スキン製作マニュアル

著作権

本書は、情報提供のみを目的としています。NHN(株)は、本書に記載された情報の完全性と正確性を検証するために細心の注意をもって作成しましたが、発生し得る内容上の誤りや記載漏れに対しては責任を負いません。したがって、本書の使用や使用の結果による責任は一切利用者のものとし、NHN(株)はこれについて明示的、黙示的を問わず何ら保証を行うものではありません。
URL情報を含め、本書で言及されている特定のソフトウェア商品、製品は該当所有者の著作権法に従うものとし、該当の著作権法の遵守は、利用者の責任で行うものとします。
NHN(株)は、本書の内容を将来予告なしに変更することがあります。

オープンソースライセンス使用告知

XEは、様々なオープンソースライセンスのうち、LGPL(GNU Lesser General Public License) v2を採用しています。LGPL v2とv3の間には、多少の違いがあるため、バージョンv2を採用している点に注意が必要です。LGPLは、基本的にはGPLと同様ですが、適用範囲がより制限的であるという点で異なります。LGPLも、GPLと同様に、該当ライセンスを保有するソフトウェアを含むソフトウェアに同じライセンスを強制する効力があります。しかし、GPLがGPLライセンスを保有したソフトウェアを含むすべてのソフトウェアに無条件にソース公開を強制するのに対し、LGPLライセンスを保有するプログラムは、特定の条件内で使用する場合であれば、ソース公開の義務はありません。したがって、LGPLライセンスを保有するソフトウェアは、独占的ソフトウェアの開発にも利用することができます。詳細については、下記のサイトを参照してください。

LGPLライセンス：http://www.gnu.org/copyleft/lesser.html
GPLライセンス：http://www.gnu.org/licenses/gpl.html

はじめに

本書について

本書では、XEスキンを製作する方法について説明します。本書の内容は、XE coreバージョン1.5をもとにされています。

読者

本書の対象読者は、XEスキンを製作し、自分だけのWebサイトを作成したいユーザーです。XEスキンは、HTMLとCSS、JavaScript、jQueryを使って作成されているため、これらの関連知識を持っていると、XEスキン製作についてよりスムーズに理解することができます。

XEをインストールし、使用する方法については、『XEユーザーマニュアル』を参照してください。

連絡先

本書の内容に誤りや不明な点がありましたら、以下の連絡先までお問い合わせください。

電子メール： developers@xpressengine.com

改版履歴

版数	日付	履歴
1.0	2010.12.31	1.0 配布
1.1	2011.12.06	XE core 1.5 アップデートに伴い改訂

表記規則

参考表記

> **参考**
> 読者が参考にすべき内容を記述します。

注意表記

> **注意**
> 読者が知っておくべき内容、システムエラーを引き起こす恐れのある内容、実行しない場合は財産上の被害を被りかねない内容を記述します。

画面名/メニュー名/入力値の表記

本書では、画面名、メニュー名、入力値を以下のように表記します。

- 画面名：**画面名**画面
- メニュー名：**メニュー > 下位メニュー**
- 入力値：*home page*を入力します。

ソースコードの表記

本書では、ソースコードを、以下のように灰地に黒文字で表記します。

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

目次

図表一覧

表一覧

図一覧

1. XE スキン製作の概要

この章では、XEスキンの定義とスキン製作に必要な要素について説明します。

1.1 XE スキンとは

XE スキンとは、XE で作成したデータをユーザー画面に表現するフォームのことです。モジュール、レイアウト、ウィジェット、ウィジェットスタイルには、ひとつ以上のスキンがあります。XE を使用するユーザーは、XE オフィシャルサイト(http://www.xpressengine.com)で提供している様々なスキンをダウンロードして利用したり、本書を参考にして自分だけのオリジナルのスキンを製作することができます。

下記図にて、ひとつの Web サイト(Textyle)にいろんなタイプのスキンを適用した例を示します。

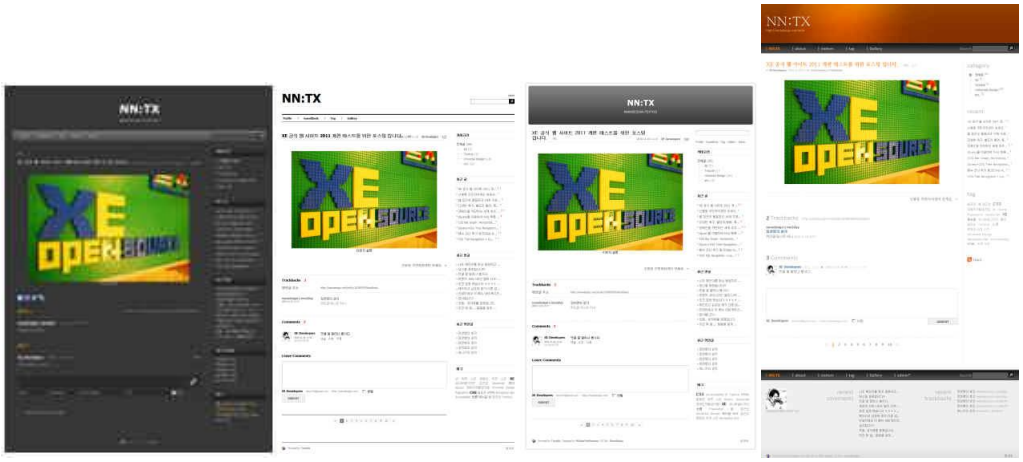


図 1-1 様々なスキンの適用

モジュールとウィジェットのスキンは、該当モジュールやウィジェットのインストールされているディレクトリ配下の skins ディレクトリに格納されます。レイアウトとウィジェットスタイルのスキンは、該当レイアウトとウィジェットスタイルがインストールされているディレクトリ内に格納されます。

1.2 XE スキン製作に必要な要素

XE スキンを製作するには、HTML と CSS、JavaScript、XE テンプレート文法についての知識が必要です。

HTML(Hyper Text Markup Language)

Web ページの構造をつくる言語です。表現するコンテンツの見出し、段落、目録などを HTML タグで作成すると、Web ブラウザがこのタグを解釈して Web ページに表示します。検索エンジンが Web ドキュメントを収集しやすく、また様々なアクセス環境で使用できるようにするためには、標準に準拠して意味のある正しい HTML ドキュメントを作成しなければなりません。

CSS(Cascading Style Sheet)

HTML が Web ページの構造のための言語なら、CSS は表現のための言語です。Web ブラウザには、HTML 要素に対する基本的な表現フォームがありますが、CSS を用いてこの表現フォームを見栄えよくすることができます。特定の HTML 要素を選択し、該当要素に CSS 属性を与えることで、画面における配置、色彩、線、形状、背景、画像などを自由に表現することができます。

JavaScript と jQuery

HTML、CSS が静的表現のための言語なら、JavaScript は、XE ドキュメントの動的表現のための言語です。JavaScript を使用すると、ページを移動することなくユーザーの実行結果を画面に即時に表示することができます。事前に読み込まれたコンテンツを、状況に合わせて、画面に表示したり、非表示にしたり、ユーザーの入力データにエラーはないか即時に確認し、エラーに対する詳細説明を提供したりします。XE は、JavaScript を使いやすくするために jQuery という JavaScript ライブラリを使用しています。

JavaScript に対応していない環境もあるため、HTML だけで実装可能な機能を全的に JavaScript に依存して実装すると相互の運用性が劣ってしまいます。JavaScript を使用するときには、常にこれを正常に対応できない障害状況も考慮しなければなりません。

XE テンプレート文法

XE テンプレート文法は、サーバーで PHP 文法として解釈されます。DB から情報を抜き出し、ユーザー画面へ表示します。XE テンプレート文法を使用すると、モジュール、またはウィジェットの機能を、許可範囲内で拡張したり、縮小したりできます。

2. XE スキン製作の基礎

XEスキン製作に必要なHTMLとCSS、JavaScript/jQuery、XEテンプレート文法について説明します。

2.1 HTML の理解

HTML は、無秩序なデータに「意味」を与え、データを有用な「情報」に変える力を持っています。HTML ドキュメントを配置し、飾るためには CSS を使用します。HTML ドキュメントは、データの「意味」と「表現」をよく切り分け、適切な言語を用いて HTML を作成することで、作成者本人でなくても容易に理解し、編集することができます。「表現」を CSS で切り分けることによって、デザインの変更のために HTML を編集する労力を削減することができます。

2.1.1 要素、属性、値

HTML は、要素と属性、値で構成されています。

要素

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

<h1>で始まり、</h1>で囲まれた全体が要素です。この要素は、<h1>要素といい、<h1>と</h1>はそれぞれ開始タグと終了タグといいます。その間にある「XE」は、この要素の「コンテンツ(contents)」といいます。

「h」は、見出しを意味する「heading」の略です。多くのタグがこのように略語を使います。数字は、見出しの階層をあらわします。「1」は、最上位レベルの見出しの意になります。ほとんどの Web ブラウザでの見出しは、太字で大きなフォントで表示していますが、Web 製作者は単に太字で大きいフォントで表現するためだけにこの要素を使用してはいけません。データに意味を与えずにただ太字で大きなフォントにするのは、「表現」にあたるため、CSS で処理しなければなりません。

属性

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

上記コードで「title」は、属性です。「属性」は、この要素が他の要素と何が違うかをあらわします。「title」は、この要素の意味を付加的に説明する「参考見出し」の役割をします。HTML 要素ごとにそれぞれ異なった使用可能な属性が規定されています。

値(Value)

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

属性は、必ず「値」を持ちます。上記コードにて、「Xpress Engine」は、title 属性の値をあらわします。値は、現在の要素がどのような属性を持っているかを具体的に定め、画面の表示に直接的な影響を与えます。属性によっては、間違った値を入力すると画面が正常に表示されなくなることもあります。値は、属性によって既定されているものとユーザーが定義しなければならないものがありますが、Xpress Engine は、ユーザーの定義する値にあたります。

2.1.2 HTML 要素の始まりと終わり

すべての HTML 要素には、データ範囲を明示するための「始まり」と「終わり」があります。

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

- この要素の始まりは、<h1>です。ここから最初の見出しが始まります。
- この要素の終わりは、</h1>です。最初の見出しは、ここで終わります。

テキストコンテンツでない場合、HTML 要素自体がデータになる場合もありますが、このような場合は、始まりと同時に終わります。下記例の場合、異なるイメージのコンテンツを含めていないため、始まりと同時にタグを閉じます。

```
<img />
```

2.1.3 親要素と子要素

HTML で要素は、他の要素を含めることができます。

例として、見出し「XE」をクリックして初期画面へ移動するコンテンツを作成してみましょう。

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

<h1>要素内に<a>要素が含まれています。このように、ひとつのデータは、複数の要素で重ねることができます。このとき、<h1>要素のように外側にある要素を、「親要素」、<a>要素のように内側にある要素を「子要素」といいます。

<a>要素は、データが違うリソースを参照するよう繋げます。href 属性は、<a>要素内で参照の対象を指定します。すなわち、上記コードは、「XE というデータは、最上位レベルの見出しであり、index.html を参照している」ということを定義しています。

ただし、要素を重畳使用する場合は、開始タグと終了タグの使用順を守らなければなりません。下記のような宣言は、重畳の規則に反した間違った使用例です。

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

2.1.4 インライン要素とブロック要素

ブロック要素は、「ひとつの独立したカタマリ」として扱い、改行します。 、<h>、<p>要素などがブロック要素にあたります。インライン要素は、'行内の一部'とし、改行なしで連続で表現します。、<img>、<a>要素などが代表的なインライン要素です。

ブロック要素は、インライン要素を含めることができますが、インライン要素はブロック要素を含めることができません。どんな要素がブロックであるか、インラインであるかは、HTML 規格を確認してください。

下記は、ブロック要素がインライン要素を囲んでいる正しいマークアップの例です。<h1>要素は、ブロック要素で、<a>要素は、インライン要素です。

```
<h1><a>XE</a></h1>      <!--正しいマークアップ-->
```

下記は、インライン要素がブロック要素を囲んでいる正しくないマークアップの例です。

```
<a><h1>XE</h1></a>      <!--正しくないマークアップ-->
```

参考

W3C HTML 4.01 規格：http://trio.co.kr/webrefer/html/cover.html
W3C HTML 4.01 要素索引：http://trio.co.kr/webrefer/html/index/elements.html
W3C HTML 4.01 属性索引：http://trio.co.kr/webrefer/html/index/attributes.html
W3C XHTML 1.0 規格：http://trio.co.kr/webrefer/xhtml/overview.html
W3C マークアップ有効性チェック：http://validator.w3.org/
W3 Schools オンライン学習：http://www.w3schools.com/

2.2 CSS の理解

HTML が Web ドキュメントに「意味」を与え、データを「情報」に変えるのに対し、CSS は、ドキュメントの「配置と表現」に関与し、コンテンツを理解しやすく、また見栄えよくします。XE スキンは、ほとんどの場合、CSS ファイルと HTML が分離されていますが、HTML ドキュメントが CSS ファイルを参照しているため、ユーザーには HTML と CSS がすべて反映された最終画面が表示されることになります。CSS 文法を学べば、ドキュメントを見栄えよくできるだけでなく、HTML を標準に沿って正しく使用することができます。

2.2.1 セレクタと属性、値

CSS は、「セレクタ」と「属性」と「値」で構成されています。「セレクタ」は、HTML の特定の「要素」を選択し、「属性」と「値」は選択された HTML 要素が画面にどのように表示されるかを定義します。

セレクタ(selector)

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

「h1」は、「セレクタ」です。<h1>要素を指します。

属性(property)

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

「font-size」は、「属性」です。<h1>要素のフォントサイズを制御する宣言です。

値(value)

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

「24px」は、「値」です。<h1>要素の font-size 属性(フォントサイズ)値を具体的に明示します。

2.2.2 CSS セレクタのタイプ

CSS セレクタは、特定の HTML 要素を選択します。セレクタを使用すると、選択した要素に固有の CSS フォームを表現することができます。CSS は、下記のように様々なタイプのセレクタをサポートします。

タイプセレクタ(type selector)

```
h1 { … }
```

HTML 要素名をそのままセレクタ名として使用することをタイプセレクタといいます。HTML ドキュメントで要素名が一致する場合、すべて選択します。標準 HTML 要素名をセレクタ名として使用することができます。

ID セレクタ(id selector)

```
#selector { … }
```

ID セレクタは、セレクタ名の前に番号記号(#)を表示します。HTML ドキュメントで ID 値が一致する要素を選択します。ID セレクタの名前は、ユーザーが定義できます。ID セレクタ名を定義する規則は、下記の通りです。

- すべての自然語をセレクタ名として使用できますが、アルファベット大文字小文字と数字の組み合わせを推奨します。
- セレクタ名として特殊文字を使用することはできませんが、アンダースコア(_)とハイフン(-)に限り、使用できます。
- 数字で開始することはできません。

HTML ドキュメントは、ひとつのページに 2 つ以上の ID 値を使用することを許可しません。ID 値は、ひとつの HTML ドキュメント内でユニークでなければなりません。

クラスセレクタ(class selector)

```
.selector { … }
```

クラスセレクタは、セレクタ名の前にピリオド(.)を表示します。HTMLドキュメントにおいてクラス値の一致する要素を選択します。クラスセレクタの名前は、ユーザーが定義できます。クラスセレクタ名を定義する規則は、下記の通りです。

- すべての言語をセレクタ名として使用できますが、アルファベット大文字小文字と数字の組み合わせを推奨します。
- セレクタ名として特殊文字を使用することはできませんが、アンダースコア(_)とハイフン(-)に限り、使用できます。
- 数字で開始することはできません。

HTMLドキュメントは、ひとつのページに 2 つ以上の同じクラス値を使用できます。

子セレクタ(child selector)

```
h1>a { … }
```

子セレクタは、セレクタの間に山括弧(>)を入れて表示します。セレクタの列挙が HTML 要素の重畳順と一致した場合に選択します。重畳が 2 階層以上になる子孫(子供の子供)は、選択しません。1階層の直下に含まれている子供のみを選択します。

> **注意**
> IE6 ブラウザは、子セレクタをサポートしません。

子孫セレクタ(descendant selector)

```
h1 a { … }
```

子孫セレクタは、セレクタの間に空きスペースを入れて表示します。セレクタの列挙が HTML 要素の重畳順と一致した場合に選択します。下記のように<h1>要素内に含まれている<a>要素のみを選択します。

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

<h1>要素の外側に存在する<a>要素は、この宣言の影響を受けません。

ユニバーサルセレクタ(universal selector)

```
* { … }
```

ユニバーサルセレクタは、アスタリスク(*)で表示します。このセレクタは、すべての HTML 要素を選択します。Web ブラウザは、通常、HTML 要素ごとに基本の CSS フォームを指定していますが、これを同じ値に初期化することができます。ユニバーサルセレクタを使用すると、ページ表示速度が遅くなるため、必ず使用すべき場合を除いては、使用は推奨されません。

擬似セレクタ(pseudo selector)

```
a:focus { … }
```

擬似セレクタは、他のセレクタの後ろにコロン(:)を付けて表示します。要素の状態を選択する「擬似クラスセレクタ」とHTML コード内にない擬似要素を選択する「擬似要素セレクタ」があります。他のセレクタが静的な状態の HTML 要素を選択するのに対し、擬似セレクタは、HTML 要素の動的な変化状態を選択したり、HTML コードにない擬似要素を選択します。

擬似クラスセレクタの種類について以下に示します。

表 2-1 擬似クラスセレクタ

擬似クラスセレクタ	説明	例題

:link	未訪問のリンクを選択します。 a というタイプセレクタと結合して使用します。	a:link
:visited	訪問済みのリンクを選択します。 a というタイプセレクタと結合して使用します。	a:visited
:hover	マウスポインタを合わせた状態を選択します。	a:hover
:active	マウスクリックの瞬間、またはリターンキーを押下した瞬間の状態を選択します。	a:active
:focus	フォーカスされた要素の状態を選択します。	a:focus
:first-child	最初の子要素を選択します。	li:first-child
:lang(language)	言語属性が一致した場合に選択します。	p:lang(en)

注意

IE6 ブラウザは、:first-child, :lang()擬似クラスセレクタをサポートしません。IE6 ブラウザでは、:hover、 :active、 :focus 擬似クラスセレクタを a 以外の要素ではサポートしません。

a 要素にいろんな状態の擬似クラスセレクタを指定するときは、:link(リンク):visited(訪問の際に) :hover(過剰に) :active(活動すると) :focus(フォーカスされます)の順を維持することを推奨します。後から宣言されたクラスが先に宣言されたものより優先順位が高いため、すべて上書きしてしまうからです。たとえば、:hover の後に a:visited を宣言すると、訪問済みのリンクの上にマウスを置いても:hover が適用されません。訪問済みのリンクの上にマウスを置いたときに反応させるためには、:visited の次に:hover クラスを宣言しなければなりません。

擬似要素セレクタの種類について以下に示します。

表 2-2 擬似要素セレクタ

擬似要素セレクタ	説明	例題
:first-line	最初の行を選択します。 ブロック要素のみ適用可能。	p:first-line
:first-letter	最初の文字を選択します。 ブロック要素のみ適用可能。	p:first-letter
:before	要素開始点の内側の擬似要素を選択します。	div:before{ content:"…" } (div 要素開始点の内側に仮想の'…'文字列を生成し選択します)
:after	要素終了点の内側の擬似要素を選択します。	div:after{ content:"…" } (div 要素終了点の内側に仮想の'…'文字列を生成し選択します)

注意

IE6 ブラウザは、擬似要素セレクタをサポートしません。

セレクタグルーピング(Selector Grouping)

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

セレクタグルーピングは、コンマ(,)を使用して複数のセレクタをひとつに括り、属性と値を一度だけ宣言します。複数のセレクタが同じ属性と値を共有するときにひとつのセレクタで括ることができます。

上記のセレクタグルーピング宣言は、下記の宣言と同じ意味をあらわします。

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

隣接セレクタ(Adjacent Sibling Selector)

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

兄弟セレクタは、セレクタの間にプラス記号(+)を入れて表示します。要素が兄弟関係にあり、同じ列挙順の場合、後からあらわれる要素を選択します。

上記の隣接兄弟セレクタ宣言によると、下記の HTML コードにて<h1>要素の次に<h2>要素が順番にあらわれているため、<h2>要素は赤色の文字で表示されることになります。

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

注意

IE6 ブラウザでは、兄弟セレクタをサポートしません。

属性セレクタ(Attribute Selector)

```
input[checked] { ... } /* input 要素に checked 属性が使用された場合、無条件で選択 */
input[type=text] { ... } /* input 要素の type 属性が text の場合にだけ選択 */
img[alt~=dog] { ... }  /* img 要素の alt 属性値に dog という単語が含まれている場合、選択 */
p[lang|=en] { ... } /* p 要素の lang 属性が en-us のように en-で始まる場合、選択 */
```

HTML 要素に宣言された属性を使用して該当要素を選択します。属性セレクタの種類について以下に示します。

表 2-3 属性セレクタ

属性セレクタ	説明	例題
[attribute]	HTML 要素に attribute という属性が使用された場合、値の有無とは関係なく要素を選択します。	input[checked]
[attribute=value]	HTML 要素に attribute という属性が使用されていて、たったひとつの値があり、そしてその値が正確に、value のときに要素を選択します。	input[type=text]
[attribute~=value]	HTML 要素に attribute という属性が使用されていて、ひとつ以上の値が空白で区切られて存在していて、その中のひとつの値が value のときに要素を選択します。	img[alt~=dog]
[attribute|= value]	HTML 要素に attribute という属性が使用されていて、値が value で始まる場合、要素を選択します。	p[lang|=en]

注意

IE6 ブラウザでは、属性セレクタをサポートしません。

参考

CSS規格： http://www.w3.org/TR/1998/REC-CSS2-19980512/
CSS Reference： http://www.w3schools.com/css/css_reference.asp

2.3 JavaScript と jQuery の使用

XE は、JavaScript を使いやすくするために jQuery という JavaScript ライブラリを使用しています。jQuery を使用すると、スキン製作者が JavaScript の専門家でなくても、JavaScript を容易に扱うことができます。

ここでは、XE スキンを製作するときに HTML ドキュメントにスクリプトをインクルードする方法について説明します。

2.3.1 jQuery ライブラリをインクルードする

XE core には、jQuery ライブラリがすでにインクルードされており、すべてのページで参照するよう宣言されているため、別途 jQuery ライブラリを呼び出す宣言を行う必要はありません。XE で作成されたすべての Web ドキュメントには、HTML コードの<head>要素の内側に、下記のように jQuery ライブラリ参照コードが含まれています。

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

JavaScript 参照コードは、文法的に HTML コードの<head>要素、または<body>要素内のどこにでも含めることができますが、XE では jQuery ライブラリ参照コードを HTML ドキュメントの<head>要素内に含めています。Web ブラウザは、JavaScript コードを宣言された順番通りに解釈していきますが、このとき JavaScript コードの解釈順序が画面表示に影響を及ぼす場合があります。HTML 解釈と同時に結果を実行しなければならない jQuery 依存コードがあった場合、jQuery ライブラリより先に解釈されるといけないため、XE は jQuery ライブラリ参照宣言を、HTML ドキュメントの<head>要素内に含めているのです。

> **参考**
> XE のインストール先によって jquery.js ファイルの参照パスは、例題とは異なることがあります。

2.3.2 XE テンプレート文法で JavaScript を宣言する

jQuery コードは、JavaScript であるため、JavaScript 文法にしたがって作成しなければなりません。JavaScript の宣言は、HTML ドキュメントの<head>要素、または<body>要素の内側で行わなければなければならず、これ以外のところで宣言を行うと HTML 文法エラーが発生します。HTML 文法の許可しない場所で JavaScript コードを作成すると、標準を準拠して厳格に実装されているブラウザでは JavaScript を解釈できなくなる場合もあります。JavaScript コードを HTML ドキュメントに直接インクルードする方法もありますが、別途の*.js ファイルで区別し、HTML ドキュメントから呼び出す方法もあります。

JavaScript は、用途によって次のように宣言する場所を区別して使用します。

<head>要素にて JS ファイルを参照

Web ブラウザが画面表示を終了する前に JavaScript でユーザー画面の一部コンテンツを表示したり、非表示にする動作を実行する場合は、JavaScript コードを HTML ドキュメントの<head>要素に含めます。この場合、JavaScript コードを<body>要素に含めると、JavaScript が HTML より後に解釈され、一時的に画面が正常に表示されないことがあります。

<head>要素に含まれた JavaScript は、HTML ドキュメントより先に解釈されますが、HTML ドキュメントのロードが完了してから実行するようコードを作成しなければなりません。

<body>要素にて JS ファイルを参照

JavaScript が HTML ドキュメントをロードする時点で画面表示のためになんらかの動作を実行しない場合は、HTML ドキュメントの<body>要素に含みます。ただし、<body>要素のいちばん下に宣言することを推奨します。Web ブラウザは、HTML コードと JavaScript コードを同時に解釈せず、作成された順番通りに解釈するため、JavaScript コードを後で解釈するようにすると、HTML ドキュメントを画面に表示するスピードが速くなります。

XE テンプレート文法で JavaScript を宣言する詳細な方法については、「JS ファイルの参照」を参照してください。

2.3.3 標準文法で JavaScript を宣言する

別途作成された JavaScript ファイルを HTML ドキュメントから呼び出す場合、XE テンプレート文法を使用すると HTML ドキュメントの<head>要素、または<body>要素の終了点のうち、いずれかを宣言位置として選択することができます。しかし、<body>要素の終了点ではなく、<body>要素内の希望する位置に正確に挿入するには、挿入したい位置に下記のように HTML 標準文法で宣言を行います。

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

別途の JavaScript ファイルを作成せずに HTML ドキュメントにて直接 JavaScript コードを作成するには、下記のように宣言します。

```
<body>
    ...
    <script type="text/javascript">
    // <![CDATA[
        JavaScript コードをここに作成します。
    // ]]>
    </script>
    ...
</body>
```

> **参考**
>
> JavaScript コードの開始点と終了点に<![CDATA[...]]>を宣言することは、JavaScript コードに含まれた文字が HTML コードで解釈されることを防止するためのものです。<![CDATA[...]]>を宣言しなければ、山括弧(<, >)と各種 JavaScript 演算子が文字通りに解釈されず、HTML タグが開始されたり、HTML エンティティが開始されるものと誤解される恐れがあります。
> <![CDATA[...]]>宣言は、JavaScript文法でないため、宣言されたラインを一行コメントで処理します。

2.3.4 HTML 解釈以後の jQuery 実行

XE で作成されたすべてのドキュメントは、jQuery ライブラリを参照するため、<script>要素内にて jQuery 文法を使用できます。しかし、HTML ドキュメントの解釈を完了する前に JavaScript が実行された場合、HTML 構造に依存している JavaScript 構文でエラーが発生する恐れがあります。このようなエラーを防止するために、jQuery ドキュメントの解釈時点と実行時点にそれぞれ異なる値を指定することができます。

下記の方法で jQuery 関数を宣言すると、HTML ドキュメントロード終了後に jQuery を実行することができます。

```
<script type="text/javascript">
// <![CDATA[
    jQuery(function($){
        jQuery 文法をここに作成します。
    });
// ]]>
</script>
```

> **注意**
>
> jQuery 文法において、「jQuery」は、ドル記号($)に置き換えて表記できます。しかし、XE は$記号を使用する他の JavaScript ライブラリとの衝突を避けるために、jQuery 関数の開始点だけは$記号への置き換えを許可しません。したがって、XE では jQuery 関数を最初に作成するときに jQuery(function($){ ... })と作成しなければなりません。

2.3.5 jQuery 動作方式の実習

jQuery の動作方式を一言で説明すると、「選択」と「実行」といえます。これを、もう少し詳しく解説すると、「HTML 要素を選択し、必要な時点に動作を実行する」の意味になります。

下記の簡単な例題を通じて実習してみましょう。

```
<head>
    <style type="text/css">
    #help { display:none; }
    </style>
    <script type="text/javascript" src="jquery.js"></script>
</head>
<body>
    <a href="#help" class="helpBtn">ヘルプ</a>
    <p id="help">この内容は、CSS 宣言により画面に表示されません。<p>
    <script type="text/javascript">
    // <![CDATA[
    jQuery(function($){
        $('a.helpBtn').click(function(){ // HTML 要素を選択し
            $('p#help').css('display','block'); //  動作を実行します。
        });
    });
    // ]]>
    </script>
</body>
```

上記コードは、p#help が、最初は display:none 状態になっていますが、ユーザーが a.helpBtn という要素をクリックして display:block へと状態の変わる動作を実装したものです。jQuery 関数内で HTML 要素を「選択」し、ユーザーのイベントを待った後、選択した要素の CSS 変更を「実行」しています。

特定 HTML 要素を画面に非表示にしたり、表示する動作は、非常に頻繁に発生するため jQuery はこのような動作をより簡単に行えるように show()、hide()という組み込み関数を提供します。上記コードは、下記のように、より簡単に作成することができます。

```
<head>
    <style type="text/css">
    #help { display:none; }
    </style>
    <script type="text/javascript" src="jquery.js"></script>
</head>
<body>
    <a href="#help" class="helpBtn">ヘルプ</a>
    <p id="help">この内容は、CSS 宣言により画面に表示されません。<p>
    <script type="text/javascript">
    // <![CDATA[
        jQuery(function($){
            $('a.helpBtn').click(function(){ // HTML 要素を選択し
                $('p#help').show(); // show()関数を実行します。
            });
        });
    // ]]>
    </script>
</body>
```

下記の例題は、a.helpBtn リンクを一回クリックすると表示し、もう一回クリックすると非表示にするトグル関数を実行するよう、動作を改善したものです。

```
<head>
    <style type="text/css">
    #help { display:none; }
    </style>
    <script type="text/javascript" src="jquery.js"></script>
</head>
```

```
<body>
    <a href="#help" class="helpBtn">ヘルプ</a>
    <p id="help">この内容は、CSS 宣言により画面に表示されません。<p>
    <script type="text/javascript">
    // <![CDATA[
        jQuery(function($){
            $('a.helpBtn').click(function(){ // HTML 要素を選択し
                $('p#help').toggle(); // toggle()関数を実行します。
            });
        });
    // ]]>
    </script>
</body>
```

これまで jQuery の使用方法と簡単な動作方式について説明しました。jQuery マニュアルや jQuery 関連書を参考にすると、より多様な方法で HTML 要素を選択し、より多くの動作を実装することができます。

参考

jQuery を使用して HTML を選択し、動作を実行する詳細な方法については、jQuery 公式 Web サイト(http://jquery.com/)を参照してください。

2.4 XE テンプレート文法

XE テンプレート文法は、サーバー側にて PHP 文法で解釈されます。DB から情報を取り出し、その情報をユーザー画面に表示します。XE テンプレート文法を使用すると、モジュール、またはウィジェットの機能や内容を許可範囲内で拡張したり縮小したりできます。

XE テンプレート文法を適用し、XE スキンを作成する方法には、以下の4つの方法があります。

- HTML 注釈<!--...-->内側に作成する方法。例) <!--@if(...)-->...<!--@end-->
- 仮想の<block>要素内側に作成する方法。例) <block>...</block>
- HTML 要素に直接作成する方法。例) ...
- 注釈や要素に依存せずに作成する方法。例) {$content}内容変数でデータを表示する。

> **参考**
>
> <block>要素は、HTML 標準の要素ではなく、XE core 1.4.4 より新しく追加された仮想の要素です。HTML 要素の形式を借りて使用してはいますが、制御文を実行するのみで、実際には画面に要素が表示されることはありません。cond 属性もまた、XE core 1.4.4 より追加された仮想の属性として、条件文の役割をします。

2.4.1 変数

変数は、プログラムで事前に宣言しておいた内容を表示したり、ユーザー入力を再度画面に表示するときに必要です。ユーザー画面に表示されるほとんどの内容は、すべて変数で表示されているといっても過言ではありません。

変数の使用形式は、下記の通りです。

基本形式

```
{$string}
```

変数の基本形式は、中括弧({ })内に変数名を書き、変数名の前にドル記号($)を付けます。変数名は、「文字列(string)」で作成しなければならず、事前に宣言された名前のみ使用できます。

変数の中の変数

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

変数は、別の変数を含めている場合がありますが、このような変数を使用するには、「ハイフンと山括弧(->)」を使用して変数と変数をつなげます。

変数の中の関数

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

変数に関数をつなげるには、「ハイフンと山括弧(->)」を使用します。関数名の後ろには、常に小括弧(( ))が付きます。小括弧には、関数の内容として文字列(string)や整数(integer)を含めることができます。関数名には、事前に宣言された名前しか使用できません。

2.4.2 XE core 変数

「XE core 変数」とは、XE core で使用する変数のことをいいます。XE core 変数は、いろんなモジュールで広く使用できます。特定のモジュールでしか使用できない変数もありますが、このような変数は、「XE core 変数」ではなくただ、「変数」といいます。XE core 変数について以下に示します。

表 2-4 XE core 変数

変数	説明
$is_logged	ユーザーのログイン有無を確認。 <!--@if($is_logged)-->Welcome!<!--@end--> Welcome!
$current_url	現在ページの URL
$request_uri	XE core インストール URL
$logged_info	ログインユーザーに自身の会員情報を表示
$logged_info->member_srl	ログインユーザーの固有番号
$logged_info->user_id	ログインユーザーの ID
$logged_info->email_address	ログインユーザーのメールアドレス
$logged_info->email_id	ログインユーザーメール ID
$logged_info->email_host	ログインユーザーメールホスト
$logged_info->user_name	ログインユーザー名
$logged_info->nick_name	ログインユーザーニックネーム
$logged_info->homepage	ログインユーザーホームページ
$logged_info->blog	ログインユーザーブログ
$logged_info->birthday	ログインユーザー生年月日(YYYYMMDD)
$logged_info->profile_image	ログインユーザープロフィールイメージ
$logged_info->image_name	ログインユーザーの名前のイメージパス
$logged_info->image_mark	ログインユーザーのグループイメージパス
$logged_info->signature	ログインユーザー署名
$logged_info->group_list	ログインユーザー入会グループリスト
$logged_info->is_admin	ログインユーザーが管理者であるかの確認
$logged_info->is_site_admin	ログインユーザーが仮想サイト管理者であるかの確認
$module_info	モジュール情報として、現在のモジュールの情報を表示
$module_info->module	モジュール名
$module_info->use_mobile	モジュールモバイルスキンの使用要否
$module_info->mid	モジュール ID
$module_info->skin	モジュールスキン名
$module_info->mskin	モジュールモバイルスキン名
$module_info->browser_title	モジュールドキュメントタイトル
$module_info->string	モジュールで作られた拡張変数

2.4.3 条件文

条件文は、与えられた条件にしたがって必要な内容を文脈にあわせて表示したり、表示してはいけないときに必要です。条件文は、「if, elseif, else, end」と「条件式」で構成されます。if 文で開始したら、必ず end 文で閉じ、条件文の

終了を宣言しなければなりません。条件式の内容は、PHP で解釈されるため、PHP で使用できるいろんな演算子(&&, ||, ==, ! など)を使用することもできます。

下記は、「Welcome XE！」という文章を表現する様々な条件文の使用方法です。

表 2-5 条件文の使用例

条件文	説明	備考
`<!--@if(条件式)-->` `<p>Welcome XE!</p>` `<!--@end-->`	条件式が真であれば、含まれている内容を表示	
`<block cond="条件式">` `<p>Welcome XE!</p>` `</block>`	条件式が真であれば、含まれている内容を表示	XE core 1.4.4 以上
`<p cond="条件式">` `Welcome XE!` `</p>`	条件式が真であれば、<p>要素と共に含まれている内容を表示	XE core 1.4.4 以上
`<p attr="value"|cond="条件式">` `Welcome XE!` `</p>`	<p>要素は、無条件で表示、条件式が真であれば、attr=″value″属性と値を表示	XE core 1.4.4 以上

参考

<block>要素は、HTML 標準の要素でなく、XE core 1.4.4 より新しく追加された仮想の要素です。HTML 要素の形式を借りて使用してはいますが、制御文を実行するのみで、実際には画面に要素が表示されることはありません。cond 属性もまた、HTML 標準ではなく、XE core 1.4.4 より追加されたテンプレート文法のひとつです。cond は、条件式を表現するときに使用する仮想の属性です。

if、elseif、else 文を同時に使用すると、真偽のみでなく、いろんな条件を与えて、条件によって異なる結果を表示させることができます。

```
<!--@if(条件 A)-->
    <p>条件 A を満たせば、この文章を表示する。</p>
<!--@elseif(条件 B)-->
    <p>条件 A は満たせられず、条件 B を満たせば、この文章を表示する。</p>
<!--@else-->
    <p>条件 A、条件 B を同時に満たせられなかったら、この文章を表示する。</p>
<!--@end-->
```

上記の条件式を、XE core 1.4.4 より新しく追加された「cond」条件式に変えると、下記のようにもっと簡単に表現することができます。

```
<p cond=″条件 A″> 条件 A を満たせば、この文章を表示する。</p>
<p cond=″条件 B″> 条件 B を満たせば、この文章を表示する。</p>
<p cond=″!条件 A && !条件 B″> 条件 A、条件 B を同時に満たせられなかったら、この文章を表示する。</p>
```

2.4.4 繰り返し文

与えられた条件にしたがって必要な内容を繰り返して表示しなければならない場合、繰り返し文が必要です。繰り返し文は「foreach, end」と「条件式」で構成されています。foreach 文で開始したら、必ず end 文で閉じ、繰り返し文の終了を宣言します。

表 2-6 繰り返し文の使用例

繰り返し文	説明	備考
<!--@foreach(変数名 as $val)--> <tr>...</tr> <!--@end-->	$key 値なしで<tr>…</tr>繰り返し	
<block loop=" 変数名 =>$val"> <tr>...</tr> </block>	$key 値なしで<tr>…</tr>繰り返し	XE core 1.4.4 以上
<tr loop=" 変数名 =>$val">...</tr>	$key 値なしで<tr>…</tr>繰り返し	XE core 1.4.4 以上
<!--@foreach(変数名 as $key => $val)--> <tr>...</tr> <!--@end-->	$key 値を含み<tr>…</tr>繰り返し	
<block loop=" 変数名 =>$key, $val"> <tr>...</tr> </block>	$key 値を含み<tr>…</tr>繰り返し	XE core 1.4.4 以上
<tr loop=" 変数名 =>$key,$val">...</tr>	$key 値を含み<tr>…</tr>繰り返し	XE core 1.4.4 以上
<!--@foreach($i=0;$i<100;$i++)--> <tr>...</tr> <!--@end-->	初期値 0 から開始し、<tr>…</tr>100 回繰り返し	
<block loop="$i=0;$i<100;$i++"> <tr>...</tr> </block>	初期値 0 から開始し、<tr>…</tr>100 回繰り返し	XE core 1.4.4 以上
<tr loop="$i=0;$i<100;$i++">...</tr>	初期値 0 から開始し、<tr>…</tr>100 回繰り返し	XE core 1.4.4 以上
<!--@while(条件文)--> … <!--@end-->	条件文が、真であれば、 … を繰り返す(条件文が偽になれないなら、無限ループエラーが発生し得る)	

> **参考**
>
> loop 属性は、HTML 標準の属性でなく、XE core 1.4.4 より新しく追加されたテンプレート文法のひとつです。loop は、foreach 文の条件式を表現するときに使用する仮想の属性です。

2.4.5 簡単な PHP 文の使用

XE スキンファイルでは、PHP 文法を使用することはできませんが、下記のように中括弧内にアットマーク(@)を含めて簡単な PHP 文を使用することができます。

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

PHP 文を使用する際には、一行に一文ずつ作成しなければなりません。たとえば、下記の文は、PHP では問題ありませんが、XE では致命的なエラーを発生させる恐れがあれます。一行に複数の文を作成しているからです。

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

上記の例文は、下記のように一行に一文ずつ作成しなければなりません。

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

2.4.6 include 文

スキン製作の際に、複数のページに渡って繰り返されるコンテンツブロックがある場合は、別途のファイルで管理すると便利です。同じ修正が複数ページに渡る場合、一括適用できるからです。include は、別途のファイルを現在のページに呼び出すコマンドです。

表 2-7 include 文の使用例

Include 文	説明	備考
`<!--#include("header.html")-->`	header.html をインクルードする	
`<include target="header.html" />`	header.html をインクルードする	XE core 1.4.4 以上

> **参考**
>
> <include />要素は、HTML 標準の要素でなく XE core 1.4.4 より新しく追加された仮想の要素です。HTML 要素の形式を借りて使用してはいますが、include 文を実行するのみで、実際には画面に<include />要素が表示されることはありません。target 属性もまた、XE core 1.4.4 より新しく追加されたテンプレート文法のひとつです。target 属性にはインクルードするファイルのパスを作成します。

2.4.7 CSS ファイル参照

CSS ファイルを HTML ドキュメントで参照する方法について説明します。XE core 1.4.4 より複数の CSS ファイルをひとつのファイルに統合する optimizer 機能が非対応となったため、ひとつのモジュールを製作する際には CSS ファイルを分離せずに、できるだけひとつのファイルにまとめて作成することを推奨します。

CSS 参照

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

HTML ドキュメントの<head>要素にて xe.css ファイルを参照します。<load />要素は、XE core 1.4.4 より使用できます。結果コードは、下記の通りです。

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

CSS ファイルは、HTML ドキュメントの<head>要素でのみ参照可能です。<body>要素で別途の CSS ファイルを参照すると HTML 文法エラーが発生します。

media 指定

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

CSS ファイルの対象となるメディアを選択して指定できます。<load />要素と media 属性は、XE core 1.4.4 より使用できます。結果コードは、下記の通りです。

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

media 属性の値にコンマを用いて複数の値を指定することができます。基本値は all で、使用可能な値は下記表の通りです。

表 2-8 media 属性値

属性値	説明
all	media 未指定の場合の基本値。すべての表示装置に対応。
aural	音声表示装置(スクリーンリーダー)
braille	点字表示装置
embossed	点字印刷装置
handheld	モバイル装置
print	プリンター
projection	投射装置
screen	スクリーン(モニター)
tty	テレタイプライター(Tele Type Writer)
tv	テレビ

> **参考**
> CSS media 属性を指定するときは、意図した装置が CSS 標準を遵守しているか、必ず確認しなければなりません。非対応装置の場合、指定した media 属性は適用されません。

IE 条件付き注釈の使用

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

targetie 属性を使用すると、IE の特定バージョンでのみ CSS ファイルを解釈できるよう、条件付き注釈で表示します。<load />要素と targetie 属性は、XE core 1.4.4 より使用可能です。結果コードは、下記の通りです。

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

このコードは、すべてのブラウザで注釈処理しますが、IE 6 ブラウザは注釈で処理せずに解釈します。

CSS ファイル参照宣言順の変更

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

index 属性を使用すると、CSS ファイルの宣言順を変更することができます。<load />要素と index 属性は、XE core 1.4.4 より使用できます。

index 属性の値には正の整数、または負の整数を使用できますが、負の整数を使用すると早く、正の整数を使用すると宣言は遅く宣言されます。index="-1"値を指定すると、他の CSS ファイルより一行速いところから宣言されます。

CSS ファイルは、同じのコマンドが衝突した場合、後で宣言された値が優先になるため、優先順位を高くする場合は、後で宣言してください。

2.4.8 JS ファイルの参照

JS ファイルを HTML ドキュメントで参照する方法について説明します。XE core 1.4.4 より複数の JS ファイルをひとつのファイルに統合する optimizer 機能が非対応となったため、ひとつのモジュールを製作する際には JS ファイルを分離せずに、できるだけひとつのファイルで作成することを推奨します。

JS ファイルを呼び出せるのは、HTML ドキュメントの<head>要素、または<body>要素のみです。<head>要素、または<body>要素以外の要素で JS ファイルを参照すると、HTML 文法エラーが発生し、JS ファイルを解釈できないブラウザがありえます。

<head>要素から JS ファイルを参照する

下記は、xe.js ファイルを HTML ドキュメントの<head>要素から呼び出す方法を示します。

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

<load />文法は、XE core 1.4.4 より使用可能です。結果コードは、下記の通りです。

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

<body>要素から JS ファイルを参照する

下記は、xe.js ファイルを HTML ドキュメントの<body>要素から呼び出す方法を示します。

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

<load />要素は、対象ファイルを HTML ドキュメントから呼び出します。属性と値は下記の通りです。

- media="all | aural | braille | embossed | handheld | print | projection | screen | tty | tv" – CSS ファイルの対象になるメディアを選択して指定することができます。コンマ(,)を用いて複数の値を指定することができます。基本値は、all で省略する場合は、media="all"です。
- targetie="IE 6 | IE 7 | IE8 | ..." – IE 条件付き注釈を使用して IE ブラウザ範囲内で CSS/JS 適用ブラウザを選択することができます。基本値は空の値で"適用しない"です。
- index="integer" – CSS/JS 参照宣言の位置を変更できます。値は、正の整数と負の整数を使用できます。正の整数は現在の位置より後で宣言され、負の整数は現在の位置より先に宣言されます。基本値は空の値で宣言順を変更せず、基本値としておく場合、他の CSS ファイルより遅く宣言されます。
- type="head | body" – JS 参照宣言位置をドキュメントの<head>要素とするか、<body>要素とするかを決定できます。基本値は、head で省略する場合、JS ファイルはドキュメント<head>要素に含まれます。

<load />文法は、XE core 1.4.4 より使用可能です。<body>要素から JS ファイルを呼び出すと、<body>要素の終了直前に JS ファイル参照が宣言されます。結果コードは下記の通りです。

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

IE 条件付き注釈の使用

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

targetie 属性を使用すると、JS ファイルを IE の特定バージョンでのみ解釈できるよう条件付き注釈で表示します。<load />要素と targetie 属性は、XE core 1.4.4 より使用可能です。結果コードは下記の通りです。

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

このコードは、すべてのブラウザで注釈処理しますが、IE 6 ブラウザは注釈で処理せずに解釈します。

JS ファイル参照宣言順の変更

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

index 属性を使用すると、JS ファイルの宣言順序を変更することができます。<load />要素と index 属性は、XE core 1.4.4 より使用できます。

index 属性の値は正の整数、または負の整数を使用できます。負の整数を使用するとより速く宣言し、正の整数を使用するとより遅く宣言します。index="-1"値を指定すると、他の CSS ファイルより一行速い位置から宣言されます。

2.4.9 XML JS フィルターの適用

XML JS フィルターは、XML 形式で入力項目を定義しておき、フォーム転送の際に自動的に有効性チェックを行う機能です。有効性チェックは、XML ベースで自動変換された JavaScript が行うため、ユーザーは複雑な JavaScript を作成する必要がありません。XML JS フィルターは、有効性チェックを通ったフォームデータをどのようなモジュールのどのようなコマンドで送るかを決める機能もサポートします。

XML JS フィルターを適用する方法は、CSS、JS ファイルを参照する文法と同様です。

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

このように XML JS フィルターを呼び出すと、XML ドキュメントは JS ドキュメントにコンパイルされ、ソースコードで表示されます。XE core 1.4.4 以前では、コンパイルされた JS ファイルを無条件で<head>要素に含めます。

表示結果は下記の通りです。

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

XE core 1.4.4 以降では、コンパイルされた JS ファイルを<body>要素の終了直前に含めます。

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

表示結果は、下記の通りです。

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

<head>要素領域に表示されず、無条件で<body>要素の終了直前に表示されるという点で、XE core 1.4.4 未満とは異なります。

2.4.10 ウィジェットを挿入する

ウィジェットは、XE core、またはモジュールに含まれた情報をユーザーにとって意味のあるかたちに加工して表示するインターフェースの役割をします。Web サイトの初期画面に最新の記事を表示するウィジェットとログインフォームを表示するウィジェットが XE core に含まれています。ウィジェットは、複雑なプログラムロジックが分からなくても、スキン製作者の希望する機能を実装しやすくサポートします。

スキン製作者は、テンプレートコードを一行作成するだけで希望する機能を実装することができます。ウィジェットを挿入するためのテンプレートコードは、<img />要素に widget という属性を含めています。<img />要素は、HTML 標準でもありますが、widget という属性を含めると、XE テンプレート文法に解釈されサーバーで標準 HTML に変換されます。

下記にて、ログインフォームを表示するウィジェット挿入コードを示します。

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

ウィジェットを挿入するためのコードは、XE Admin ページにて、**サイト設定 > ウィジェット**を選択し、**コード生成**機能を使用すると、自動生成されます。詳細については、『XE ユーザーマニュアル』を参照してください。

2.4.11 XE core 1.4.4 からの新しいテンプレート文法

ここでは、XE core 1.4.4 より新しく追加されたテンプレート文法についてのみ述べます。XE core1.4.4 より古いバージョンを使用していて、XE core 1.4.4 以上にアップグレードした場合には、下記表を参照してください。

表 2-9 XE core 1.4.4 より追加されたテンプレート文法

追加されたテンプレート文法	説明
<block cond="条件式"> Welcome XE! </block>	条件式が真であれば、含まれている内容を表示
   Welcome XE! 	条件式が真であれば、 要素と共に含まれている内容を表示
<p attr="value"|cond="条件式"> Welcome XE! 	要素は、無条件で表示するが、条件式が真であれば、attr="value"属性と値を共に表示
<block loop="変数名=>$val"> <tr>...</tr> </block>	$key 値なしで<tr>…</tr>繰り返し
<tr loop=" 変数名 =>$val">...</tr>	$key 値なしで<tr>…</tr> 繰り返し
<block loop=" 変数名 =>$key, $val"> <tr>...</tr> </block>	$keyatai 値を含み<tr>…</tr> 繰り返し
<tr loop=" 変数名 =>$key,$val">...</tr>	$key 値を含み<tr>…</tr> 繰り返し
<block loop="$i=0;$i<100;$i++"> <tr>...</tr> </block>	初期値 0 から開始し<tr>…</tr> 100 回繰り返し
<tr loop="$i=0;$i<100;$i++">...</tr>	初期値 0 から開始し<tr>…</tr> 100 回繰り返し
<include target="header.html" />	header.html を含む(include)
<load target="xxx.xxx" />	CSS/JS/SML JS フィルターファイルを<head>に含む
<load target="xxx.xxx" type="body" />	JS/XML JS フィルターファイルをドキュメント<body>に含む
<unload target="xxx.xxx" />	パスの一致する CSS/JS/XML JS フィルターファイルを除く

3. レイアウトスキンをつくる

この章では、例題レイアウトスキンを使用してレイアウトスキンを製作し、適用する方法について説明します。

3.1 レイアウトスキンとは

XE は、レイアウトとコンテンツが分離されています。レイアウトは、XE コンテンツを収める構造体、または骨組みです。一般の Web サイトは、ヘッダー、グローバルナビゲーション、ローカルナビゲーション、コンテンツ、フッターで構成されていますが、このような要素の画面配置を決定するのが、このレイアウトスキンの役割です。掲示板やウィキのような特定のコンテンツをユーザーに表示する際に、掲示板やウィキ以外のコンテンツを表示する必要がないなら、レイアウトスキンは使用しなくても構いません。しかし、掲示板やウィキのようなコンテンツ以外に、Web サイトを一貫性をもって探索できるよう、ヘッダー、グローバルナビゲーション、ローカルナビゲーション、フッターエリアなどを配置すべきならレイアウトスキンが必ず必要となります。

下記図にて、レイアウトスキンを使用した一般的な画面構成をあらわします。

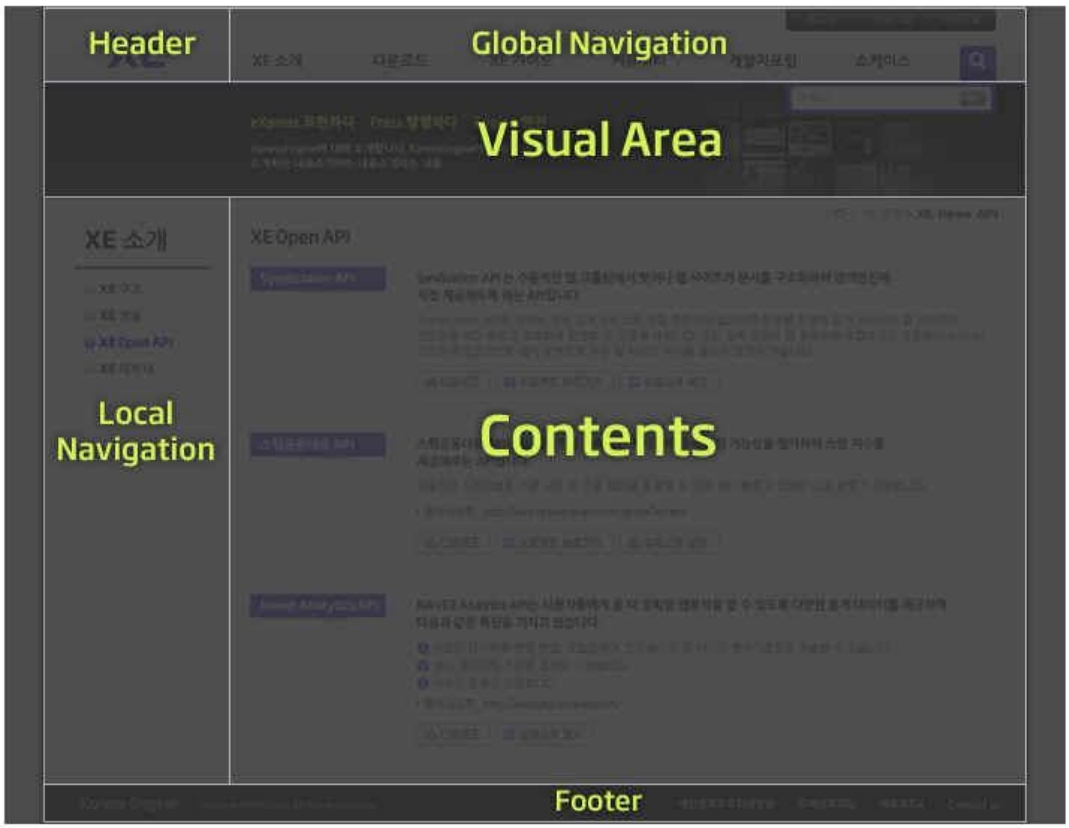


図 3-1 レイアウトスキンを使用した一般的な画面構成

XE core は、ひとつ以上のレイアウトスキンを含めています。基本的に含まれているレイアウトスキンは、XE core インストール先の/layouts/にあります。スキン製作者は、XE core に含まれている基本レイアウトスキンを使用したり、新しく製作することもできます。

3.2 例題レイアウトスキンのダウンロード

新しいレイアウトスキンを製作するには、レイアウトスキンの構造に合わせてディレクトリとファイルを作成しなければなりません。本書では、スキン製作者の便宜のために例題レイアウトスキンを提供します。例題レイアウトスキンでは、レイアウトスキンを製作する際に必要なディレクトリ構造とファイルを備えており、例題レイアウトスキンを使って思い通りのスキンに便利に修正することができます。

例題レイアウトスキンのダウンロード先は、下記の通りです。

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

3.3 レイアウトスキンの位置とディレクトリ構造

3.3.1 レイアウトスキンの位置確認

/xe/というユーザー定義ディレクトリに XE core をインストールした場合、レイアウトスキンのディレクトリは、下記の場所になります。

```
/xe/layouts/
```

ダウンロードした例題レイアウトスキン(user_layout)を解凍し、/xe/layouts/配下にコピーします。

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

> **参考**
>
> レイアウトスキンをローカル PC で修正する際には、修正内容を Web サイトのレイアウトスキンディレクトリに反映しなければなりません。

3.3.2 レイアウトスキンのディレクトリ構造

レイアウトスキンを製作する際に必ず遵守すべきディレクトリ構造があります。この構造を維持しなければ、レイアウトスキンを製作できません。

例題レイアウトスキン(user_layout)の構造を以下に示します。

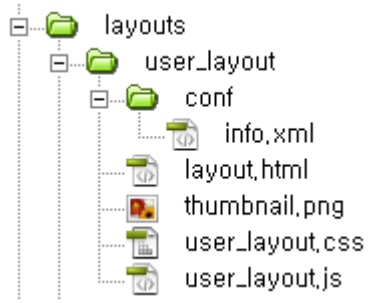


図 3-2 レイアウトスキンのディレクトリ構造

info.xml

info.xml には、レイアウトスキンの基本情報が含まれます。管理者画面に表示する説明とオプションを提供します。info.xml は、必ず下記のパスに存在していなければなりません。なお、conf ディレクトリ名と info.xml ファイル名は、変更できません。

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

layout.html

layout.html には、レイアウトスキンの HTML と CSS、JS 参照情報が含まれます。layout.html ファイル名は、変更できません。

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

thumbnail.png

thumbnail.png は、レイアウトプレビュー用のイメージファイルです。横幅 180 ピクセル、高さ 120 ピクセル以上で製作しておくと、管理者画面にプレビューイメージとして表示されます。“thumbnail.png”というファイル名は、変更できません。

/xe/layouts/user_layout/**thumbnail.png**

CSS、JS、IMG

CSS、JS、IMG ファイルはレイアウトスキンの必須要素ではありません。必要によって作成して使用します。conf ディレクトリを除き、どの場所にあっても構いません。管理すべきファイルの数が多くなったら、/user_layout/ディレクトリ配下に css、js、img と同じ名前のディレクトリを追加作成して管理することができます。

3.4 レイアウトスキン情報作成

レイアウトスキン情報は、info.xml にて作成します。info.xml の基本構造について以下に示します。

```
<?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">デザインの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>
```

上記コードの説明を以下に示します。

コード	説明
`<?xml version="1.0" encoding="UTF-8"?>`	XML ドキュメント形式の宣言
`<layout version="0.2">`	レイアウト情報についてのドキュメントであることを宣言。version には XE core でサポートするバージョンを表示しなければなりません。XE core 1.4.4.2 基準での対応レイアウトバージョンは、0.2 です。
`<title xml:lang="ko">...</title>`	レイアウト名
`<description xml:lang="ko">...</description>`	レイアウト説明
`<version>...</version>`	レイアウトスキンのバージョン
`<date>YYYY-MM-DD</date>`	レイアウトスキンの製作日。年-月-日(YYYY-MM-DD)形式で作成します。
`<author email_address="..." link="...">` `<name xml:lang="ko">...</name>` `</author>`	レイアウトスキン製作者情報。メールアドレス、Web サイトアドレス、製作者名を作成します。
`<menus>` `<menu name="main_menu" maxdepth="2" default="true">` `<title xml:lang="ko">...</title>` `</menu>`	XE Admin ページのコントロールパネルにて、作成したメニューとつなげることができます。<menus>...</menus>要素内に 2 つ以上の<menu>...</menu>を作成することができます。

コード	説明
</menus>	

このほかにも、info.xml にはユーザー定義が可能ないろんなタイプの変数を、<extra_vars>要素内に追加的に含めることができます。下記は、スキン製作者が select、image、text、textarea タイプのデータ入力を受け、変数として使用できるよう拡張した例題です。

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

    ...


    <extra_vars>
        <var name="colorset" type="select">
            <title xml:lang="ko">カラーセット</title>
            <description xml:lang="ko">希望するカラーセットを選択してください。</description>
            <options value="black">
                <title xml:lang="ko">Black (基本)</title>
            </options>
            <options value="white">
                <title xml:lang="ko">White</title>
            </options>
        </var>
        <var name="logo_image" type="image">
            <title xml:lang="ko">ロゴイメージ</title>
            <description xml:lang="ko">ページ上部に表示するロゴイメージを入力してください。</description>
        </var>
        <var name="logo_url" type="text">
            <title xml:lang="ko">ロゴイメージリンク</title>
            <description xml:lang="ko">ロゴイメージをクリックした際に移動する URL を入力してください。</description>
        </var>
        <var name="title_description" type="textarea">
            <title xml:lang="ko">本文上部内容</title>
            <description xml:lang="ko">本文上部に表示されるエリアの内容を入力してください(HTML 使用可)。</description>
        </var>
    </extra_vars>

    ...

</layout>
```

上記コードの説明を以下に示します

コード	説明
<extra_vars>...</extra_vars>	拡張変数コンテナ
<var name="colorset" type="select">...</var>	select タイプの拡張変数。スキン製作者は、{$layout_info->colorset}形式で表示可。
<var name="logo_image" type="image">...</var>	Image タイプの拡張変数。スキン製作者は、{$layout_info->image}形式で表示可。
<var name="logo_url" type="text">...</var>	text タイプの拡張変数。スキン製作者は、{$layout_info->logo_url}形式で表示可。
<var name="title_description" type="textarea">...</var>	textarea タイプの拡張変数。スキン製作者は、{$layout_info->title_description}形式で表示可。

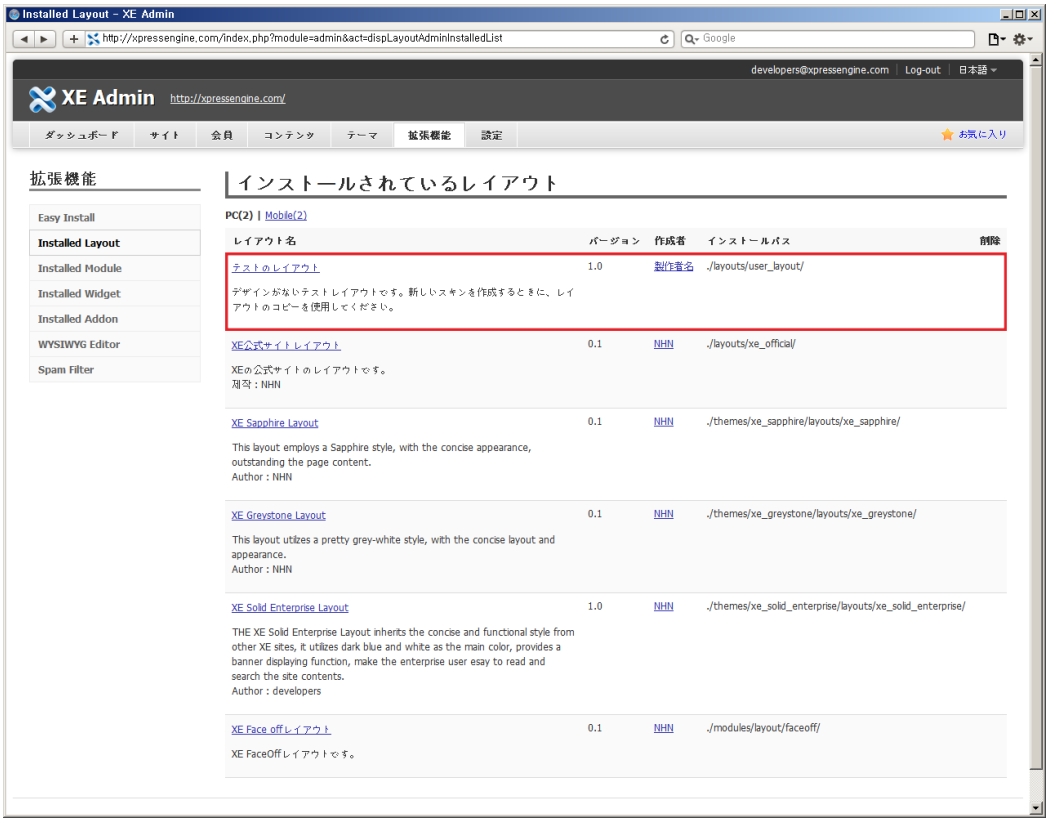
info.xml に追加した拡張変数は、レイアウト生成以降、**レイアウト設定**ページに表示されます。ウェブサイト管理者が拡張変数になんらかの値を入力すると、その結果をスキンに表示することができます。

3.5 レイアウト作成

ユーザー定義レイアウトの確認

info.xml を正しく作成したかどうかを確認する方法について説明します。

1. XE Admin ページにて**拡張機能 > Installed Layout** を選択します。
2. **テストのレイアウト**が正常に表示されていることを確認します。



メニュー数は、info.xml にて<menu>要素を1つしか含めていないため、**1** で表示されます。<menu>要素を複数作成すると、**メニュー数**は、<menu>要素の数のぶんだけ増えます。

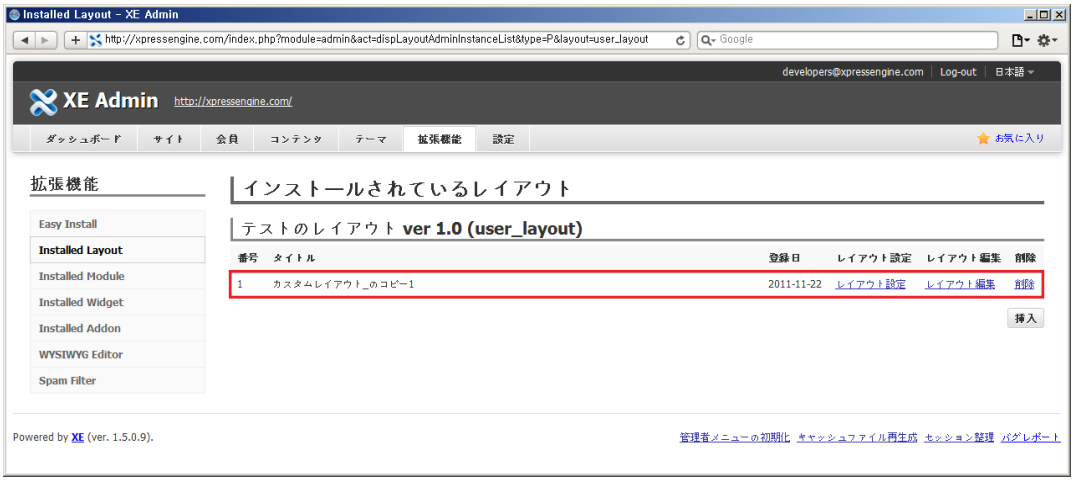
レイアウトコピーの作成

Installed Layout である「テストのレイアウト」にて、レイアウトの「コピー」を生成すると、使用可能な状態になります。ユーザーは、user_layout というコピーを複数、繰り返し作成することもできます。

レイアウトコピーを作成する方法について説明します。

1. **テストのレイアウト**ページを開き、**挿入**をクリックします。
2. **タイトル**にコピーの名前を入力し、**挿入**をクリックします。**タイトル**は、user_layout の別のコピーと区別できるように入力しなければなりません。この例題では、**テストのレイアウト**のコピー名として、*カスタムレイアウト__のコピー1* を入力しました。

3. **テストのレイアウト**ページにて下記図のように、**カスタムレイアウト_のコピー1** が作成されていることを確認します。



これで、このレイアウトを使用するモジュールの設定画面にて、**カスタムレイアウト_のコピー1** を選択できるようになりました。

参考

layout.html を作成しない状態で、このレイアウトを適用したモジュールページへアクセスすると"Err : "./layouts/user_layout/layout.html" template file does not exist."というエラーメッセージが画面に表示されます。

3.6 レイアウトスキン作成

レイアウトスキンの内容は、layout.html にて作成します。

XE は、DOCTYPE、html、head、body 要素を XE core で自動表示するという特徴があります。したがって、スキン製作者は、この要素をスキンファイルで作成する必要がありません。スキンファイルにこの要素を含めると、重畳文法として処理され、HTML エラーが発生したり、画面が非正常表示されたりする現象が発生します。

レイアウトスキンを適用した Web ページの基本構造を以下に示します。

```
<!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" />
    <!--ここに含む内容は、layout.HTMLドキュメント、または管理者モードで作成される。-->
</head>
<body>
    <!-- layout.HTMLドキュメントのコードと内容は、ここに表示される。 -->
</body>
</html>
```

layout.html ページにて作成する HTML コードと内容は、すべて<body>要素の内側に HTML コードとして表示されます。したがって、スキン製作者は、<body>要素内側で使用できる有効な HTML 文法で layout.html を作成しなければなりません。<head>要素に含むべき内容は、XE テンプレート文法を使用して<body>要素内部で作成できます。

レイアウトスキンのドキュメント構造

例題レイアウトスキンは、header、gnb(global navigation bar)、body、lnb(local navigation bar)、content、footer で構成されています。したがって、layout.html のドキュメント構造は下記のようになります。

```
<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>
```

HTML で作成された 要素にクラス名を指定した理由は、CSS ファイルで HTML 要素を選択し、希望する配置に実装できるようにするためです。

{$content}変数で本文を表示する

layout.html は、まだユーザー画面では確認できません。まだこのレイアウトを選択したモジュールがないからです。レイアウトスキンページは、自らは存在できず、特定のモジュールで使用されたときに該当モジュールのページにアクセスして確認することができます。レイアウトスキンは、モジュールに依存します。

例題レイアウトスキンをユーザー画面で確認するには、特定モジュール(ページ、掲示板、ウィキなど)を1つ作成し、このレイアウトスキンとつなげなければなりません。その前にスキン製作者は、レイアウトスキンとつながっているモジュール(ページ、掲示板、ウィキなど)にて、レイアウトスキンを画面に正常表示できるよう、「内容変数」を作成しなければなりません。

例題レイアウトスキンでは、下記のようにレイアウトの本文エリアに{$content}変数を用いて「内容変数」を作成しました。

```
<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}は、レイアウトスキンで使用できる最も重要な変数で、現在のレイアウトスキンがどの種類のモジュールとつながっていても、つながっているモジュールの内容を本文エリアに表示します。つながるモジュールによって静的なページになることも、動的な掲示板やウィキページになることもあります。

XE において、モジュールとレイアウトの関係は、「レイアウトスキンにいろんな種類のモジュールを搭載する」のではなく、「様々なモジュールに、レイアウトスキンをつなげる」という意味合いの関係です。XE で作成したモジュールページには固有の URL がありますが、レイアウトスキンには URL がなく、「モジュール」につながっている状況でしか表示されないということを記憶しておいてください。

グローバルメニューを表示する

.gnb 要素内部にレイアウトとつながっているグローバルメニューを表示することができます。XE Admin ページにて、レイアウトとメニューをつなげると、レイアウトは、常につながっているメニューを画面に表示しますが、このときつながっているメニューを表示できるよう、レイアウトスキンで事前にコードを作成しておかなければなりません。

例題レイアウトスキンでは、下記のように、つながっているメニューを画面に表示するコードを作成しました。メニューを最大で 2 階層まで表示します。

```
<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>
```

上記コードで使用されているテンプレート文法と変数は、下記の通りです。

テンプレート文法/変数	説明	構文
`loop="$global_menu->list=>$key1,$val1"`	つながっているメニューリストがある場合、表示	繰り返し文
`cond="$val1['list']"`	つながっているメニューが下位リストを含んでいる	条件文

テンプレート文法/変数	説明	構文
	場合、表示	
loop="$val1['list']=>$key2,$val2"	つながっているメニューが下位リストを含んでいる場合、表示	繰り返し文
class="active"|cond="$val1['selected']" class="active"\|cond="$val2['selected']"	つながっているメニュー、またはつながっているメニューの下位リストと現在ページが一致した場合、 要素に class="active"属性を追加	条件文
target="_blank"|cond="$val1['open_window']=='Y'" target="_blank"\|cond="$val2['open_window']=='Y'"	つながっているメニューのリンクオプションが新しいウィンドウで表示であれば、target="_blank"表示	条件文
{$val1['href']} {$val2['href']}	つながっているメニューリンクの URL	変数
{$val1['link']} {$val2['link']}	つながっているメニューリンクのテキスト	変数

ローカルメニューを表示する

.lnb 要素内にレイアウトとつながっているローカルメニューを表示できます。ローカルメニューを表示する方法は、グローバルメニューの表示方法とほぼ同様です。

グローバルメニュー表示とローカルメニュー表示の違いは、ローカルメニューが 1 階層メニューを表示しないという点です。グローバルメニューが .gnb エリアにて、すでに 1~2 階層を表示しているため、ローカルメニューは、現在ページを基準に、関連している 2~3 階層メニューを表示します。

ローカルメニューは、メニューにつながっているページがあるときのみ表示されるため、メニューを作成するときに実際に接続可能なページの URL を入力しなければなりません。

例題レイアウトスキンでは、下記のようにローカルメニュー表示コードを作成しました。

```
<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>
```

上記コードで使用されているテンプレート文法と変数は、下記の通りです。

テンプレート文法/変数	説明	構文
loop="$global_menu->list=>$key1,$val1"	つながっているメニューリストがある場合、表示	繰り返し文
cond="$val1['selected']"	つながっているメニューと現在ページが一致した場合、内容を表示	条件文

テンプレート文法/変数	説明	構文
`cond="$val1['selected'] && $val1['list']"`	つながっているメニューと現在ページが一致し、同時に下位リストを含んでいる場合、表示	
`loop="$val1['list']=>$key2,$val2"` `loop="$val2['list']=>$key3,$val3"`	つながっているメニューが下位リストを含んでいる場合、表示	繰り返し文
`cond="$val2['list']"`	つながっているメニューが下位リストを含んでいる場合、表示	条件文
`class="active"|cond="$val2['selected']"` `class="active"\|cond="$val3['selected']"`	つながっているメニュー、またはつながっているメニュー下位リストと現在ページが一致した場合、 要素にclass="active"属性を追加	条件文
`target="_blank"|cond="$val1['open_window']=='Y'"` `target="_blank"\|cond="$val2['open_window']=='Y'"` `target="_blank"|cond="$val3['open_window']=='Y'"`	つながっているメニューリンクオプションが新しいウィンドウで表示であれば、 target="_blank"表示	条件文
`{$val1['href']}` `{$val2['href']}` `{$val3['href']}`	つながっているメニューリンクの URL	変数
`{$val1['link']}` `{$val2['link']}` `{$val3['link']}`	つながっているメニューリンクのテキスト	変数

統合検索フォームを表示する

Web サイトの統合検索は、検索の対象を特定のモジュールのみに限らず、作成されたすべてのモジュールを検索結果に含める機能です。統合検索機能を使用するには、統合検索フォームを提供しなければなりませんが、統合検索フォームは通常、レイアウトスキンで提供します。統合検索フォームコードは、XE Admin ページの**拡張機能 > Installed Module > 統合検索**にて提供しています。

例題レイアウトスキンでは、下記のように ... 要素に統合検索フォームコード<form class="search">...</form>を追加しました。

```
<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>
```

上記コードで使用されている変数は、下記の通りです。

変数	説明
{getUrl()}	ユーザーの検索語を転送するページの URL
{$vid}	仮想サイト ID
{$mid}	モジュール ID
{$is_keyword}	ユーザーが入力した統合検索キーワード。検索結果ページにてユーザーが入力したキーワードを再表示します。
{$lang->cmd_search}	言語変数。「検索」という単語が変数に割り当てられています。

ログインフォームを表示する

運営する Web サイトが会員制である場合、会員の入会を受け付け、会員がログインできなければなりません。XE core では、すでに入会ページとログインフォームが用意されています。ログインフォームをレイアウトスキンに埋め込むには、ログインフォームを表示したい位置にログインフォーム表示コードを追加します。

例題レイアウトスキンでは、下記のように ... 内部のエリア上部にログインウィジェットを表示するよう、作成しました。

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

3.7 サイトマップ作成

サイトマップとは、サイトのメニューのことです。XE Admin ページにてサイトマップを作成し、レイアウトのコピーとつなげると、レイアウトを表示するときにサイトマップをメニューとして表示することができます。

サイトマップを作成する方法について説明します。

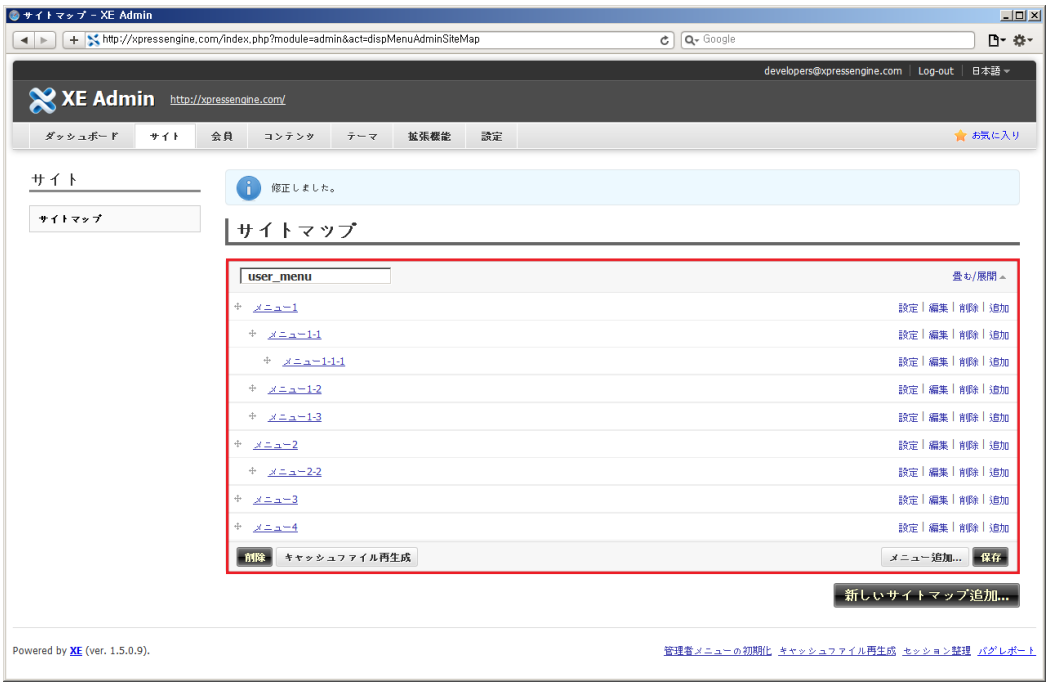
1. XE Admin ページにて、**サイト** > **サイトマップ**を選択します。
2. サイトマップページにて、**新しいサイトマップ追加**をクリックします。

サイトマップでは、複数のセットを追加することはできますが、レイアウトとつなげる際に他のサイトマップと区別できるように**タイトル**を必ず入力しなければなりません。

3. **タイトル**に *user_menu* と入力し、**保存**をクリックします。

4. **user_menu** にて**メニュー追加**をクリックし、ウェブサイトのグローバルメニュー構造を完成させます。詳細なメニュー作成方法については、『XE ユーザーマニュアル』を参照してください。

5. user_menuという名前のサイトマップが生成されたら、**保存**をクリックします。



まだ user_menu というメニューをユーザー画面で確認することはできません。レイアウトにて、メニューを参照しない限りユーザー画面には表示されません。

参考

メニューを作成する際に、**モジュール、または URL** 項目に有効な**モジュール ID**、または**接続 URL** を指定しなければなりません。**モジュール ID**、または**接続 URL** が有効でないとレイアウトにメニューが正常に表示されないことがあります。

3.8 レイアウトにサイトマップをつなげる

user_menu というサイトマップが作成されたため、これでレイアウトからこのサイトマップを参照することができるようになりました。レイアウトに user_menu をつなげる方法について説明します。

1. XE Admin ページにて、**拡張機能 > Installed Layout** を選択します。
2. **テストのレイアウト**項目を開き、**カスタムレイアウト_のコピー1** の**レイアウト設定**をクリックします。

3. **メニュー > グローバルメニュー(global menu)**にて、**user_menu** を選択し**レイアウトの一括適用**を選択後に**保存**をクリックします。

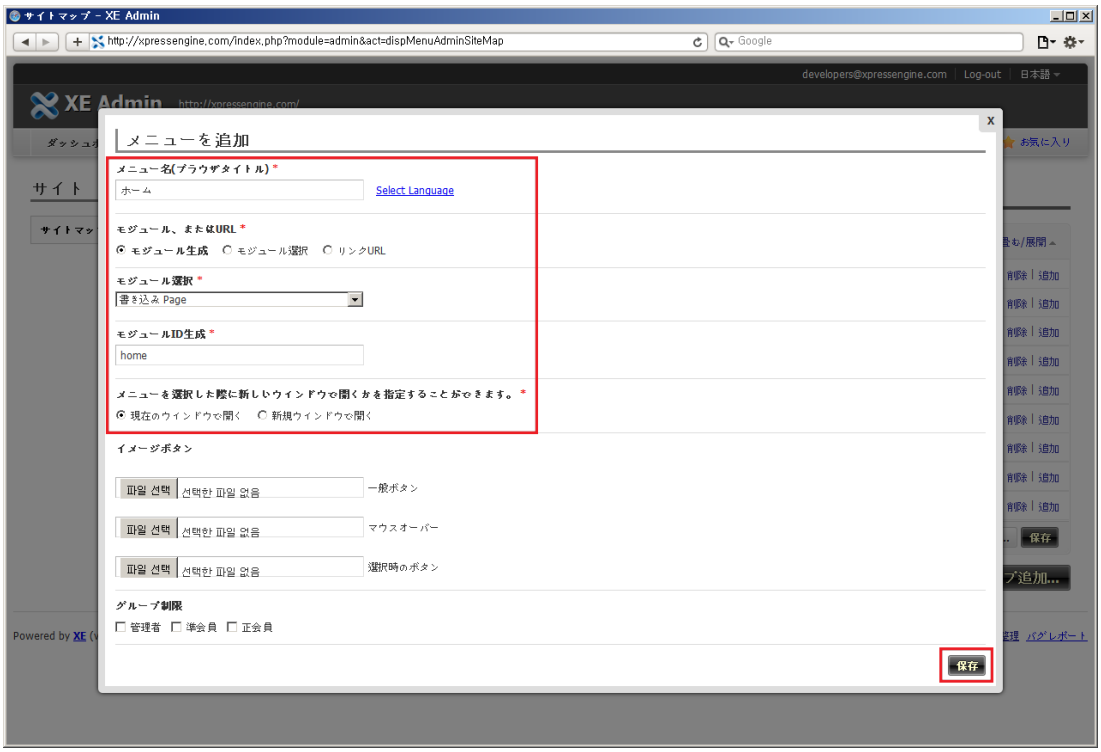
   レイアウトの一括適用を選択すると、メニューでつながっているすべてのモジュールのレイアウトが現在のレイアウトに一括変更されます。以後、**user_menu** は常にこのレイアウトで表示されます。

3.9 ページモジュールにレイアウトをつなげる

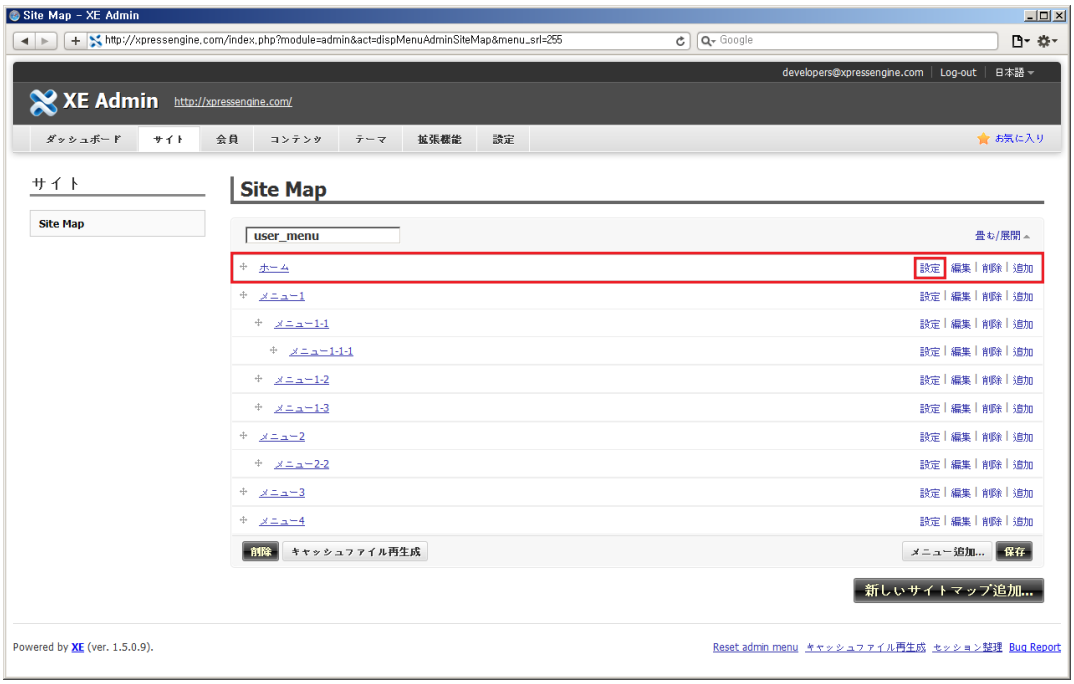
ページモジュールにレイアウトスキンを適用して、例題レイアウトスキンの内容が画面に正常表示されるかどうか確認してみましょう。

ページ作成

1. XE Admin ページにて、**サイト** > **サイトマップ**を選択します。
2. **user_menu** の下の**メニュー追加**をクリックします。
3. 下記図のように **Add Menu** 画面が表示されたら、**メニュー名(ブラウザタイトル)**、**モジュール選択**、**モジュール ID 生成**を設定し、**現在のウィンドウで開く**を選択します。この例題では、次のように設定し、**保存**をクリックします。
   - **メニュー名**: *ホーム*
   - **モジュール選択**: *書き込み Page*
   - **モジュール ID 生成**: *home*



4. **サイトマップ** > **user_menu** に**ホーム**が生成されたことを確認のうえ、**ホーム**の**設定**をクリックして、**Manage of page** 画面へ移動します。



5. **Manage of page** の**レイアウト**項目にて、**カスタムレイアウト_のコピー1(user_layout)**を選択して**保存**をクリックします。

レイアウトに**カスタムレイアウト_のコピー1(user_layout)**項目が表示されない場合は、user_layout のコピーがまだ生成されていないことを意味します。user_layout のコピーを生成する方法については、「レイアウトコピーの作成」を参照してください。

ページ確認

作成されたページにユーザー画面からアクセスすると、レイアウトスキンが適用された結果を確認できます。モジュール名を home としたため、このページのアクセスパスは、下記のようになります。下記のパスにて、example.com は、ユーザーの Web サイトがインストールされているドメインアドレスを意味します。

- mode_rewrite を使用する場合：http://example.com/home/
- mode_rewrite を使用しない場合：http://example.com/?mid=home

home ページを開くと、下記図のような画面が表示されます。

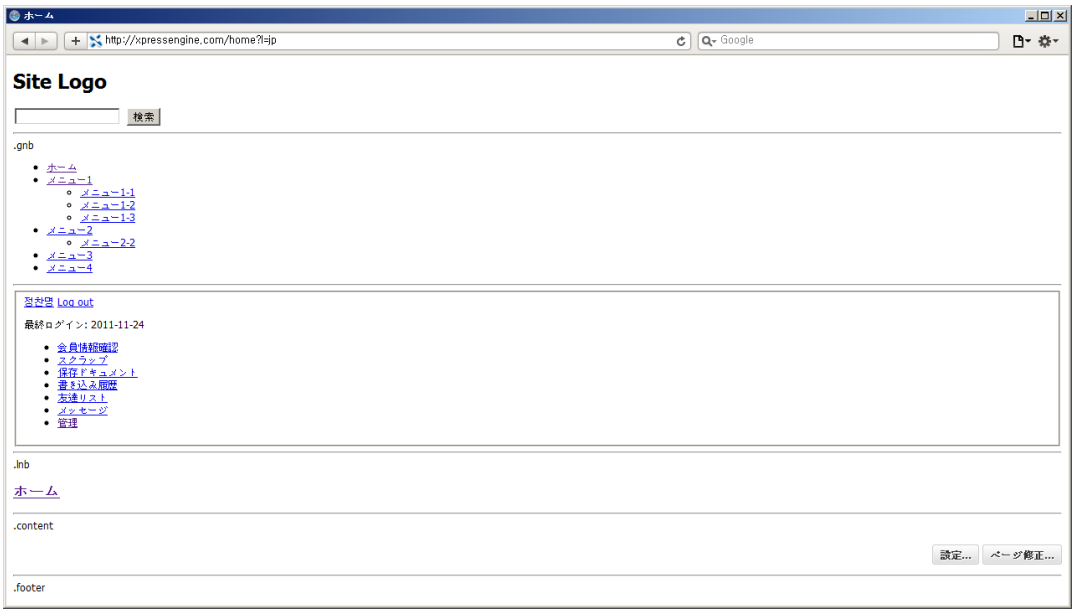


図 3-3 レイアウトスキンが適用されたページ

ページ本文には、まだ内容が何も作成されていないため表示する内容がなく、本文以外のエリアに作成されたレイアウトコンテンツしか見えません。「Site Logo, gnb, lnb, content, footer」というテキストは、スキンファイルである layout.html ドキュメントで作成したものです。このほか、レイアウトスキンに含まれている統合検索フォーム、グローバルメニュー、ログインフォームウィジェット、ローカルメニューが正常に表示されていることが確認できます。Content エリアにはページ本文を編集できるボタンがありますが、これは管理者だけに公開される機能です。このボタンが見えないなら、管理者ログアウト状態か、本文エリアの{$content}変数の作成に間違いがあるということになります。

> **参考**
>
> メニューを作成する際に、**モジュール、または URL** 項目に有効な**モジュール ID**、または**接続 URL** を指定しない限りローカルメニューは正常に表示されません。

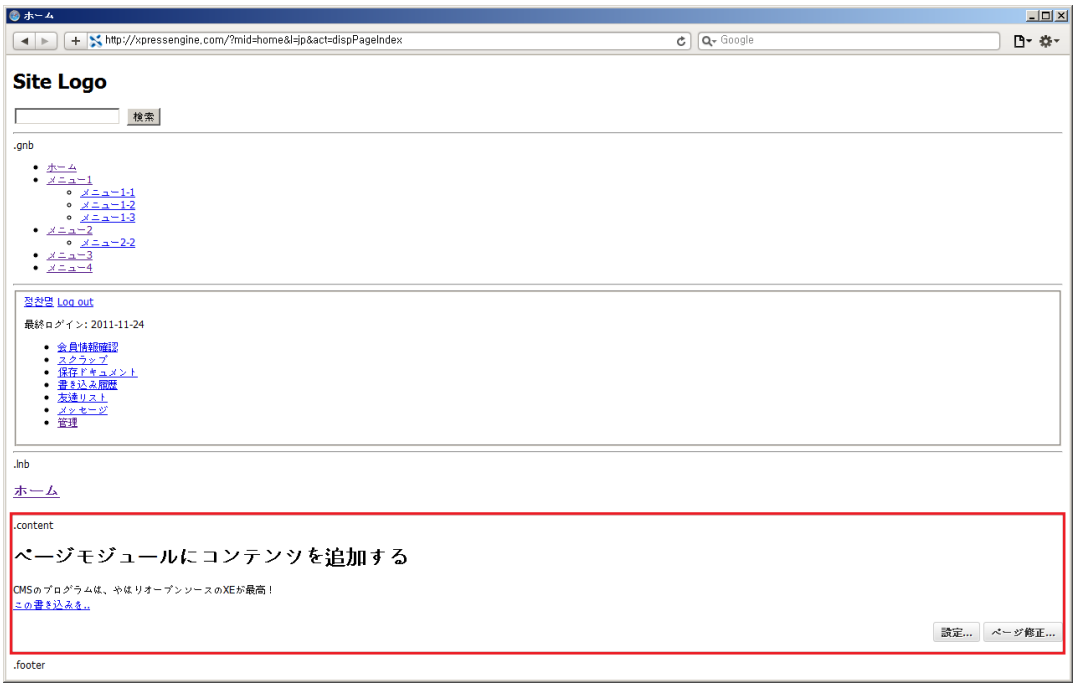
ページ修正

図 3-3 のような画面では、ページに新しい内容を入力できます。ページを修正する方法について説明します。

1. レイアウトスキンが適用されたページにて、画面右下にある**ページ修正**をクリックします。
2. タイトルと本文の内容を追加し、**登録**をクリックします。

追加した内容がページに反映されます。

3. ユーザーに表示される結果画面は、下記図の通りです。content エリアの右下に表示される**設定**、**ページ修正**は、現在ページモジュールに対する編集権限を持つ管理者だけに公開されます。ユーザー画面には表示されません。



まだレイアウトといえる配置ではありません。CSS コードを作成して適切な配置を実装しなければなりません。

3.10 CSS 適用

ここでは、レイアウトスキンに CSS コードを適用する方法について説明します。本書では CSS コードの作成方法については述べません。スキン製作者は、製作意図に合わせて CSS コードを修正して使用します。

1. 例題レイアウトスキンの user_layout.css ファイルでは、下記のように.lnb エリアと.content エリアを左右のカラム形式で配置するよう、作成しました。

```
@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.html にて、下記のように user_layout.css を参照するよう宣言します。

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

詳細な説明は、「CSS ファイル参照」を参照してください。

3. 下記のパスへアクセスし、ページに CSS が正常に適用されたかどうかを確認します。下記のパスにて、「example.com」は、ユーザーWeb サイトがインストールされているドメインアドレスを意味します。
   - mod_rewrite を使用する場合：http://example.com/home/
   - mod_rewrite を使用しない場合：http://example.com/?mid=home

下記図にて、user_layout.css を正常に参照し、画面の構成要素が正しく表示された様子を示します。

図 3-4 CSS が正常に適用されたページ

もし、作成したコードが画面に反映されていないなら、CSS を参照する<load />テンプレート文法が正しいか、または CSS 参照パス(target)が正しく指定されているかどうか確認してください。

3.11 JavaScript 適用

ここでは、レイアウトスキンに JavaScript コードを追加する方法について説明します。本書では jQuery 実行コードの作成方法については述べません。スキン製作者は、製作意図に合ったコードを作成し、追加します。

1. user_layout.js にて、jQuery 文法のコードを作成します。

```
// <![CDATA[
jQuery(function($){
    // ここに jQuery コードを作成します。
});
// ]]>
```

2. layout.html にて、下記のように user_layout.js を参照するよう宣言します。

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

詳細な説明については、「JS ファイルの参照」を参照してください。

3. 下記のパスへアクセスし、JavaScript が正常に実行されるかどうかを確認します。下記のパスにて、「example.com」は、ユーザーWeb サイトがインストールされているドメインアドレスを意味します。
   - mod_rewrite を使用する場合：http://example.com/home/
   - mod_rewrite を使用しない場合：http://example.com/?mid=home

もし、作成されたコードが画面に反映されていないなら、user_layout.js を参照する<load />テンプレート文法が正しいか、または user_layout.js 参照パス(target)が正しく指定されているかどうか確認してください。

> **参考**
>
> XE における jQuery の基本的な使用方法については、「JavaScript と jQuery の使用」を参照してください。
> jQuery を使用して、より多彩な効果を実装するには、http://jquery.com/Web サイトを参照してください。

4. 掲示板スキンをつくる

この章では、例題掲示板スキンを使用して掲示板スキンを製作し、適用する方法について説明します。

4.1 掲示板スキンとは

掲示板スキンとは、掲示板モジュールをユーザー画面に表示するインターフェースです。掲示板スキンは、ユーザー画面を基準にリスト、閲覧、書き込み、削除、コメント、コメント削除、トラックバック削除、権限ガイド、パスワード入力ページで構成されます。

4.2 掲示板モジュールのインストール

XE で掲示板を製作するには、掲示板モジュールを別途インストールしなければなりません。掲示板モジュールは、XE Admin ページのイージーインストール機能を利用して、もしくは掲示板モジュールのソースファイルをサーバーにアップロードしてインストールできます。

イージーインストール

XE Admin ページにて、**拡張機能 > Easy Install > モジュール**を選択して掲示板モジュールをサーバーにダイレクトにインストールします。

イージーインストールで掲示板モジュールをインストールした場合、掲示板スキンを作成するには、FTP を通じて掲示板モジュールがインストールされているディレクトリ(/modules/board/)をローカル PC にダウンロードしなければなりません。

掲示板モジュールのディレクトリ構造は、下記図にて示します。掲示板モジュールディレクトリに「skins」ディレクトリが含まれているかどうか確認します。掲示板スキンは、「skins」ディレクトリに格納されます。

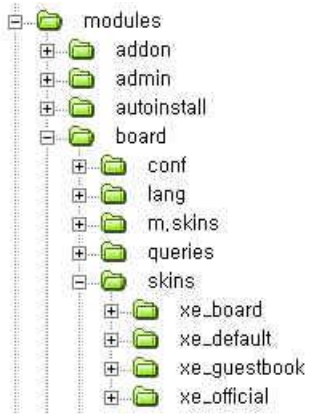


図 4-1 掲示板モジュールのディレクトリ構造

ソースファイルのアップロード

XE 公式 Web サイトにて、掲示板モジュールのソースファイルをローカル PC にダウンロードした後、FTP プログラムを通して Web サーバーの「/modules/」パスにソースファイルをアップロードします。たとえば、/xe/というユーザー定義ディレクトリに XE core をインストールした場合、「/xe/modules/」パスにソースファイルをアップロードしなければなりません。

掲示板モジュールが正常にアップーロードされたら、掲示板モジュールのディレクトリパスは、「/modules/board/」、または「/xe/modules/board/」になります。

4.3 例題掲示板スキンのダウンロード

掲示板スキンを製作するには、掲示板スキンに必要な必須ファイルを作成しなければなりません。本書では、スキン製作者の便宜のために必須ファイルを含んだ例題掲示板スキンを提供します。スキン製作者は、例題掲示板スキンを利用して思い通りにスキンの修正ができます。

例題掲示板スキンのダウンロード先は、下記の通りです。

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

4.4 掲示板スキンの位置と必須ファイル

4.4.1 掲示板スキンの位置確認

掲示板スキンディレクトリは、下記の場所にあります。

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

ダウンロードした例題掲示板スキン(user_board)を解凍して、「/modules/board/skins/」配下にコピーします。

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

> **参考**
>
> 掲示板スキンをローカル PC で修正する際には、修正内容を Web サイトの掲示板スキンディレクトリに反映しなければなりません。

4.4.2 掲示板スキン必須ファイル

掲示板スキンを製作するには、下記の必須ファイルが必要です。

表 4-1 掲示板スキン必須ファイル

ファイル	説明	備考
skin.xml	掲示板スキン情報	ファイル名変更不可
list.html	記事リスト	ファイル名変更不可
write_form.html	記事書き込みフォーム	ファイル名変更不可
delete_form.html	記事削除フォーム	ファイル名変更不可
comment_form.html	コメント登録フォーム	ファイル名変更不可
delete_comment_form.html	コメント削除フォーム	ファイル名変更不可
delete_trackback_form.html	トラックバック削除フォーム	ファイル名変更不可
input_password_form.html	パスワード入力フォーム	ファイル名変更不可
message.html	お知らせメッセージ	ファイル名変更不可
_header.html	掲示板ヘッダー表示	別ページにインクルードされる
_footer.html	掲示板フッター表示	別ページにインクルードされる
_read.html	記事閲覧ページ表示	別ページにインクルードされる
_comment.html	コメント表示	別ページにインクルードされる
_trackback.html	トラックバック表示	別ページにインクルードされる
user_board.css	掲示板スキンスタイルシート	

4.5 掲示板スキン情報作成

skin.xml は、掲示板スキンの基本情報を含んでおり、管理者画面に表示する説明とオプションを提供します。
skin.xml の名前は、変更できません。

skin.xml の基本構造を以下に示します。

```
<?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_board 입니다.</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>
```

上記コードの内容は、下記表の通りです。

コード	説明
`<?xml version="1.0" encoding="UTF-8"?>`	XML ドキュメント形式の宣言
`<skin version="0.2">`	スキン情報についてのドキュメントであることを宣言。version には、XE core でサポートするバージョンを表示しなければなりません。XE core 1.4.4.2 基準での対応バージョンは、0.2 です。
`<title xml:lang="ko">...</title>`	掲示板スキンの名前
`<description xml:lang="ko">...</description>`	掲示板スキンの説明
`<version>...</version>`	掲示板スキンのバージョン
`<date>YYYY-MM-DD</date>`	掲示板スキンの製作日。年-月-日(YYYY-MM-DD)形式で作

コード	説明
	成します。
<author email_address="..." link="..."> <name xml:lang="ko">...</name> </author>	掲示板スキン製作者情報。メールアドレス、Web サイトアドレス、製作者名を作成します。
<license>LGPL v2</license>	掲示板スキンライセンス情報。XE core 同様、LGPL v2 ライセンスを推奨します。
<extra_vars>...</extra_vars>	掲示板モジュールでサポートする拡張変数を使用するには、タグの内側に内容を追加します。
<var name="title" type="text"> <title xml:lang="ko">掲示板タイトル</title> <description xml:lang="ko">作成すると画面に表示される</description> </var>	掲示板タイトルの入力を受け、画面に表示します。
<var name="title" type="textarea"> <title xml:lang="ko">掲示板の説明</title> <description xml:lang="ko">作成すると画面に表示される</description> </var>	掲示板説明の入力を受け、画面に表示します。

「user_board」掲示板スキンの skin.xml ドキュメントが正常に作成されていることを確認するには、掲示板をひとつ作成し、「user_board」スキンを使用するよう設定します。

この他にも、skin.xml には、ユーザーが定義可能ないろんなタイプの変数を<extra_vars>要素内に追加することができます。下記は、スキン製作者が select、image タイプのデータの入力を受け、変数で使用できるよう拡張した例題です。

```
<?xml version="1.0" encoding="UTF-8"?>
<layout version="0.2">
    ...
    <extra_vars>
        <var name="colorset" type="select">
            <title xml:lang="ko">カラーセット</title>
            <description xml:lang="ko">希望するカラーセットを選択してください</description>
            <options value="black">
                <title xml:lang="ko">Black (基本)</title>
            </options>
            <options value="white">
                <title xml:lang="ko">White</title>
            </options>
        </var>
        <var name="board_image" type="image">
            <title xml:lang="ko">掲示板ヘッダーイメージ</title>
            <description xml:lang="ko">掲示板上部に表示するイメージを入力してください。</description>
        </var>
    </extra_vars>
    ...
</layout>
```

上記コードで使用された拡張変数は、下記の通りです。

変数	説明
`<var name="colorset" type="select">...</var>`	select タイプの拡張変数。{$module_info->colorset}形式で表示可能。
`<var name="board_image" type="image">...</var>`	image タイプの拡張変数。{$modul_info->image}形式で表示可能。

skin.xml にて追加された拡張変数は、XE Admin ページの**拡張機能 > Installed Module > 掲示板 >設定 > スキン管理**ページに表示されます。掲示板管理者が該当変数に値を入力すると、スキンに表示できます。

4.6 掲示板作成、およびスキン適用

新しい掲示板を作成し、スキンを適用する方法について説明します。

1. XE Admin ページにて、**拡張機能** > **Installed Module** > **掲示板**を選択します。
2. **掲示板管理** > **掲示板リスト**ページが表示されたら、**作成**をクリックします。

3. 下記のように各項目を入力し、**登録**をクリックします。実際には、もっと多くのオプション項目がありますが、掲示板を作成する際に必ずしも確認すべき項目ではないため、この例題画面では省略しています。
   - **モジュール名**: *test_board*
   - **ブラウザタイトル**: *掲示板スキン実習*
   - **スキン**: user_board

4. 作成された test_board 掲示板の**スキン管理**をクリックし、ユーザー画面に表示する**掲示板タイトル**、**掲示板の詳しい説明**を入力して**登録**をクリックします。
   - **掲示板タイトル**: *XE 実習*
   - **掲示板の詳しい説明**: *XE 掲示板スキン製作実習のための掲示板です。*


Board - XE Admin
http://xpressengine.com/index.php?module=admin&act=dispBoardAdminSkinInfo&
Google
developers@xpressengine.com | Log-out | 日本語
XE Admin
http://xpressengine.com/
ダッシュボード
サイト
会員
コンテンツ
テーマ
拡張機能
設定
お気に入り
掲示板の管理
test_board | 表示
掲示板リスト | 掲示板情報 | カテゴリ管理 | 拡張変数 | リストの設定 | 権限管理 | 追加設定 | スキン管理
スキン基本情報
スキン
user_board
スキン作者
製作者名 (http://user-define.com/, user@user.com)
日付
2010-12-24
ライセンス
LGPL v2
説明
掲示板スキン製作実習のためのuser_boardです。
拡張変数
掲示板タイトル
XE実習
言語コード検索
作成すると画面に表示される
掲示板詳細説明
XE掲示板スキン製作実習のための掲示板です。
言語コード検索
作成すると画面に表示される
登録
Powered by XE (ver. 1.5.0.9).
管理者メニューの初期化
キャッシュファイル再生成
セッション整理
バグレポート

4.7 掲示板ヘッダー/フッター作成

4.7.1 ヘッダー作成

掲示板ヘッダーは、ユーザーが掲示板のどのページにアクセスしても、常に同じ内容を画面上部に表示します。

例題掲示板スキンでは、XE Admin ページで入力した**掲示板タイトル、掲示板の詳しい説明**を常に表示し、CSS ファイルを参照するよう、下記のように_header.html を作成しました。掲示板のすべてのページにこの_header.html をインクルードする予定です。

```
<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>
```

要素を閉じていない点に留意してください。この要素の終了タグは、_footer.html で作成します。 要素は、掲示板のすべての要素を一度にまとめ、CSS を便利に作成、管理できるようにします。

上記コードにて、使用されているテンプレート文法と変数は、下記の通りです。

テンプレート文法/変数	説明
`cond="$module_info->title || $module_info->comment"`	**掲示板タイトル**、または**掲示板の詳しい説明**のうち、ひとつでもある場合、含まれている内容を表示
`cond="$module_info->title"`	**掲示板タイトル**がある場合、表示
`{$module_info->title}`	**掲示板タイトル**表示
`cond="$module_info->comment"`	**掲示板の詳しい説明**がある場合、表示
`{$module_info->comment}`	**掲示板の詳しい説明**表示
`{getUrl('','mid',$mid)}`	掲示板 URL

XE core 1.4.4 より追加された新しいテンプレート文法を使用しました。新しいテンプレート文法についての詳細な説明は、「XE テンプレート文法」を参照してください。

4.7.2 フッター作成

掲示板フッターは、ユーザーが掲示板のどのページにアクセスしても常に同じ内容を画面下部に表示します。

例題掲示板スキンでは、_header.html で終了していなかった の終了タグのみを作成しました。

```
</div>
```

_footer.html は、_header.html と共に、すべてのページにインクルードされます。

4.8 リストページ作成

掲示板にアクセスした際に最初に表示されるページは、記事リストを表示するリストページです。リストページは、list.html にて作成します。

下記図は、例題掲示板スキンで作成したリストページの完成画面です。掲示板ヘッダーで作成した通りに「XE 実習」という掲示板タイトルと「XE 掲示板のスキンを作成実習のための一時的な掲示板です。」という掲示板の詳しい説明が掲示板上部に常に表示されます。

図 4-2 掲示板リストページ完成画面

上記の掲示板リストページは、表示可能なすべてのカラム項目を表示するよう、設定した状態です。掲示板管理者がリストページにどの項目を表示させるか分からないため、スキンを作成するときにすべての項目を表示するように設定することを推奨します。

list.html を作成する方法について、以下に説明します。

_header.html、_footer.html をインクルードする

例題掲示板スキンの list.html では、下記のように_header.html と_footer.html をインクルードするよう、作成しました。

```
<include target="_header.html" />
    ここに記事リストを表示する予定です。
<include target="_footer.html" />
```

上記コードにて使用されたテンプレート文法は、下記の通りです。

XE テンプレート文法	説明
`<include target="_header.html" />`	_header.html インクルード
`<include target="_footer.html" />`	_footer.html インクルード

XE core 1.4.4 バージョンより追加された新しいテンプレート文法を使用しました。新しいテンプレート文法についての詳細な説明は、「XE テンプレート文法」を参照してください。

記事がないときにメッセージを表示する

条件文を使用して、記事がないときにメッセージを表示させることができます。例題掲示板スキンの list.html では、記事がないときに「登録された記事がありません」というメッセージを表示するよう、下記のように条件文を作成しました。

```
<include target="_header.html" />
<p cond="!$document_list && !$notice_list" class="no_document">{$lang->no_documents}</p>
<include target="_footer.html" />
```

上記コードにて使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明
`cond="!$document_list && !$notice_list"`	記事とお知らせリストが両方ともない場合、表示
`{$lang->no_documents}`	「登録された記事がありません」という言語変数

XE core 1.4.4 より追加された新しいテンプレート文法を使用しました。新しいテンプレート文法についての詳細な説明は、「XE テンプレート文法」を参照してください。

記事リストを表で作成する

通常、記事リストは、表で作成します。さまざまな条件文と変数を含める前に HTML を使用して下記のように掲示板の骨組みを完成させます。表は、リストヘッダー(LIST HEADER)、お知らせ(NOTICE)、リスト(LIST)行で大別し、コードを見やすくするために注釈表示しておきます。

```
<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">番号</th>
            <th scope="col">タイトル</th>
            <th scope="col">投稿者</th>
            <th scope="col">ID</th>
            <th scope="col">名前</th>
            <th scope="col">日付</th>
            <th scope="col">最終更新日</th>
            <th scope="col">最新の記事</th>
            <th scope="col">閲覧数</th>
            <th scope="col">推奨数</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" />
```

上記コードにて使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明
`cond="$document_list || $notice_list"`	記事やお知らせがある場合、表と共に含まれている内容を表示

XE core 1.4.4 より追加された新しいテンプレート文法を使用しました。新しいテンプレート文法についての詳細な説明は、「XE テンプレート文法」を参照してください。

記事リストヘッダーを表示する

記事リストヘッダーとは、番号、タイトル、投稿者、日付、閲覧数などのタイトルのセルのことです。掲示板管理者がどの項目を表示するかは分からないため、スキン製作者は、掲示板モジュールがサポートするほとんどの内容を条件文で作成しておくことを推奨します。

例えば、掲示板管理者は、XE Admin ページの**拡張機能 > Installed Module > 掲示板**にて**設定**をクリックし、**リストの設定**を開いて、すべての項目を表示するように設定することができます。このとき、スキンがすべての項目に対応していないと、掲示板管理者はスキンに問題があると誤解する恐れがあります。

図 4-3 掲示板リスト設定

XE 掲示板モジュールは、基本的に上記のように 12 種類の情報をリスト画面に表示できます。**サムネール**と**要約**は通常、別途のカラムにせず、**タイトル**セルに一緒に表示するため、記事リストのカラム数には含めません。したがって、すべての項目を表示できるよう対応するには、10 種類の項目に対して条件文を作成しなければなりません。掲示板管理者が全項目表示を設定する場合、合計 10 個のカラムが生成されます。

例題掲示板スキンの list.html では、下記のように<thead>要素の内部に 10 個の項目に対する条件文を作成しました。

```
...
<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>
```

```
...
```

上記コードで使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明
`<block loop="$list_config=>$key,$val">...</block>`	表示項目を表示
`<th scope="col" cond="$val->type=='no'">{$lang->no}</th>`	記事番号
`<th scope="col" class="title" cond="$val->type=='title'">` `{$lang->title}` `</th>`	記事タイトル
`<th scope="col" cond="$val->type=='nick_name'">` `{$lang->writer}` `</th>`	投稿者
`<th scope="col" cond="$val->type=='user_id'">` `{$lang->user_id}` `</th>`	ID
`<th scope="col" cond="$val->type=='user_name'">` `{$lang->user_name}` `</th>`	名前
`<th scope="col" cond="$val->type=='regdate'">` `<a href="{getUrl('sort_index','regdate','order_type',$order_type)}">` `{$lang->date}` `</a>` `</th>`	日付(整列順変更可)
`<th scope="col" cond="$val->type=='last_update'">` `<a` `href="{getUrl('sort_index','last_update','order_type',$order_type)}">` `{$lang->last_update}` `</a>` `</th>`	最終更新日(整列順変更可)
`<th scope="col" cond="$val->type=='last_post'">` `<a` `href="{getUrl('sort_index','last_update','order_type',$order_type)}">` `{$lang->last_post}` `</a>` `</th>`	最新記事(整列順変更可)
`<th scope="col" cond="$val->type=='readed_count'">` `<a` `href="{getUrl('sort_index','readed_count','order_type',$order_type)}">` `{$lang->readed_count}` `</a>` `</th>`	閲覧数(整列順変更可)
`<th scope="col" cond="$val->type=='voted_count'">` `<a` `href="{getUrl('sort_index','voted_count','order_type',$order_type)}">` `{$lang->voted_count}` `</a>` `</th>`	推奨数(整列順変更可)

XE core 1.4.4 より追加された新しいテンプレート文法を使用しました。新しいテンプレート文法についての詳細な説明は、「XE テンプレート文法」を参照してください。

記事リストを表示する

リストには、お知らせリストと一般の記事リストがあります。お知らせリストは、記事リストとは違って、常に表の上部に表示されます。

例題掲示板スキンの list.html では、下記のように<tbody>要素にてお知らせリストと記事リストを表示するよう作成しました。

```
...
<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>
...
```

上記コードで使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明
`<tr class="notice" loop="$notice_list=>$no,$document">`	お知らせリスト行を繰り返す
`<tr loop="$document_list=>$no,$document">`	記事リスト行を繰り返す
`<block loop="$list_config=>$key,$val">`	記事表示項目列を繰り返す
`<block cond="$document_srl==$document->document_srl">»</block>`	記事の固有番号と現在ページ記事の固有番号が一致の場合、右側二重山括弧(>>)を表示
`<block cond="$document_srl!=$document->document_srl">{$lang->notice}</block>`	記事の固有番号と現在ページ記事の固有番号が不一致の場合、「お知らせ」を表示
`<block cond="$document_srl!=$document->document_srl">{$no}</block>`	記事の固有番号と現在ページ記事の固有番号が不一致の場合、「記事番号」を表示
`<td class="title" cond="$val->type=='title'">`	記事タイトルセル
`<a href="{getUrl('document_srl',$document->document_srl, 'listStyle', $listStyle, 'cpage','')}">...</a>`	記事リンク
`{$document->getTitle()}`	記事タイトル表示
`<a cond="$document->getCommentCount()" href="{getUrl('document_srl', $document->document_srl)}#comment" class="replyNum"`	記事コメントリンク

XE テンプレート文法/変数	説明
title="Replies"> [...] 	
{$document->getCommentCount()}	記事コメント数
getTrackbackCount()" href="{getUrl('document_srl', $document->document_srl)}#trackback" class="trackbackNum" title="Trackbacks"> [...] 	記事トラックバックリンク
{$document->getTrackbackCount()}	記事トラックバック数
<td class="author" cond="$val->type=='nick_name'">{$document->getNickName()}</td>	ニックネーム
<td class="author" cond="$val->type=='user_id'">{$document->getUserID()}</td>	ID
<td class="author" cond="$val->type=='user_name'">{$document->getUserName()}</td>	名前
<td class="time" cond="$val->type=='regdate'">{$document->getRegdate('Y.m.d')}</td>	日付
<td class="time" cond="$val->type=='last_update'">{zdate($document->get('last_update'),'Y.m.d')}</td>	最終更新日
<td class="lastReply" cond="$val->type=='last_post'">...</td>	最新記事
<block cond="(int)($document->get('comment_count'))>0">...</block>	コメントがひとつ以上ある場合、表示
<block cond="(int)($document->get('comment_count'))==0">...</block>	コメントがない場合、表示
getPermanentUrl()}#comment" title="Last Reply"> {zdate($document->get('last_update'),'Y.m.d')} 	最新のコメント日付 + リンク
<span cond="$document->get('last_updater')"> _{by} {htmlspecialchars($document->get('last_updater'))} 	最新のコメント登録者
<td class="readNum" cond="$val->type=='readed_count'"> {$document->get('readed_count')>0?$document->get('readed_count'):'0'} </td>	閲覧数
<td class="voteNum" cond="$val->type=='voted_count'"> {$document->get('voted_count')!=0?$document->get('voted_count'):'0'} </td>	推奨数

XE core 1.4.4 より追加された新しいテンプレート文法を使用しました。新しいテンプレート文法についての詳細な説明は、「XE テンプレート文法」を参照してください。

> **参考**
>
> 記事リストの表示数は、XE Admin ページの**拡張機能 > Installed Module > 掲示板**にて修正するモジュールの右にある**設定**をクリックし、**掲示板情報**にて**リスト数**を設定して変更できます。基本値は、20 です。

ページ番号リンクを表示する

記事の数量が1ページを超えると、過去の記事を検索できるよう、ページ移動リンクを提供しなければなりません。

例題掲示板スキンの list.html では、記事リスト下部に下記のようにページ番号リンクを表示するように作成しました。ページ番号は ... 要素で2度囲んで、この要素の内側に**書き込む**ボタンと**検索**入力フォームを追加する予定です。

```
<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" />
```

上記コードで使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明
`<div class="pagination" cond="$document_list || $notice_list">...</div>`	記事があったり、お知らせがひとつでもある場合、ページ番号を表示
`<a href="{getUrl('page','','document_srl','','division',$division,'last_division',$last_division)}" class="prevEnd">{$lang->first_page}</a>`	最初のページリンク
`<a href="{getUrl('page',$page_navigation->last_page,'document_srl','','division',$division,'last_division',$last_division)}" class="nextEnd">{$lang->last_page}</a>`	最後のページリンク
`<block loop="$page_no=$page_navigation->getNextPage()">...</block>`	ページ番号を繰り返す
`<strong cond="$page==$page_no">{$page_no}</strong>`	ページ番号が現在ページと一致の場合、表示
`<a cond="$page!=$page_no" href="{getUrl('page',$page_no,'document_srl','','division',$division,'last_division',$last_division)}">{$page_no}</a>`	ページ番号が現在ページと不一致の場合、表示

XE core 1.4.4 より追加された新しいテンプレート文法を使用しました。新しいテンプレート文法についての詳細な説明は、「XE テンプレート文法」を参照してください。

書き込むボタンを表示する

list.html は、リストページのみでなく、閲覧ページでも表示されます。**書き込む**ボタンをクリックすると、write_form.html ページへ移動します。

例題掲示板スキンの list.html では、下記のように書き込みページリンクボタンを作成しました。

```
<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>
```

上記コードで使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明
`<a href="{getUrl('act','dispBoardWrite','document_srl','')}" class="btn">{$lang->cmd_write}</a>`	書き込みページリンクボタン。リストページと閲覧ページに表示されます。

XE core 1.4.4 より追加された新しいテンプレート文法を使用しました。新しいテンプレート文法についての詳細な説明は、「XE テンプレート文法」を参照してください。

掲示板検索入力フォームを表示する

掲示板の内容を検索できる検索入力フォームを表示します。例題掲示板スキンの list.html では、下記のように検索入力フォームを表示するよう作成しました。

```
<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>
```

上記コードで使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明
`<form cond="$grant->view" action="{getUrl()}" method="get" onsubmit="return procFilter(this, search)" id="board_search" class="board_search">`	記事閲覧権限がある場合、検索入力フォームを表示
`<input type="hidden" name="vid" value="{$vid}" />`	仮想サイト ID 転送要素(hidden type)
`<input type="hidden" name="mid" value="{$mid}" />`	モジュール ID 転送要素(hidden type)
`<input type="hidden" name="category" value="{$category}" />`	カテゴリ情報転送要素(hidden type)
`<input type="text" name="search_keyword" value="{htmlspecialchars($search_keyword)}" accesskey="S" title="{$lang->cmd_search}" class="iText" />`	検索キーワード入力、および表示要素。アクセスキー「S」が割り当てられる。
`<select name="search_target">`	検索範囲選択コントロール
`<option loop="$search_option=>$key,$val" value="{$key}" selected="selected"|cond="$search_target==$key">{$val}</option>`	検索範囲オプション表示
`<input type="submit" onclick="xGetElementById('board_search').submit();return false;" value="{$lang->cmd_search}" class="btn" />`	検索入力フォーム転送ボタン

XE core 1.4.4 より追加された新しいテンプレート文法を使用しました。新しいテンプレート文法についての詳細な説明は、「XE テンプレート文法」を参照してください。

掲示板リストの画面表示結果を確認する

下記のパスへアクセスし、掲示板リスト画面を確認します。下記パスにて、「example.com」は、ユーザーの Web サイトがインストールされているドメインアドレスを意味します。

- mod_rewrite を使用する場合：http://example.com/test_board/
- mod_rewrite を使用しない場合：http://example.com/?mid=test_board

作成された記事がない場合、下記図のように「登録された記事がありません」というメッセージが表示されます。

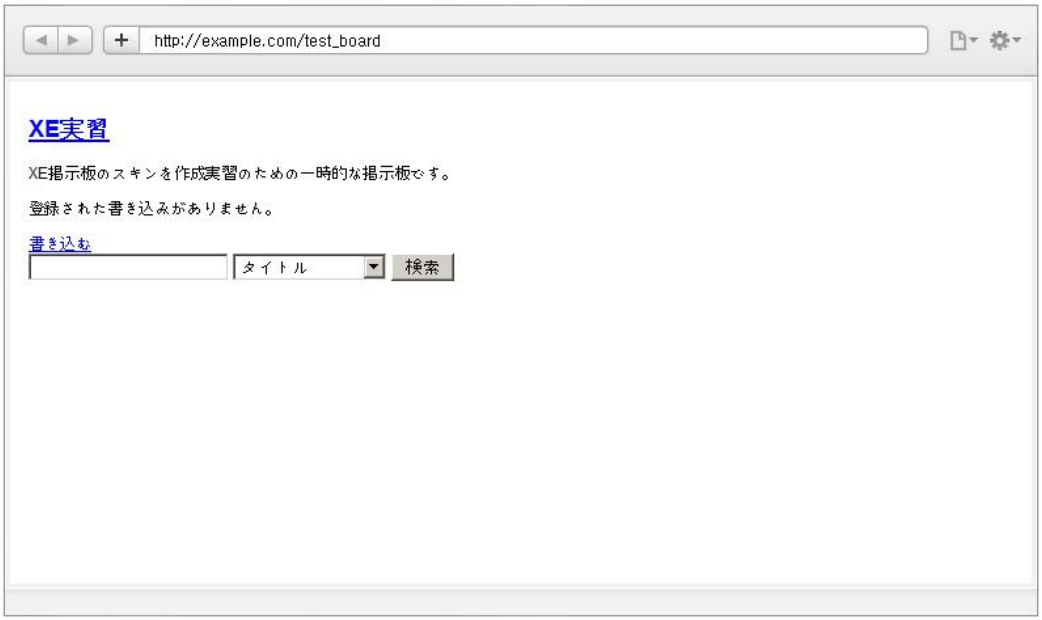


図 4-4 掲示板リスト画面 – 作成された記事がない場合

作成された記事があれば、下記図のように記事リスト画面が表示されます。

図 4-5 掲示板リスト画面 – 作成された記事がある場合

まだ write_form.html ページを作成していないため、「test_board」掲示板に記事を作成することはできません。しかし、「test_board」掲示板にテスト用の記事データをコピーして入れることができます。上記画面は、他の掲示板のテスト

用の記事を「test_board」掲示板へコピーした様子です。XE Admin ページにて、**コンテンツ** > **Document** を選択し、**選択した書き込みを管理**をクリックして他の掲示板の書き込みを「test_board」掲示板へコピーすることができます。

4.9 書き込みページ作成

書き込みページは、新しい記事を書いたり、既存の記事を修正する画面として使用します。書き込みページは、write_form.html にて作成します。

書き込みページは、_header.html、_footer.html、記事タイトル入力ウィンドウ、記事内容入力ウィンドウ、投稿者情報入力ウィンドウ(名前、パスワード、ホームページ)、登録ボタンで構成されています。XE core に完成版の WYSIWYG エディターが含まれています。書き込みページでは、このエディターモジュールを呼び出して使用します。

下記図は、例題掲示板スキンで作成した書き込みページです。

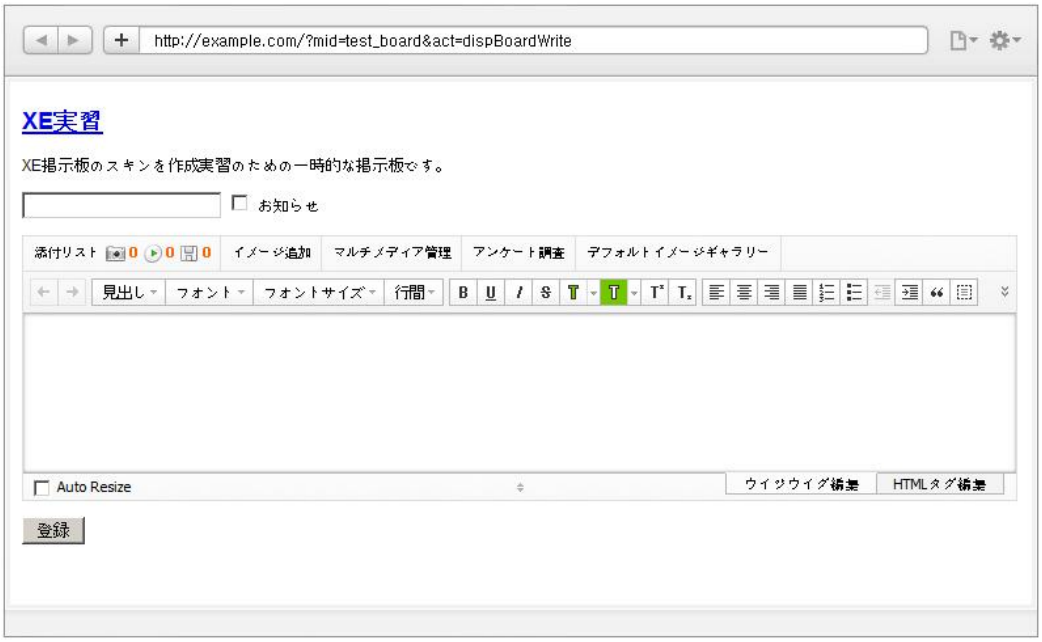


図 4-6 書き込みページ完成画面

write_form.html を作成する方法について、以下に説明します。

_header.html、_footer.html をインクルードする

例題掲示板スキンの write_form.html では、下記のように_header.html と_footer.html をインクルードするよう、作成しました。

```
<include target="_header.html" />
    ここに書き込み入力フォームを作成する予定です。
<include target="_footer.html" />
```

上記コードにて使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明
`<include target="_header.html" />`	_header.html インクルード
`<include target="_footer.html" />`	_footer.html インクルード

XE core 1.4.4 より追加された新しいテンプレート文法を使用しました。新しいテンプレート文法についての詳細な説明は、「XE テンプレート文法」を参照してください。

書き込み画面の HTML 構造

書き込み画面の HTML 構造を以下に示します。

```
<include target="_header.html" />
<form action="" class="board_write">
    <div class="write_header">タイトル入力ウィンドウ</div>
    <div class="write_editor">内容入力ウィンドウ</div>
    <div class="write_author">投稿者情報入力ウィンドウ(名前、パスワード、ホームページ)</div>
    <div class="write_footer">登録ボタン</div>
</form>
<include target="_footer.html" />
```

作成内容をサーバーへ転送するには、form 要素で作成しなければなりません。転送する内容は、「タイトル、内容、投稿者情報」です。**登録**をクリックすると、form が転送されます。

書き込みフォームを作成する

onsubmit 属性を通じて、書き込みページに入力された内容が正しく作成されたか、クライアント環境でチェックすることができます。書き込みページにて記事を修正するときは、ユーザーが作成した既存のデータを呼び出します。

例題掲示板スキンの write_form.html では、下記のように書き込みフォームを作成しました。

```
<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" />
```

上記コードで使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明
`onsubmit="return procFilter(this, window.insert)"`	ユーザー入力データ有効性チェック、および form 転送
`<input type="hidden" name="mid" value="{$mid}" />`	モジュール ID 転送要素(hidden type)
`<input type="hidden" name="content" value="{$oDocument->getContentText()}" />`	内容転送要素(hidden type)
`<input type="hidden" name="document_srl" value="{$document_srl}" />`	ドキュメント固有番号転送要素(hidden type)
`<input type="hidden" name="allow_comment" value="Y" />`	コメント登録許可要素(hidden type)。 許可しない場合は、value="N"を入力します。
`<input type="hidden" name="allow_trackback" value="Y" />`	トラックバック許可要素(hidden type)。許可しない場合は、value="N"を入力します。

XE core 1.4.4 バージョンより追加された新しいテンプレート文法を使用しました。新しいテンプレート文法についての詳細な説明は、「XE テンプレート文法」を参照してください。

タイトル入力ウィンドウを作成する

タイトル入力ウィンドウは、書き込みページが「新しい記事」のときと「記事修正」のときの両方を考慮しなければなりません。作成する記事をお知らせ記事として登録するかどうかも選択できなくてはなりません。

例題掲示板スキンの write_form.html では、下記のようにタイトル入力ウィンドウを作成しました。

```
...
```

```
<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>
...
```

上記コードで使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明
`cond="$oDocument->getTitleText()"`	すでにタイトルのある「修正」画面の場合、表示
`cond="!$oDocument->getTitleText()"`	タイトルのない「新しい記事」画面の場合、表示
`{htmlspecialchars($oDocument->getTitleText())}`	修正画面の場合、ドキュメントのタイトルテキストを表示
`{$lang->title}`	「タイトル」言語変数
`cond="$grant->manager"`	管理者の場合、お知らせ確認入力ウィンドウとラベルを表示
`checked="checked"|cond="$oDocument->isNotice()"`	修正画面の場合、この記事がお知らせであれば、checked 属性と値を表示
`{$lang->notice}`	「お知らせ」言語変数

XE core 1.4.4 より追加された新しいテンプレート文法を使用しました。新しいテンプレート文法についての詳細な説明は、「XE テンプレート文法」を参照してください。

内容入力（編集）ウィンドウを作成する

内容入力（編集）ウィンドウは、{$oDocument->getEditor()}という変数を宣言します。この変数は、**拡張機能 > Installed Module > 掲示板**にて編集するモジュールの右にある**設定**をクリックし、**追加設定**の**ウイジウイグエディター**で選択したエディター(WYSIWYG エディター)を表示します。

例題掲示板スキンの write_form.html では、下記のように WYSIWYG エディターを呼び出すよう宣言しました。

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

上記コードで使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明
`{$oDocument->getEditor()}`	エディター(WYSIWYG エディター)表示変数

投稿者情報入力ウィンドウを作成する

投稿者情報入力ウィンドウは、非ログインユーザーにだけ表示されます。入力ウィンドウは、「投稿者、パスワード、ホームページ」で構成されます。パスワードは、記事を修正、または削除する際に必要です。

例題掲示板スキンの write_form.html では、下記のように投稿者情報入力ウィンドウを作成しました。

```
...
<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>
...
```

上記コードで使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明
`<div class="write_author" cond="!$is_logged">`	ログインしていない場合、投稿者情報入力ウィンドウを表示
`{$lang->writer}`	「投稿者」言語変数
`{htmlspecialchars($oDocument->get('nick_name'))}`	書き込みページが修正ページとして使用された場合、既存記事の投稿者名を表示
`{$lang->password}`	「パスワード」言語変数
`{$lang->homepage}`	「ホームページ」言語変数
`{htmlspecialchars($oDocument->get('homepage'))}`	書き込みページが修正ページとして使用された場合、既存記事のホームページアドレスを表示

登録ボタンを表示する

ユーザーの作成した form をサーバーに転送するには、**登録**ボタンが必要です。

例題掲示板スキンの write_form.html では、下記のように**登録**ボタンを表示するよう作成しました。

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

上記コードで使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明
`{$lang->cmd_registration}`	「登録」言語変数

XE core 1.4.4 より追加された新しいテンプレート文法を使用しました。新しいテンプレート文法についての詳細な説明は、「XE テンプレート文法」を参照してください。

書き込みページの表示結果を確認する

掲示板リストにて**書き込む**をクリックしたり、下記のパスで掲示板書き込み画面へアクセスします。下記のパスにて、「example.com」は、ユーザーの Web サイトがインストールされているドメインアドレスを意味します。

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

非ログイン状態で、書き込みページにアクセスすると、下記図のように投稿者情報入力ウィンドウが表示されます。

図 4-7 非ログイン状態での書き込みページ

非ログイン状態で書き込みページにアクセスすると、下記の画面が表示されます。投稿者情報入力ウィンドウは表示されません。管理者の場合、お知らせを作成できる**お知らせ**チェックボックスが表示されます。

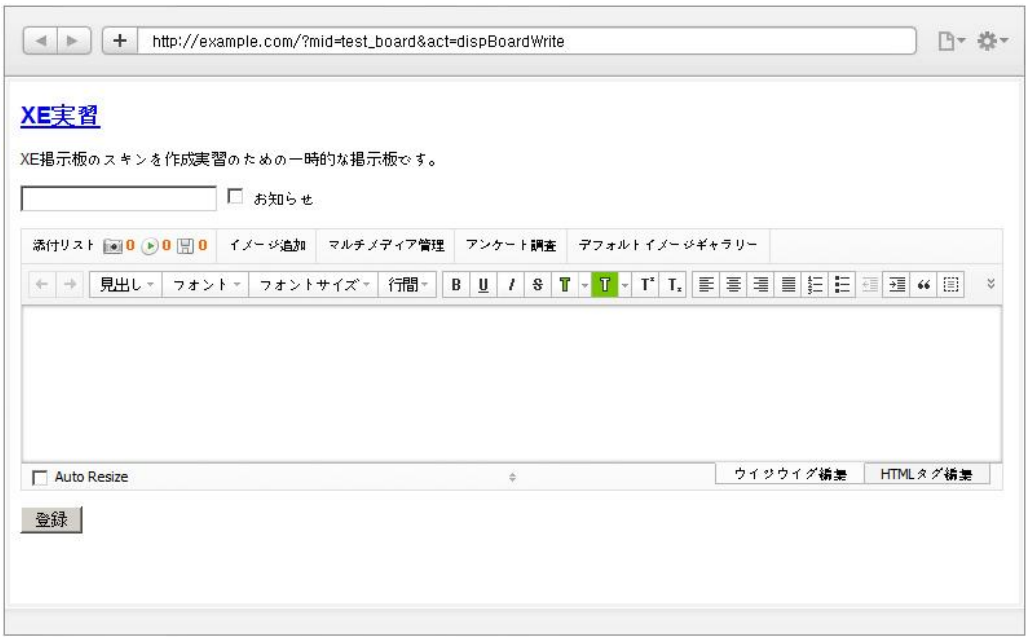


図 4-8 ログイン状態での書き込みページ

タイトルと内容を作成し、**登録**をクリックして作成された記事が掲示板リストに表示されるかどうか確認します。ログインユーザーの必須入力項目は、タイトルと内容です。非ログインユーザーの必須入力項目は、タイトルと内容、投稿者、パスワードです。

4.10 閲覧ページ作成

閲覧ページは、記事の本文以外に、トラックバックリスト、コメントリスト、コメント登録、記事リスト画面で構成され、_read.html にて作成します。

閲覧ページは、リストページ(list.html)にインクルードします。閲覧ページ下部に記事リストを提供し、ユーザーが容易に記事を探せるようにするためです。

下記図にて、例題掲示板スキンで作成した閲覧ページ完成画面をあらわします。

図 4-9 閲覧ページ完成画面

_read.html を作成する方法について説明します。

list.html に_read.html をインクルードする

list.html は、_read.html を条件にしたがってインクルードします。ユーザーが閲覧ページを呼び出すと list.html は、_read.html をインクルードします。

例題掲示板スキンの list.html では、下記の位置に条件文の含まれた include 文を作成しました。ユーザーが閲覧ページへアクセスすると、閲覧エリア下部に記事リストを一緒に表示します。

```
<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" />
```

上記コードで使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明
`<include cond="$oDocument->isExists()" target="_read.html" />`	閲覧ページが呼び出されたら_read.html をインクルード

XE core 1.4.4 より追加された新しいテンプレート文法を使用しました。新しいテンプレート文法についての詳細な説明は、「XE テンプレート文法」を参照してください。

閲覧ページ構造

閲覧ページは、下記のようにヘッダー(タイトル、投稿者、閲覧数、推薦数、日付)、本文、フッター(添付ファイル、リストボタン、修正ボタン、削除ボタン)、トラックバックリスト、コメントリスト、コメント登録フォームで構成されます。

```
<div class="board_read">
    <!-- READ HEADER -->
    <div class="read_header">
        タイトル、投稿者、閲覧数、推薦数、日付
    </div>
    <!-- /READ HEADER -->
    <!-- READ BODY -->
    <div class="read_body">
        記事本文
    </div>
    <!-- /READ BODY -->
    <!-- READ FOOTER -->
    <div class="read_footer">
        添付ファイル、リストボタン、修正ボタン、削除ボタン
    </div>
    <!-- /READ FOOTER -->
</div>
トラックバックリストをインクルード
コメントリストをインクルード
<!-- WRITE COMMENT -->
<form class="write_comment">
    コメント登録フォーム
</form>
<!-- /WRITE COMMENT -->
```

タイトル、投稿者を表示する

例題掲示板スキンの_read.html では、下記のようにタイトルと投稿者を呼び出すよう作成しました。

```
...
```

```
<!-- 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 -->
...
```

上記コードで使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明
`<a href="{$oDocument->getPermanentUrl()}">`	記事の永久 URL
`{$oDocument->getTitle()}`	記事タイトル
`<a cond="$oDocument->getHomepageUrl()" href="{$oDocument->getHomepageUrl()}" class="author">...</a>`	ホームページアドレスがある場合、リンクとニックネームを表示
`<strong cond="!$oDocument->getHomepageUrl()" class="author">...</strong>`	ホームページアドレスがない場合、名前のみ表示
`{$oDocument->getNickName()}`	投稿者のニックネームを表示

XE core 1.4.4 より追加された新しいテンプレート文法を使用しました。新しいテンプレート文法についての詳細な説明は、「XE テンプレート文法」を参照してください。

閲覧数、推奨数、日付を表示する

例題掲示板スキンの_read.html では、閲覧数、推奨数、日付を表示するよう、下記のように作成しました。

```
...
<!-- 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 -->
...
```

上記コードで使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明
`{$lang->readed_count}`	「閲覧数」言語変数
`{$oDocument->get('readed_count')}`	閲覧数表示
`{$lang->voted_count}`	「推薦数」言語変数
`{$oDocument->get('voted_count')}`	推奨数表示
`{$oDocument->getRegdate('Y.m.d H:i')}`	記事作成日、および時刻表示(秒単位まで表示するには、H:i:s)

記事の本文を表示する

例題掲示板スキンの_read.html では、記事本文を表示するよう、下記のように作成しました。

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

上記コードで使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明
{$oDocument->getContent(false)}	記事本文内容表示

添付ファイルを表示する

例題掲示板スキンの_read.html では、添付ファイルを表示するよう、下記のように作成しました。

```
...
<!-- 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 -->
...
```

上記コードで使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明
<div cond="$oDocument->hasUploadedFiles()" class="fileList">	添付ファイルがある場合、含まれている内容を表示
{$lang->uploaded_file}	「添付」言語変数
{$oDocument->get('uploaded_count')}	添付されているファイル数
<li loop="$oDocument->getUploadedFiles()=>$key,$file">	添付されているファイルリストがある場合、取得後に表示する繰り返し文
<a href="{getUrl('')}{$file->download_url}">	添付ファイルダウンロードリンク
{$file->source_filename}	添付ファイル名
{FileHandler::filesize($file->file_size)}	添付ファイルサイズ
{number_format($file->download_count)}	添付ファイルダウンロード回数

XE core 1.4.4 より追加された新しいテンプレート文法を使用しました。新しいテンプレート文法についての詳細な説明は、「XE テンプレート文法」を参照してください。

リスト、修正、削除ボタンを表示する

例題掲示板スキンの_read.html では、**リスト、修正、削除**ボタンを表示するよう、下記のように作成しました。

```
...
<!-- 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 -->
...
```

上記コードで使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明
`<a href="#board_list" class="btn">...</a>`	下部の記事リストへ移動
`{$lang->cmd_list}`	「リスト」言語変数
`<a cond="$oDocument->isEditable()" ...>...</a>`	編集権限がある場合、含まれている内容を表示
`{getUrl('act','dispBoardWrite','document_srl',$oDocument->document_srl,'comment_srl','')}`	修正ページ URL 表示
`{$lang->cmd_modify}`	「修正」言語変数
`{getUrl('act','dispBoardDelete','document_srl',$oDocument->document_srl,'comment_srl','')}`	削除ページ URL 表示
`{$lang->cmd_delete}`	「削除」言語変数

XE core 1.4.4 より追加された新しいテンプレート文法を使用しました。新しいテンプレート文法についての詳細な説明は、「XE テンプレート文法」を参照してください。

トラックバックリスト、コメントリストをインクルードする

例題掲示板スキンの_read.html では、下記のように_trackback.html と_comment.html をインクルードするよう作成しました。

```
<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 -->
```

上記コードで使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明

XE テンプレート文法/変数	説明
`<include target="..." />`	ファイルをインクルード
`cond="$oDocument->allowTrackback()"`	トラックバックを許可した場合、トラックバックリストファイル(_trackback.html)をインクルード
`cond="$oDocument->allowComment()"`	コメント登録を許可した場合、コメントリストファイル(_comment.html)をインクルード

XE core 1.4.4 より追加された新しいテンプレート文法を使用しました。新しいテンプレート文法についての詳細な説明は、「XE テンプレート文法」を参照してください。

記事に対するコメント登録フォームを表示する

例題掲示板スキンの_read.html では、下記のように記事に対するコメント登録フォームを表示するよう作成しました。コメント返し登録、およびコメント修正ページは、comment_form.html にて作成します。

```
<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 -->
```

上記コードで使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明
`<form cond="$grant->write_comment && $oDocument->isEnableComment()"> ...</form>`	コメント登録権限があり、コメント登録が許可されている場合、コメント登録フォームを表示
`onsubmit="return procFilter(this, insert_comment)"`	ユーザー入力データ有効性チェック、および form 転送
`<input type="hidden" name="mid" value="{$mid}" />`	モジュール ID 値転送要素(hidden type)

XE テンプレート文法/変数	説明
`<input type="hidden" name="document_srl" value="{$oDocument->document_srl}" />`	ドキュメント固有番号転送要素(hidden type)
`<input type="hidden" name="comment_srl" value="" />`	コメント固有番号転送要素(hidden type)
`<textarea name="content" rows="5" cols="50"></textarea>`	コメント登録ウィンドウ
`<div class="write_author" cond="!$is_logged">...</div>`	非ログインの場合、含まれた内容(名前入力ウィンドウ、パスワード入力ウィンドウ、ホームページ入力ウィンドウ)を表示
`{$lang->writer}`	「投稿者」言語変数
`{$lang->password}`	「パスワード」言語変数
`{$lang->homepage}`	「ホームページ」言語変数
`{$lang->cmd_comment_registration}`	「コメント登録」言語変数

XE core 1.4.4 より追加されている新しいテンプレート文法を使用しました。新しいテンプレート文法についての詳細な説明は、「XE テンプレート文法」を参照してください。

4.11 トラックバック/コメント関連ページ作成

4.11.1 トラックバックリスト作成

トラックバックリストは、_trackback.html にて作成します。トラックバックリストは、_read.html にインクルードし、閲覧ページに表示します。

例題掲示板スキンの_trackback.html では、下記のようにトラックバックリストを作成します。

```
<!-- 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 -->
```

上記コードで使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明
{$lang->trackback}	「トラックバック」言語変数
{$oDocument->getTrackbackCount()}	トラックバック数を表示
{$oDocument->getTrackbackUrl()}	トラックバックするための URL 表示
cond="$oDocument->getTrackbackCount()"	トラックバックがある場合、含まれている内容を表示（条件）
loop="$oDocument->getTrackbacks()=>$key,$val"	トラックバックがある場合、含まれている内容を表示（繰り返し）
{$val->trackback_srl}	トラックバック固有番号
{$val->url}	トラックバックページの URL
{htmlspecialchars($val->blog_name)}	トラックバックサイト名
{htmlspecialchars($val->title)}	トラックバックタイトル
{zdate($val->regdate, "Y.m.d H:i")}	トラックバック時刻
{$val->excerpt}	トラックバック内容
cond="$grant->manager"	管理者の場合、含まれている内容を表示
{getUrl('act','dispBoardDeleteTrackback','trackback_srl',$val->trackback_srl)}	トラックバック削除ページ URL
{$lang->cmd_delete}	「削除」言語変数

XE core 1.4.4 より追加された新しいテンプレート文法を使用しました。新しいテンプレート文法についての詳細な説明は、「XE テンプレート文法」を参照してください。

4.11.2 コメントリスト作成

コメントリストは、_comment.html にて作成します。コメントリストは、_read.html にインクルードして閲覧ページに表示します。

例題掲示板スキンの_comment.html では、下記のようにコメントリストを作成しました。

```
<!-- 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 -->
```

上記コードで使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明
{$lang->comment}	「コメント」言語変数
{$oDocument->getCommentcount()}	コメント数を表示
cond="$oDocument->getCommentcount()"	コメントがある場合、含まれている内容を表示(条件)
loop="$oDocument->getComments()=>$key,$comment"	コメントがある場合、含まれている内容を表示(繰り返し)
{($comment->get('depth'))*15}	コメントの階層数から 15 を掛ける。
|cond="$comment->get('depth')"	コメントに階層がある場合、属性を表示
{$comment->comment_srl}	コメント固有番号

XE テンプレート文法/変数	説明
cond="$comment->homepage"	ホームページがある場合、含まれている内容を表示
{$comment->homepage}	ホームページ URL
{$comment->getNickName()}	投稿者名を表示
cond="!$comment->homepage"	ホームページがない場合、含まれている内容を表示
{$comment->getRegdate('Y.m.d H:i')}	コメント作成日、および時刻を表示
{$comment->getContent(false)}	コメント本文を表示
{getUrl('act','dispBoardReplyComment','comment_srl',$comment->comment_srl)}	コメント返し登録ページ URL
{$lang->cmd_reply}	「コメント」言語変数
cond="$comment->isGranted()\|\|!$comment->get('member_srl')"	コメント編集権限があったり、会員が書き込んだ記事でない場合、含まれている内容(修正、削除)を表示
{getUrl('act','dispBoardModifyComment','comment_srl',$comment->comment_srl)}	記事修正ページの URL
{$lang->cmd_modify}	「修正」言語変数
{getUrl('act','dispBoardDeleteComment','comment_srl',$comment->comment_srl)}	記事削除ページの URL
{$lang->cmd_delete}	「削除」言語変数
cond="$oDocument->comment_page_navigation"	コメントが多すぎて、ページリンクが必要な場合、表示
{getUrl('cpage',1)}	コメントの最初のページ URL
{getUrl('cpage',$oDocument->comment_page_navigation->last_page)}	コメントの最終のページ URL
{$lang->first_page}	コメントの「最初のページ」言語変数
{$lang->last_page}	コメントの「最終のページ」言語変数
<block loop="$page_no=$oDocument->comment_page_navigation->getNextPage()">...</block>	コメントページ番号リストを表示(繰り返し)
cond="$cpage==$page_no"	コメント現在ページとページ番号が一致の場合、含まれている内容を表示
cond="$cpage!=$page_no"	コメント現在ページとページ番号が不一致の場合、含まれている内容を表示
{getUrl('cpage',$page_no)}	コメントページ URL
{$page_no}	コメントページ番号

XE core 1.4.4 より追加された新しいテンプレート文法を使用しました。新しいテンプレート文法についての詳細な説明は、「XE テンプレート文法」を参照してください。

4.11.3 コメント返し登録、およびコメント修正ページ作成

ユーザーがコメントに対するコメントを作成するときやコメントを修正するときに表示されるページです。元記事に対するコメント登録フォームは、閲覧ページ(_read.html)に含まれています。コメント返し登録、およびコメント修正ページは、comment_form.html にて作成します。

下記図は、例題掲示板スキンで作成したコメント返し登録ページです。元コメントの登録者名、コメント作成時刻、元コメントを上部に表示し、コメントを修正する際には、自身の作成した書き込みを textarea エリアに表示します。

図 4-10 コメント返し登録ページ完成画面

例題掲示板スキンの comment_form.html では、下記のようにコメント返し登録、およびコメント修正ページを作成しました。

```
<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" />
```

上記コードで使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明
<include target="_header.html" />	_header.html インクルード
<include target="_footer.html" />	_footer.html インクルード
cond="$oSourceComment->isExists()"	元コメントがある場合、表示
cond="$oSourceComment->homepage"	ホームページアドレスがある場合、含まれている内容を表示
cond="!$oSourceComment->homepage"	ホームページアドレスがない場合、含まれている内容を表示
{$oSourceComment->homepage}	ホームページアドレス
{$oSourceComment->getNickName()}	元コメントの登録者名
{$oSourceComment->getRegdate('Y.m.d H:i')}	元コメントの作成時刻
{$oSourceComment->getContent(false)}	コメント本文表示
onsubmit="return procFilter(this, insert_comment)"	ユーザー入力データ有効性チェック、および form 転送
<input type="hidden" name="mid" value="{$mid}" />	掲示板モジュール ID 転送要素(hidden type)
<input type="hidden" name="document_srl" value="{$oComment->get('document_srl')}" />	記事固有番号転送要素(hidden type)
<input type="hidden" name="comment_srl" value="{$oComment->get('comment_srl')}" />	コメント固有番号転送要素(hidden type)
<input type="hidden" name="parent_srl" value="{$oComment->get('parent_srl')}" />	元コメントの固有番号転送要素(hidden type)
<textarea name="content" rows="5" cols="50">{htmlspecialchars($oComment->get('content'))}</textarea>	コメント登録、および編集ポップアップ。textarea 内部に含まれている変数は、コメント修正時に元コメントを表示する。
cond="!$is_logged"	非ログインの場合、含まれている内容を表示
{$lang->writer}	「投稿者」言語変数
{$lang->password}	「パスワード」言語変数
{$lang->homepage}	「ホームページ」言語変数
{htmlspecialchars($oComment->get('nick_name'))}	登録者名
{htmlspecialchars($oComment->get('homepage'))}	ホームページ URL
{$lang->cmd_comment_registration}	「コメント登録」言語変数

XE core 1.4.4 より追加された新しいテンプレート文法を使用しました。新しいテンプレート文法についての詳細な説明は、「XE テンプレート文法」を参照してください。

4.12 削除ページ作成

4.12.1 記事削除ページ作成

権限を持つ者が記事を削除しようとした際に、削除する前にもう一度確認を行うページです。記事削除ページは、delete_form.html にて作成します。

下記図にて、例題掲示板スキンで作成した記事削除ページの完成画面をあらわします。

図 4-11 記事削除ページ完成画面

例題掲示板スキンの delete_form.html では、下記のように記事削除ページを作成しました。

```
<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" />
```

上記コードで使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明
`<include target="_header.html" />`	_header.html インクルード
`<include target="_footer.html" />`	_footer.html インクルード

XE テンプレート文法/変数	説明
cond="$oDocument->isExists()"	元記事がある場合、表示
{$oDocument->getTitle()}	元記事のタイトル
{$oDocument->getNickName()}	元記事の投稿者
onsubmit="return procFilter(this, delete_document)"	入力データ有効性チェック、および form 転送
<input type="hidden" name="mid" value="{$mid}" />	モジュール ID 転送要素(hidden type)
<input type="hidden" name="page" value="{$page}" />	ページ番号転送要素(hidden type)
<input type="hidden" name="document_srl" value="{$document_srl}" />	ドキュメント固有番号転送要素(hidden type)
{$lang->confirm_delete}	「削除しますか?」言語変数
{$lang->cmd_delete}	「削除」言語変数
{$lang->cmd_cancel}	「取り消し」言語変数

XE core 1.4.4 より追加された新しいテンプレート文法を使用しました。新しいテンプレート文法についての詳細な説明は、「XE テンプレート文法」を参照してください。

4.12.2 コメント削除ページ作成

ユーザーがコメント**削除**ボタンをクリックした際に、削除する前にもう一度確認を行うページです。コメント削除ページは、delete_comment_form.html にて作成します。

下記図は、例題掲示板スキンで作成したコメント作成ページです。削除するコメントの登録者、コメント作成時刻、元コメントを表示し、その下部に**削除**と**取り消し**ボタンを表示します。

図 4-12 コメント削除ページ完成画面

例題掲示板スキンの delete_comment_form.html では、下記のようにコメント削除ページを作成しました。

```
<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" />
```

上記コードで使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明
`<include target="_header.html" />`	_header.html インクルード
`<include target="_footer.html" />`	_footer.html インクルード
`cond="$oComment->isExists()"`	元コメントがある場合、表示
`cond="$oComment->homepage"`	元コメントに登録者のホームページがある場合、表示
`cond="!$oComment->homepage"`	元コメントに登録者のホームページがない場合、表示
`{$oComment->homepage}`	元コメント登録者のホームページ URL
`{$oComment->getNickName()}`	登録者名
`{$oComment->getRegdate('Y.m.d H:i')}`	元コメント作成時刻
`{$oComment->getContent(false)}`	元コメントの本文表示
`onsubmit="return procFilter(this, delete_comment)"`	入力データ有効性チェック、および form 転送
`<input type="hidden" name="mid" value="{$mid}" />`	モジュール ID 転送要素(hidden type)
`<input type="hidden" name="page" value="{$page}" />`	ページ番号転送要素(hidden type)
`<input type="hidden" name="document_srl" value="{$oComment->get('document_srl')}" />`	元コメントが属する記事の固有番号転送要素(hidden type)
`<input type="hidden" name="comment_srl" value="{$oComment->get('comment_srl')}" />`	元コメントの固有番号転送要素(hidden type)
`{$lang->confirm_delete}`	「削除しますか」言語変数
`{$lang->cmd_delete}`	「削除」言語変数
`{$lang->cmd_cancel}`	「取り消し」言語変数

XE core 1.4.4 より追加された新しいテンプレート文法を使用しました。新しいテンプレート文法についての詳細な説明は、「XE テンプレート文法」を参照してください。

4.12.3 トラックバック削除ページ作成

ユーザーがトラックバック削除ボタンをクリックした際に、削除する前にもう一度確認を行うページです。トラックバック削除ページは、delete_trackback_form.html にて作成します。

下記図にて、例題掲示板スキンで作成したトラックバック削除ページの完成画面をあらわします。

図 4-13 トラックバック削除ページ完成画面

例題掲示板スキンの delete_trackback_form.html では、下記のようにトラックバック削除ページを作成しました。

```
<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" />
```

上記コードで使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明
`<include target="_header.html" />`	_header.html インクルード
`<include target="_footer.html" />`	_footer.html インクルード
`onsubmit="return procFilter(this, delete_trackback)"`	入力データ有効性チェック、および form 転送
`<input type="hidden" name="mid" value="{$mid}" />`	モジュール ID 転送要素(hidden type)
`<input type="hidden" name="page" value="{$page}" />`	ページ番号転送要素(hidden type)
`<input type="hidden" name="document_srl" value="{document_srl}" />`	記事の固有番号転送要素(hidden type)
`<input type="hidden" name="trackback_srl" value="{$trackback_srl}" />`	トラックバックの固有番号転送要素(hidden type)
`{$lang->confirm_delete}`	「削除しますか?」言語変数
`{$lang->cmd_delete}`	「削除」言語変数

XE テンプレート文法/変数	説明
{$lang->cmd_cancel}	「取り消し」言語変数

XE core 1.4.4 より追加された新しいテンプレート文法を使用しました。新しいテンプレート文法についての詳細な説明は、「XE テンプレート文法」を参照してください。

4.13 権限ガイドページ作成

ユーザーが権限を持たないページへアクセスした際に、権限がないことを知らせ、ログインページへ移動したり、以前ページへ戻ったりするボタンを提供するページです。権限ガイドページは、message.html にて作成します。

下記図にて、例題掲示板スキンで作成した権限案内ページの完成画面をあらわします。

図 4-14 権限案内ページ完成画面

例題掲示板スキンの message.html では、下記のように権限案内ページを作成します。

```
<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" />
```

上記コードで使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明
`<include target="_header.html" />`	_header.html インクルード
`<include target="_footer.html" />`	_footer.html インクルード
`{$message}`	記事に対する閲覧権限がない場合、「権限がありません。」という案内メッセージを表示
`cond="!$is_logged"`	非ログインの場合、含まれている内容を表示
`{getUrl('act','dispMemberLoginForm')}`	ログインページ URL
`{$lang->cmd_login}`	「ログイン」言語変数
`{$lang->cmd_back}`	「戻る」言語変数

XE core 1.4.4 より追加された新しいテンプレート文法を使用しました。新しいテンプレート文法についての詳細な説明は、「XE テンプレート文法」を参照してください。

4.14 パスワード入力ページ作成

ユーザーがパスワード確認の必要なページにアクセスした際に、パスワードを問うページです。非会員ユーザーが自身の書き込んだ記事を修正したり、削除する際に必要です。パスワード入力ページは、input_passowrd_form.html にて作成します。

下記図にて、例題掲示板スキンで作成したパスワード入力ページの完成画面をあらわします。

図 4-15 パスワード入力ページ完成画面

例題掲示板スキンの input_passowrd_form.html では、下記のようにパスワード入力ページを作成しました。

```
<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" />
```

上記コードで使用されたテンプレート文法と変数は、下記の通りです。

XE テンプレート文法/変数	説明
`<include target="_header.html" />`	_header.html インクルード
`<include target="_footer.html" />`	_footer.html インクルード
`onsubmit="return procFilter(this, input_password)"`	入力データ有効性チェック、および form 転送
`{$lang->msg_input_password}`	「パスワードを入力して下さい。」言語変数
`<input type="hidden" name="mid" value="{$mid}" />`	モジュール ID 転送要素(hidden type)
`<input type="hidden" name="page" value="{$page}" />`	ページ番号転送要素(hidden type)

XE テンプレート文法/変数	説明
`<input type="hidden" name="document_srl" value="{$document_srl}" />`	記事固有番号転送要素(hidden type)
`<input type="hidden" name="comment_srl" value="{$comment_srl}" />`	コメント固有番号転送要素(hidden type)
`{$lang->password}`	「パスワード」言語変数
`{$lang->cmd_input}`	「入力」言語変数

XE core 1.4.4 より追加された新しいテンプレート文法を使用しました。新しいテンプレート文法についての詳細な説明は、「XE テンプレート文法」を参照してください。

4.15 CSS の適用

ここでは、掲示板スキンに CSS コードを適用する方法について説明します。本書では、CSS コードの作成方法については、述べません。スキン製作者は、製作意図に合わせて CSS コードを修正して使用します。

1. 例題掲示板スキンの CSS ファイルを作成します。

   下記の例題コードは、user_board.css にて例題掲示板スキン製作の際に使用したクラスリストを構造化したものです。掲示板スキン製作者は、製作意図に合わせて CSS コードを追加して使用することができます。

```
@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.html にて、下記のように user_board.css を参照するよう、宣言します。

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

詳細な説明は、「CSS ファイル参照」を参照してください。

3. 下記のパスにアクセスし、ページに CSS が適用されているかどうかを確認します。下記パスにて、「example.com」は、ユーザーの Web サイトがインストールされているドメインアドレスを意味します。
   - mod_rewrite を使用する場合：http://example.com/test_board/
   - mod_rewrite を使用しない場合：http://example.com/?mid=test_board

作成したコードが画面に反映されていなかったら、CSS を参照する<load />テンプレート文法が正しいかどうか、または CSS 参照パス(target)が正しく指定されているかどうか確認してください。

4.16 JavaScript 適用

ここでは、掲示板スキンに JavaScript コードを追加する方法について説明します。本書では、jQuery 実行コードの作成方法については、述べません。スキン製作者は、製作意図に合ったコードを作成して追加します。

1. user_board.js にて、jQuery 文法のコードを作成します。

```
// <![CDATA[
jQuery(function($){
    // ここに jQuery コードを作成します。
});
// ]]>
```

2. _header.html にて、下記のように user_board.js を参照するよう宣言します。

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

詳細なファイル参照方法については、「JS ファイルの参照」を参照してください。

3. 下記のパスにアクセスし、JavaScript が正常に実行されているかどうか確認します。下記パスにて、「example.com」は、ユーザーの Web サイトがインストールされているドメインアドレスを意味します。
   - mod_rewrite を使用する場合：http://example.com/test_board
   - mod_rewrite を使用しない場合：http://example.com/?mid=test_board

作成したコードが画面に反映されていなかったら、user_board.js を参照する<load />テンプレート文法が正しいかどうか、または user_board.js 参照パス(target)が正しく指定されているかどうか確認してください。

> **参考**
>
> XE における jQuery の基本的な使用方法については、「JavaScript と jQuery の使用」を参照してください。
> jQuery を使用してより多彩な効果を実装するには、http://jquery.com/ Web サイトを参照してください。