력하세요.</description>
```

```
        </var>
    </extra vars>
    ...
</layout>
```

위 코드에서 사용된 확장 변수는 다음과 같습니다.

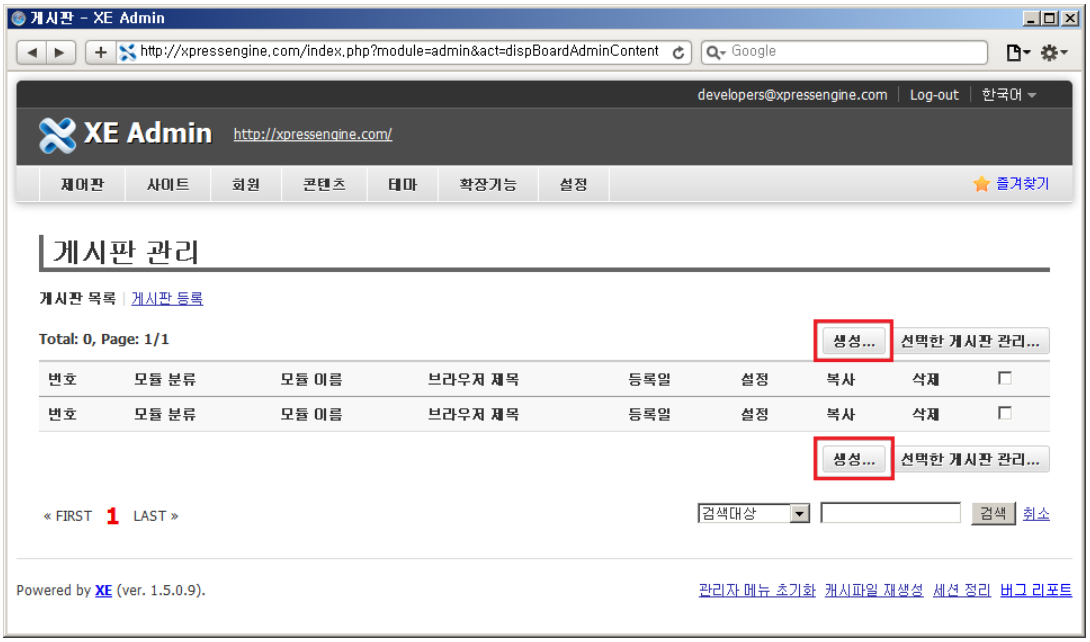
변수	설명
`<var name="colorset" type="select">...</var>`	select 타입의 확장 변수. {$module_info->colorset} 형식으로 출력 가능
`<var name="board_image" type="image">...</var>`	image 타입의 확장 변수. {$modul_info->image} 형식으로 출력 가능

skin.xml에서 추가된 확장 변수는 XE 관리자 페이지의 **확장기능 > 설치된 모듈 > 게시판 > 설정 > 스킨 관리** 페이지에 표시됩니다. 게시판 관리자가 해당 변수에 값을 입력하면 스킨에서 출력할 수 있습니다.

4.6 게시판 생성 및 스킨 적용

새 게시판을 생성하고 스킨을 적용하는 방법은 다음과 같습니다.

1. XE 관리자 페이지에서 **확장기능 > 설치된 모듈 > 게시판**을 선택합니다.
2. **게시판 관리 > 게시판 목록** 페이지가 열리면 **생성**을 클릭합니다.



3. 다음과 같이 각 항목을 입력하고 **등록**을 클릭합니다. 실제로는 더 많은 옵션 항목이 있지만 게시판을 생성할 때 반드시 확인해야 하는 것은 아니기 때문에 이 예제 화면에서는 생략했습니다.
   - **모듈 이름**: *test_board*
   - **브라우저 제목:** *게시판 스킨 실습*
   - **스킨: user_board**
4. 생성된 test_board 게시판의 **스킨 관리**를 열고 사용자 화면에 표시할 **게시판 제목, 게시판 상세 설명**을 입력하고 **등록**을 클릭합니다.
   - **게시판 제목**: *XE 실습*
   - **게시판 상세 설명**: *XE 게시판 스킨 제작 실습을 위한 게시판입니다.*


게시판 - XE Admin
http://xpressengine.com/index.php?module=admin&act=dispBoardAdminSkinInfo&
Google
developers@xpressengine.com | Log-out | 한국어
XE Admin
http://xpressengine.com/
제어판 사이트 회원 콘텐츠 테마 확장기능 설정
즐겨찾기
게시판 관리
test_board | 보기
게시판 목록 | 게시판 정보 | 분류 관리 | 확장변수 | 목록설정 | 권한 관리 | 추가 설정 | 스킨 관리
스킨 기본정보
스킨 user_board
스킨 제작자 User Name (http://user-define.com/, user@user.com)
날짜 2010-12-24
라이선스 LGPL v2
설명 게시판 스킨 제작 실습을 위한 user_board 입니다.
확장변수
게시판 제목 XE 실습 언어 코드 찾기
작성하면 화면에 표시 됨
게시판 상세 설명 XE 게시판 스킨 제작 실습을 위한 게시판 입니다. 언어 코드 찾기
작성하면 화면에 표시 됨
등록
Powered by XE (ver. 1.5.0.9).
관리자 메뉴 초기화 캐시파일 재생성 세션 정리 버그 리포트

4.7 게시판 헤더/푸터 작성

4.7.1 헤더 작성

게시판 헤더는 사용자가 게시판의 어떤 페이지에 접근해도 항상 같은 내용을 화면 윗부분에 출력합니다. 예제 게시판 스킨에서는 XE 관리자 페이지에서 입력한 **게시판 제목, 게시판 상세 설명**을 항상 표시하고 CSS 파일도 참조하도록 다음과 같이 _header.html을 작성했습니다. 게시판의 모든 페이지에서 이 _header.html을 포함(include)할 예정입니다.

```
<div class="user_board">
   <div class="board_header" cond="$module_info->title || $module_info->comment">
      <h2 cond="$module info->title">
         <a href="{getUrl('','mid',$mid)}">{$module info->title}</a>
      </h2>
      <p cond="$module_info->comment">{$module_info->comment}</p>
   </div>
```

 요소를 닫지 않았다는 점에 유의하시기 바랍니다. 이 요소의 종료 태그는 _footer.html에서 작성합니다. 요소는 게시판의 모든 요소를 한꺼번에 묶어서 CSS를 편리하게 작성 및 관리하게 도와줍니다.

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

템플릿 문법/변수	설명
`cond="$module_info->title || $module_info->comment"`	**게시판 제목** 또는 **게시판 상세 설명** 가운데 하나라도 있으면 포함된 내용 출력
`cond="$module_info->title"`	**게시판 제목**이 있으면 출력
`{$module_info->title}`	**게시판 제목** 출력
`cond="$module_info->comment"`	**게시판 상세 설명**이 있으면 출력
`{$module_info->comment}`	**게시판 상세 설명** 출력
`{getUrl('','mid',$mid)}`	게시판 URL

XE core 1.4.4 버전에 추가된 새 템플릿 문법을 사용했습니다. 새 템플릿 문법에 대한 자세한 설명은 "XE 템플릿 문법"을 참조하십시오.

4.7.2 푸터 작성

게시판 푸터는 사용자가 게시판의 어떤 페이지에 접근해도 항상 같은 내용을 화면 아랫부분에 출력합니다. 예제 게시판 스킨에서는 별도의 내용을 포함하지 않고 _header.html에서 종료하지 않았던 의 종료 태그만 작성했습니다.

```
</div>
```

_footer.html은 _header.html과 함께 모든 페이지에서 포함(include)합니다.

4.8 목록 페이지 작성

게시판에 접근했을 때 처음 보이는 페이지는 게시물 목록을 보여주는 목록 페이지입니다. 목록 페이지는 list.html에서 작성합니다.

다음은 예제 게시판 스킨으로 만든 목록 페이지의 완성 화면입니다. 게시판 헤더에서 작성한 대로 'XE 실습'이라는 게시판 제목과 "XE 게시판 스킨 제작 실습을 위한 임시 게시판입니다."라는 게시판 상세 설명이 게시판 윗부분에 항상 표시됩니다.

그림 4-2 게시판 목록 페이지 완성 화면

위의 게시판 목록 페이지는 가능한 모든 칼럼 항목을 표시하도록 설정한 상태입니다. 게시판 관리자가 목록 페이지에서 어떤 항목을 표시하려고 할지 알 수 없기 때문에 스킨을 만들 때 모든 항목을 출력하도록 설정하는 것이 좋습니다.

list.html을 작성하는 방법은 다음과 같습니다.

_header.html, _footer.html 포함(include)

예제 게시판 스킨의 list.html에서는 다음과 같이 _header.html과 _footer.html을 포함(include)하도록 작성했습니다.

```
<include target="_header.html" />
    이곳에 게시물 목록을 출력할 예정입니다.
```

```
<include target="_footer.html" />
```

위 코드에서 사용된 템플릿 문법은 다음과 같습니다.

XE 템플릿 문법	설명
`<include target="_header.html" />`	_header.html 포함(include)
`<include target="_footer.html" />`	_footer.html 포함(include)

XE core 1.4.4 버전에 추가된 새 템플릿 문법을 사용했습니다. 새 템플릿 문법에 대한 자세한 설명은 "XE 템플릿 문법"을 참조하십시오.

게시물이 없을 때 메시지 출력

조건문을 사용해서 게시물이 없을 때 메시지를 출력할 수 있습니다. 예제 게시판 스킨의 list.html에서는 게시물이 없을 때 "등록된 글이 없습니다"라는 메시지를 출력하도록 다음과 같이 조건문을 작성했습니다.

```
<include target="_header.html" />
<p cond="!$document list && !$notice list" class="no document">{$lang->no documents}</p>
<include target="_footer.html" />
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명
`cond="!$document_list && !$notice_list"`	게시물과 공지사항 목록이 모두 없으면 출력
`{$lang->no_documents}`	"등록된 글이 없습니다"라는 언어 변수

XE core 1.4.4 버전에 추가된 새 템플릿 문법을 사용했습니다. 새 템플릿 문법에 대한 자세한 설명은 "XE 템플릿 문법"을 참조하십시오.

게시물 목록을 표로 작성

일반적으로 게시물 목록은 표로 작성합니다. 다양한 조건문과 변수를 포함하기 전에 HTML을 사용하여 다음과 같이 게시판의 골격을 완성합니다. 표는 크게 목록 헤더(LIST HEADER), 공지사항(NOTICE), 목록(LIST) 행으로 구분하고, 코드의 가독성을 높이기 위하여 주석으로 표시해 둡니다.

```
<include target=" header.html" />
<p cond="!$document list && !$notice list">{$lang->no documents}</p>
<table width="100%" border="1" cellspacing="0" summary="List of Articles" id="board list"
class="board_list" cond="$document_list || $notice_list">

   <thead>
      <!-- LIST HEADER -->
      <tr>
         <th scope="col">번호</th>
         <th scope="col">제목</th>
         <th scope="col">글쓴이</th>
         <th scope="col">아이디</th>
         <th scope="col">이름</th>
         <th scope="col">날짜</th>
```

```
            <th scope="col">최근 수정일</th>
            <th scope="col">최종 글</th>
            <th scope="col">조회 수</th>
            <th scope="col">추천 수</th>
        </tr>
        <!-- /LIST HEADER -->
    </thead>
    <tbody>
        <!-- NOTICE -->
        <tr>
            <td> </td>
            <td> </td>
            <td> </td>
            <td> </td>
            <td> </td>
            <td> </td>
            <td> </td>
            <td> </td>
            <td> </td>
            <td> </td>
        </tr>
        <!-- /NOTICE -->
        <!-- LIST -->
        <tr>
            <td> </td>
            <td> </td>
            <td> </td>
            <td> </td>
            <td> </td>
            <td> </td>
            <td> </td>
            <td> </td>
            <td> </td>
            <td> </td>
        </tr>
        <!-- /LIST -->
    </tbody>
</table>
<include target="_footer.html" />
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명
`cond="$document_list || $notice_list"`	게시물이나 공지사항이 있으면 표와 함께 포함된 내용을 출력

XE core 1.4.4 버전에 추가된 새 템플릿 문법을 사용했습니다. 새 템플릿 문법에 대한 자세한 설명은 "XE 템플릿 문법"을 참조하십시오.

게시물 목록 헤더 출력

게시물 목록 헤더란 번호, 제목, 글쓴이, 날짜, 조회 수와 같은 내용의 제목 셀입니다. 게시판 관리자가 어떤 항목을 표시할 것인지 알 수 없기 때문에 스킨 제작자는 게시판 모듈이 지원하는 대부분의 내용을 조건문으로 작성해 두는 것이 좋습니다.

예를 들어, 게시판 관리자는 XE 관리자 페이지의 **확장기능 > 설치된 모듈 > 게시판**에서 **설정**을 클릭하고 **목록설정**을 열어 모든 항목을 표시하도록 설정할 수 있습니다. 이때 스킨에서 모든 항목을 지원하지 않으면 게시판 관리자는 스킨에 문제가 있는 것으로 오해할 수 있습니다.

그림 4-3 게시판 목록설정

XE 게시판 모듈은 기본적으로 위와 같이 12가지 정보를 목록 화면에 출력할 수 있습니다. **섬네일**과 **요약** 내용은 통상 별도의 칼럼으로 두지 않고 **제목** 셀에 함께 출력하기 때문에 게시물 목록의 칼럼 개수에 포함시키지 않습니다. 따라서 모든 항목을 표시할 수 있도록 지원하려면 10가지 항목에 대한 조건문을 작성해야 합니다. 게시판 관리자가 모든 항목을 표시하도록 설정하는 경우 총 10개의 칼럼이 생성됩니다.

예제 게시판 스킨의 list.html에서는 다음과 같이 <thead> 요소 내부에 10가지 항목에 대한 조건문을 작성했습니다.

```
...
<thead>
   <!-- LIST HEADER -->
   <tr>
      <block loop="$list config=>$key,$val">
      <th scope="col" cond="$val->type=='no'">{$lang->no}</th>
      <th scope="col" class="title" cond="$val->type=='title'">{$lang->title}</th>
      <th scope="col" cond="$val->type=='nick_name'">{$lang->writer}</th>
      <th scope="col" cond="$val->type=='user_id'">{$lang->user_id}</th>
      <th scope="col" cond="$val->type=='user name'">{$lang->user name}</th>
      <th scope="col" cond="$val->type=='regdate'"><a
href="{getUrl('sort index','regdate','order type',$order type)}">{$lang->date}</a></th>
      <th scope="col" cond="$val->type=='last_update'"><a
href="{getUrl('sort_index','last_update','order_type',$order_type)}">{$lang-
>last_update}</a></th>
      <th scope="col" cond="$val->type=='last post'"><a
href="{getUrl('sort index','last update','order type',$order type)}">{$lang-
>last_post}</a></th>
      <th scope="col" cond="$val->type=='readed_count'"><a
href="{getUrl('sort_index','readed_count','order_type',$order_type)}">{$lang-
>readed_count}</a></th>
      <th scope="col" cond="$val->type=='voted count'"><a
href="{getUrl('sort_index','voted_count','order_type',$order_type)}">{$lang-
>voted_count}</a></th>
      </block>
   </tr>
```

```
   <!-- /LIST HEADER -->
</thead>
...
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명
<block loop="$list_config=>$key,$val">...</block>	표시 항목을 출력
<th scope="col" cond="$val->type=='no'">{$lang->no}</th>	게시물 번호
<th scope="col" class="title" cond="$val->type=='title'"> {$lang->title} </th>	게시물 제목
<th scope="col" cond="$val->type=='nick_name'"> {$lang->writer} </th>	글쓴이
<th scope="col" cond="$val->type=='user_id'"> {$lang->user_id} </th>	아이디
<th scope="col" cond="$val->type=='user_name'"> {$lang->user_name} </th>	이름
<th scope="col" cond="$val->type=='regdate'"> {$lang->date} </th>	날짜(정렬 순서 변경 가능)
<th scope="col" cond="$val->type=='last_update'"> {$lang->last_update} </th>	최근 수정일(정렬 순서 변경 가능)
<th scope="col" cond="$val->type=='last_post'"> {$lang->last_post} </th>	최종 글(정렬 순서 변경 가능)
<th scope="col" cond="$val->type=='readed_count'"> {$lang->readed_count} </th>	조회 수(정렬 순서 변경 가능)
<th scope="col" cond="$val->type=='voted_count'"> {$lang->voted_count} </th>	추천 수(정렬 순서 변경 가능)

XE core 1.4.4 버전에 추가된 새 템플릿 문법을 사용했습니다. 새 템플릿 문법에 대한 자세한 설명은 "XE 템플릿 문법"을 참조하십시오.

게시물 목록 출력

목록에는 공지사항 목록과 일반 게시물 목록이 있습니다. 공지사항 목록은 일반 게시물 목록과 달리 항상 표의 윗부분에 출력됩니다.

예제 게시판 스킨의 list.html에서는 다음과 같이 <tbody> 요소에 공지사항 목록과 일반 게시물 목록을 출력하도록 작성했습니다.

```
...
<tbody>
   <!-- NOTICE -->
   <tr class="notice" loop="$notice list=>$no,$document">
      <block loop="$list_config=>$key,$val">
      <td class="notice" cond="$val->type=='no'">
         <block cond="$document srl==$document->document srl">&raquo;</block>
         <block cond="$document srl!=$document->document srl">{$lang->notice}</block>
      </td>
      <td class="title" cond="$val->type=='title'">
         <a href="{getUrl('document_srl',$document->document_srl, 'listStyle',
$listStyle, 'cpage','')}">
            {$document->getTitle()}
         </a>
         <a cond="$document->getCommentCount()" href="{getUrl('document srl', $document-
>document_srl)}#comment" class="replyNum" title="Replies">
            [{$document->getCommentCount()}]
         </a>
         <a cond="$document->getTrackbackCount()" href="{getUrl('document srl',
$document->document srl)}#trackback" class="trackbackNum" title="Trackbacks">
            [{$document->getTrackbackCount()}]
         </a>
      </td>
      <td class="author" cond="$val->type=='nick name'">{$document->getNickName()}</td>
      <td class="author" cond="$val->type=='user id'">{$document->getUserID()}</td>
      <td class="author" cond="$val->type=='user name'">{$document->getUserName()}</td>
      <td class="time" cond="$val->type=='regdate'">{$document->getRegdate('Y.m.d')}</td>
      <td class="time" cond="$val->type=='last_update'">{zdate($document-
>get('last_update'),'Y.m.d')}</td>
      <td class="lastReply" cond="$val->type=='last post'">
         <block cond="(int)($document->get('comment count'))>0">
            <a href="{$document->getPermanentUrl()}#comment" title="Last Reply">
               {zdate($document->get('last_update'),'Y.m.d')}
            </a>
            <span cond="$document->get('last_updater')">
               <sub>by</sub>
               {htmlspecialchars($document->get('last updater'))}
            </span>
         </block>
         <block cond="(int)($document->get('comment_count'))==0"> </block>
      </td>
      <td class="readNum" cond="$val->type=='readed count'">{$document-
>get('readed count')>0?$document->get('readed count'):'0'}</td>
      <td class="voteNum" cond="$val->type=='voted_count'">{$document-
>get('voted_count')!=0?$document->get('voted_count'):'0'}</td>
      </block>
   </tr>
   <!-- /NOTICE -->
   <!-- LIST -->
   <tr loop="$document_list=>$no,$document">
      <block loop="$list_config=>$key,$val">
      <td class="no" cond="$val->type=='no'">
         <block cond="$document srl==$document->document srl">&raquo;</block>
         <block cond="$document srl!=$document->document srl">{$no}</block>
      </td>
      <td class="title" cond="$val->type=='title'">
```

```
          <a href="{getUrl('document_srl',$document->document_srl, 'listStyle',
$listStyle, 'cpage','')}">
             {$document->getTitle()}
          </a>
          <a cond="$document->getCommentCount()" href="{getUrl('document_srl', $document-
>document_srl)}#comment" class="replyNum" title="Replies">
             [{$document->getCommentCount()}]
          </a>
          <a cond="$document->getTrackbackCount()" href="{getUrl('document_srl',
$document->document_srl)}#trackback" class="trackbackNum" title="Trackbacks">
             [{$document->getTrackbackCount()}]
          </a>
       </td>
       <td class="author" cond="$val->type=='nick_name'">{$document->getNickName()}</td>
       <td class="author" cond="$val->type=='user_id'">{$document->getUserID()}</td>
       <td class="author" cond="$val->type=='user_name'">{$document->getUserName()}</td>
       <td class="time" cond="$val->type=='regdate'">{$document->getRegdate('Y.m.d')}</td>
       <td class="time" cond="$val->type=='last_update'">{zdate($document-
>get('last_update'),'Y.m.d')}</td>
       <td class="lastReply" cond="$val->type=='last_post'">
          <block cond="(int)($document->get('comment_count'))>0">
             <a href="{$document->getPermanentUrl()}#comment" title="Last Reply">
                {zdate($document->get('last_update'),'Y.m.d')}
             </a>
             <span cond="$document->get('last_updater')">
                <sub>by</sub>
                {htmlspecialchars($document->get('last_updater'))}
             </span>
          </block>
          <block cond="(int)($document->get('comment_count'))==0"> </block>
       </td>
       <td class="readNum" cond="$val->type=='readed_count'">{$document-
>get('readed_count')>0?$document->get('readed_count'):'0'}</td>
       <td class="voteNum" cond="$val->type=='voted_count'">{$document-
>get('voted_count')!=0?$document->get('voted_count'):'0'}</td>
       </block>
   </tr>
   <!-- /LIST -->
</tbody>
...
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명
`<tr class="notice" loop="$notice_list=>$no,$document">`	공지 게시물 목록 행을 반복
`<tr loop="$document_list=>$no,$document">`	일반 게시물 목록 행을 반복
`<block loop="$list_config=>$key,$val">`	게시물 표시 항목 열을 반복
`<block cond="$document_srl==$document->document_srl">»</block>`	게시물의 고유 번호와 현재 페이지 게시물의 고유 번호가 일치하면 오른쪽 이중 꺾은 괄호(>>) 출력
`<block cond="$document_srl!=$document->document_srl">{$lang->notice}</block>`	게시물의 고유 번호와 현재 페이지 게시물의 고유 번호가 일치하지 않으면 '공지' 출력
`<block cond="$document_srl!=$document->document_srl">{$no}</block>`	게시물의 고유 번호와 현재 페이지 게시물의 고유 번호가 일치하지 않으면 '게시물 번호' 출력
`<td class="title" cond="$val->type=='title'">`	게시물 제목 셀

XE 템플릿 문법/변수	설명
`<a href="{getUrl('document_srl',$document->document_srl, 'listStyle', $listStyle, 'cpage','')}">...</a>`	게시물 링크
`{$document->getTitle()}`	게시물 제목 출력
`<a cond="$document->getCommentCount()" href="{getUrl('document_srl', $document->document_srl)}#comment" class="replyNum" title="Replies">` `[...]` `</a>`	게시물 댓글 링크
`{$document->getCommentCount()}`	게시물 댓글 수
`<a cond="$document->getTrackbackCount()" href="{getUrl('document_srl', $document->document_srl)}#trackback" class="trackbackNum" title="Trackbacks">` `[...]` `</a>`	게시물 엮인글 링크
`{$document->getTrackbackCount()}`	게시물 엮인글 수
`<td class="author" cond="$val->type=='nick_name'">{$document->getNickName()}</td>`	닉네임
`<td class="author" cond="$val->type=='user_id'">{$document->getUserID()}</td>`	아이디
`<td class="author" cond="$val->type=='user_name'">{$document->getUserName()}</td>`	이름
`<td class="time" cond="$val->type=='regdate'">{$document->getRegdate('Y.m.d')}</td>`	날짜
`<td class="time" cond="$val->type=='last_update'">{zdate($document->get('last_update'),'Y.m.d')}</td>`	최종 수정일
`<td class="lastReply" cond="$val->type=='last_post'">...</td>`	최종 글
`<block cond="(int)($document->get('comment_count'))>0">...</block>`	댓글이 하나 이상이면 출력
`<block cond="(int)($document->get('comment_count'))==0">...</block>`	댓글이 없으면 출력
`<a href="{$document->getPermanentUrl()}#comment" title="Last Reply">` `{zdate($document->get('last_update'),'Y.m.d')}` `</a>`	최종 댓글 날짜 + 링크
`<span cond="$document->get('last_updater')">` `<sub>by</sub>` `{htmlspecialchars($document->get('last_updater'))}` `</span>`	최종 댓글 작성자
`<td class="readNum" cond="$val->type=='readed_count'">` `{$document->get('readed_count')>0?$document->get('readed_count'):'0'}` `</td>`	조회 수

XE 템플릿 문법/변수	설명
<td class="voteNum" cond="$val->type=='voted_count'"> {$document->get('voted_count')!=0?$document->get('voted_count'):'0'} </td>	추천 수

XE core 1.4.4 버전에 추가된 새 템플릿 문법을 사용했습니다. 새 템플릿 문법에 대한 자세한 설명은 "XE 템플릿 문법"을 참조하십시오.

> **참고**
>
> 일반 게시물 목록의 출력 개수는 XE 관리자 페이지의 **확장기능 > 설치된 모듈 > 게시판**에서 수정할 모듈 오른쪽의 **설정**을 클릭하고 **게시판 정보** 탭에서 **목록 수**를 설정하여 변경할 수 있습니다. 기본값은 20입니다.

페이지 번호 링크 출력

게시물의 수가 한 페이지를 넘어가면 과거에 작성된 게시물을 탐색할 수 있도록 페이지 이동 링크를 제공해야 합니다.

예제 게시판 스킨의 list.html에서는 게시물 목록 아래쪽에 다음과 같이 페이지 번호 링크를 출력하도록 작성했습니다. 페이지 번호를 ... 요소로 한 번 더 감싼 이유는 이 요소 안쪽에 **쓰기** 버튼과 **검색** 입력 양식을 추가로 포함할 예정이기 때문입니다.

```
<table summary="List of Articles" id="board_list" class="board_list">
   <!-- LIST HEADER -->
   <!-- /LIST HEADER -->
   <!-- NOTICE -->
   <!-- /NOTICE -->
   <!-- LIST -->
   <!-- /LIST -->
</table>
<div class="list_footer">
   <!-- PAGINATION -->
   <div class="pagination" cond="$document_list || $notice_list">
      <a
href="{getUrl('page','','document_srl','','division',$division,'last_division',$last_division)}" class="prevEnd">{$lang->first_page}</a>
      <block loop="$page_no=$page_navigation->getNextPage()">
            <strong cond="$page==$page_no">{$page_no}</strong>
            <a cond="$page!=$page_no"
href="{getUrl('page',$page_no,'document_srl','','division',$division,'last_division',$last_division)}">{$page_no}</a>
      </block>
      <a href="{getUrl('page',$page_navigation->last_page,'document_srl','','division',$division,'last_division',$last_division)}"
class="nextEnd">{$lang->last_page}</a>
   </div>
   <!-- /PAGINATION -->
</div>
<include target="_footer.html" />
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명
...	게시물이 있거나 공지가 하나라도 있을 때 페이지 번호를 출력

XE 템플릿 문법/변수	설명
`<a href="{getUrl('page','','document_srl','','division',$division,'last_division',$last_division)}" class="prevEnd">{$lang->first_page}</a>`	첫 페이지 링크
`<a href="{getUrl('page',$page_navigation->last_page,'document_srl','','division',$division,'last_division',$last_division)}" class="nextEnd">{$lang->last_page}</a>`	끝 페이지 링크
`<block loop="$page_no=$page_navigation->getNextPage()">...</block>`	페이지 번호를 반복
`<strong cond="$page==$page_no">{$page_no}</strong>`	페이지 번호가 현재 페이지와 일치하면 출력
`<a cond="$page!=$page_no" href="{getUrl('page',$page_no,'document_srl','','division',$division,'last_division',$last_division)}">{$page_no}</a>`	페이지 번호가 현재 페이지와 일치하지 않으면 출력

XE core 1.4.4 버전에 추가된 새 템플릿 문법을 사용했습니다. 새 템플릿 문법에 대한 자세한 설명은 "XE 템플릿 문법"을 참조하십시오.

쓰기 버튼 출력

list.html은 목록 페이지뿐만 아니라 읽기 페이지에서도 출력됩니다. **쓰기** 버튼을 클릭하면 write_form.html 페이지로 이동합니다.

예제 게시판 스킨의 list.html에서는 다음과 같이 쓰기 페이지 링크 버튼을 작성했습니다.

```
<div class="list footer">
   <!-- PAGINATION -->
   <div class="pagination">
      ...
   </div>
   <!-- /PAGINATION -->
   <div class="btnArea">
      <a href="{getUrl('act','dispBoardWrite','document_srl','')}" class="btn">{$lang->cmd_write}</a>
   </div>
</div>
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명
`<a href="{getUrl('act','dispBoardWrite','document_srl','')}" class="btn">{$lang->cmd_write}</a>`	쓰기 페이지 링크 버튼. 목록 페이지와 읽기 페이지에서 출력됩니다.

XE core 1.4.4 버전에 추가된 새 템플릿 문법을 사용했습니다. 새 템플릿 문법에 대한 자세한 설명은 "XE 템플릿 문법"을 참조하십시오.

게시판 검색 입력 양식 출력

게시판 내용을 검색할 수 있는 검색 입력 양식을 출력합니다. 예제 게시판 스킨의 list.html에서는 다음과 같이 검색 입력 양식을 출력하도록 작성했습니다.

```
<div class="list_footer">
   <!-- PAGINATION -->
```

```
    <div class="pagination">
        ...
    </div>
    <!-- /PAGINATION -->
    <a ...>{$lang->cmd_write}</a>
    <!-- SEARCH -->
    <form cond="$grant->view" action="{getUrl()}" method="get" onsubmit="return
procFilter(this, search)" class="board_search">
        <input type="hidden" name="vid" value="{$vid}" />
        <input type="hidden" name="mid" value="{$mid}" />
        <input type="hidden" name="category" value="{$category}" />
        <input type="text" name="search keyword"
value="{htmlspecialchars($search keyword)}" accesskey="S" title="{$lang->cmd search}"
class="iText" />
        <select name="search_target">
            <option loop="$search option=>$key,$val" value="{$key}"
selected="selected"|cond="$search target==$key">{$val}</option>
        </select>
        <input type="submit" onclick="xGetElementById('fo search').submit();return false;"
value="{$lang->cmd_search}" class="btn" />
    </form>
    <!-- /SEARCH -->
</div>
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명
`<form cond="$grant->view" action="{getUrl()}" method="get" onsubmit="return procFilter(this, search)" id="board_search" class="board_search">`	게시물 열람 권한이 있으면 검색 입력 양식을 출력
`<input type="hidden" name="vid" value="{$vid}" />`	가상 사이트 아이디 전송 요소(hidden type)
`<input type="hidden" name="mid" value="{$mid}" />`	모듈 아이디 전송 요소(hidden type)
`<input type="hidden" name="category" value="{$category}" />`	카테고리 정보 전송 요소(hidden type)
`<input type="text" name="search_keyword" value="{htmlspecialchars($search_keyword)}" accesskey="S" title="{$lang->cmd_search}" class="iText" />`	검색 키워드 입력 및 출력 요소. 접근키 'S'가 할당됨.
`<select name="search_target">`	검색 범위 선택 컨트롤
`<option loop="$search_option=>$key,$val" value="{$key}" selected="selected"|cond="$search_target==$key">{$val}</option>`	검색 범위 옵션 출력
`<input type="submit" onclick="xGetElementById('board_search').submit();return false;" value="{$lang->cmd_search}" class="btn" />`	검색 입력 양식 전송 버튼

XE core 1.4.4 버전에 추가된 새 템플릿 문법을 사용했습니다. 새 템플릿 문법에 대한 자세한 설명은 "XE 템플릿 문법"을 참조하십시오.

게시판 목록 화면 출력 결과 확인

다음 경로로 접근하여 게시판 목록 화면을 확인합니다. 아래 경로에서 'example.com'은 사용자의 웹 사이트가 설치된 도메인 주소를 의미합니다.

- mod_rewrite를 사용할 경우: http://example.com/test_board/

- mod_rewrite를 사용하지 않는 경우: http://example.com/?mid=test_board

작성된 게시물이 없으면 다음과 같이 "등록된 글이 없습니다."라는 메시지를 출력합니다.

그림 4-4 게시판 목록 화면 - 작성된 게시물이 없을 때

작성된 게시물이 있으면 다음과 같이 게시물 목록 화면을 출력합니다.

그림 4-5 게시판 목록 화면 - 작성된 게시물이 있을 때

아직 write_form.html 페이지를 작성하지 않았기 때문에 'test_board' 게시판에 직접 글을 작성할 수 없습니다. 그러나 'test_board' 게시판에 테스트용 게시물 데이터를 복사해 넣을 수 있습니다. 위 화면은 다른 게시판의 테스트용 게시물을 'test_board' 게시판으로 복사한 것입니다. XE 관리자 페이지에서 **콘텐츠 > 문서**를 선택하고, 문서를 선택한 후 **선택한 글 관리**를 클릭하여 다른 게시판의 게시물을 'test_board' 게시판으로 복사할 수 있습니다.

4.9 쓰기 페이지 작성

쓰기 페이지는 새 글을 쓰거나 기존에 작성한 게시물을 수정하는 화면으로 사용합니다. 쓰기 페이지는 write_form.html에서 작성합니다.

쓰기 페이지는 _header.html, _footer.html, 게시물 제목 입력 창, 게시물 내용 입력 창, 글쓴이 정보 입력 창(이름, 비밀번호, 홈페이지), 등록 버튼으로 구성됩니다. XE core에 완성된 위지윅 에디터가 포함되어 있으므로, 쓰기 페이지에서는 이 에디터 모듈을 불러와서 사용합니다.

다음은 예제 게시판 스킨으로 만든 쓰기 페이지입니다.

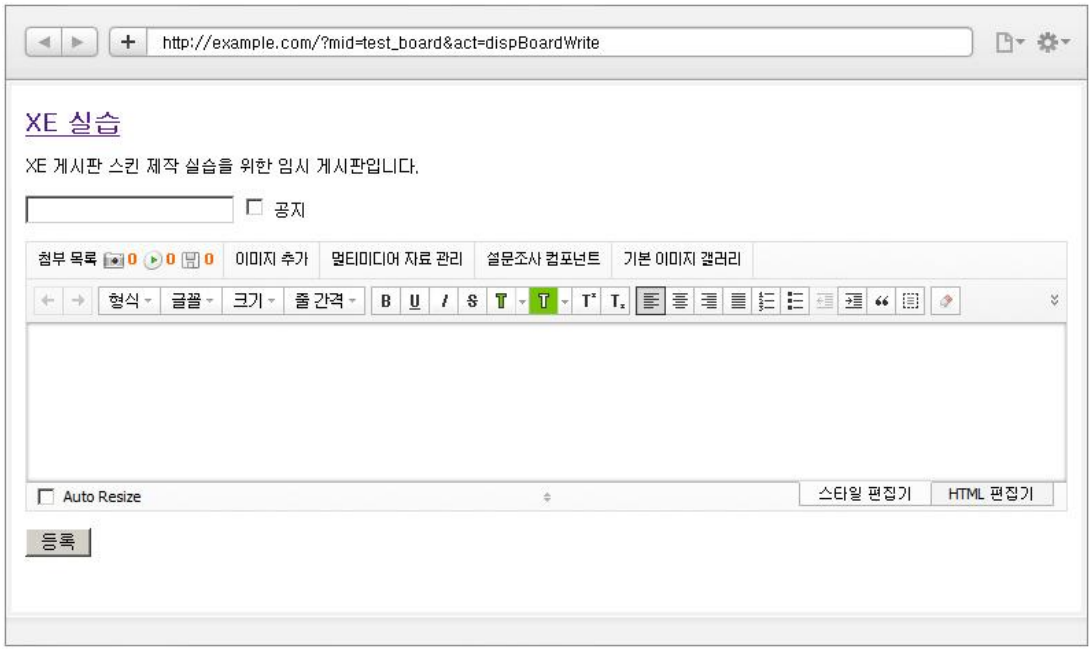


그림 4-6 쓰기 페이지 완성 화면

write_form.html을 작성하는 방법은 다음과 같습니다.

_header.html, _footer.html 포함(include)

예제 게시판 스킨의 write_form.html에서는 다음과 같이 _header.html과 _footer.html을 포함(include)하도록 작성했습니다.

```
<include target="_header.html" />
   이곳에 글쓰기 입력 양식을 작성할 예정입니다.
<include target="_footer.html" />
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명
`<include target="_header.html" />`	_header.html 포함(include)
`<include target="_footer.html" />`	_footer.html 포함(include)

XE core 1.4.4 버전에 추가된 새 템플릿 문법을 사용했습니다. 새 템플릿 문법에 대한 자세한 설명은 "XE 템플릿 문법"을 참조하십시오.

쓰기 화면의 HTML 구조

쓰기 화면의 HTML 구조는 다음과 같습니다.

```
<include target="_header.html" />
<form action="" class="board_write">
    <div class="write_header">제목 입력 창</div>
    <div class="write_editor">내용 입력 편집 창</div>
    <div class="write_author">글쓴이 정보 입력 창 (이름, 비밀번호, 홈페이지)</div>
    <div class="write_footer">등록 버튼</div>
</form>
<include target="_footer.html" />
```

작성한 내용을 서버에 전송할 수 있도록 form 요소로 작성해야 하고 전송할 내용은 '제목, 내용, 글쓴이 정보'입니다. **등록**을 클릭하면 form이 전송됩니다.

쓰기 양식 작성

onsubmit 속성을 통해 쓰기 페이지에 입력된 내용이 올바르게 작성됐는지 클라이언트 환경에서 검사할 수 있습니다. 쓰기 페이지에서 게시물을 수정할 때는 사용자가 작성한 기존 데이터를 불러옵니다.

예제 게시판 스킨의 write_form.html에서는 다음과 같이 쓰기 양식을 작성했습니다.

```
<include target=" header.html" />
<form action="./" method="post" onsubmit="return procFilter(this, window.insert)"
class="board_write">
    <input type="hidden" name="mid" value="{$mid}" />
    <input type="hidden" name="content" value="{$oDocument->getContentText()}" />
    <input type="hidden" name="document srl" value="{$document srl}" />
    <input type="hidden" name="allow comment" value="Y" />
    <input type="hidden" name="allow_trackback" value="Y" />
    ...
</form>
<include target="_footer.html" />
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명
`onsubmit="return procFilter(this, window.insert)"`	사용자 입력 데이터 유효성 검사 및 form 전송
`<input type="hidden" name="mid" value="{$mid}" />`	모듈 아이디 전송 요소(hidden type)
`<input type="hidden" name="content" value="{$oDocument->getContentText()}" />`	내용 전송 요소(hidden type)
`<input type="hidden" name="document_srl" value="{$document_srl}" />`	문서 고유번호 전송 요소(hidden type)
`<input type="hidden" name="allow_comment" value="Y" />`	댓글 허용 요소(hidden type). 허용하지 않으려면 value="N"을 입력합니다.
`<input type="hidden" name="allow_trackback" value="Y" />`	엮인글 허용 요소(hidden type). 허용하지 않으려면 value="N"을 입력합니다.

XE core 1.4.4 버전에 추가된 새 템플릿 문법을 사용했습니다. 새 템플릿 문법에 대한 자세한 설명은 "XE 템플릿 문법"을 참조하십시오.

제목 입력 창 작성

제목 입력 창은 쓰기 페이지가 '새 글쓰기'일 때와 '글 수정'일 때를 모두 고려해야 합니다. 작성하려는 게시물을 공지사항 게시물로 등록할 것인지도 선택할 수 있어야 합니다.

예제 게시판 스킨의 write_form.html에서는 다음과 같이 제목 입력 창을 작성했습니다.

```
...
<div class="write header">
    <input cond="$oDocument->getTitleText()" type="text" name="title" class="iText"
title="{$lang->title}" value="{htmlspecialchars($oDocument->getTitleText())}" />
    <input cond="!$oDocument->getTitleText()" type="text" name="title" class="iText"
title="{$lang->title}" />
    <input cond="$grant->manager" type="checkbox" name="is notice" value="Y"
checked="checked"|cond="$oDocument->isNotice()" id="is notice" />
    <label cond="$grant->manager" for="is notice">{$lang->notice}</label>
</div>
...
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명
`cond="$oDocument->getTitleText()"`	이미 제목이 있는 '수정' 화면일 때 출력
`cond="!$oDocument->getTitleText()"`	제목이 없는 '새 글쓰기' 화면일 때 출력
`{htmlspecialchars($oDocument->getTitleText())}`	수정 화면일 때 문서의 제목 텍스트를 출력
`{$lang->title}`	'제목' 언어 변수
`cond="$grant->manager"`	관리자이면 공지사항 확인 입력 창과 레이블 출력
`checked="checked"|cond="$oDocument->isNotice()"`	수정 화면일 때 이 글이 공지사항이면 checked 속성과 값을 출력
`{$lang->notice}`	'공지' 언어 변수

XE core 1.4.4 버전에 추가된 새 템플릿 문법을 사용했습니다. 새 템플릿 문법에 대한 자세한 설명은 "XE 템플릿 문법"을 참조하십시오.

내용 입력 창(편집 창) 작성

내용 입력 창과 관련된 코드는 {$oDocument->getEditor()}라는 변수 하나만 선언하면 됩니다. 이 변수는 **확장기능 > 설치된 모듈 > 게시판**에서 편집할 모듈 오른쪽의 **설정**을 클릭하고 **추가 설정** 탭의 **위지윅 에디터**에서 선택한 편집기(위지윅 에디터)를 출력합니다.

예제 게시판 스킨의 write_form.html에서는 다음과 같이 위지윅 에디터를 불러오도록 선언했습니다.

```
...
<div class="write editor">
    {$oDocument->getEditor()}
</div>
...
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명
{$oDocument->getEditor()}	편집기(위지윅 에디터) 출력 변수

글쓴이 정보 입력 창 작성

글쓴이 정보 입력 창은 로그인하지 않은 사람에게만 보입니다. 입력 창은 '글쓴이, 비밀번호, 홈페이지'로 구성됩니다. 비밀번호는 글을 수정 또는 삭제할 때 필요합니다.

예제 게시판 스킨의 write_form.html에서는 다음과 같이 글쓴이 정보 입력 창을 작성했습니다.

```
...
<div class="write_author" cond="!$is_logged">
    <label for="userName">{$lang->writer}</label>
    <input type="text" name="nick name" id="userName" class="iText userName"
value="{htmlspecialchars($oDocument->get('nick name'))}" />
    <label for="userPw">{$lang->password}</label>
    <input type="password" name="password" id="userPw" class="iText userPw" />
    <label for="homePage">{$lang->homepage}</label>
    <input type="text" name="homepage" id="homePage" class="iText homePage"
value="{htmlspecialchars($oDocument->get('homepage'))}" />
</div>
...
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명
	로그인하지 않았으면 글쓴이 정보 입력 창을 출력
{$lang->writer}	'글쓴이' 언어 변수
{htmlspecialchars($oDocument->get('nick_name'))}	쓰기 페이지가 수정 페이지로 사용되면 이미 작성된 글쓴이 이름을 출력
{$lang->password}	'비밀번호' 언어 변수
{$lang->homepage}	'홈페이지' 언어 변수
{htmlspecialchars($oDocument->get('homepage'))}	쓰기 페이지가 수정 페이지로 사용되면 이미 작성된 홈페이지 주소를 출력

등록 버튼 출력

사용자가 작성한 form을 서버에 전송하려면 **등록** 버튼이 필요합니다.

예제 게시판 스킨의 write_form.html에서는 다음과 같이 **등록** 버튼을 출력하도록 작성했습니다.

```
...
<div class="btnArea">
    <input type="submit" value="{$lang->cmd_registration}" class="btn" />
</div>
...
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명
{$lang->cmd_registration}	'등록' 언어 변수

XE core 1.4.4 버전에 추가된 새 템플릿 문법을 사용했습니다. 새 템플릿 문법에 대한 자세한 설명은 "XE 템플릿 문법"을 참조하십시오.

쓰기 페이지 출력 결과 확인

게시판 목록에서 **쓰기**를 클릭하거나 다음 경로로 게시판 쓰기 화면에 접근합니다. 아래 경로에서 'example.com'은 사용자의 웹 사이트가 설치된 도메인 주소를 의미합니다.

- http://example.com/?mid=test_board&act=dispBoardWrite

로그인하지 않은 상태에서 쓰기 페이지에 접근하면 다음과 같이 글쓴이 정보 입력 창이 나타납니다.

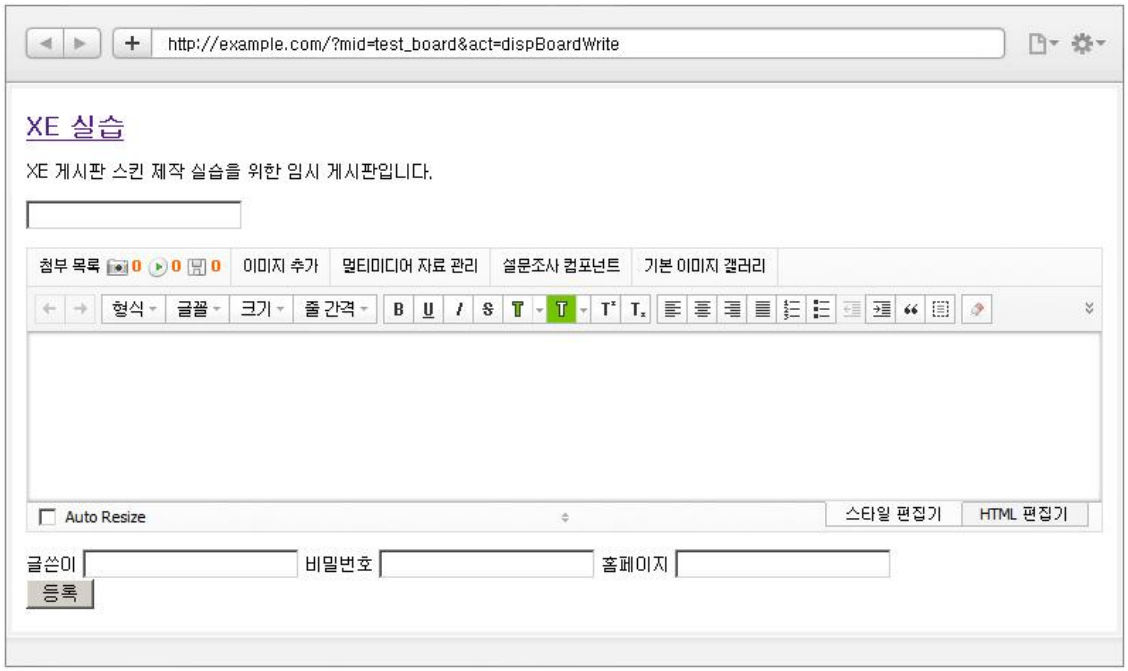


그림 4-7 로그인하지 않은 상태의 쓰기 페이지

로그인한 상태에서 쓰기 페이지에 접근하면 다음과 같은 화면이 나타납니다. 글쓴이 정보 입력 창은 표시되지 않고, 관리자인 경우 공지사항을 작성할 수 있는 **공지** 확인란이 표시됩니다.

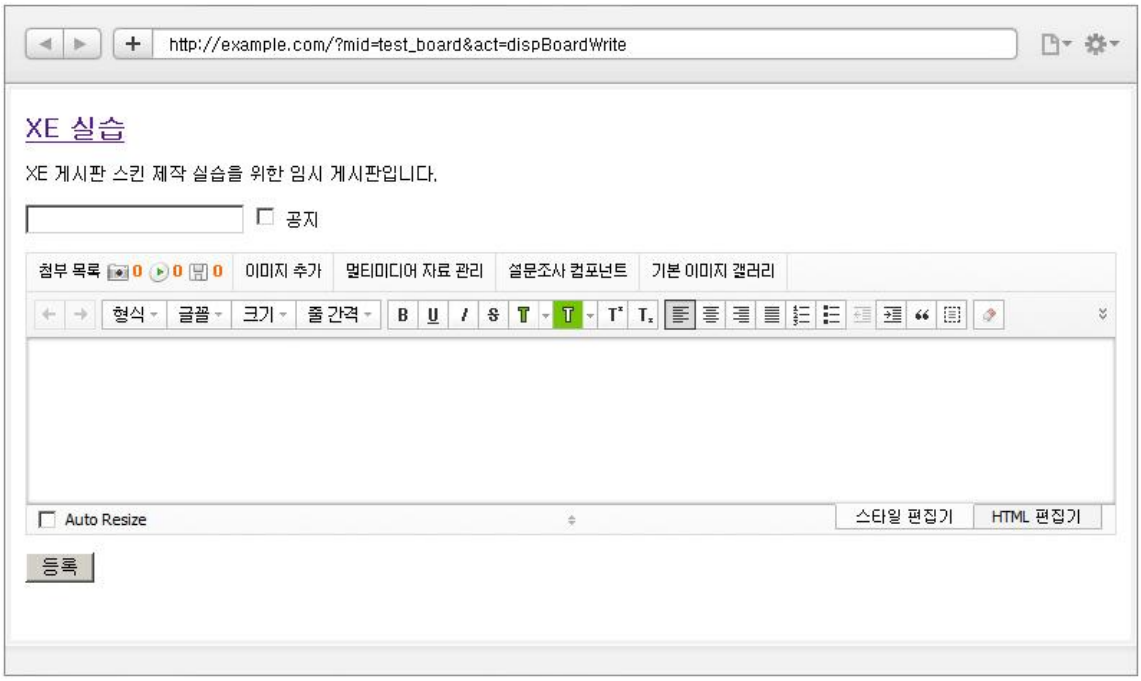


그림 4-8 로그인한 상태의 쓰기 페이지

제목과 내용을 작성하고 **등록**을 클릭해서 작성된 글이 게시판 목록에 보이는지 확인합니다. 로그인한 사용자의 필수 입력 항목은 제목과 내용이고, 로그인하지 않은 사용자의 필수 입력 항목은 제목과 내용, 글쓴이, 비밀번호입니다.

4.10 읽기 페이지 작성

읽기 페이지는 게시물의 본문 외에 엮인글 목록, 댓글 목록, 댓글 쓰기, 게시물 목록 화면으로 구성되며, _read.html에서 작성합니다.

읽기 페이지는 목록 페이지(list.html)에서 포함(include)합니다. 읽기 페이지 아래에 게시물 목록을 제공하여 사용자들이 쉽게 게시물을 탐색할 수 있게 하기 위해서입니다.

다음은 예제 게시판 스킨으로 만든 읽기 페이지의 완성 화면입니다.

그림 4-9 읽기 페이지 완성 화면

_read.html을 작성하는 방법은 다음과 같습니다.

list.html 에 _read.html 포함(include)

list.html은 _read.html을 조건에 따라 포함(include)해야 합니다. 사용자가 읽기 페이지를 호출하면 list.html은 _read.html을 포함(include)합니다.

예제 게시판 스킨의 list.html에서는 다음과 같은 위치에 조건문이 포함된 include문을 작성했습니다. 사용자가 읽기 페이지에 접근하면 읽기 영역 아래쪽에 게시물 목록을 함께 출력합니다.

```
<include target=" header.html" />
<include cond="$oDocument->isExists()" target="_read.html" />
<p ...>{$lang->no documents}</p>
<table ...  summary="List of Articles" id="board_list" class="board_list">
    ...
</table>
<div class="list footer">
    ...
</div>
<include target="_footer.html" />
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명
`<include cond="$oDocument->isExists()" target="_read.html" />`	읽기 페이지가 호출되면 _read.html 을 포함(include)

XE core 1.4.4 버전에 추가된 새 템플릿 문법을 사용했습니다. 새 템플릿 문법에 대한 자세한 설명은 "XE 템플릿 문법"을 참조하십시오.

읽기 페이지 구조

읽기 페이지는 다음과 같이 헤더(제목, 글쓴이, 조회 수, 추천 수, 날짜), 본문, 푸터(첨부 파일, 목록 버튼, 수정 버튼, 삭제 버튼), 엮인글 목록, 댓글 목록, 댓글 입력 양식으로 구성됩니다.

```
<div class="board read">
    <!-- READ HEADER -->
    <div class="read header">
        제목, 글쓴이, 조회 수, 추천 수, 날짜
    </div>
    <!-- /READ HEADER -->
    <!-- READ BODY -->
    <div class="read_body">
        게시물 본문
    </div>
    <!-- /READ BODY -->
    <!-- READ FOOTER -->
    <div class="read_footer">
        첨부 파일, 목록 버튼, 수정 버튼, 삭제 버튼
    </div>
    <!-- /READ FOOTER -->
</div>
엮인글 목록 포함(include)
댓글 목록 포함(include)
<!-- WRITE COMMENT -->
<form class="write_comment">
    댓글 입력 양식
</form>
```

```
<!-- /WRITE COMMENT -->
```

제목, 글쓴이 출력

예제 게시판 스킨의 _read.html에서는 다음과 같이 제목과 글쓴이를 출력하도록 작성했습니다.

```
...
<!-- READ HEADER -->
<div class="read_header">
   <h1><a href="{$oDocument->getPermanentUrl()}">{$oDocument->getTitle()}</a></h1>
   <a cond="$oDocument->getHomepageUrl()" href="{$oDocument->getHomepageUrl()}"
class="author">{$oDocument->getNickName()}</a>
   <strong cond="!$oDocument->getHomepageUrl()" class="author">{$oDocument-
>getNickName()}</strong>
</div>
<!-- /READ HEADER -->
...
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명
`<a href="{$oDocument->getPermanentUrl()}">`	게시물의 영구 URL
`{$oDocument->getTitle()}`	게시물 제목
`<a cond="$oDocument->getHomepageUrl()" href="{$oDocument->getHomepageUrl()}" class="author">...</a>`	홈페이지 주소가 있으면 링크와 닉네임을 출력
`<strong cond="!$oDocument->getHomepageUrl()" class="author">...</strong>`	홈페이지 주소가 없으면 이름만 출력
`{$oDocument->getNickName()}`	글쓴이의 닉네임을 출력

XE core 1.4.4 버전에 추가된 새 템플릿 문법을 사용했습니다. 새 템플릿 문법에 대한 자세한 설명은 "XE 템플릿 문법"을 참조하십시오.

조회 수, 추천 수, 날짜 출력

예제 게시판 스킨의 _read.html에서는 조회 수, 추천 수, 날짜를 출력하도록 다음과 같이 작성했습니다.

```
...
<!-- READ HEADER -->
<div class="read header">
   ...
   <p class="sum">
      <span class="read">{$lang->readed_count}
         <span class="num">{$oDocument->get('readed_count')}</span>
      </span>
      <span class="vote">{$lang->voted count}
         <span class="num">{$oDocument->get('voted count')}</span>
      </span>
      <span class="time">{$oDocument->getRegdate('Y.m.d H:i')}</span>
   </p>
</div>
<!-- /READ HEADER -->
...
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명
{$lang->readed_count}	'조회 수' 언어 변수
{$oDocument->get('readed_count')}	조회 수 출력
{$lang->voted_count}	'추천 수' 언어 변수
{$oDocument->get('voted_count')}	추천 수 출력
{$oDocument->getRegdate('Y.m.d H:i')}	글 쓴 날짜 및 시간 출력(시간에 초 단위까지 출력하려면 H:i:s)

게시물 본문 출력

예제 게시판 스킨의 _read.html에서는 게시물 본문을 출력하도록 다음과 같이 작성했습니다.

```
...
<!-- READ BODY -->
<div class="read body">
    {$oDocument->getContent(false)}
</div>
<!-- /READ BODY -->
...
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명
{$oDocument->getContent(false)}	게시물 본문 내용 출력

첨부 파일 출력

예제 게시판 스킨의 _read.html에서는 첨부 파일을 출력하도록 다음과 같이 작성했습니다.

```
...
<!-- READ FOOTER -->
<div class="read_footer">
    <div cond="$oDocument->hasUploadedFiles()" class="fileList">
        <button type="button" class="toggleFile"
onclick="jQuery(this).next('ul.files').toggle();">{$lang->uploaded_file} ({$oDocument-
>get('uploaded_count')})</button>
        <ul class="files">
            <li loop="$oDocument->getUploadedFiles()=>$key,$file"><a
href="{getUrl('')}{$file->download url}">{$file->source filename} <span
class="fileSize">[File Size:{FileHandler::filesize($file-
>file_size)}/Download:{number_format($file->download_count)}]</span></a></li>
        </ul>
    </div>
</div>
<!-- /READ FOOTER -->
...
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명
<div cond="$oDocument->hasUploadedFiles()" class="fileList">	첨부 파일이 있으면 포함된 내용을 출력
{$lang->uploaded_file}	'첨부' 언어 변수

XE 템플릿 문법/변수	설명
{$oDocument->get('uploaded_count')}	첨부된 파일 수
<li loop="$oDocument->getUploadedFiles()=>$key,$file">	첨부된 파일 목록이 있으면 받아와서 출력하는 반복문
<a href="{getUrl('')}{$file->download_url}">	첨부 파일 다운로드 링크
{$file->source_filename}	첨부 파일 이름
{FileHandler::filesize($file->file_size)}	첨부 파일 크기
{number_format($file->download_count)}	첨부 파일 다운로드 횟수

XE core 1.4.4 버전에 추가된 새 템플릿 문법을 사용했습니다. 새 템플릿 문법에 대한 자세한 설명은 "XE 템플릿 문법"을 참조하십시오.

목록, 수정, 삭제 버튼 출력

예제 게시판 스킨의 _read.html에서는 **목록, 수정, 삭제** 버튼을 출력하도록 다음과 같이 작성했습니다.

```
...
<!-- READ FOOTER -->
<div class="read footer">
    ...
    <div class="btnArea">
        <span class="goList"><a href="#board_list" class="btn">{$lang->cmd_list}</a></span>
        <span class="goEdit">
            <a cond="$oDocument->isEditable()" class="btn"
href="{getUrl('act','dispBoardWrite','document srl',$oDocument->document srl,'comment srl','')}">{$lang->cmd modify}</a>
            <a cond="$oDocument->isEditable()" class="btn"
href="{getUrl('act','dispBoardDelete','document_srl',$oDocument->document srl,'comment srl','')}">{$lang->cmd delete}</a>
        </span>
    </div>
</div>
<!-- /READ FOOTER -->
...
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명
...	아래쪽 게시물 목록으로 이동
{$lang->cmd_list}	'목록' 언어 변수
isEditable()" ...>...	편집 권한이 있으면 포함된 내용을 출력
{getUrl('act','dispBoardWrite','document_srl',$oDocument->document_srl,'comment_srl','')}	수정 페이지 URL 출력
{$lang->cmd_modify}	'수정' 언어 변수
{getUrl('act','dispBoardDelete','document_srl',$oDocument->document_srl,'comment_srl','')}	삭제 페이지 URL 출력
{$lang->cmd_delete}	'삭제' 언어 변수

XE core 1.4.4 버전에 추가된 새 템플릿 문법을 사용했습니다. 새 템플릿 문법에 대한 자세한 설명은 "XE 템플릿 문법"을 참조하십시오.

엮인글 목록, 댓글 목록 포함(include)

예제 게시판 스킨의 _read.html에서는 다음과 같이 _trackback.html과 _comment.html을 포함(include)하도록 작성했습니다.

```
<div class="board_read">
   <!-- READ HEADER -->
   ...
   <!-- /READ HEADER -->
   <!-- READ BODY -->
   ...
   <!-- /READ BODY -->
   <!-- READ FOOTER -->
   ...
   <!-- /READ FOOTER -->
</div>
<include cond="$oDocument->allowTrackback()" target="_trackback.html" />
<include cond="$oDocument->allowComment()" target=" comment.html" />
<!-- WRITE COMMENT -->
...
<!-- /WRITE COMMENT -->
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명
`<include target="..." />`	파일 포함(include)
`cond="$oDocument->allowTrackback()"`	엮인글 쓰기를 허용했다면 엮인글 목록 파일(_trackback.html)을 포함(include)
`cond="$oDocument->allowComment()"`	댓글 쓰기를 허용했다면 댓글 목록 파일(_comment.html)을 포함(include)

XE core 1.4.4 버전에 추가된 새 템플릿 문법을 사용했습니다. 새 템플릿 문법에 대한 자세한 설명은 "XE 템플릿 문법"을 참조하십시오.

게시물에 대한 댓글 입력 양식 출력

예제 게시판 스킨의 _read.html에서는 다음과 같이 게시물에 대한 댓글 입력 양식을 출력하도록 작성했습니다. 댓글의 댓글 쓰기 및 댓글 수정하기 페이지는 comment_form.html에서 작성합니다.

```
<div class="board_read">
   <!-- READ HEADER -->
   ...
   <!-- /READ HEADER -->
   <!-- READ BODY -->
   ...
   <!-- /READ BODY -->
   <!-- READ FOOTER -->
   ...
   <!-- /READ FOOTER -->
</div>
<include cond="$oDocument->allowTrackback()" target="_trackback.html" />
<include cond="$oDocument->allowComment()" target="_comment.html" />
<!-- WRITE COMMENT -->
```

```
<form cond="$grant->write comment && $oDocument->isEnableComment()" action="./"
method="post" onsubmit="return procFilter(this, insert comment)" class="write comment">
    <input type="hidden" name="mid" value="{$mid}" />
    <input type="hidden" name="document_srl" value="{$oDocument->document_srl}" />
    <input type="hidden" name="comment_srl" value="" />
    <textarea name="content" rows="5" cols="50"></textarea>
    <div class="write author" cond="!$is logged">
        <label for="userName">{$lang->writer}</label>
        <input type="text" name="nick_name" id="userName" class="iText userName" />
        <label for="userPw">{$lang->password}</label>
        <input type="password" name="password" id="userPw" class="iText userPw" />
        <label for="homePage">{$lang->homepage}</label>
        <input type="text" name="homepage" id="homePage" class="iText homePage" />
    </div>
    <div class="btnArea">
        <input type="submit" value="{$lang->cmd comment registration}" class="btn" />
    </div>
</form>
<!-- /WRITE COMMENT -->
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명
`<form cond="$grant->write_comment && $oDocument->isEnableComment()"> ...</form>`	댓글 쓰기 권한이 있고 댓글 쓰기가 허용되어 있으면 댓글 쓰기 양식을 출력
`onsubmit="return procFilter(this, insert_comment)"`	사용자 입력 데이터 유효성 검사 및 form 전송
`<input type="hidden" name="mid" value="{$mid}" />`	모듈 아이디 값 전송 요소(hidden type)
`<input type="hidden" name="document_srl" value="{$oDocument->document_srl}" />`	문서 고유 번호 전송 요소(hidden type)
`<input type="hidden" name="comment_srl" value="" />`	댓글 고유 번호 전송 요소(hidden type)
`<textarea name="content" rows="5" cols="50"></textarea>`	댓글 입력 창
`<div class="write_author" cond="!$is_logged">...</div>`	로그인되어 있지 않으면 포함된 내용(이름 입력 창, 비밀번호 입력 창, 홈페이지 입력 창) 출력
`{$lang->writer}`	'글쓴이' 언어 변수
`{$lang->password}`	'비밀번호' 언어 변수
`{$lang->homepage}`	'홈페이지' 언어 변수
`{$lang->cmd_comment_registration}`	'댓글 쓰기' 언어 변수

XE core 1.4.4 버전에 추가된 새 템플릿 문법을 사용했습니다. 새 템플릿 문법에 대한 자세한 설명은 "XE 템플릿 문법"을 참조하십시오.

4.11 엮인글/댓글 관련 페이지 작성

4.11.1 엮인글 목록 작성

엮인글 목록은 _trackback.html에 작성합니다. 엮인글 목록은 _read.html에서 포함(include)해서 읽기 페이지에 출력합니다.

예제 게시판 스킨의 _trackback.html에서는 다음과 같이 엮인글 목록을 작성했습니다.

```
<!-- TRACKBACK -->
<div class="feedback" id="trackback">
   <div class="fbHeader">
      <h2>{$lang->trackback} <em>'{$oDocument->getTrackbackCount()}'</em></h2>
      <p class="trackbackURL"><a href="{$oDocument->getTrackbackUrl()}" onclick="return
false;">{$oDocument->getTrackbackUrl()}</a></p>
   </div>
   <ul cond="$oDocument->getTrackbackCount()" class="fbList">
      <li class="fbItem" loop="$oDocument->getTrackbacks()=>$key,$val"
id="trackback_{$val->trackback_srl}">
         <h3 class="author"><a href="{$val->url}" title="{htmlspecialchars($val-
>blog name)}">{htmlspecialchars($val->title)}</a></h3>
         <p class="time">{zdate($val->regdate, "Y.m.d H:i")}</p>
         <p class="xe_content">{$val->excerpt}</p>
         <p class="action" cond="$grant->manager"><a
href="{getUrl('act','dispBoardDeleteTrackback','trackback_srl',$val-
>trackback srl)}">{$lang->cmd delete}</a></p>
      </li>
   </ul>
</div>
<!-- /TRACKBACK -->
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명
{$lang->trackback}	'엮인글' 언어 변수
{$oDocument->getTrackbackCount()}	엮인글의 개수를 출력
{$oDocument->getTrackbackUrl()}	엮인글을 달기 위한 URL 출력
cond="$oDocument->getTrackbackCount()"	엮인글이 있으면 포함된 내용을 출력(조건)
loop="$oDocument->getTrackbacks()=>$key,$val"	엮인글이 있으면 포함된 내용을 출력(반복)
{$val->trackback_srl}	엮인글 고유번호
{$val->url}	엮인 페이지의 URL
{htmlspecialchars($val->blog_name)}	엮인 사이트 이름
{htmlspecialchars($val->title)}	엮인 제목
{zdate($val->regdate, "Y.m.d H:i")}	엮인 시각
{$val->excerpt}	엮인글 내용
cond="$grant->manager"	관리자이면 포함된 내용을 출력
{getUrl('act','dispBoardDeleteTrackback','trackback_srl',$val->trackback_srl)}	엮인글 삭제 페이지 URL

XE 템플릿 문법/변수	설명
{$lang->cmd_delete}	'삭제' 언어 변수

XE core 1.4.4 버전에 추가된 새 템플릿 문법을 사용했습니다. 새 템플릿 문법에 대한 자세한 설명은 "XE 템플릿 문법"을 참조하십시오.

4.11.2 댓글 목록 작성

댓글 목록은 _comment.html에 작성합니다. 댓글 목록은 _read.html에서 포함(include)해서 읽기 페이지에 출력합니다.

예제 게시판 스킨의 _comment.html에서는 다음과 같이 댓글 목록을 작성했습니다.

```
<!-- COMMENT -->
<div class="feedback" id="comment">
   <div class="fbHeader">
      <h2>{$lang->comment} <em>'{$oDocument->getCommentcount()}'</em></h2>
   </div>
   <ul cond="$oDocument->getCommentcount()" class="fbList">
      <li loop="$oDocument->getComments()=>$key,$comment" class="fbItem" style="padding-
left:{($comment->get('depth'))*15}px"|cond="$comment->get('depth')"
id="comment_{$comment->comment_srl}">
         <h3 class="author">
            <a cond="$comment->homepage" href="{$comment->homepage}">{$comment-
>getNickName()}</a>
            <strong cond="!$comment->homepage">{$comment->getNickName()}</strong>
         </h3>
         <p class="time">{$comment->getRegdate('Y.m.d H:i')}</p>
         {$comment->getContent(false)}
         <p class="action">
            <a href="{getUrl('act','dispBoardReplyComment','comment srl',$comment-
>comment_srl)}">{$lang->cmd_reply}</a>
            <a cond="$comment->isGranted()||!$comment->get('member_srl')"
href="{getUrl('act','dispBoardModifyComment','comment_srl',$comment-
>comment srl)}">{$lang->cmd modify}</a>
            <a cond="$comment->isGranted()||!$comment->get('member srl')"
href="{getUrl('act','dispBoardDeleteComment','comment srl',$comment-
>comment_srl)}">{$lang->cmd_delete}</a>
         </p>
      </li>
   </ul>
   <div cond="$oDocument->comment page navigation" class="pagination">
      <a href="{getUrl('cpage',1)}#comment" class="prevEnd">{$lang->first_page}</a>
      <block loop="$page_no=$oDocument->comment_page_navigation->getNextPage()">
         <strong cond="$cpage==$page_no">{$page_no}</strong>
         <a cond="$cpage!=$page_no"
href="{getUrl('cpage',$page no)}#comment">{$page no}</a>
      </block>
      <a href="{getUrl('cpage',$oDocument->comment_page_navigation->last_page)}#comment"
class="nextEnd">{$lang->last_page}</a>
   </div>
</div>
<!-- /COMMENT -->
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명
{$lang->comment}	'댓글' 언어 변수
{$oDocument->getCommentcount()}	댓글 수를 출력

XE 템플릿 문법/변수	설명
cond="$oDocument->getCommentcount()"	댓글이 있으면 포함된 내용을 출력(조건)
loop="$oDocument->getComments()=>$key,$comment"	댓글이 있으면 포함된 내용을 출력(반복)
{($comment->get('depth'))*15}	댓글의 계층 수에서 15 를 곱함
|cond="$comment->get('depth')"	댓글에 계층이 있으면 속성을 출력
{$comment->comment_srl}	댓글 고유번호
cond="$comment->homepage"	홈페이지가 있으면 포함된 내용을 출력
{$comment->homepage}	홈페이지 URL
{$comment->getNickName()}	글쓴이 이름을 출력
cond="!$comment->homepage"	홈페이지가 없으면 포함된 내용을 출력
{$comment->getRegdate('Y.m.d H:i')}	댓글 작성 날짜 및 시각을 출력
{$comment->getContent(false)}	댓글 본문을 출력
{getUrl('act','dispBoardReplyComment','comment_srl',$comment->comment_srl)}	댓글의 댓글 쓰기 페이지 URL
{$lang->cmd_reply}	'댓글' 언어 변수
cond="$comment->isGranted()\|\|!$comment->get('member_srl')"	댓글 편집 권한이 있거나 회원이 쓴 글이 아닐 때 포함된 내용(수정, 삭제)을 출력
{getUrl('act','dispBoardModifyComment','comment_srl',$comment->comment_srl)}	글 수정 페이지 URL
{$lang->cmd_modify}	'수정' 언어 변수
{getUrl('act','dispBoardDeleteComment','comment_srl',$comment->comment_srl)}	글 삭제 페이지 URL
{$lang->cmd_delete}	'삭제' 언어 변수
cond="$oDocument->comment_page_navigation"	댓글이 너무 많아서 페이지 링크가 필요하면 출력
{getUrl('cpage',1)}	댓글 첫 페이지 URL
{getUrl('cpage',$oDocument->comment_page_navigation->last_page)}	댓글 끝 페이지 URL
{$lang->first_page}	댓글 '첫 페이지' 언어 변수
{$lang->last_page}	댓글 '끝 페이지' 언어 변수
<block loop="$page_no=$oDocument->comment_page_navigation->getNextPage()">...</block>	댓글 페이지 번호 목록을 출력(반복)
cond="$cpage==$page_no"	댓글 현재 페이지와 페이지 번호가 일치하면 포함된 내용을 출력
cond="$cpage!=$page_no"	댓글 현재 페이지와 페이지 번호가 일치하지 않

XE 템플릿 문법/변수	설명
	으면 포함된 내용을 출력
{getUrl('cpage',$page_no)}	댓글 페이지 URL
{$page_no}	댓글 페이지 번호

XE core 1.4.4 버전에 추가된 새 템플릿 문법을 사용했습니다. 새 템플릿 문법에 대한 자세한 설명은 "XE 템플릿 문법"을 참조하십시오.

4.11.3 댓글의 댓글 쓰기 및 댓글 수정 페이지 작성

사용자가 댓글에 또 다른 댓글을 작성할 때나 댓글을 수정할 때 나타나는 페이지입니다. 게시물 원본 글에 대한 댓글 쓰기 양식은 읽기 페이지(_read.html)에 포함되어 있습니다. 댓글의 댓글 쓰길 및 댓글 수정 페이지는 comment_form.html에 작성합니다.

다음은 예제 게시판 스킨으로 만든 댓글의 댓글 쓰기 페이지입니다. 댓글 원문의 글쓴이 이름, 댓글 작성 시간, 댓글 원문을 윗부분에 출력하고, 댓글을 수정할 때는 자신이 작성한 글을 textarea 영역에 출력합니다.

그림 4-10 댓글의 댓글 쓰기 페이지 완성 화면

예제 게시판 스킨의 comment_form.html에서는 다음과 같이 댓글의 댓글 쓰기 및 댓글 수정 페이지를 작성했습니다.

```
<include target=" header.html" />
<div cond="$oSourceComment->isExists()" class="context data">
   <h3 class="author">
      <a cond="$oSourceComment->homepage" href="{$oSourceComment-
>homepage}">{$oSourceComment->getNickName()}</a>
      <strong cond="!$oSourceComment->homepage">{$oSourceComment->getNickName()}</strong>
```

```
    </h3>
    <p class="time">{$oSourceComment->getRegdate('Y.m.d H:i')}</p>
    {$oSourceComment->getContent(false)}
</div>
<!-- WRITE COMMENT -->
<form action="./" method="post" onsubmit="return procFilter(this, insert comment)"
class="write comment">
    <input type="hidden" name="mid" value="{$mid}" />
    <input type="hidden" name="document_srl" value="{$oComment->get('document_srl')}" />
    <input type="hidden" name="comment_srl" value="{$oComment->get('comment_srl')}" />
    <input type="hidden" name="parent_srl" value="{$oComment->get('parent_srl')}" />
    <textarea name="content" rows="5" cols="50">{htmlspecialchars($oComment-
>get('content'))}</textarea>
    <div class="write_author" cond="!$is_logged">
        <label for="userName">{$lang->writer}</label>
        <input type="text" name="nick name" id="userName" class="iText userName"
value="{htmlspecialchars($oComment->get('nick name'))}" />
        <label for="userPw">{$lang->password}</label>
        <input type="password" name="password" id="userPw" class="iText userPw" />
        <label for="homePage">{$lang->homepage}</label>
        <input type="text" name="homepage" id="homePage" class="iText homePage"
value="{htmlspecialchars($oComment->get('homepage'))}" />
    </div>
    <div class="btnArea">
        <input type="submit" value="{$lang->cmd_comment_registration}" class="btn" />
    </div>
</form>
<!-- /WRITE COMMENT -->
<include target="_footer.html" />
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명
`<include target="_header.html" />`	_header.html 포함(include)
`<include target="_footer.html" />`	_footer.html 포함(include)
`cond="$oSourceComment->isExists()"`	댓글 원문이 있으면 출력
`cond="$oSourceComment->homepage"`	홈페이지 주소가 있으면 포함된 내용을 출력
`cond="!$oSourceComment->homepage"`	홈페이지 주소가 없으면 포함된 내용을 출력
`{$oSourceComment->homepage}`	홈페이지 주소
`{$oSourceComment->getNickName()}`	댓글 원문의 글쓴이 이름
`{$oSourceComment->getRegdate('Y.m.d H:i')}`	댓글 원문 작성 시각
`{$oSourceComment->getContent(false)}`	댓글 본문 출력
`onsubmit="return procFilter(this, insert_comment)"`	사용자 입력 데이터 유효성 검사 및 form 전송
`<input type="hidden" name="mid" value="{$mid}" />`	게시판 모듈 아이디 전송 요소(hidden type)
`<input type="hidden" name="document_srl" value="{$oComment->get('document_srl')}" />`	게시물 고유번호 전송 요소(hidden type)
`<input type="hidden" name="comment_srl" value="{$oComment->get('comment_srl')}" />`	댓글 고유번호 전송 요소(hidden type)
`<input type="hidden" name="parent_srl" value="{$oComment->get('parent_srl')}" />`	댓글 원문의 고유번호 전송 요소(hidden type)
`<textarea name="content" rows="5" cols="50">{htmlspecialchars($oComment-`	댓글 입력 및 편집 창. textarea 내부에 포함된

XE 템플릿 문법/변수	설명
>get('content'))}</textarea>	변수는 댓글 수정 시 원본 글을 보여 줌.
cond="!$is_logged"	로그인하지 않았으면 포함된 내용을 출력
{$lang->writer}	'글쓴이' 언어 변수
{$lang->password}	'비밀번호' 언어 변수
{$lang->homepage}	'홈페이지' 언어 변수
{htmlspecialchars($oComment->get('nick_name'))}	글쓴이 이름
{htmlspecialchars($oComment->get('homepage'))}	홈페이지 URL
{$lang->cmd_comment_registration}	'댓글 등록' 언어 변수

XE core 1.4.4 버전에 추가된 새 템플릿 문법을 사용했습니다. 새 템플릿 문법에 대한 자세한 설명은 "XE 템플릿 문법"을 참조하십시오.

4.12 삭제 페이지 작성

4.12.1 게시물 삭제 페이지 작성

권한이 있는 사람이 게시물을 삭제하려고 할 때 한 번 더 확인하는 페이지입니다. 게시물 삭제 페이지는 delete_form.html에 작성합니다.

다음은 예제 게시판 스킨으로 만든 게시물 삭제 페이지의 완성 화면입니다.

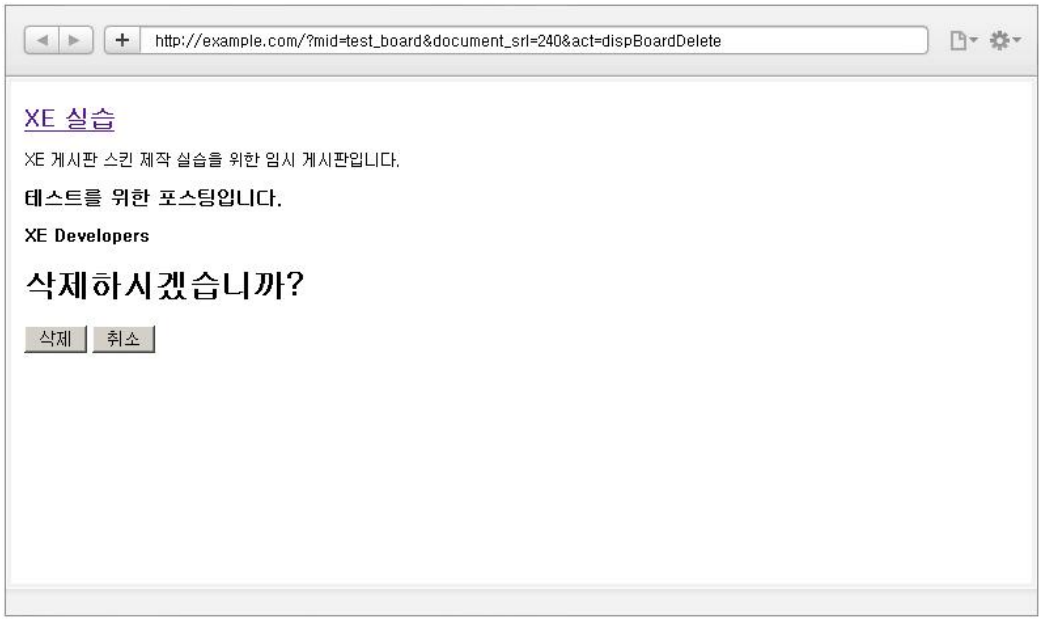


그림 4-11 게시물 삭제 페이지 완성 화면

예제 게시판 스킨의 delete_form.html에서는 다음과 같이 게시물 삭제 페이지를 작성했습니다.

```
<include target="_header.html" />
<div cond="$oDocument->isExists()" class="context_data">
    <h3 class="title">{$oDocument->getTitle()}</h3>
    <p class="author">
        <strong>{$oDocument->getNickName()}</strong>
    </p>
</div>
<form action="./" method="get" onsubmit="return procFilter(this, delete_document)"
class="context message">
    <input type="hidden" name="mid" value="{$mid}" />
    <input type="hidden" name="page" value="{$page}" />
    <input type="hidden" name="document_srl" value="{$document_srl}" />
    <h1>{$lang->confirm_delete}</h1>
    <div class="btnArea">
        <input type="submit" value="{$lang->cmd delete}" class="btn" />
        <button type="button" onclick="history.back()" class="btn">{$lang-
>cmd_cancel}</button>
    </div>
</form>
<include target="_footer.html" />
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명

XE 템플릿 문법/변수	설명
`<include target="_header.html" />`	_header.html 포함(include)
`<include target="_footer.html" />`	_footer.html 포함(include)
`cond="$oDocument->isExists()"`	게시물 원문이 있으면 출력
`{$oDocument->getTitle()}`	게시물 원문 제목
`{$oDocument->getNickName()}`	게시물 원문 글쓴이
`onsubmit="return procFilter(this, delete_document)"`	입력 데이터 유효성 검사 및 form 전송
`<input type="hidden" name="mid" value="{$mid}" />`	모듈 아이디 전송 요소(hidden type)
`<input type="hidden" name="page" value="{$page}" />`	페이지 번호 전송 요소(hidden type)
`<input type="hidden" name="document_srl" value="{$document_srl}" />`	문서 고유번호 전송 요소(hidden type)
`{$lang->confirm_delete}`	'삭제하시겠습니까?' 언어 변수
`{$lang->cmd_delete}`	'삭제' 언어 변수
`{$lang->cmd_cancel}`	'취소' 언어 변수

XE core 1.4.4 버전에 추가된 새 템플릿 문법을 사용했습니다. 새 템플릿 문법에 대한 자세한 설명은 "XE 템플릿 문법"을 참조하십시오.

4.12.2 댓글 삭제 페이지 작성

사용자가 댓글 **삭제** 버튼을 눌렀을 때 한 번 더 확인하는 페이지입니다. 댓글 삭제 페이지는 delete_comment_form.html에서 작성합니다.

다음은 예제 게시판 스킨으로 만든 댓글 삭제 페이지입니다. 삭제하려는 댓글의 작성자, 댓글 작성 시각, 댓글 원문을 출력하고 그 아래에 **삭제**와 **취소** 버튼을 출력합니다.

그림 4-12 댓글 삭제 페이지 완성 화면

예제 게시판 스킨의 delete_comment_form.html에서는 다음과 같이 댓글 삭제 페이지를 작성했습니다.

```
<include target="_header.html" />
<div cond="$oComment->isExists()" class="context_data">
    <h3 class="author">
        <a cond="$oComment->homepage" href="{$oComment->homepage}">{$oComment-
>getNickName()}</a>
        <strong cond="!$oComment->homepage">{$oComment->getNickName()}</strong>
    </h3>
    <p class="time">{$oComment->getRegdate('Y.m.d H:i')}</p>
    {$oComment->getContent(false)}
</div>
<form action="./" method="get" onsubmit="return procFilter(this, delete_comment)"
class="context_message">
    <input type="hidden" name="mid" value="{$mid}" />
    <input type="hidden" name="page" value="{$page}" />
    <input type="hidden" name="document_srl" value="{$oComment->get('document_srl')}" />
    <input type="hidden" name="comment_srl" value="{$oComment->get('comment_srl')}" />
    <h1>{$lang->confirm_delete}</h1>
    <div class="btnArea">
        <input type="submit" value="{$lang->cmd_delete}" class="btn" />
        <button type="button" onclick="history.back()" class="btn">{$lang-
>cmd_cancel}</button>
    </div>
</form>
<include target="_footer.html" />
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명
`<include target="_header.html" />`	_header.html 포함(include)
`<include target="_footer.html" />`	_footer.html 포함(include)
`cond="$oComment->isExists()"`	댓글 원문이 있으면 출력
`cond="$oComment->homepage"`	댓글 원문에 글쓴이의 홈페이지가 있으면 출력

XE 템플릿 문법/변수	설명
cond="!$oComment->homepage"	댓글 원문에 글쓴이의 홈페이지가 없으면 출력
{$oComment->homepage}	댓글 원문에 대한 글쓴이의 홈페이지 URL
{$oComment->getNickName()}	글쓴이 이름
{$oComment->getRegdate('Y.m.d H:i')}	댓글 원문 작성 시각
{$oComment->getContent(false)}	댓글 원문의 본문 출력
onsubmit="return procFilter(this, delete_comment)"	입력 데이터 유효성 검사 및 form 전송
<input type="hidden" name="mid" value="{$mid}" />	모듈 아이디 전송 요소(hidden type)
<input type="hidden" name="page" value="{$page}" />	페이지 번호 전송 요소(hidden type)
<input type="hidden" name="document_srl" value="{$oComment->get('document_srl')}" />	댓글 원문이 속한 게시물의 고유번호 전송 요소(hidden type)
<input type="hidden" name="comment_srl" value="{$oComment->get('comment_srl')}" />	댓글 원문의 고유번호 전송 요소(hidden type)
{$lang->confirm_delete}	'삭제하시겠습니까?' 언어 변수
{$lang->cmd_delete}	'삭제' 언어 변수
{$lang->cmd_cancel}	'취소' 언어 변수

XE core 1.4.4 버전에 추가된 새 템플릿 문법을 사용했습니다. 새 템플릿 문법에 대한 자세한 설명은 "XE 템플릿 문법"을 참조하십시오.

4.12.3 엮인글 삭제 페이지 작성

사용자가 엮인글 삭제 버튼을 눌렀을 때 한 번 더 확인하는 페이지입니다. 엮인글 삭제 페이지는 delete_trackback_form.html에서 작성합니다.

다음은 예제 게시판 스킨으로 만든 엮인글 삭제 페이지의 완성 화면입니다.

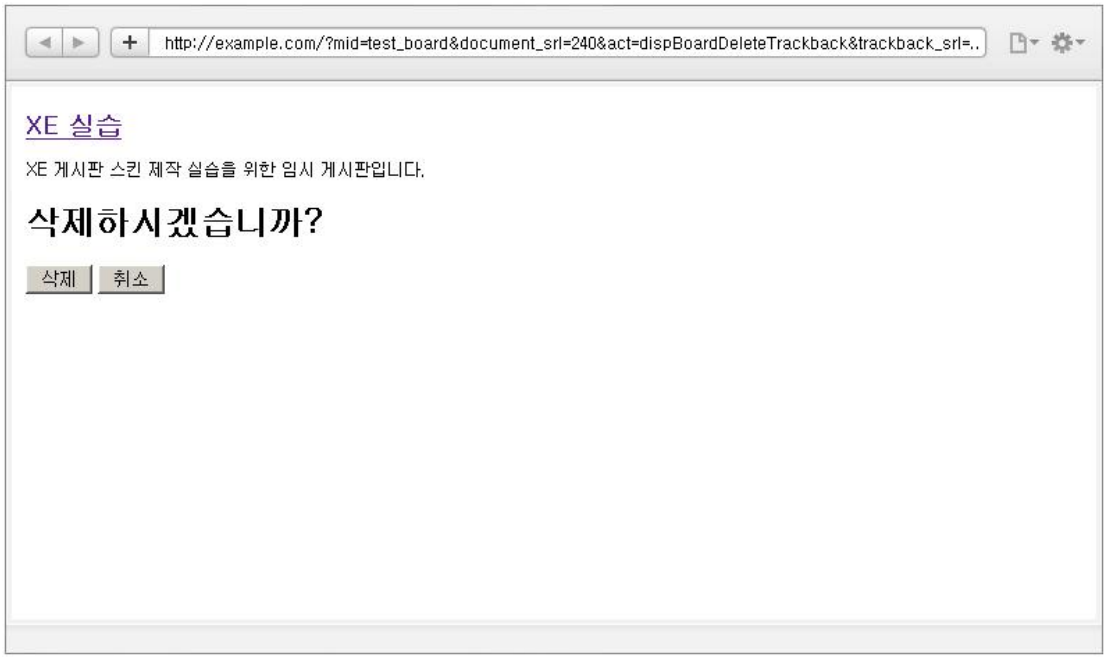


그림 4-13 엮인글 삭제 페이지 완성 화면

예제 게시판 스킨의 delete_trackback_form.html에서는 다음과 같이 엮인글 삭제 페이지를 작성했습니다.

```
<include target=" header.html" />
<form action="./" method="get" onsubmit="return procFilter(this, delete trackback)"
class="context_message">
   <input type="hidden" name="mid" value="{$mid}" />
   <input type="hidden" name="page" value="{$page}" />
   <input type="hidden" name="document srl" value="{document srl}" />
   <input type="hidden" name="trackback srl" value="{$trackback srl}" />
   <h1>{$lang->confirm delete}</h1>
   <div class="btnArea">
      <input type="submit" value="{$lang->cmd_delete}" class="btn" />
      <button type="button" onclick="history.back()" class="btn">{$lang-
>cmd cancel}</button>
   </div>
</form>
<include target="_footer.html" />
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명
`<include target="_header.html" />`	_header.html 포함(include)
`<include target="_footer.html" />`	_footer.html 포함(include)
`onsubmit="return procFilter(this, delete_trackback)"`	입력 데이터 유효성 검사 및 form 전송
`<input type="hidden" name="mid" value="{$mid}" />`	모듈 아이디 전송 요소(hidden type)
`<input type="hidden" name="page" value="{$page}" />`	페이지 번호 전송 요소(hidden type)
`<input type="hidden" name="document_srl" value="{document_srl}" />`	게시물의 고유번호 전송 요소(hidden type)
`<input type="hidden" name="trackback_srl" value="{$trackback_srl}" />`	엮인글의 고유번호 전송 요소(hidden

XE 템플릿 문법/변수	설명
	type)
{$lang->confirm_delete}	'삭제하시겠습니까?' 언어 변수
{$lang->cmd_delete}	'삭제' 언어 변수
{$lang->cmd_cancel}	'취소' 언어 변수

XE core 1.4.4 버전에 추가된 새 템플릿 문법을 사용했습니다. 새 템플릿 문법에 대한 자세한 설명은 "XE 템플릿 문법"을 참조하십시오.

4.13 권한 안내 페이지 작성

사용자가 권한이 없는 페이지에 접근했을 때 권한이 없다는 것을 알려주고 로그인 페이지로 이동하거나 뒤로 이동할 수 있는 버튼을 제공하는 페이지입니다. 권한 안내 페이지는 message.html에서 작성합니다.

다음은 예제 게시판 스킨으로 만든 권한 안내 페이지의 완성 화면입니다.

그림 4-14 권한 안내 페이지 완성 화면

예제 게시판 스킨의 message.html에서는 다음과 같이 권한 안내 페이지를 작성했습니다.

```
<include target=" header.html" />
<div class="context message">
   <h1>{$message}</h1>
   <div class="btnArea">
      <a cond="!$is_logged" href="{getUrl('act','dispMemberLoginForm')}"
class="btn">{$lang->cmd login}</a>
      <button type="button" onclick="history.back()" class="btn">{$lang-
>cmd_back}</button>
   </div>
</div>
<include target="_footer.html" />
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명
`<include target="_header.html" />`	_header.html 포함(include)
`<include target="_footer.html" />`	_footer.html 포함(include)
`{$message}`	게시물 열람 권한이 없으면 '권한이 없습니다'라는 안내 메시지를 출력

XE 템플릿 문법/변수	설명
`cond="!$is_logged"`	로그인하지 않았으면 포함된 내용을 출력
`{getUrl('act','dispMemberLoginForm')}`	로그인 페이지 URL
`{$lang->cmd_login}`	'로그인' 언어 변수
`{$lang->cmd_back}`	'돌아가기' 언어 변수

XE core 1.4.4 버전에 추가된 새 템플릿 문법을 사용했습니다. 새 템플릿 문법에 대한 자세한 설명은 "XE 템플릿 문법"을 참조하십시오.

4.14 비밀번호 입력 페이지 작성

사용자가 비밀번호 확인이 필요한 페이지에 접근했을 때 비밀번호를 묻는 페이지입니다. 회원이 아닌 사용자가 자신이 쓴 글을 수정하거나 삭제할 때 필요합니다. 비밀번호 입력 페이지는 input_passowrd_form.html에서 작성합니다.

다음은 예제 게시판 스킨으로 만든 비밀번호 입력 페이지의 완성 화면입니다.

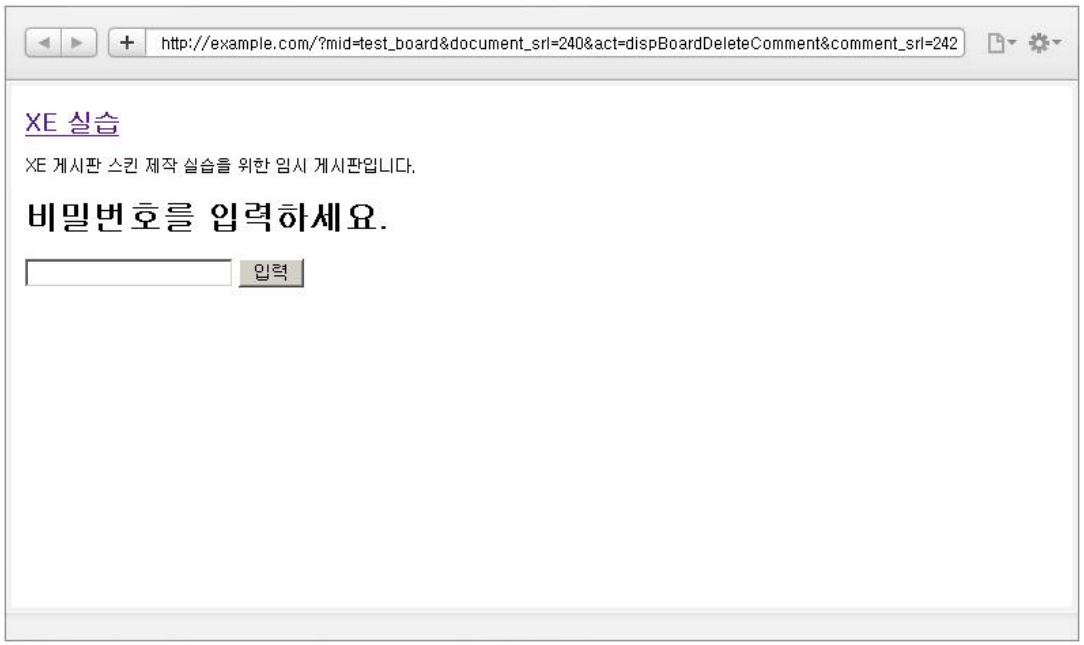


그림 4-15 비밀번호 입력 페이지 완성 화면

예제 게시판 스킨의 input_passowrd_form.html에서는 다음과 같이 비밀번호 입력 페이지를 작성했습니다.

```
<include target="_header.html" />
<form action="./" method="get" onsubmit="return procFilter(this, input password)"
class="context message">
   <input type="hidden" name="mid" value="{$mid}" />
   <input type="hidden" name="page" value="{$page}" />
   <input type="hidden" name="document_srl" value="{$document_srl}" />
   <input type="hidden" name="comment_srl" value="{$comment_srl}" />
   <h1>{$lang->msg input password}</h1>
   <input type="password" name="password" title="{$lang->password}" class="iText" />
   <input type="submit" value="{$lang->cmd input}" class="btn" />
</form>
<include target="_footer.html" />
```

위 코드에서 사용된 템플릿 문법과 변수는 다음과 같습니다.

XE 템플릿 문법/변수	설명
`<include target="_header.html" />`	_header.html 포함(include)
`<include target="_footer.html" />`	_footer.html 포함(include)
`onsubmit="return procFilter(this, input_password)"`	입력 데이터 유효성 검사 및 form 전송

XE 템플릿 문법/변수	설명
{$lang->msg_input_password}	'비밀번호를 입력하세요.' 언어 변수
<input type="hidden" name="mid" value="{$mid}" />	모듈 아이디 전송 요소(hidden type)
<input type="hidden" name="page" value="{$page}" />	페이지 번호 전송 요소(hidden type)
<input type="hidden" name="document_srl" value="{$document_srl}" />	게시물 고유번호 전송 요소(hidden type)
<input type="hidden" name="comment_srl" value="{$comment_srl}" />	댓글 고유번호 전송 요소(hidden type)
{$lang->password}	'비밀번호' 언어 변수
{$lang->cmd_input}	'입력' 언어 변수

XE core 1.4.4 버전에 추가된 새 템플릿 문법을 사용했습니다. 새 템플릿 문법에 대한 자세한 설명은 "XE 템플릿 문법"을 참조하십시오.

4.15 CSS 적용

이 절에서는 게시판 스킨에 CSS 코드를 적용하는 방법을 설명합니다. 이 문서에서 CSS 코드를 작성하는 방법은 설명하지 않습니다. 스킨 제작자는 자신의 의도에 맞게 CSS 코드를 수정해서 사용할 수 있습니다.

1. 예제 게시판 스킨의 CSS 파일을 작성합니다.

   다음은 user_board.css에서 예제 게시판 스킨 제작에 사용된 클래스 목록을 구조화하여 제공한 예제입니다. 게시판 스킨 제작자는 자신의 의도에 알맞게 CSS 코드를 추가해서 사용할 수 있습니다.

```
@charset "utf-8";

/* User Board */
.user board{}

/* Board Header */
.board_header{}
.board_header h2{}
.board_header p{}

/* Form Control */
/* list.html |  read.html | write form.html | comment form.html */
.user_board .iText{}
.user_board .iCheck{}
.user_board textarea{}

/* Button Area */
/* list.html | write form.html | comment form.html |  read.html | delete form.html |
delete_comment_form.html | delete_trackback_form.html | message.html */
.user_board .btnArea{}
.user_board .btnArea .goList{}
.user board .btnArea .goEdit{}
.user board .btn{}

/* Board List */
/* list.html */
.no document{}
.board list{}
.board list th{}
.board_list th.title{}
.board_list tr.notice{}
.board_list td{}
.board list td.notice{}
.board list td.no{}
.board_list td.title{}
.board_list td.title a{}
.board_list td.title a.replyNum{}
.board_list td.title a.trackbackNum{}
.board list td.author{}
.board list td.time{}
.board_list td.lastReply{}
.board_list td.lastReply a{}
.board list td.lastReply span{}
.board list td.lastReply sub{}
.board list td.readNum{}
.board list td.voteNum{}
.list_footer{}
.list_footer .pagination{}
.list footer .btnArea{}
.list footer .board search{}
.list footer .board search select{}
.list_footer .board_search option{}

/* Board Write */
/* write form.html */
.board_write{}
```

```
.write header{}
.write header .iText{}
.write_header .iCheck{}
.write_header label{}
.write_editor{}
.board write .write author{}
.board write .btnArea{}

/* Board Read */
/* _read.html */
.board_read{}
.read header{}
.read header h1{}
.read_header h1 a{}
.read_header a.author{}
.read header strong.author{}
.read header .sum{}
.read header .sum .read{}
.read header .sum .vote{}
.read_header .sum .time{}
.read_body{}
.read body .xe content{}
.read footer{}
.read footer .fileList{}
.read_footer .toggleFile{}
.read_footer .files{}
.read_footer .files li{}
.read footer .btnArea{}

/* Feedback (Trackback+Comment) */
/* _trackback.html | _comment.html */
.feedback{}
.feedback .fbHeader{}
.feedback .fbHeader h2{}
.feedback .fbHeader .trackbackURL{}
.feedback .fbList{}
.feedback .fbItem{}
.feedback .author{}
.feedback .author a{}
.feedback .author strong{}
.feedback .time{}
.feedback .xe_content{}
.feedback .action{}
.feedback .action a{}
.feedback .pagination{}

/* Pagination */
/* list.html | _comment.html */
.user_board .pagination{}
.user_board .pagination a{}
.user board .pagination a.prevEnd{}
.user_board .pagination a.nextEnd{}
.user_board .pagination strong{}

/* Write Author */
/*  read.html | write form.html | comment form.html */
.write author{}
.write_author label{}
.write_author .iText{}
.write_author .userName{}
.write_author .userPw{}
.write author .homePage{}

/* Write Comment */
/* _read.html | comment_form.html */
.write_comment{}
.write_comment textarea{}
.write comment .write author{}
.write comment .btnArea{}

/* Context Data | Context Message */
/* comment form.html | delete form.html | delete comment form.html |
input_password_form.html | message.html */
```

```
.context data{}
.context data h3.author{}
.context_data h3.author a{}
.context_data h3.author strong{}
.context_data h3.title{}
.context data p.author{}
.context data p.author strong{}
.context_data .time{}
.context_data .xe_content{}
.context_message{}
.context_message h1{}
.context message .iText{}
.context_message .btnArea{}
```

2. _header.html에 다음과 같이 user_board.css를 참조하도록 선언합니다.

```
<load target="user_board.css" />
```

   자세한 설명은 "CSS 파일 참조"를 참조하십시오.

3. 다음 경로로 접근하여 페이지에 CSS가 적용되었는지 확인합니다. 아래 경로에서 'example.com'은 사용자의 웹 사이트가 설치된 도메인 주소를 의미합니다.
   - mod_rewrite를 사용할 경우: http://example.com/test_board/
   - mod_rewrite를 사용하지 않는 경우: http://example.com/?mid=test_board

만약 작성된 코드가 화면에 반영되지 않았다면 CSS를 참조하는 <load /> 템플릿 문법이 올바른지, 또는 CSS 참조 경로(target)가 바르게 지정되었는지 확인하십시오.

4.16 자바스크립트 적용

이 절에서는 게시판 스킨에 자바스크립트 코드를 추가하는 방법을 설명합니다. 이 문서에서 jQuery 실행 코드를 작성하는 방법은 설명하지 않습니다. 스킨 제작자는 자신의 의도에 맞는 코드를 작성해서 추가합니다.

1. user_board.js에 jQuery 문법의 코드를 작성합니다.

```
// <![CDATA[
jQuery(function($){
    // 이곳에 jQuery 코드를 작성합니다.
});
// ]]>
```

2. _header.html에 다음과 같이 user_board.js를 참조하도록 선언합니다.

```
<load target="user_board.js" type="body" />
```

   자세한 파일 참조 방법은 "JS 파일 참조"를 참조하십시오.

3. 다음 경로로 접근하여 자바스크립트가 제대로 실행되는지 확인합니다. 아래 경로에서 'example.com'은 사용자의 웹 사이트가 설치된 도메인 주소를 의미합니다.
   - mod_rewrite를 사용할 경우: http://example.com/test_board
   - mod_rewrite를 사용하지 않는 경우: http://example.com/?mid=test_board

만약 작성된 코드가 화면에 반영되지 않았다면 user_board.js를 참조하는 <load /> 템플릿 문법이 올바른지, 또는 user_board.js 참조 경로(target)가 바르게 지정되었는지 확인하십시오.

참고

XE에서 jQuery를 사용하는 기본적인 방법은 "자바스크립트와 jQuery 사용"을 참조하십시오.
jQuery를 사용해서 다양한 효과를 구현하려면 http://jquery.com/ 웹 사이트를 참조하십시오.