目次

1 この資料について

この資料は『CMS Designer』の詳しい利用方法を解説しています。
CMS Designer そのものについての詳細は、Web サイトをご覧下さい。
http://cms.al-design.jp/

また、はじめての方はこの資料を読む前に「CMS Designer チュートリアル１」をお読みになることをお勧めします。

※用語について

当資料中では、CMS Designer を使ってサイトを設計／構築する人を総称して「ウェブデザイナー」又は単に「デザイナー」と呼んでいます。

これは当ツールが、デザイナー自らがプログラマーや SE の力を借りることなしに、独力で「更新可能なウェブサイト」を構築することを目的として作られている為です。

ですから、もし本来の意味でのウェブデザイナーではない、例えばプログラマーなどがこのツールを使う場合は、文中に「この作業はデザイナーが行います」と書いてあったとしても、それはサイトを構築するあなたの事を常に指しています。

「デザイナー」以外に登場する役割としては「サイト管理者」があります。サイト管理者は、完成したウェブサイトを実際に更新していく人のことを指します。

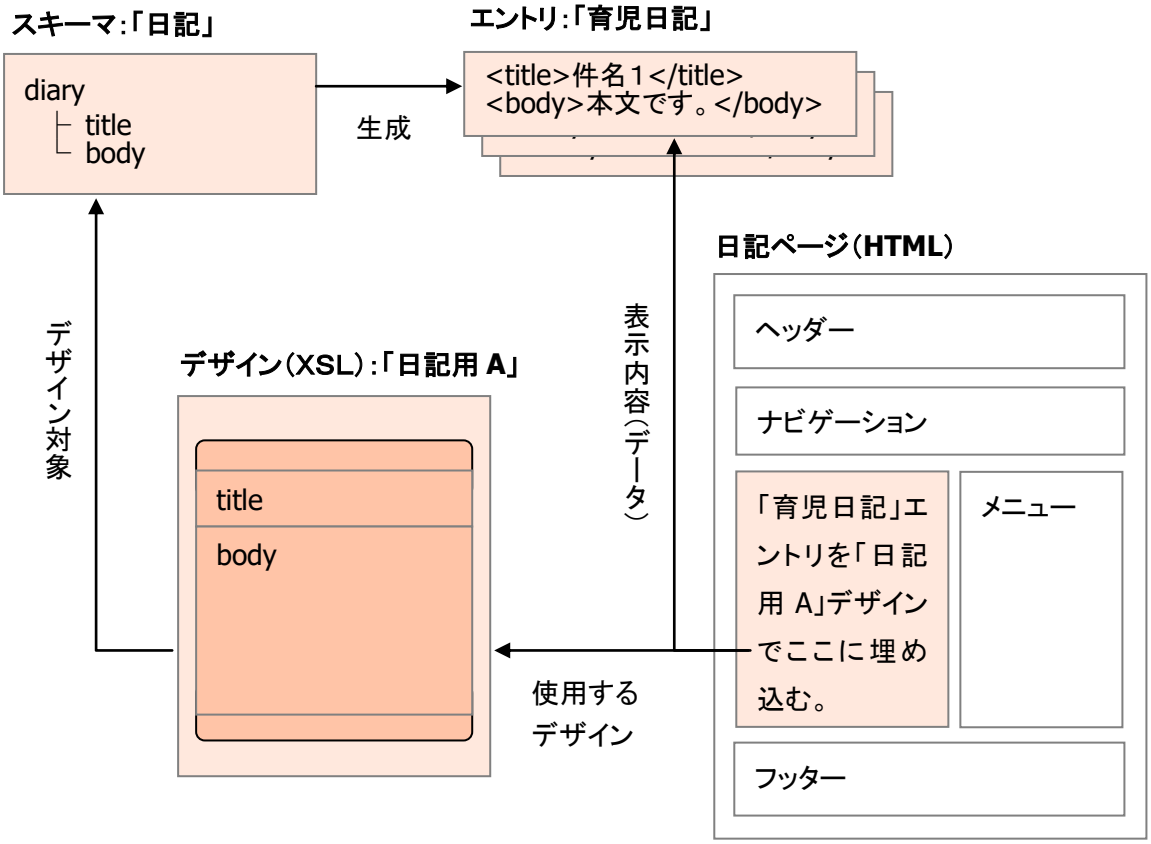
通常は、ウェブサイト納入先の企業の担当者などがそれに該当します。

2 CMS Designer の基本的な仕組み

2．1 概要

CMS Designer の構成を、「日記」を例にして図で表します。

（※diray は、インストールフォルダには含まれていません。あくまで説明用の例です。）



(1) エントリ

投稿し、蓄積していく「データ」そのもの、つまり「コンテンツ」です。

1つの投稿が1つのエントリファイルとして保存されます。このファイルはシステムによって自動的に生成され、サイト管理者やデザイナーが直接編集することはありません。

(2) スキーマ

エントリの「型」となる情報です。「このエントリには、"件名"と"本文"、そして"写真"と"URL"の記入欄がある」というような情報を定義します。HTML の form 定義みたいなものです。

コンテンツ管理画面はこのスキーマの情報を読み込んで、入力画面を構成します。

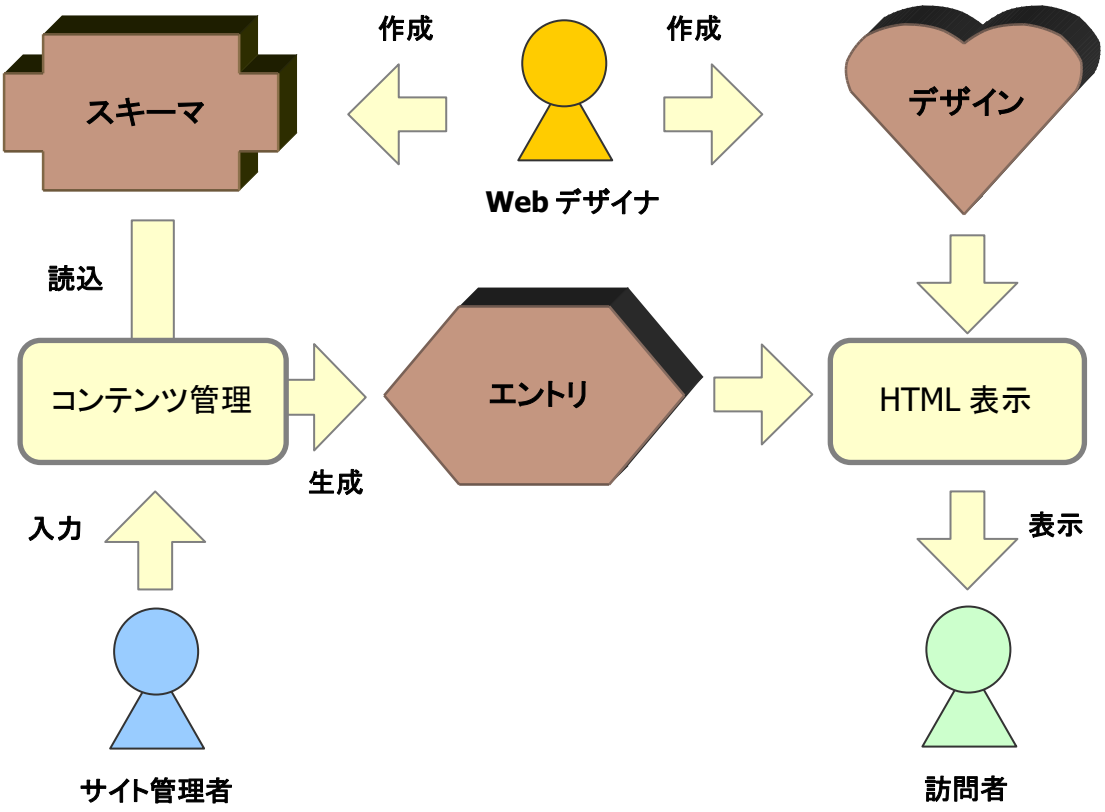
このファイルはデザイナーが作成します。

(3) デザイン

エントリをどのように HTML として表示するかを指定します。テンプレートのようなものです。

一覧表示用、単票表示用、連票表示用など、表示したい方法やシチュエーションに合わせてそれぞれデザインファイルを作成します。デザイナーが作成します。

デザイナーはまずスキーマを作成し、次にそのスキーマをどのように HTML に変換して表示するかをデザインします。最後に、そのデザインを、好きな画面（HTML）に埋め込んで表示します。



図：CMS Designer とその関係者

２．２ 主なフォルダ構成

CMS Designer の主なフォルダ構成は以下の通りです。デザイナーが意識する必要のある箇所は**太字**の部分です。

/cmsdesigner
システムを格納するフォルダです。
/config
site.config.xml
エントリ定義をや全体的な設定ここに記述します。デザイナーが直接編集します。
users.config.xml
ユーザー名やパスワード（暗号化）がここに設定されます。システムが直接更新します。
/schema
スキーマ毎にフォルダを作成して、その中にスキーマ定義とデザイン定義を格納します。
/diary（例）
このように、スキーマ毎にフォルダを作成します。デザイナーが作成します。
diary.schema.xml
スキーマ定義ファイルです。デザイナーが作成します。
diary.default.design.xsl
エントリ1件表示用のデザインファイル（標準用）です。デザイナーが作成します。
diary.list.default.design.xsl
エントリー覧表示用のデザインファイル（標準用）です。デザイナーが作成します。

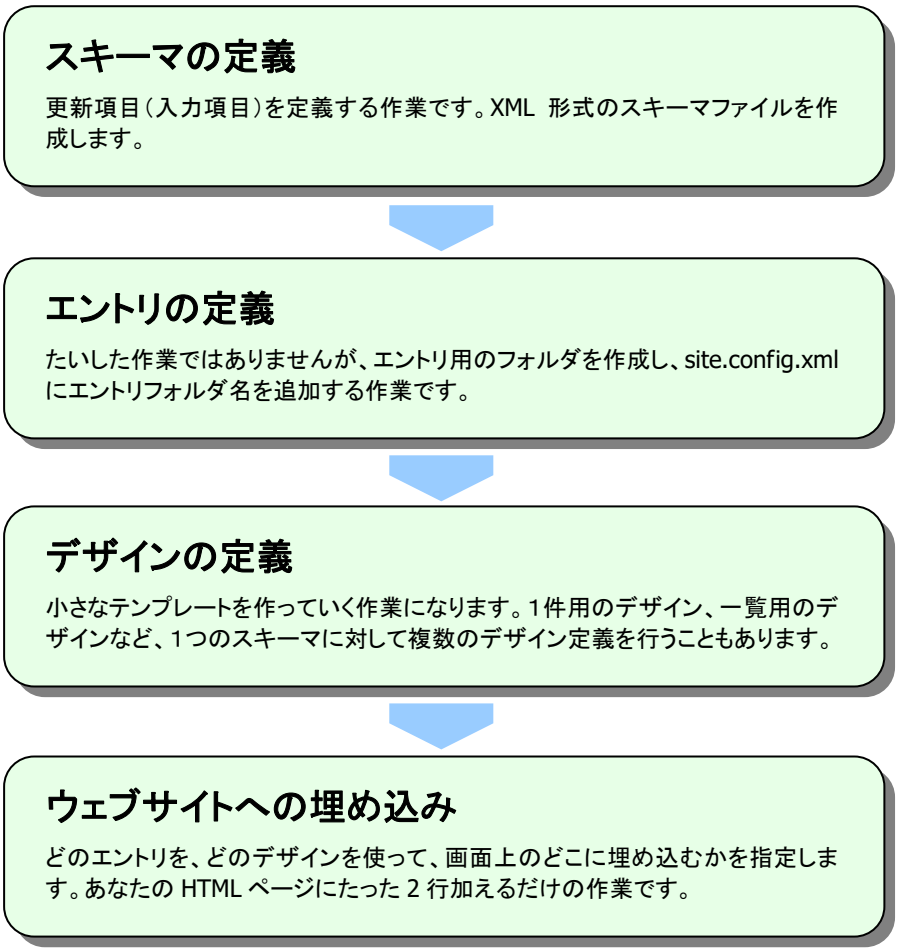
/data
/entry
エントリ毎にフォルダを作成して、その中にエントリデータを格納します。
/baby_diary（例）
このように、エントリ毎にフォルダを作成します。デザイナーが作成します。
フォルダだけ作れば、あとはシステムが自動的にここへエントリデータを追加していきます。

/include
特にデザイナーが意識することはありません。システムの本体部分です。

２．３ 作業の流れ

CMS Designer を使ったサイト構築作業は、大まかに以下の流れで進みます。

最初に「更新させる内容を決めて（スキーマ定義、エントリ定義）」、「その内容をどのように表示するかを決めて（デザイン定義）」、「それをどこに表示するかを決める（埋め込み）」という流れになります。



２．４ XML と文字コードについて

CMS Designer で扱う各ファイルのうち、拡張子が xml、又は xsl となっているファイルは全て XML 形式のデータです。XML 形式のファイルは文字コードを UTF-8 で保存する必要があります。

XML 形式や UTF-8 文字コードについて詳しくない方は、以下の注意事項をお読みください。

コラム：XML って何？

XML について分からない方はご心配なく。HTML と書き方はほとんど同じです。

<name>名前</name>

のように、<タグ名>データ</タグ名>の形式になっているテキストデータのことです。

理解しておく必要がある HTML との違いは次の２点だけです。

① 閉じタグが無いタグの場合について。

HTML の IMG タグ（例：<img src="xxx.jpg">）のように「閉じタグが無い」タグは、 のように書かなければいけません。「>」の前に「/」をつけます。

も、
になります。もちろん、閉じタグとセットにして「
</br>」と書けば OK ですが、ちょっと冗長な書き方ですね。
だけの方がシンプルです。

② 属性は必ず""で囲う

HTML では、<form action=post >のように書けましたが、属性値は必ず""で囲って、<form action="post">と書かなければいけません。

気をつける必要があるのはこれぐらいです。まずは、とりあえずやってみましょう！

[！] 注意：UTF-8 について

CMS Designer では、スキーマファイルやデザインファイルなどの XML ファイルは UTF-8 の文字コード（漢字コード）で保存する必要があります。UTF-8 でファイルを保存するには、Windows 標準添付の「メモ帳でファイルを保存する際に「文字コード」から「UTF-8」を選んで保存すれば OK です。UTF-8 が扱えるエディタとしては Windows 上では「**秀丸エディタ**」などが有名です。

3 スキーマ定義

３．１ スキーマとは何か

スキーマは、エントリの「型」となる情報です。

スキーマを定義する作業は HTML の form タグを作るのに似ています。form タグは input というタグを使って、入力項目を定義します。フォームから送信されたデータは、メールやサーバなどに送られます。

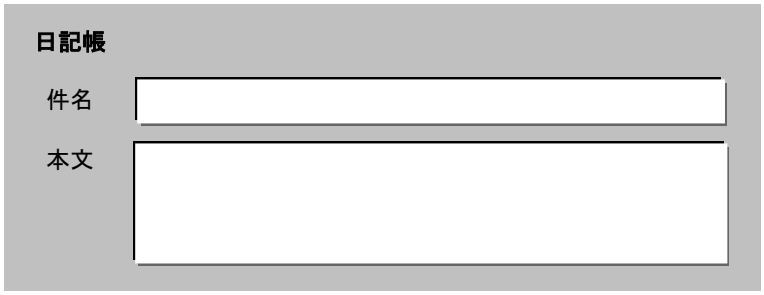
これと同様にスキーマは入力項目を定義し、CMS Designer がそれを HTML フォームとして表示します。フォームから送信されたデータは、エントリデータとしてサーバに保存されます。

スキーマ

```
<?xml version="1.0" encoding="UTF-8"?>
<schema name="diary" caption="日記帳" >
    <data name="title" type="text" caption="件名" />
    <data name="body" type="textarea" caption="本文" />
</schema>
```

対応する

入力項目



３．２ スキーマファイルの作成

まず、スキーマ名を決定します。スキーマ名は半角英数字でつけます。日本語や全角文字は使用できません。例えば日記帳ならば"diary"のようにつけます。

スキーマファイルには、

「スキーマ名」＋「.schema.xml」

の形式でファイル名をつけます。拡張子は"xml"です。

例えばスキーマ名が"diary"なら、"diary.schema.xml"がスキーマファイル名になります。

３．３ スキーマファイルの編集

スキーマファイルは以下のような構成で記述します。

```
<?xml version="1.0" encoding="UTF-8"?>                    ← ヘッダ
<schema name="diary" caption="日記帳" >                    ← スキーマ定義
    <data name="title" type="text" caption="件名" />       ← 項目定義
    <data name="body" type="textarea" caption="本文" />    ← 項目定義
</schema>
```

文字コードは UTF-8 で保存します。

ヘッダはそのままコピーしてください。

スキーマ定義部の詳細はは「３．５ スキーマ リファレンス」を参照してください。

３．４ スキーマファイルの設置

スキーマファイルをサーバに設置する為に、以下の作業を行います。

① /cmsdesigner/config/schema フォルダ以下に、新規にスキーマ名と同名のフォルダを作成してください。

例えばスキーマ名が"diary"なら、/cmsdesigner/config/schema/diary というフォルダを作成します。

② ①で作成したフォルダに、スキーマファイルをアップロードしてください。

以上でスキーマの設定は完了です。

３．５ スキーマ リファレンス

３．５．１ schema タグの設定

schema タグには、以下の属性を設定します。

属性値	省略	内容
name	不可	スキーマ名。半角英数字で設定（先頭は必ず英字）。
caption	不可	日本語名称。
title	可	エントリー覧の見出しに使う項目名。詳しくは「３．５．１５ エントリ見出し項目名の指定」を参照して下さい。

３．５．２ data タグの種類

data タグには以下の種類があります。詳細についてはそれぞれの種別(type)の詳細説明を参照してください。

種別(type)	説明
text	１行テキスト項目。
textarea	複数行テキスト項目。
int	整数値項目。
menu	ドロップダウンメニューからの選択式入力項目。
img	画像アップロード項目。
file	ファイルアップロード項目。
date	日付項目。西暦年＋月＋日の入力用。
time	時刻項目。時（24H）＋分の入力用。
datetime	日時項目。西暦年＋月＋日＋時（24H）＋分の入力用。
year	年項目。西暦年の入力用。
month	年月項目。西暦年＋月の入力用。
list	リスト項目。
separator	※特殊。入力項目ではなく、見た目上の区切り。

3．5．3 data タグ - text

text 項目は、1行分のテキストを入力するのに適しています。

何かの名称、ちょっとしたコメントなどの入力にはこれを使用すると良いでしょう。

text 項目の場合には、data タグに以下の属性を設定します。

属性値	省略	内容
name	不可	データ名。半角英数字で設定します（先頭は必ず英字）。 同一スキーマ内で一意の（他とかぶらない）名前をつけてください。
type	不可	"text"と指定します。
caption	不可	日本語名称。入力項目の横にラベルとして表示されます。サイト管理者が分かりやすい名称をつけてください。
size	可	テキストボックスの大きさを指定します。HTML の size 属性と同じです。
maxlength	可	入力できる文字の数に制限を設けたい時に指定します。 省略した場合、無制限になります。 size 省略時の入力欄のテキストボックスの大きさにも影響します。
minlength	可	入力しなければならない最低文字数を指定します。1 以上にすれば必須入力項目になります。指定しなかった場合は 0 になります。
output	可	入力されたテキストデータの HTML への出力方法を指定します。 -"**text1**" 入力された内容をそのまま出力します。HTML タグなどが入力された場合もそのまま出力しますので、場合によっては HTML が崩れることがあります。HTML タグなどが入力されても問題ない項目に指定してください。 -"**text2**" 通常はこれを指定してください。 改行を自動的に BR タグに変換して出力します（自動改行）。サイト管理者がいちいち BR タグを使って改行する必要はありません。 また、"<"や">"、"&"など、HTML として表示できない文字を自動的に"&"などのコードに変換して出力します。 サイト管理者が不用意にこれらの文字を入力してしまっても、HTML が崩れることはありません。 -"**html1**" 入力された内容をそのまま HTML として出力します。HTML つまり、HTML タグが入力されたら、そのタグが有効になります。HTML タグを有効にしたい場合に使用してください。 サイト管理者に HTML の知識がある場合のみ、注意してこれを利用してください。 -"**html2**" 基本は html1 と同じ動作ですが、改行を BR タグ（ ）に変換します（自動改行）。HTML を有効にしつつ、サイト管理者が BR をいちいち記述する手間を省きます。 サイト管理者に HTML の知識がある場合のみ、注意してこれを利用してください。 尚、以下の場合には行末に BR タグを付けません。 ・行末が">"で終わっている。 ・又は、<pre></pre>で囲ってある範囲。 これは、例えばTABLEタグなどを入力した場合には行末に が付くと困るし、PREタグの範囲は BR をつける必要が無い為です。
initdata	可	エントリの新規作成時に初期値として入力される値を指定します。[ver.1.2.1a 以降]

例） <data name="url" type="text" caption="ホームページ" maxlength="1000" />

3．5．4 data タグ - textarea

textarea 項目は、複数行のまとまったテキストを入力するのに適しています。

本文や説明文などの入力にはこれを使用すると良いでしょう。

textarea 項目の場合には、data タグに以下の属性を設定します。

※type 属性以外、text 項目と同じです。

属性値	省略	内容
name	不可	データ名。半角英数字で設定します（先頭は必ず英字）。 同一スキーマ内で一意の（他とかぶらない）名前をつけてください。
type	不可	"textarea"と指定します。
caption	不可	日本語名称。入力項目の横にラベルとして表示されます。サイト管理者が分かりやすい名称をつけてください。
cols	可	テキストボックスの幅を指定します。HTML の属性と同じです。
rows	可	テキストボックスの高さを指定します。HTML の属性と同じです。
maxlength	可	入力できる文字の数に制限を設けたい時に指定します。 省略した場合、無制限になります。 cols、rows 省略時の入力欄のテキストボックスの大きさにも影響します。
minlength	可	入力しなければならない最低文字数を指定します。1 以上にすれば必須入力項目になります。指定しなかった場合は 0 になります。
output	可	※text 項目と同じです。
autolink	可	"True"を設定すると、入力された文章中の URL と思われる部分を自動的にリンクに変換します。output 属性が"text2"又は"html2"の場合のみ有効です。
initdata	可	エントリの新規作成時に初期値として入力される値を指定します。 [ver.1.2.1a 以降]

```
例）<data name="description2" type="textarea" caption="所在地"
      output="html1" maxlength="100" />
```

3．5．5 data タグ - int

int 項目は、整数値を入力するのに適しています。

個数、評価値、番号などの入力にはこれを使用すると良いでしょう。

int 項目の場合には、data タグに以下の属性を設定します。

属性値	省略	内容
name	不可	データ名。半角英数字で設定します（先頭は必ず英字）。 同一スキーマ内で一意の（他とかぶらない）名前をつけてください。
type	不可	"int"と指定します。
caption	不可	日本語名称。入力項目の横にラベルとして表示されます。サイト管理者が分かりやすい名称をつけてください。
min	可	入力できる範囲の下限を指定します。 0 以上の値を指定できます。 例えば"5"と指定すると、4 以下の値は入力できなくなります。
max	可	入力できる範囲の上限を指定します。 0 以上の値を指定できます。 例えば"100"と指定すると、101 以上の値は入力できなくなります。
initdata	可	エントリの新規作成時に初期値として入力される値を指定します。 [ver.1.2.1a 以降]

```
例） <data name="count" type="int" caption="個数" min="0" max="50" />
```

３．５．６ data タグ - menu

menu 項目は、選択項目を入力するのに適しています。

何かの種別、カテゴリ名などの選択入力にはこれを使用すると良いでしょう。

menu 項目の場合には、data タグに以下の属性を設定します。

属性値	省略	内容
name	不可	データ名。半角英数字で設定します（先頭は必ず英字）。 同一スキーマ内で一意の（他とかぶらない）名前をつけてください。
type	不可	"menu"と指定します。
caption	不可	日本語名称。入力項目の横にラベルとして表示されます。サイト管理者が分かりやすい名称をつけてください。
initdata	可	エントリの新規作成時に初期値として入力される値を指定します。 [ver.1.2.1a 以降]

また、選択項目の値として、menuitem タグを子に持ちます。

例えば、以下のように設定します。

```
<data name="shopkind" type="menu" caption="お店種別" >
  <menuitem id="1">中華</menuitem>
  <menuitem id="2">ラーメン</menuitem>
  <menuitem id="3">和食</menuitem>
  <menuitem id="4">洋食</menuitem>
  <menuitem id="5">スイーツ</menuitem>
</data>
```

menuitem タグの id 属性に指定されている値が、実際にデータとして保存されます。

３．５．７ data タグ - img

img 項目は、画像ファイルをアップロードさせるのに用います。

img 項目の場合には、data タグに以下の属性を設定します。

属性値	省略	内容
name	不可	データ名。半角英数字で設定します（先頭は必ず英字）。 同一スキーマ内で一意の（他とかぶらない）名前をつけてください。
type	不可	"img"と指定します。
caption	不可	日本語名称。入力項目の横にラベルとして表示されます。サイト管理者が分かりやすい名称をつけてください。
alt	可	HTML の img タグの alt 属性を入力させるかどうかを指定します。省略した場合、"False"になります。 -"False" 　入力させない。 -"True" 　入力させる。
maxfilesize	可	アップロードする画像ファイルのサイズを指定します。 単位は「byte」です。1KB(KBytes)=1024bytes で計算します。 "20KB"や"3MB"のように書くこともできます。
width	可	アップロードする画像の幅が指定した width より大きかった場合、自動的に width まで幅を縮小します（縦横比は保持されます）。
height	可	アップロードする画像の高さが指定した height より大きかった場合、自動的に height まで高さを縮小します（縦横比は保持されます）。

例）<data name="shopphoto" type="img" caption="写真" />

３．５．８ data タグ - file

file 項目は、画像以外の添付ファイルをアップロードさせるのに用います。

PDF ファイルや各種資料ファイルなどはこの項目を使用してください。

file 項目の場合には、data タグに以下の属性を設定します。

属性値	省略	内容
name	不可	データ名。半角英数字で設定します（先頭は必ず英字）。 同一スキーマ内で一意の（他とかぶらない）名前をつけてください。
type	不可	"file"と指定します。
caption	不可	日本語名称。入力項目の横にラベルとして表示されます。サイト管理者が分かりやすい名称をつけてください。
maxfilesize	可	アップロードするファイルのサイズを指定します。 単位は「byte」です。1KB(KBytes)=1024bytes で計算します。 "20KB"や"3MB"のように書くこともできます。

例）<data name="file1" type="file" caption="添付ファイル" />

３．５．９ data タグ - date, time, datetime, year, month

date, time, datetime, year, month 項目は、日時を入力するのに用います。

年月日を入力したい場合は date、時刻を入力したい場合は time、日時を入力したい場合は datetime を使います。

また、西暦年のみを入力したい場合は year、西暦年と月を入力したい場合は month を使ってください。

これらの日時項目は、初期値として現在の日時が設定されます。

日時項目の場合には、data タグに以下の属性を設定します。

属性値	省略	内容
name	不可	データ名。半角英数字で設定します（先頭は必ず英字）。 同一スキーマ内で一意の（他とかぶらない）名前をつけてください。
type	不可	"date"又は"time"又は"datetime"又は"year"又は"month"と指定します。
caption	不可	日本語名称。入力項目の横にラベルとして表示されます。サイト管理者が分かりやすい名称をつけてください。

例）<data name="birthday" type="date" caption="生年月日" />

３．５．１０ data タグ - geolocation

※Ver.1.1.5a 以降で使用可能です。

geolocation 項目は、地図の緯度経度情報を入力する際に使用します。

使用の為には Google Maps API が必要です。詳しい使用方法は、CMS Designer 公式サイトのダウンロードページより「Google Maps API v3 を使った地図表示サンプル」をダウンロードして、何ある説明テキストファイルを参照してください。

geolocation 項目の場合には、data タグに以下の属性を設定します。

属性値	省略	内容
name	不可	データ名。半角英数字で設定します（先頭は必ず英字）。 同一スキーマ内で一意の（他とかぶらない）名前をつけてください。
type	不可	"geolocation"と指定します。
caption	不可	日本語名称。入力項目の横にラベルとして表示されます。サイト管理者が分かりやすい名称をつけてください。

例） <data name="location" type="geolocation" caption="所在地" />

３．５．１１ data タグ - separator

separator は、data タグの中で唯一「意味のないタグ」です。このタグはコンテンツ管理画面上で各種項目の間に「区切り（横線）」を入れて見やすくする為のもので、入力内容にはまったく影響を与えません。

例えば「基本情報」と「追加情報」のようにセクションを分けたい場合などに、区切りを使って領域を分けることができます。

name 属性と type 属性だけを指定した場合、細い横線が入ります。caption 属性を指定すると、大きな文字でその caption が区切り線内に表示されます。description 属性を指定すると、小さな文字でその description が区切り線内に表示されます。caption と description を両方同時に指定することもできます。用途によって使い分けて下さい。

区切り項目の場合には、data タグに以下の属性を設定します。

属性値	省略	内容
name	不可	データ名。半角英数字で設定します（先頭は必ず英字）。 同一スキーマ内で一意の（他とかぶらない）名前をつけてください。
type	不可	"separator"と指定します。
caption	可能	区切り線の内部に大きな文字で見出しを表示します。separator を「大見出し」代わりに使いたい場合に指定します。
description	可能	区切り線の内部に小さな文字で説明文を表示します。 separator を「小見出し」又は「補足表示」代わりに使いたい場合に指定します。

```
例） <data name="separator1" type="separator" caption="基本情報" description="必ず入力する必要がある項目群です。" />
```

3．5．12 data タグ - list

list 項目は、繰り返し項目を入力する際に使用する特殊なタグです（使用するケースは稀ですので、必要無い場合は読み飛ばしてください）。list 項目の中に、繰り返したい項目を定義します。

例えば、画像データを1つのエントリの中にたくさん登録した場合、このような方法を考えるかもしれません。

```
<?xml version="1.0" encoding="UTF-8"?>
<schema name="写真たくさん" caption="写真たくさん" >
  <data name="title" type="text" caption="タイトル" />
  <data name="photo1" type="img" caption="写真 1" />
  <data name="photo2" type="img" caption="写真 2" />
  <data name="photo3" type="img" caption="写真 3" />
  <data name="photo4" type="img" caption="写真 4" />
  <data name="photo5" type="img" caption="写真 5" />
  <data name="photo6" type="img" caption="写真 6" />
</schema>
```

入力画面には、画像ファイルのアップロード項目が 6 つ並ぶことになります。

しかしこの方法だと、最大 6 件まで画像を登録することができますが、逆に言えば 7 件以上は登録できません。

list 項目を使うと、「0 件～何件でも」同じ項目を繰り返し登録させることができます。

```
<?xml version="1.0" encoding="UTF-8"?>
<schema name="写真たくさん" caption="写真たくさん" >
  <data name="title" type="text" caption="タイトル" />
  <data name="shopphotolist" type="list" caption="写真一覧" >
    <listitem caption="写真の説明文" >
      <data name="photo" type="img" caption="写真" />
      <data name="photodesc" type="textarea" caption="写真の説明文" />
    </listitem>
  </data>
</schema>
```

list 項目の場合には、data タグに以下の属性を設定します。

属性値	省略	内容
name	不可	データ名。半角英数字で設定します（先頭は必ず英字）。 同一スキーマ内で一意の（他とかぶらない）名前をつけてください。
type	不可	"list"と指定します。
caption	不可	日本語名称。入力項目の横にラベルとして表示されます。サイト管理者が分かりやすい名称をつけてください。

繰り返したい入力項目（data タグ）を、listitem タグで囲みます。尚、listitem タグの中には複数の項目を含めることができます。この中にさらに list 項目を含めることも可能です。

listitem 要素に title 属性を指定することで、コンテンツ管理画面のリスト項目一覧の見出しに表示する内容を任意に指定することができます。

```
<?xml version="1.0" encoding="UTF-8"?>
<schema name="写真たくさん" caption="写真たくさん" >
  <data name="title" type="text" caption="タイトル" />
  <data name="shopphotolist" type="list" caption="写真一覧" >
    <listitem caption="写真の説明文" title="photodesc" >
      <data name="photo" type="img" caption="写真" />
      <data name="photodesc" type="textarea" caption="写真の説明文" />
    </listitem>
  </data>
</schema>
```

上記の例では、photodesc 項目の内容が見出しとして利用されます。

title 属性を省略した場合には、list 項目内の内部項目のうち先頭の項目が見出しとして使用されます。

また、title属性にはカンマ区切りで複数の項目を指定することができます。その場合には各項目の値が「｜」で区切られて、指定した順序で表示されます。

3．5．13 ソート（並べ替え）指定

エントリを登録すると、通常は更新された順に（降順で）ソート（並べ替え）されます。

つまり、新しいエントリほど先頭に来ます。

しかし、場合によっては「件名をあいうえお順に」という場合もあったり、「商品番号順に」という場合もあるでしょう。

CMS Designer では、どの項目でソートするか、昇順か降順かを指定することができます。

例えば、次のようなスキーマがあるとして、

```
<?xml version="1.0" encoding="UTF-8"?>
<schema name="diary" caption="日記帳" >
    <data name="title" type="text" caption="件名" />
    <data name="body" type="textarea" caption="本文" />
</schema>
```

「件名で昇順にソートしたい」という場合は、次のように、schema タグに sortkey 属性と sortorder 属性を追加します。

```
<?xml version="1.0" encoding="UTF-8"?>
<schema name="diary" caption="日記帳" sortkey="title" sortorder="asc" >
    <data name="title" type="text" caption="件名" />
    <data name="body" type="textarea" caption="本文" />
</schema>
```

sortkey 属性には、ソート項目にしたい項目名を指定します。省略すると更新日（@date）でソートされます。

sortorder 属性には、昇順か降順かで以下のように設定します。省略すると"desc"になります。

sortorder 属性	説明
"asc"	昇順でソートする。例）4,7,10,11,15...
"desc"	降順でソートする。例）15,11,10,7,4...

尚、sortkey として指定できるデータ項目は、スキーマのルート階層（schema タグの直下の階層）のデータ項目だけです。また、複数のソート項目を指定することはできません。

[！]注意

ソート機能は、エントリ投稿時にソートが実行されるものです。表示の際にソート順を指定して「昇順／降順」を入れ替えたりというような事はできません。

また、ソート項目を設定してエントリを投稿した後でソート項目やソート順を変更しても、既存のエントリの並び順は変更されません。その場合は既存のエントリを全て保存しなおせば正しくソートされますが、基本的には運用中のソート項目の変更は**しないでください**。

３．５．１４ グループ（絞込み）指定

エントリー覧に全てのエントリを表示するのではなく、全エントリの中から「この種類のエントリだけ」を一覧したい場合があります。

例えば「商品」エントリを、「商品種別」が「雑貨」のものだけ表示したい、という場合などです。

CMS Designer には「グループ」という、簡単な絞込み機能があります。

例えば、次のようなスキーマがあるとして、

```
<?xml version="1.0" encoding="UTF-8"?>
<schema name="shop" caption="お店情報" >
    <data name="shopname" type="text" caption="店名" maxlength="10" />
    <data name="url" type="text" caption="ホームページ" />
    <data name="description" type="textarea" caption="説明" output="text2" maxlength="100" />
    <data name="shopkind" type="menu" caption="お店種別">
        <menuitem id="1">中華</menuitem>
        <menuitem id="2">ラーメン</menuitem>
        <menuitem id="3">和食</menuitem>
        <menuitem id="4">洋食</menuitem>
        <menuitem id="5">スイーツ</menuitem>
    </data>
</schema>
```

このスキーマのエントリの中から「お店種別」が「中華」や「ラーメン」などのエントリだけを一覧表示するには、まず、「お店種別」項目をグループに指定します。

```
    <data name="shopkind" type="menu" caption="お店種別" group="True">
```

上記の太字の部分のように、データ項目に group 属性を追加し、値を"True"と記入します。

実際に絞込みをする方法については6章を参照してください。

※ソート項目と同じく、グループ項目も運用中に変更することは**できません**。

※group 属性が指定できるデータ項目は、**スキーマのルート階層**（schema タグの直下の階層）のデータ項目だけです。

※group 属性は、複数の項目に指定して使用できます。使用する際には、異なる二つのグループに絞込み条件を指定することもできます（例えば、「お店種別」が"和食"で且つ、「評価」が"5"のエントリ、など）。

※グループを増やせば増やすほど処理のパフォーマンスは落ちていきます。絞込み機能を使わなくても、グループ項目が存在するだけで多少負荷がかかります。

※textarea 項目をグループ項目に指定するのは極力避けてください。グループ指定された項目は、index ファイルという絞込み検索用の別ファイルにデータ内容が全てコピーされます。

※グループ項目はあくまでも「〇〇項目の値が△△のもの」という絞込みしかできません。「△△でないもの」とか「△△以下」などの複雑な絞込みはできませんのでご了承ください。

３．５．１５ エントリ見出し項目名の指定

既定では、コンテンツ管理画面のエントリー覧の「エントリ見出し」には「スキーマの最初に指定された項目の内容」が表示されます。

例えば以下のような場合には、エントリ見出しとして shopname の内容が表示されます。

```
<?xml version="1.0" encoding="UTF-8"?>
<schema name="shop" caption="お店情報" >
    <data name="shopname" type="text" caption="店名" maxlength="10" />
    <data name="url" type="text" caption="ホームページ" />
    <data name="description" type="textarea" caption="説明" output="text2" maxlength="100" />
    <data name="shopkind" type="menu" caption="お店種別">
        <menuitem id="1">中華</menuitem>
        <menuitem id="2">ラーメン</menuitem>
        <menuitem id="3">和食</menuitem>
        <menuitem id="4">洋食</menuitem>
        <menuitem id="5">スイーツ</menuitem>
    </data>
</schema>
```

しかし、サイト管理者によっては「お店の名前ではなく、説明の内容を表示して欲しい」というような場合もあるかもしれません。また、「shopname 項目より前に shopkind 項目を配置して欲しい」という要望も有り得ます。shopkind 項目を先頭に持ってくると、エントリー覧には「中華」とか「ラーメン」などが並ぶ事になり、非常に困ります。

そんなときはエントリ見出しとして用いたい項目名を明示的に指定することができます。
schema タグの title 属性に、エントリ見出し用のデータ項目名を指定します。

```
<?xml version="1.0" encoding="UTF-8"?>
<schema name="shop" caption="お店情報" title="description">
    <data name="shopname" type="text" caption="店名" maxlength="10" />
    <data name="url" type="text" caption="ホームページ" />
    <data name="description" type="textarea" caption="説明" output="text2" maxlength="100" />
    <data name="shopkind" type="menu" caption="お店種別">
        <menuitem id="1">中華</menuitem>
        <menuitem id="2">ラーメン</menuitem>
        <menuitem id="3">和食</menuitem>
        <menuitem id="4">洋食</menuitem>
        <menuitem id="5">スイーツ</menuitem>
    </data>
</schema>
```

この title 属性には、複数のデータ項目名をカンマ区切りで指定することも可能です。複数指定した場合には、「｜」で各項目が連結されたものが見出しとして使用されます。

```
<schema name="shop" caption="お店情報" title="shopname,shopkind,description">
```

尚、見出しは最長５０文字でカットされて表示されます。

4 エントリ定義

4．1 エントリ定義とは何か

スキーマは単なるデータの「型」である為、実際にエントリを登録する為には、指定したスキーマ（型）を使って生成したエントリを保管してゆく「場所」を作らなければなりません。

それが「エントリ定義」です。

4．2 エントリ定義手順

エントリ定義は二つの作業が必要になります。

① エントリ保存用フォルダの作成とパーミッションの設定。
② site.config.xml へのエントリ定義の追加。

4．3 エントリ保存用フォルダの作成とパーミッションの設定

エントリ保存用フォルダを以下の場所に作成し、パーミッションを"707"以上に設定します。フォルダ名は「エントリフォルダ名」として各所で使いますので、覚えておいてください。

/cmsdesigner/data/entry

例えば"mydiary"というエントリフォルダ名ならば、

/cmsdesigner/data/entry/mydiary　**・・・(※パーミッションを 707 に設定)**

というフォルダを新規に作成します。

作成後、mydiary フォルダのパーミッションを"707"以上に設定してください。

パーミッションの設定作業は忘れやすいのでご注意ください。

4．4 site.config.xml へのエントリ定義の追加

次に、/public-html/cmsdesigner/config/site.config.xml ファイルをエディタで編集します。このファイルも UTF-8 の文字コードですので、UTF-8 対応のエディタで編集してください。

以下は、site.config.xml の例です。

site.config.xml（例）

```
<?xml version="1.0" encoding="UTF-8"?>
<site>
  <entries>
    <entry name="news1" schema="news" caption="新着情報" />
    <entry name="news2" schema="news" caption="お知らせ" />
  </entries>
</site>
```

entries タグの中に設定されている entry タグが、それぞれエントリ定義です。

entry タグの設定内容

属性値	省略	内容
name	不可	エントリフォルダ名。半角英数字で設定します（先頭は必ず英字）。システム内で一意の（他とかぶらない）名前をつけてください。
schema	不可	このエントリ定義の型となるスキーマ名を指定します。
caption	不可	日本語名称。エントリ選択の際の見出しとして表示されます。サイト管理者が分かりやすい名称をつけてください。
publishcontrol	可	エントリ毎の公開／非公開を管理するかどうかを設定します。 省略すると"on-off"を指定したことになります。 **"full"** - 「公開／非公開」に加え、いつからいつまで公開、のような公開予約を指定することができます。 **"on-off"** - 「公開／非公開」を設定できるようにします。また、fullよりも若干処理速度が向上します。公開予約までは必要ない場合はこちらを指定して下さい。 **"none"** - 公開管理を行いません。全てのエントリが自動的に公開されます。その代わり、処理速度が向上します。処理速度が気になる場合はこちらを指定して下さい。
entrypageurl	可	エントリ 1 件表示を行うページの URL を指定する事で、コンテンツ管理画面から「表示の確認」をすることができるようになります。 "http://"から指定してください。 「表示の確認」では、非公開エントリも表示させることができる為、簡易的なプレビューとしても利用できて便利です。 *例）entrypageurl="http://cms.al-design.jp/newsentry.php"*
listpageurl	可	エントリー覧表示を行うページの URL を指定する事で、コンテンツ管理画面から「表示の確認」をすることができるようになります。 "http://"から指定してください。 *例）listpageurl="http://cms.al-design.jp/newslist.php"*

前のページの例で、同じスキーマに対して二つのエントリ定義が入っていることに注目してください。この例では、news というスキーマを元にして news1 と news2 というエントリ定義を作っています。

片方は新着情報としてサイトの更新情報を、もう片方はお知らせとしてお店からのお知らせを配信する為に用いることができます。

このように、1つのスキーマに対して複数のエントリ定義を作ることができます。スキーマは「型」ですから、その型を元に別の用途のエントリ定義を作ることで、スキーマを再利用することができます。

4．5「区切り」表示用のエントリ定義について

補足的な機能として、コンテンツ管理画面のエントリフォルダー覧（エントリ定義一覧）に表示する際の「区切り」を設定することができます。

例えば、サイトに「商品関係」のエントリ定義が4種類、「お知らせ関係」のエントリ定義が2種類、「お客様の声」のエントリ定義が3種類あったとしたら、エントリフォルダー覧には9個のエントリ定義が並ぶ事になり、サイト管理者にとって多少見づらい画面になってしまいます。
そこで、それらのエントリ定義のグループの間に区切りを入れてやることで、見やすい画面を作ることができます。

区切りの設定は、エントリ定義と同じ entry タグを用います。但し、以下のように属性を設定する必要があります。表示の為の設定なので、schema 属性は不要です。

name 属性と type 属性だけを指定した場合、細い横線が入ります。caption 属性を指定すると、大きな文字でその caption が区切り線内に表示されます。description 属性を指定すると、小さな文字でその description が区切り線内に表示されます。caption と description を両方同時に指定することもできます。用途によって使い分けて下さい。

「区切り」の場合の entry タグの設定内容

属性値	省略	内容
name	不可	半角英数字で設定します（先頭は必ず英字）。本来不要ですが、他の entry タグと合わせる為、"separator1"などの適当な名前をつけて下さい。 システム内で一意の（他とかぶらない）名前をつけてください。
type	不可	"separator"と指定します。
caption	可能	区切り線の内部に大きな文字で見出しを表示します。 separator を「大見出し」代わりに使いたい場合に指定します。
description	可能	区切り線の内部に小さな文字で説明文を表示します。 separator を「小見出し」又は「補足表示」代わりに使いたい場合に指定します。

例） <entry name="separator1" type="separator" caption="商品関係" description="各種商品情報の追加や削除、更新の際に編集します。" />

4．6 画像キャッシュの指定

CMS Designer にはアップロードした画像を表示する際に指定した width と height で縮小して表示する機能があります。サムネイルなどを別途用意する必要がない為便利な機能ですが、サーバへ要求がある度に毎回画像を縮小する処理が実行される為、サーバの負荷が高く、一度に大量のサムネイルを表示するようなページを作った場合には表示の遅れが目立つことがありました。

これを回避する為、一度生成した縮小画像をキャッシングしておき、次に同じ width と height で縮小表示要求があった場合には、元画像から縮小処理を行うのではなく、キャッシュ画像をそのままブラウザへ送り返す機能を利用することができます。

画像キャッシュの指定はエントリフォルダ単位に行います。例えば、次のような例で説明します。

とあるデザイン定義にて、次のような画像縮小表示を行っているとします。

```
例） <img src="{photo}&amp;amp;w=120&amp;amp;h=180" />
```

width=120、height=180 の範囲内に収まるように画像が縮小して表示されます。この縮小画像をキャッシングする場合には、site.config.xml の entry タグに次のような imagecache タグを追加します。

```
<?xml version="1.0" encoding="UTF-8"?>
<site>
  <entries>
      <entry name="news1" schema="news" caption="新着情報" >
        <imagecache width="120" height="180" />
      </entry>
  </entries>
</site>
```

entry タグが、これまでのように /> で終わる空要素ではなく、開始タグと終了タグに分かれている点に注意して下さい。entry の開始タグと終了タグの間に、imagecache タグを記述します。

画像キャッシュが作成されるのは、この例では news1 のエントリフォルダ内の画像に対して「w=120&h=180」の縮小表示指定がされた場合のみです。

全ての画像に対して自動的にキャッシュを作るような仕様になっていない理由は、悪意のある攻撃によって大量の（ちょっとサイズを変えただけの）画像キャッシュが勝手に作成されるようなことがないようにする為です。

imagecache タグの width と height は、どちらか片方だけを指定することもできます。width="120" とだけ指定した場合は、同様に「w=120」の縮小指定がされた場合のみキャッシュが作成されます（「w=120&h=180」のように両方指定された場合には作成されません）。また、width="all"のように記述することで、全ての縮小幅指定に対して画像キャッシュを作るように指示することもできます。"all"を指定した場合、先ほどの攻撃に対して脆弱になりますので注意して下さい。

画像キャッシュは以下のフォルダに自動的に作成されます。cache フォルダのパーミッションは 707 以上に設定して下さい。

```
cmsdesigner/cache/      (707)
```

作成された画像キャッシュは、対象となる画像を含むエントリが更新されたり削除された時などに自動的に削除されます。

4．7 ユーザー権限

4．7．1 ユーザー権限について

エントリフォルダ単位で、「このエントリフォルダはこのユーザーしか見れない」「一覧表示はできるが編集はできない」「編集はできるが新規追加や削除はできない」といったユーザー権限を設定することができます。これにより、複数人でサイトを更新する場合でも、あるユーザーには編集させたくないようなエントリフォルダを作ることが出来ます。

4．7．2 ユーザー権限の指定方法

指定方法は、個々のエントリフォルダに対して「ユーザー1 は、一覧表示と編集ができる」のような形で指定します。何も指定しなかった場合、そのエントリフォルダは「誰もが管理者権限を持つ状態」となります。尚、管理者権限を持つユーザーは、どのエントリフォルダに対しても必ず全ての権限を持ちます。

指定は site.config.xml の entry 要素の子要素として指定します。

例えば、user1、user2、user3、及び admin の 4 つのユーザーが存在し（admin 以外は全員「編集者ユーザー」です）、新着情報のエントリフォルダについては「user1 以外には全く触らせたくない」という場合、次のように指定します。

例）user1 に news1 へのアクセス権限を与える

```
<entry name="news1" schema="news" caption="新着情報" >
  <permit users="user1" />
</entry>
```

ここが /> になっているとエラーになるので注意。

permit 要素は、users に指定したユーザーに対して、そのエントリフォルダにアクセスする権限を与える設定を行う為の要素です。上記の指定だと、user2 と user3 は、このエントリフォルダへアクセスすることができなくなります（コンテンツ一覧自体にもこのエントリフォルダ名が表示されません）。

管理者ユーザーである admin については、指定がなくとも全てのエントリフォルダへアクセスすることが可能です。

複数のユーザーに同時に権限を与える場合は、次のようにカンマ区切りで複数のユーザーID を指定することができます。

例）user1 と user2 に news1 へのアクセス権限を与える

```
<entry name="news1" schema="news" caption="新着情報" >
  <permit users="user1,user2 " />
</entry>
```

4．7．3 詳細な指定方法

もう少し細かい権限設定を行う事もできます。

「一覧表示」「新規追加」「編集」「削除」の権限を、「自分が投稿したエントリ」「他の人が投稿したエントリ」のそれぞれに対して設定することができます。

例えば、「そのユーザーの投稿したエントリについては全ての操作が可能だが、他の人の投稿したエントリについては一覧にすら表示させない」という指定は、次のようにします。

```
<entry name="news1" schema="news" caption="新着情報" >
  <permit users="user1" mine="all" others="none" />
</entry>
```

こうすると、user1 が「新着情報」エントリフォルダを開くと、自分（user1）が投稿したエントリのみが一覧に表示されるようになります。

mine 属性には、「自分が投稿したエントリに対する権限」を指定します。others 属性には「他の人が投稿したエントリに対する権限」を指定します。"all"を指定すると、全ての権限を与える事になります。"none"を指定すると、全ての権限を与えないという意味になります。

指定できる権限の種類は次の通りです。カンマ区切りで複数指定できます。

permit タグの mine 属性、others 属性の設定内容

属性値	意味
list	エントリー覧にエントリを表示できる。 ※edit,add,delete のいずれかを指定した場合は、list は省略可能
edit	エントリを編集／保存できる。
add	エントリを新規追加できる。
delete	エントリを削除できる。
all	全ての権限を許可する。
none	全ての権限を許可しない。
deny	全ての権限を拒否する。
（省略時）	省略時は"all"を指定した事になります。

mine 属性、others 属性に空文字を指定したり属性そのものを省略したりすると、"all"を指定したのと同じ意味になります。"none"ではありませんのでご注意下さい。

例えば先ほどの例を少し変更して、「自分の投稿したエントリは全ての操作が可能だが、他の人の投稿したエントリについては一覧に表示されるだけ」としたい場合には、次のようにします。

```
<entry name="news1" schema="news" caption="新着情報" >
  <permit users="user1" mine="all" others="list" />
</entry>
```

もちろん、ユーザー毎に別々の権限を設定することもできます。

```
<entry name="news1" schema="news" caption="新着情報" >
  <permit users="user1" mine="all" others="list" />
  <permit users="user2" mine="all" others="none" />
</entry>
```

4．7．4 ユーザー全体への権限設定

次のように括弧付きで(publisher)と指定すると、ユーザー個別ではなく、「全ての管理者以外のユーザー」、つまり「編集者ユーザー」に対する権限をまとめて行う事ができます。

```
<entry name="news1" schema="news" caption="新着情報" >
  <permit users="(publisher)" mine="all" others="list" />
</entry>
```

(publisher)の指定を使うと、「全てのユーザーは基本的に自分のエントリを編集することしかできないが、user1 だけは他のユーザーのエントリも編集できる」というような複雑な指定が可能となります。

```
<entry name="news1" schema="news" caption="新着情報" >
  <permit users="(publisher)" mine="edit" others="list" />
  <permit users="user1" mine="edit" others="edit" />
</entry>
```

ここで、「各ユーザーに編集（edit）権限だけ与えても、新規追加（add）が出来ない状態では意味がないのではないか？」という疑問を持つ方もいらっしゃると思います。実は管理者（Administrator）ユーザーは、エントリ投稿時に「作成者ユーザー名」を選択することができます。これを使って例えば、複合店舗（ショッピングモール）サイトを運営する際などに、管理者ユーザーで各店舗のエントリをそれぞれの店舗ユーザー名で1つずつ作成しておき、店舗ユーザーにはmine="edit"のみ与えておくという利用方法が考えられます。**こうすることで、店舗ユーザーが間違えて自分のエントリを削除してしまったり、勝手に別の店舗エントリを作成してしまうというようなミスを防ぐことができます**。

4．7．5 権限の「拒否（deny）」

permit 要素では、権限の「許可」以外に「拒否」を指定することもできます。「拒否」は、既に与えられた権限を取り消す指定です。

権限の拒否は、"-edit"とか"-list"のように、mine 属性や others 属性の設定値に対してマイナス（-）をつけて指定します。全ての権限を拒否する場合には、"deny"と指定します（"-all"ではないので注意して下さい）。

通常は許可を指定しなければ自動的に権限が無い状態となる為、「拒否」を使うことはありません。これは、ユーザー全体への権限設定と組み合わせた場合に使用します。

例を示します。「全ユーザーに対して全ての権限を与えるが、user1 に対しては権限を与えない」としたい場合、次のようにしても無効となります。

```
<entry name="news1" schema="news" caption="新着情報" >
  <permit users="(publisher)" mine="all" others="all" />
  <permit users="user1" mine="none" others="none" />
</entry>
```

この指定は無効となる。

none は「権限を与えない」という指定ですが、既に与えられた権限を取り消すまでの効果は持ちません。この例だと、(publisher)を使って全ユーザーに all の権限が与えられている為、user1 で別途 none を指定しても all 権限は消えません。

この場合、次のように deny を使います。

```
<entry name="news1" schema="news" caption="新着情報" >
  <permit users="(publisher)" mine="all" others="all" />
  <permit users="user1" mine="deny" others="deny" />
</entry>
```

全権限拒否とまではいかずとも、自分のエントリの編集(edit）以外はできないようにしたい、という場合には、次のように edit 以外の権限をマイナス付きで指定します（-list を指定すると一覧表示自体が出来なくなるのでご注意下さい）。

```
<entry name="news1" schema="news" caption="新着情報" >
  <permit users="(publisher)" mine="all" others="all" />
  <permit users="user1" mine="edit,-add,-delete" others="deny" />
</entry>
```

ここで、user1 の mine 属性を"-add,-delete"のように edit を省略することも可能です（publisher に対して既に与えられている為です）。しかし、場合によっては、user1 に何をさせたいかを明確にする為に、敢えて指定して置いても害はありません。

4．7．6「許可しない(none)」と「拒否(deny)」の違い

権限の指定は、許可しない(none)→許可→拒否（deny）の順に優先度が高くなります。

許可を none で取り消すことはできません。また、拒否を指定されたものを許可で上書きすることもできません。permit 要素の記述順序は結果に影響しません。

組み合わせの例と結果を表で表します。（間違え易い部分には色をつけてあります）

編集者(publisher)	user1	user1 に対する edit に対する権限の結果（○：編集可能、×：編集不可）
edit	edit	○
edit	none	○
none	edit	○
none	none	×
list	list	×
list	edit	○
edit	list	○
edit	-edit	×
-edit	edit	×
all	deny	×
deny	all	×

4．7．7 コンテンツ一覧への表示／非表示

あるユーザーに対してコンテンツ一覧にエントリフォルダを表示するかどうかは自動的に決まります。具体的にはそのユーザーが該当エントリフォルダに対していかなる権限も持っていない場合にのみ、非表示となります。

4．7．8 権限設定事例集

（1）特定ユーザー専用エントリフォルダ

※user1 は「サイト更新者」ユーザー。

```
<entry name="news1" schema="news" caption="新着情報" >
  <permit users="user1" />
</entry>
```

（2）管理者専用エントリフォルダ

```
<entry name="news1" schema="news" caption="新着情報" >
  <permit users="(administrator)" />
</entry>
```

（3）user1：準管理者、user2：アルバイト（既存エントリの編集のみ可能）

※user1,user2 共に「サイト更新者」ユーザー。

```
<entry name="news1" schema="news" caption="新着情報" >
  <permit users="user1" />
  <permit users="user2" mine="none" others="edit" />
</entry>
```

（4）複合店舗サイト。各店舗担当者（user1～user3）、事務局担当者（center）、admin。

※user1～user3 には、自分の店舗のエントリしか編集させない（新規、削除は不可）。

※center は、全店舗のエントリを編集できる（新規、削除は不可）。

※新規店舗の追加は admin のみが行える（該当ユーザー名でエントリを作る）。

※user1～user3、center は「サイト更新者」ユーザー、admin は「システム管理者」ユーザー。

```
<entry name="shop" schema="shop" caption="各店舗情報" >
  <permit users="(publisher)" mine="edit" others="none" />
  <permit users="center" mine="deny" others="edit" />
</entry>
```

4．7．9 その他の留意事項

（1）エントリ作成者名について

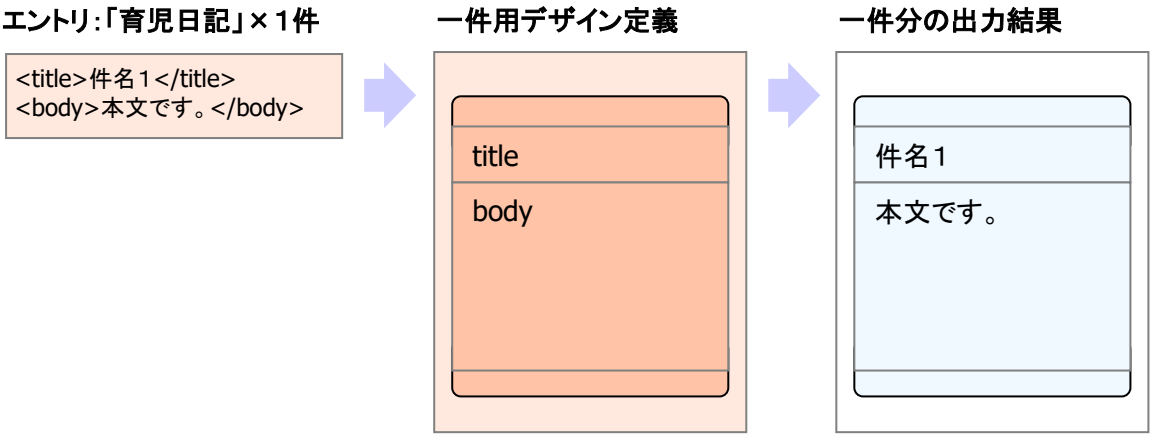
ver.1.1.7a より前のバージョンで保存したエントリには作成者名が含まれていない為、そのエントリの作成者名は「未設定」としてコンテンツ管理画面上で表示されます。

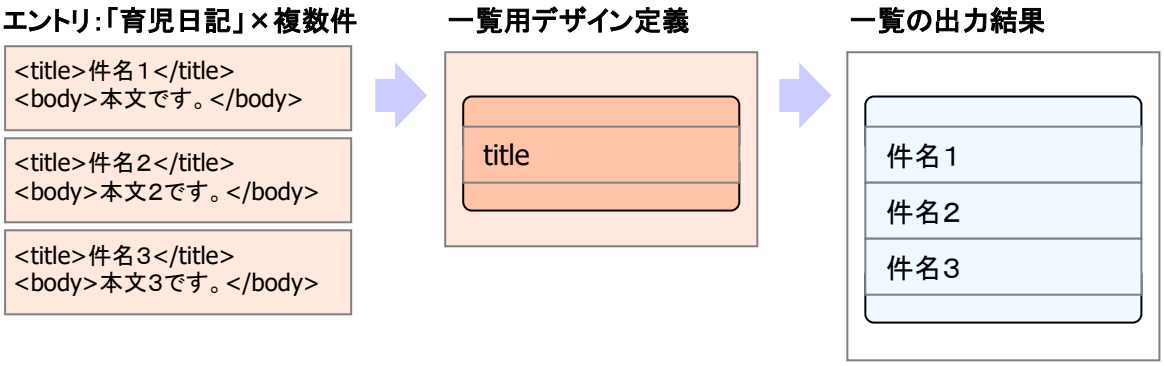
5 デザイン定義

5．1 デザイン定義とは何か。

コンテンツ管理画面から入力されたエントリデータは、あくまでデータでしかない為、見栄え良くHTML タグで飾りつけをする必要があります。

デザイン定義とは、エントリデータ(XML データ)をHTMLに変換する際のテンプレートを作成する作業です。エントリー件用のデザイン定義、もしくは複数のエントリを表示する為のエントリー覧用のデザイン定義を作成することができます。





デザイン定義には、**画面の一部分のみ**を記述します。デザイン定義に<html>～</html>を全て含めるような事もできますが、通常は更新したい範囲のみを記述します。

画面全体の HTML の記述はこれまで通り dreamweaver などのツールを使って作成し、更新したい一部分のみを「デザイン定義」として別に作成するようにします。そして、作成したデザイン定義を、画面全体の HTML へと埋め込みます。

画面全体の HTML にデザイン定義を埋め込む方法については「6．ウェブサイトへの埋め込み」を参照してください。

５．２ XSLT（XML Stylesheet Language Transformations）について

CMS Designer では、XML 形式で保存されたエントリデータを HTML に変換するための「テンプレート」を記述する仕組みとして、XSLT を利用しています。

XSLT は XML 技術の標準技術で、CMS Designer だけでなく他の様々な XML システムで利用されている、使い勝手の良い技術です。

XSLT は奥が深く、非常にきめ細かい処理を行うことができますが、デザイナーが覚えなければならない機能はごくわずかです（もちろん、XSLT を極めてもらっても構いませんよ！）。

例えば、次のようなエントリデータがあったとします（これは CMS Designer が実際に出力するエントリデータの一例です）。

```
<?xml version="1.0" encoding="UTF-8" ?>
<entry>
  <title>件名１。</title>
  <body>本文1です。</body>
</entry>
```

これを、次のような HTML に変換したいとします。

```
<table border="1">
  <tr><td>件名</td><td>件名１。</td></tr>
  <tr><td>本文</td><td>本文1です。</td></tr>
</table>
```

この場合、次のようなデザイン定義（XSLT）を記述します。

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="html" encoding="UTF-8" omit-xml-declaration="yes" />

  <xsl:template match="/entry">
    <table border="1">
      <tr><td>件名</td><td><xsl:value-of select="title" /></td></tr>
      <tr><td>本文</td><td><xsl:value-of select="body" /></td></tr>
    </table>
  </xsl:template>

</xsl:stylesheet>
```

注目して欲しいのは、5 行目から 10 行目あたりの xsl:template タグで囲まれた部分です。

出力した HTML がほとんどそのまま記述されています。データを出力したい部分だけ、xsl:value-of というタグが記述されています。

基本はこれだけです。おそらく、すぐに理解できるでしょう。

[！] HTML 形式、XML（XHTML）形式での出力

xsl ファイル中の<xsl:output method="html" ... />の method 属性によって、最終的に出力される HTML/XML データに若干違いが生まれます。

method="html"と指定すると、HTML4.0 の規約に沿ったタグが出力されます。xsl 中で
と記述していても、
と出力されます。<input .... /> なども、<input ... >のようになります。これは HTML4.0 としては便利ですが、XHTML としては正しくありません。

method="xml"と指定すると、通常の XML の規約に沿った形でタグが出力されます。よって、XHTML として出力したい場合にはこちらを指定して下さい。もちろん、XML として出力したい場合もこちらになります。

一部のサーバ環境では、method="xhtml"という指定が使える場合もあります。これは、method="xml"指定だと空要素が例えば
ではなく
のように「/>」の前の空白が詰められてしまうような仕様を回避できます。method="xml"の場合のこの仕様は、XML としては正しいのですが、HTML4.0 との互換性を意識した XHTML の仕様としては正しくありません。但し、CMSD は method="xml"の場合でも、内部的に
や<hr/>を
<hr />のように空白を空けたものに変換して出力していますので、method="xml"でも問題はありません（ver.1.1.3a 以降のみ）。

５．３ エントリ１件用のデザイン定義

例えばお店情報のエントリがあったとして、一覧ではなく、１件分のお店の詳細情報を画面に表示したい場合があります。この場合、「エントリ１件用のデザイン定義」を作成します。

エントリ１件用のデザイン定義は以下のように行います。

まず、デザイン名を決めます。デザイン名は半角英数でつけます（全角文字は不可）。

そのデザイン名によって、デザイン定義ファイル名は次のように決まります。

「スキーマ名」. デザイン名. design.xsl

例えば、"diary"スキーマに"default"という名前の一件用デザインを作成する場合は、

diary.default.design.xsl

というファイル名になります。

エントリ１件分を画面に表示する場合のデザイン定義は以下のようになります。

```
<?xml version="1.0" encoding="UTF-8" ?>                                    ヘッダ
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="html" encoding="UTF-8" omit-xml-declaration="yes" />
  <xsl:template match="/entry">
    <table border="1">                                                     デザイン
      <tr><td>件名</td><td><xsl:value-of select="title" /></td></tr>
      <tr><td>本文</td><td><xsl:value-of select="body" /></td></tr>
    </table>
  </xsl:template>                                                          フッタ
</xsl:stylesheet>
```

ヘッダ部とフッタ部は、常にこのまま記述すれば OK です。

デザイン部の記述方法は、「５．５ デザイン リファレンス」を参照してください。

このファイルは、スキーマ定義ファイルと同じフォルダにアップロードしてください。

５．４ エントリー覧用のデザイン定義

例えばお店情報のエントリがあったとして、登録されている全てのエントリについて、お店の名前を一覧表示したい場合があります。この場合、「エントリー覧用のデザイン定義」を作成します。
エントリー覧用のデザイン定義は以下のように行います。

まず、デザイン名を決めます。デザイン名は半角英数でつけます（全角文字は不可）。
そのデザイン名によって、デザイン定義ファイル名は次のように決まります。

「スキーマ名」. list. デザイン名. design.xsl

スキーマ名の後に「list」という文字が入っているのがミソです。
例えば、"diary"スキーマに"default"という名前の一覧用デザインを作成する場合は、
diary.list.default.design.xsl
というファイル名になります。

エントリー覧用のデザイン定義は以下のようになります。

```
<?xml version="1.0" encoding="UTF-8" ?>                                    ← ヘッダ
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="html" encoding="UTF-8" omit-xml-declaration="yes" />
  <xsl:template match="/entrylist">
    <xsl:for-each select="entry">
      <table border="1">                                                   ← デザイン
        <tr><td>件名</td><td><xsl:value-of select="title" /></td></tr>
        <tr><td>本文</td><td><xsl:value-of select="body" /></td></tr>
      </table>
    </xsl:for-each>                                                        ← フッタ
  </xsl:template>
</xsl:stylesheet>
```

ヘッダ部とフッタ部は、常にこのまま記述すれば OK です。
デザイン部の記述方法は、「５．５ デザイン リファレンス」を参照してください。

このファイルは、スキーマ定義ファイルと同じフォルダにアップロードしてください。

５．５ デザイン リファレンス

５．５．１ 指定の箇所へデータ項目を埋め込む。

指定の箇所へスキーマの text 項目、textarea 項目、int 項目を埋め込む場合、xsl:value-of タグを使います。

```
<xsl:value-of select="データ名" />
```

```
例）　<tr><td>件名</td><td><xsl:value-of select="title" /></td></tr>
```

最も使用頻度の高いタグです。

注意事項として、出力したい text 項目又は textarea の output 属性が"text1"以外の場合は、次のように disable-output-escaping 属性を"yes"に設定しなければなりません。

```
<xsl:value-of select="データ名" disable-output-escaping="yes" />
```

```
例）
<tr><td>件名</td><td><xsl:value-of select="title" disable-output-escaping="yes" /></td></tr>
```

disable-output-escaping 属性の意味を考える必要は特にありませんが、気になる方の為に説明します。disable-output-escaping 属性は、データ中の"<"や">"などの「HTML として表示できない文字」を自動的に「<」などのコードに変換して表示しないことを指定します。"yes"と指定すると、変換「しません」。disable-output-escaping 属性を省略すると"no"を指定したことになり、変換されてしまいます。スキーマの output 属性を"html1"などにした場合、そのまま出力したい HTML コードまで変換されてしまう為、その場合はここで"yes"を指定する必要があります。

５．５．２ 画像項目（img 項目）を出力する。

img 項目を画像として表示したい場合、以下のようにします。

img 項目はエントリデータ中では画像への URL として保存されている為、次のように記述できます。

```
<img src="{データ項目名}" />
```

例）

ここで、次のように書くことはできません。

<img src="<xsl:value-of select="photo" />" />

属性の中に画像の URL を入れればいい訳ですから上記のように書きたくなりますが、XML ではタグの途中にタグが出現することはできない為、これはエラーとなります。

属性の中にデータを埋め込みたい場合、そのデータ名を"{}"で囲えば OK です。"{}"を使った書き方は、属性の中でしか使用できません。

例えば、画像をそのまま表示するのではなくリンクとして表示し、リンクをクリックしたら別ウィンドウで画像を表示したい場合は次のように書きます。

```
<a href="{データ項目名}" target="_blank">画像を表示する！</a>
```

スキーマで alt 属性を"True"に指定した場合、alt 用の文字列を使うことができます。

```
<img src="{データ項目名}" alt="{データ項目名/@alt}"/>
```

例）

ここで"photo/@alt"という表現が出てきましたが、これは「photo 項目の alt 属性を取得する」という XSLT の表現です。スキーマの img 項目からは、alt 属性以外に width 属性や height 属性も取得できます。

```
<img src="{データ項目名}" width="{データ項目名/@width}"
height="{データ項目名/@height}" />
```

画像ファイルが存在する時だけ画像を表示する、等の方法については、『５．５．６ データ値の内容によって処理を変える。』を参照してください。

５．５．３ 画像項目（img 項目）を縮小／拡大して表示する（サムネイル等）。

img 項目を拡大／縮小して表示したい場合、次のように幅か高さを指定します。

```
<img src="{データ項目名}&amp;w=幅" />
又は
<img src="{データ項目名}&amp;h=高さ" />
```

```
例） <img src="{photo}&amp;w=240" />
```

「&」という文字は、そのまま入力してください。XML 文書中で「&」という文字列を出力したい場合はこのように書きます（HTML と同じですね）。

幅又は高さを指定すると、その幅又は高さにピッタリ収まるように拡大又は縮小されて表示されます。

尚、拡大／縮小表示した場合、width や height は取得できません（拡大／縮小前の値が取得されます）のでご注意ください。

サンプルとして、サムネイルをクリックすると元の大きさの画像を別ウィンドウで表示するデザイン定義の例を示します。

```
<a href="{photo}" target="_blank"><img src="{photo}&amp;w=240" /></a>
```

簡単な応用ですね。

ところで、ここで HTML の img タグの書き方が、

```
<img src="xxx">
```

ではなく、

```
<img src="xxx" />
```

であることに注意してください。前述の通り、XML では「開始タグと終了タグ」が必ず対になっていなければなりません。しかし、HTML では img タグのように終了タグがないタグも存在する為、

```
<img src="xxx" />
```

のように、"/>"でタグを終わるようにします。

この書き方は、他にも HTML の input タグや br タグも同じようにしなくてはいけません。

「４．６ 画像キャッシュの指定」も併せてご覧下さい。

５．５．４ ファイル項目（file 項目）を出力する。

file 項目を表示したい場合、以下のようにします。

file 項目はエントリデータ中ではファイルへの URL として保存されている為、次のように記述できます。ファイル項目の URL をブラウザ上でクリックすると、該当ファイルのダウンロードが開始されます（ダウンロードでなくブラウザにインライン表示したい場合については後述します）。

```
<a href="{データ項目名}" ><xsl:value-of select="データ項目名/@org" /></a>
```

```
例） <a href="{myfile}" ><xsl:value-of select="myfile/@org" /></a>
```

ここで"myfile/@org"という表現が出てきましたが、これは「myfile 項目の org 属性を取得する」という XSLT の表現です。org 属性には、アップロードした際のファイル名が入っています。

他、"myfile/@filesize"でファイルサイズを取得できます。

```
例） <xsl:value-of select="myfile/@filesize" />bytes
```

ファイルサイズはバイト数で格納されている為、そのままですと見難い場合があります。

もしこれを MB 単位で表示したい場合は、1024 バイト=1KB、1024KB=1MB、という関係を使い、1024*1024=1048576 で割り、結果を format-number という XSL の命令文で整形します。

```
例） <xsl:value-of select="format-number(myfile/@filesize div 1048576,'0.00') " />MB
```

format-number の二つ目の値として「'0.00'」というものを指定していますが、これは「1 桁目と、小数点以下 2 桁までは必ず表示する」というフォーマット指定です。結果がちょうど 8 でも「8.00」のように出力されます。これが嫌な場合は、0 の代わりに#を指定し、'0.##'のようにして下さい。

@filesize は画像項目でも使用できます。

ときにはファイル項目を、ダウンロードさせずにブラウザにインライン表示させたいことがあります。例えばPDFファイルをブラウザ内に表示させたり（ブラウザが対応している必要があります）、音楽や動画ファイルを再生したり、といった用途です。この場合、次のように URL の末尾に「&disp=inline」を付けてください。HTML 中では「&」を「&」と記述しなければいけない点にご注意ください。

```
例1） <a href="{myfile}&amp;disp=inline" ><xsl:value-of select="myfile/@org" /></a>
```

```
例2） <iframe src="{myfile}&amp;disp=inline" height="450" width="500" ></iframe>
```

```
例2） <video src="{myfile}&amp;disp=inline"></video>
```

５．５．５ 繰り返し項目（list 項目）を出力する。

※「list 項目」とは、1件のエントリ内に存在する繰り返し項目の事です。エントリー覧の事ではありません。エントリー覧の出力方法は『５．４ エントリー覧用のデザイン定義』をご覧下さい。

list 項目を出力する場合、以下のように記述します。

```
<xsl:for-each select="繰り返し項目データ名/listitem" >
  <xsl:value-of select="繰り返す部分のデータ名" />
      :
</xsl:for-each>
```

```
例）
<table>
  <xsl:for-each select="photolist/listitem">
    <tr>
      <td><xsl:value-of select="phototitle" /></td>
      <td><img src="{photo}" /></td>
    </tr>
  </xsl:for-each>
</table>
```

尚、この例のデザインは次のような繰り返し項目を持つスキーマを対象にしています。

```
<data name="photolist" type="list" caption="画像リスト" >
  <listitem caption="画像">
    <data name="phototitle" type="text" caption="画像のタイトル" />
    <data name="photo" type="img" caption="画像" />
  </listitem>
</data>
```

画像1、画像ファイル1

画像2、画像ファイル2

画像3、画像ファイル3

というデータがあった場合、この例のデザインは以下のように出力されます。

```
<table>
  <tr>
    <td>画像1</td>
    <td><img src="画像ファイル 1 の URL" /></td>
  </tr>
  <tr>
    <td>画像2</td>
    <td><img src="画像ファイル2の URL" /></td>
  </tr>
  <tr>
    <td>画像3</td>
    <td><img src="画像ファイル3の URL" /></td>
  </tr>
</table>
```

５．５．６ データ値の内容によって処理を変える。

例えば、画像が登録されている場合は画像を表示し、登録されていない場合は表示したくない場合など、「データ値の内容によって出力する内容を変えたい」場合があります。

そんな時は xsl:if タグを使用します。

```
<xsl:if test="条件" >
  ....出力したい内容をこの中に記述。
</xsl:if>
```

「条件」の部分には様々な条件を書くことができます。

例えば、「画像が登録されていない場合」という書き方は、言い換えれば「画像データの中身が空っぽなら」という意味になります。

「**データに中身が空っぽだったら**」と指定したい場合は、

test="データ名＝''"

と書きます。等号の右辺には、シングルコーテーション(')を連続して 2 つ入力します。

シングルコーテーション 2 つで「空っぽの文字」を表します。

```
例）
<xsl:if test="photo=''">
  <img src="noimage.gif" />
</xsl:if>
```

逆に、「**データの中身が入っていたら**」と書きたい場合は、

test="not(データ名＝'')"

と書きます。"not(....)"を付けると、その中の条件の意味が逆になります。

画像データの中身があれば、img タグを出力、中身がなければ、「no image」という文字を出力したい場合、次のように書きます。

```
例）
<xsl:if test="not(photo='')">
  <img src="{photo}" />
</xsl:if>
<xsl:if test=" photo=''">
  no image
</xsl:if>
```

また、数値を何かと比較する事もできます。

「データが〇〇より上の場合」「データが〇〇と同じ」「データが〇〇未満」などです。

25 より上 ： test="データ名>25"

25 と同じ: test="データ名＝25"

25 未満 ： test="データ名<25"

25 以上 ： test="データ名>=25"

25 以下 ： test="データ名<=25"

※<は"<"(lower than)を、>は">"(greater than)を表します。

例えば、点数(score)というデータ項目が 60 点ぴったりなら「ギリギリ合格！」、60 点未満なら「不合格」、60 点より上なら「合格！」と表示する場合、次のように書きます。

```
例）
<xsl:if test="score=60">
  ギリギリ合格！
</xsl:if>
<xsl:if test="score&lt;60">
  不合格
</xsl:if>
<xsl:if test="score&gt;60">
  合格！
</xsl:if>
```

もし、比較対象が数値ではなく文字だった場合は、比較する値をシングルコーテーション(')で囲う必要があります。

```
例）
<xsl:if test="username='admin'">
  管理者
</xsl:if>
```

５．５．７ メニュー項目を表示する。

メニュー（menu）項目は、エントリファイル中では ID しか格納されていません。

例えば次のようなスキーマ定義があるとします。

```
<data name="shopkind" type="menu" caption="お店種別" >
  <menuitem id="1">中華</menuitem>
  <menuitem id="2">ラーメン</menuitem>
  <menuitem id="3">和食</menuitem>
  <menuitem id="4">洋食</menuitem>
</data>
```

このメニュー項目を表示する為に、次のようなデザイン定義をします。

```
<xsl:value-of select="shopkind" />
```

しかし、この出力結果は、

```
4
```

のように、ID しか表示されません。

ID から元の選択肢を表示するには、xsl:if タグを使って以下のようにします。

```
<xsl:if test="shopkind='1'">中華</xsl:if>
<xsl:if test="shopkind='2'">ラーメン</xsl:if>
<xsl:if test="shopkind='3'">和食</xsl:if>
<xsl:if test="shopkind='4'">洋食</xsl:if>
```

もちろん、スキーマ定義と表示内容を変えることもできます。この例では画像を表示しています。

```
<xsl:if test="shopkind='1'"><img src="tyuka.gif" /></xsl:if>
<xsl:if test="shopkind='2'"><img src="ramen.gif" /></xsl:if>
<xsl:if test="shopkind='3'"><img src="washoku.gif" /></xsl:if>
<xsl:if test="shopkind='4'"><img src="youshoku.gif" /></xsl:if>
```

【※Ver.1.1.18a 以降】

Ver1.1.18a にて、メニュー項目の ID だけでなく、元の選択肢ラベルをデザイン定義中で簡単に取得して表示することが可能となりました。次のように、xsl:value-of タグの select 属性の中に、「メニュー項目名/@label」のように「**/@label**」を追加して記述します。

```
<xsl:value-of select="shopkind/@label" />
```

出力は例えば次のようになります。

```
洋食
```

メニュー項目の選択肢リスト全体を取得して表示するには、「５．５．１７ メニュー項目の選択肢リストを表示する。」をご覧ください。

５．５．８ エントリー覧から個別のエントリへリンクを張る

「６．ウェブサイトへの埋め込み」で説明しますが、個別のエントリを画面に表示する場合、その画面の URL に以下のように「エントリ ID」をパラメータとして渡します。

```
mypage.php?eid=00001
```

ここで mypage.php は、あなたが作成する、１件分のエントリを埋め込んだ HTML 画面です（作り方の詳細は「６．ウェブサイトへの埋め込み」を参照してください）。
eid に指定されているのがエントリ ID です。

よって、エントリー覧から個別のエントリへリンクを張る為には、エントリ ID を取得して出力する必要があります。
エントリ ID は、"@id"というデータ名で取得することができます。

具体的には次のように記述できます。

```
<a href="mypage.php?eid={@id}">
  <xsl:value-of select="title" />
</a>
```

この結果は、例えば以下のように出力されます。

```
<a href="mypage.php?eid=00001">件名１。</a>
```

尚、もっと簡単な方法として、@href を使う方法もあります。

```
<a href="mypage.php{@href}">
  <xsl:value-of select="title" />
</a>
```

上記のように、mypage.php の後に続けて{@href}と書きます。
@href には、"?eid=00001"のように、パラメータ名も含めた値が入っています。
@href の場合、グループによる絞込みを行った場合には絞り込み条件も自動的にパラメータとして追加される為、特に便利です。

通常は@href を使うようにしてください。

５．５．９「次のエントリへ」「前のエントリ」へのリンクをつける

blog などでは、各記事に「前の記事」「次の記事」のようなリンクがついていて、次々と記事を読んでいくことができるようになっています。これを便宜上、「記事のナビゲーション」と呼ぶことにします。

<<前の記事へ | 次の記事へ>>

CMS Designer でも blog と同様のナビゲーションをつけることができます。

尚、ナビゲーションを表示するにはデザイン定義を変更するだけでなく、６章で説明する埋め込み命令として cms:: navi_entry 命令を使用する必要があります。「６．ウェブサイトへの埋め込み」も合わせて参照してください。

（１）１件用のデザイン定義にナビゲーションを追加する。

CMS Designer は、記事のナビゲーション情報を出力することができます。デザイン定義ではその情報を好きなように表示させることができます。

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="html" encoding="UTF-8" omit-xml-declaration="yes" />
  <xsl:template match="/entry">

    <xsl:for-each select="navi">
      <xsl:for-each select="prev">
        <a href="{@href}">&lt;&lt;前へ</a> |
      </xsl:for-each>
      <xsl:for-each select="next">
        <a href="{@href}">次へ&gt;&gt;</a>
      </xsl:for-each>
    </xsl:for-each>

    <table border="1">
      <tr><td>件名</td><td><xsl:value-of select="title" /></td></tr>
      <tr><td>本文</td><td><xsl:value-of select="body" /></td></tr>
    </table>
  </xsl:template>
</xsl:stylesheet>
```

<xsl:template match="/entry">の下に、太字の部分を追加します。

<xsl:for-each select="navi">から始まる部分がそれです。

この部分に、記事のナビゲーションのデザインを記述します。よくわからない場合はこのまま太字部分をコピーしてください。

```
<xsl:for-each select="prev">
  <a href="{@href}">&lt;&lt;前へ</a> |
</xsl:for-each>
```

この部分が、「前の記事」へ移動するリンクを作成する部分です。

尚、HTML でも同じなのでご存知だとは思いますが、"<"という暗号みたいな文字は、"<"(lower than)という文字を表す記号です。@href には、例えば"?eid=00002"というような、次のエントリの URL パラメータ文字列が入っています。同じページへジャンプするので、href のページ名は省略できます。

```
<xsl:for-each select="next">
  <a href="{@href}">次へ&gt;&gt;</a>
</xsl:for-each>
```

この部分は、次の記事へ移動するリンクを作成する部分です。

">"というのは、">"(greater than)という文字を表す記号です。

ところで、「前の記事」や「次の記事」が存在しない場合どうなるかというと、CMS Designer は prev や next データを出力しません。

しかし、上記の例のように<xsl:for-each select="next"></xsl:for-each>で囲ってあるので、prev や next データが存在しない場合はリンクが出力されないだけで、エラーにはなりません。

（2）前の記事や次の記事が存在しない場合でもナビゲーションが消えないようにする。

もし、「前や次の記事が存在しない場合にはリンクなしの"<<前へ"や"次へ>>"という文字を表示したい」という場合は、xsl:if の"not"を使って次のようにします（"not"の意味については「５．５．６ データ値の内容によって処理を変える。」を参照してください）。

```
<xsl:for-each select="navi">
  <xsl:for-each select="prev">
    <a href="{@href}">&lt;&lt;前へ</a> |
  </xsl:for-each>
  <xsl:if test="not(prev)">
    &lt;&lt;前へ |
  </xsl:if>
  <xsl:for-each select="next">
    <a href="{@href}">次へ&gt;&gt;</a>
  </xsl:for-each>
  <xsl:if test="not(next)">
    次へ&gt;&gt;
  </xsl:if>
</xsl:for-each>
```

（3）好きな位置へナビゲーションを配置する。

ナビゲーションは、好きな位置へ配置することができます。

（1）で挙げた例のままだと、ナビゲーションはコンテンツの上に表示されますが、デザイン中の下の方へ移動すれば、ナビゲーションを下へ持ってくる事もできます。

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="html" encoding="UTF-8" omit-xml-declaration="yes" />
  <xsl:template match="/entry">

    <table border="1">
      <tr><td>件名</td><td><xsl:value-of select="title" /></td></tr>
      <tr><td>本文</td><td><xsl:value-of select="body" /></td></tr>
    </table>

    <xsl:for-each select="navi">
      <xsl:for-each select="prev">
        <a href="{@href}">&lt;&lt;前へ</a> |
      </xsl:for-each>
      <xsl:for-each select="next">
        <a href="{@href}">次へ&gt;&gt;</a>
      </xsl:for-each>
    </xsl:for-each>

  </xsl:template>
</xsl:stylesheet>
```

また、次のように書けば、コンテンツ中にリンクを配置することもできます。

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="html" encoding="UTF-8" omit-xml-declaration="yes" />
  <xsl:template match="/entry">

    <table border="1">
      <tr><td>件名</td><td><xsl:value-of select="title" /></td></tr>
      <tr><td>本文</td><td><xsl:value-of select="body" /></td></tr>
      <tr>
        <td colspan="2">
          <xsl:for-each select="navi">
            <xsl:for-each select="prev">
              <a href="{@href}">&lt;&lt;前へ</a> |
            </xsl:for-each>
            <xsl:for-each select="next">
              <a href="{@href}">次へ&gt;&gt;</a>
            </xsl:for-each>
          </xsl:for-each>
        </td>
      </tr>
    </table>

  </xsl:template>
</xsl:stylesheet>
```

５．５．１０ 一覧表示で「次のページへ」「前のページへ」のリンクをつける

５．５．９ では、エントリ1件表示の場合に表示を前後の記事へ切り替える為のナビゲーションを作る方法を解説しました。

似たようなケースで、エントリー覧表示の場合にも1ページに表示できないほど大量の数のエントリがあった場合、1ページには規定で 10 件分しか表示されません（表示件数は変更できます。詳しくは6章をご覧下さい）。この時に、「次のページ」「前のページ」のリンクをつけて、次の 10 件、前の 10 件へとページを切り替えることができます。

これを便宜上、「ページ切替のナビゲーション」と呼ぶことにします。

<<前のページへ | 次のページへ>>

デザイン定義では以下のようにページ切替のナビゲーションをデザインします。

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="html" encoding="UTF-8" omit-xml-declaration="yes" />
  <xsl:template match="/entrylist">

    <xsl:for-each select="navi">
      <xsl:for-each select="prev">
        <a href="{@href}">&lt;&lt;前のページへ</a> |
      </xsl:for-each>
      <xsl:for-each select="next">
        <a href="{@href}">次のページへ&gt;&gt;</a>
      </xsl:for-each>
    </xsl:for-each>

    <xsl:for-each select="entry">
      <table border="1">
        <tr><td>件名</td><td><xsl:value-of select="title" /></td></tr>
        <tr><td>本文</td><td><xsl:value-of select="body" /></td></tr>
      </table>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

<xsl:template match="/entry">の下に、太字の部分を追加します。

<xsl:for-each select="navi">から始まる部分がそれです。

ほとんど「５．５．９ 」と同じですので、詳しくはそちらを参考にしてください。

５．５．１１ エントリの更新日付や、日付項目を表示する。

エントリの更新日付は、"@date"というデータ名で取得できます。

@date は、エントリの更新日付を"年-月-日 時:分"の形式で保存しています。

例えば更新日付を表示したい場合、

```
更新日付 ：<xsl:value-of select="@date" />
```

のように書くと、"2004-12-24 08:20"のような形式で表示できます。

年、月、日、時、分、曜日をそれぞれ個別に取得したい場合は、それぞれ以下のデータ名で取得できます。

データ名	説明
@year	年。
@month	月。
@day	日。
@hour	時。
@minute	分。
@weekday	曜日。

例えば更新日付を「2004 年 12 月 08 日」とだけ表示したい場合は、次のように記述します。

```
<xsl:value-of select="@year" />年
<xsl:value-of select="@month" />月
<xsl:value-of select="@day" />日
```

```
2004 年 12 月 24 日
```

同様に、日付項目（スキーマの date、time、datetime、 year、month 型）を表示する際には、項目名だけで参照すると、例えば date 項目は"2004-12-24"のように表示されます。

```
<xsl:value-of select="birthday" />
```

```
2004-12-24
```

年や月などを別々に取得したい場合は、エントリ更新日付と同様に@year や@month などを使ってください。

```
<xsl:value-of select="birthday/@year" />年
<xsl:value-of select="birthday/@month" />月
<xsl:value-of select="birthday/@day" />日生
```

```
2004 年 12 月 24 日生
```

曜日を取得したい場合は、次のようにします。

```
<xsl:value-of select="@weekday" />
```

曜日に応じて、"sun"、"mon"、"tue"・・・のような文字が取得できます。

もし、「日曜」「月曜」「火曜」・・・のような表示をしたい場合は、xsl:if を使って次のようにします。

```
<xsl:if test="@weekday='sun'">日曜</xsl:if>
<xsl:if test="@weekday='mon'">月曜</xsl:if>
<xsl:if test="@weekday='tue'">火曜</xsl:if>
<xsl:if test="@weekday='wed'">水曜</xsl:if>
<xsl:if test="@weekday='thu'">木曜</xsl:if>
<xsl:if test="@weekday='fri'">金曜</xsl:if>
<xsl:if test="@weekday='sat'">土曜</xsl:if>
```

５．５．１２ エントリ ID を表示する。

エントリ ID は、"@id"というデータ名で取得できます。

エントリ ID は 5 桁の数字で、先頭はゼロ埋めされています。（例：00002）

```
ID：<xsl:value-of select="@id" />
```

```
00002
```

５．５．１３ グループ絞込み条件を表示する。

エントリー一覧などでは、グループによる絞込み条件を指定することができます。

この方法については『６．３．５ 絞込みの指定』や『６．３．６ URL パラメータからの動的な絞込みの指定』をご覧下さい。

絞込み条件を指定した場合、デザイン定義側でその条件を表示したい場合がよくあります。

特にパラメータからの絞込み指定をした場合、条件が動的に変わることになるので、指定された条件をデザイン定義側で取得できると便利です。

絞込み条件は、「項目名」と「値（完全一致条件）」の対で取得できます。

例えば、"shopkind"という項目名で絞込みをしている場合、デザイン定義側では次のように絞込み条件を取得し、結果を表示することができます。

```
<xsl:template match="/entrylist">
        :
  <xsl:for-each select="group">
    <xsl:if test="@key='shopkind'">
      「
      <xsl:if test="@value='1'">
        中華
      </xsl:if>
      <xsl:if test="@value='2'">
        ラーメン
      </xsl:if>
      <xsl:if test="@value='3'">
        和食
      </xsl:if>
      <xsl:if test="@value='4'">
        洋食
      </xsl:if>
      <xsl:if test="@value='5'">
        スイーツ
      </xsl:if>
      」の一覧<br />
    </xsl:if>
  </xsl:for-each>
      :
```

shopkind に 3 を指定した場合は、

```
「和食」の一覧
```

のように表示されます。

2 行目の<xsl:if test="@key='項目名'">に項目名を指定し、4 行目以降の xsl:if でそれぞれの条件の場合の出力内容を記述します。

５．５．１４「NEW!」表示をする。

「７日以内に更新されたエントリに"NEW!"と付けたり、目立つマーク画像をつけたりしたい」という場合には、"@daysago"を使います。

@daysago には、そのエントリが「今日から数えて何日前に更新されたか」という数字が入っています。今日が 2005 年 10 月 10 日で、エントリ更新日付が 2005 年 10 月 5 日なら、「5」という数字が入っています。

この情報を使うと、以下のように書く事ができます。

```
<xsl:if test="@daysago&lt;7">New!</xsl:if>
```

上記のように記述することで、「７日以内の更新ならば"New!"と表示する」という事が可能です。

画像を表示する場合は次のようにします。

```
<xsl:if test="@daysago&lt;7"><img src="new.gif" alt="new!"/></xsl:if>
```

【注意！】
@daysago は、表示の際に動的に付与される情報であり、保存されているエントリデータには存在しません。よって、@daysago を使って絞込み表示をしたり、ソート処理を行うことはできません。
※XSLT 側でソートや絞込みを行なうのはもちろん可能です。これについての詳しい説明はユーザーズフォーラムでご質問頂くか、XSLT の解説書等をご覧下さい。

５．５．１５ 現在日時を取得する。

XSLT には通常、現在日時を取得する為の機能はありません。

そこで CMS Designer では、エントリ情報として現在日時を出力し、XSLT（デザイン定義）側から取得できるようになっています（ver.1.1.5a からの新機能）。

（1）today 要素

下記の情報をデザイン定義側から取得可能です。

データ名	説明
today	日時表示文字列。 例）"2010-10-27 19:30"
@time	1970 年 1 月 1 日からの経過秒数。 日付項目の time 属性と組み合わせて、日付間の日時差を求めるのに利用する。
@month	月。
@day	日。
@hour	時。
@minute	分。
@second	秒。
@weekday	曜日。"sun"や"mon"等の値が入る。
@tzoffset	タイムゾーンのオフセット（時差）が秒数で格納されている。 日本では 32400（9 時間）が入る。

（2）today 要素の参照方法

１件用デザイン定義と一覧用デザイン定義とで参照パスが異なります。

●1件用デザイン定義の場合

```
<xsl:value-of select="/entry/today" />　又は
<xsl:value-of select="/entry/today/@time" />　等
```

●一覧用デザイン定義の場合

```
<xsl:value-of select="/entrylist/today" />　又は
<xsl:value-of select="/entrylist/today/@time" />　等
```

XSLT に詳しい方は、次のような表記法でも利用可能です。この場合は1件用、一覧用とで共通になります。

```
<xsl:value-of select="/*/today" />　又は
<xsl:value-of select="/*/today/@time" />　等
```

（3） 利用例1：開催日までの日数を表示する。

スキーマに「開催日」という日付項目(type="date")があるとします。項目名は"eventdate"とします。日付項目にも、ver.1.1.5a より time 属性が追加されています（1.1.5a より前のバージョンで保存されたエントリデータには time 属性は付与されていません。データを上書き保存すれば追加されます）。

これを利用して、開催日と現在日時の「日数差」を求めます。

eventdate/@time から /*/today/@time を引き算すると、開催日と今日の時間差が秒で求められます。

この秒数を 60[秒]で割れば「分」が求められます。

さらに 60[分]で割れば「時間」が求められます。

さらに 24[時間]で割れば「日数」が求められます。

つまり、以下のような式により、開催日までの日数を求める事ができます。

```
あと
<xsl:value-of select="(eventdate/@time - /*/today/@time) div (60*60*24)" />
日
```

「div」は、両辺を割り算せよという命令です。「*」は両辺を掛け算せよという命令です。

しかしこのままだと、出力結果が

```
あと 5.68625 日
```

のように小数になってしまいます。正しい日数ではありますが、これでは困るので整数に丸めます。整数に丸めるには floor 関数という XSLT(XPath)の命令を使います。

```
あと
<xsl:value-of
      select="floor((eventdate/@time - /*/today/@time) div (60*60*24))" />日
```

結果は次のようになります。

```
あと 5 日
```

尚、この例だと、開催日を過ぎてしまった場合に次のような表示になってしまいます。

```
あと-1 日
```

エントリの公開予約機能を使って自動的に非公開にしている場合は問題ないのですが、そうしていない場合には、これが問題となることもあるかもしれません（このままでも充分意味は通じますが）。

これを避けるには、次のように if 文を使って表示を変える事ができます。

```
<xsl:if
  test="floor((eventdate/@time - /*/today/@time) div (60*60*24)) &gt;= 0" />
  あと<xsl:value-of
    select="floor((eventdate/@time - /*/today/@time) div (60*60*24))" />日
</xsl:if>
```

「日時差がマイナスの場合には？」という条件式を xsl:if で作り、その中に先ほどの表示命令を入れています。

日時差がマイナスの場合に「このイベントは終了しました」と表示したい場合には次のデザインを追加します。

```
<xsl:if test="floor((eventdate/@time - /*/today/@time) div (60*60*24)) &lt; 0" />
  このイベントは終了しました
</xsl:if>
```

全て組み合わせると次のようになります。

```
<xsl:if test="floor((eventdate/@time - /*/today/@time) div (60*60*24)) &gt;= 0" />
  あと
  <xsl:value-of
    select="floor((eventdate/@time - /*/today/@time) div (60*60*24))" />日
</xsl:if>
<xsl:if test="floor((eventdate/@time - /*/today/@time) div (60*60*24)) &lt; 0" />
  このイベントは終了しました
</xsl:if>
```

何度も「floor((eventdate/@time - /*/today/@time) div (60*60*24))」が出てきて見にくいと思う場合には、xslt の命令である xsl:variable を使って、一時的にデータを変数に格納して使いまわす事ができます。

この場合だと、

```
<xsl:variable name="timelag"
  select="floor((eventdate/@time - /*/today/@time) div (60*60*24))" />
```

のように書いておき、必要な箇所で

```
<xsl:value-of select="$timelag" />
```

のように、name 属性で名づけた変数名に「$」をつけて呼び出す事ができるようになります。

先ほどのサンプルを xsl:variable を使って書き直すと次のように書くことができます。

```
<xsl:variable name="timelag"
  select="floor((eventdate/@time - /*/today/@time) div (60*60*24))" />
<xsl:if test="$timelag &gt;= 0" />
  あと<xsl:value-of select="$timelag" />日
</xsl:if>
<xsl:if test="$timelag &lt; 0" />
  このイベントは終了しました
</xsl:if>
```

（4） 利用例2：開催日までの日数と時間を表示する。

「利用例1：開催日までの日数を表示する」の例だと、eventdate 項目の type が"datetime"だった場合に困る事があります。

例えば開催日の値が「2007 年 10 月 10 日 10 時 00 分」となっていたとします。

先ほどの「あと〇日」のデザインを仕込んだページを 2007 年 10 月 9 日 9 時 00 分」にみると、

```
あと 1 日
```

と確かに表示されます。しかし、これを 2007 年 10 月 9 日 11 時 00 分に見ると、

```
あと 0 日
```

と表示されてしまいます。これは CMS Designer の不具合ではなく、時間差が 24 時間未満になった為に「1 日」とはカウントされなくなったのです。

これだと、実際には明日開催なのに、もう終了したと思われてしまう可能性があります。これを回避するには「当日かどうか」等の判定を行う必要があり、多少面倒です。

これに対する最も簡単な解決方法は、日数と時間を表示することです。

```
あと 0 日と 23 時間
```

のように表示されれば、誤解されることもないでしょう。

これは次のように行います。

```
あと<xsl:value-of
  select="floor((eventdate/@time - /*/today/@time) div (60*60*24))" />日と
<xsl:value-of
  select="floor(((eventdate/@time - /*/today/@time) mod (60*60*24)) div
(60*60))" />時間
```

「mod」は左辺を右辺で割った余り（剰余）を求める命令です。

60*60*24、つまり一日の秒数で割った余りを求めています。結果として、1 日以下の半端な秒数が得られます。

この結果を「時」に変換する為に、さらに 60*60（1 時間の秒数）で割り、結果を floor 関数で丸

めています。

詳しく理解していると難しいので、基本的にこの形のまま、項目名を差し替えて使って頂ければ幸いです。

５．５．１６ エントリ作成者名を表示する。

ver.1.1.7a より、エントリを作成したユーザーID を取得できるようになりました。

ver.1.1.7a より前のバージョンで保存したエントリには、この情報は存在しませんのでご注意下さい。その場合、システム管理者ユーザーでログインして保存しなおせば、ユーザーID を保存できます。

エントリ作成者名を取得するには、@author を使用します。

```
作成者 ：<xsl:value-of select="@author" />
```

ver.1.1.7a より前のバージョンで保存したエントリの場合、ここが空欄になってしまう為、これを回避する為には次のように記述します。

```
<!-- 作成者が保存されている場合のみ作成者名を表示 -->
<xsl:if test="@author)">
作成者 ：<xsl:value-of select="@author" />
</xsl:if>
```

又は、次のようにも書けます。

```
<!-- 作成者を表示。作成者名が無い場合「（未設定）」と表示-->
作成者 ：<xsl:value-of select="@author" /><xsl:if test="not(@author)">（未設定）</xsl:if>
```

５．５．１７ メニュー項目の選択肢リストを表示する。

Ver.1.1.18a にて、メニュー項目の選択肢リストをデザイン定義上で取得して表示できるようになりました。例えば以下のようなスキーマのメニュー項目があった場合に、「中華」「ラーメン」「和食」「洋食」といった選択肢の一覧をデザイン定義上で取得することができず、スキーマに定義した選択肢リストを再度デザイン定義上でも記載する必要があり、面倒でした。

```
<data name="shopkind" type="menu" caption="お店種別" >
  <menuitem id="1">中華</menuitem>
  <menuitem id="2">ラーメン</menuitem>
  <menuitem id="3">和食</menuitem>
  <menuitem id="4">洋食</menuitem>
</data>
```

今後は、デザイン定義から上記の選択肢リスト全体を取得して表示することが可能となります。
以下の例は、選択肢リストを li 要素でリスト表示させるデザイン定義です。

選択肢リストを li 要素でリスト表示させるデザイン定義（エントリー覧用）

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="xml" encoding="UTF-8" omit-xml-declaration="yes" />
  <xsl:template match="/entrylist">

    <xsl:for-each select="menuinfo[@name='shopkind']">
      <ul>
        <xsl:for-each select="menuitem">
          <li><xsl:value-of select="." disable-output-escaping="yes" />
              (<xsl:value-of select="@id" />)</li>
        </xsl:for-each>
      </ul>
    </xsl:for-each>

  </xsl:template>
</xsl:stylesheet>
```

出力は次のようになります。

```
<ul>
  <li>中華(1)</li>
  <li>ラーメン(2)</li>
  <li>和食(3)</li>
  <li>洋食(4)</li>
</ul>
```

エントリー件用のデザイン定義の場合は、上記の「xsl:template match=”/entrylist”」の部分を「xsl:template match=”/entry”」に変更してください。

次のページに、グループ絞り込み機能と組み合わせたページ・タブのサンプルをご紹介します（かなり複雑ですので、XSL に慣れた方向けです）。

グループ絞り込み機能を使ってページ・タブを実現するデザイン定義（エントリー一覧用）

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="xml" encoding="UTF-8" omit-xml-declaration="yes" />
  <xsl:template match="/entrylist">

    <!-- お店種別毎のページ・タブ表示 -->
    <xsl:for-each select="menuinfo[@name='shopkind']">
      <ul>
        <xsl:for-each select="menuitem">
          <li>
            <!-- 絞り込み条件と合致する場合は class に selected を指定する -->
      <xsl:if test="@id=/entrylist/group[@key='shopkind']/@value">
        <xsl:attribute name="class">selected</xsl:attribute>
      </xsl:if>
              <a href="articlelist.php?shopkind={@id}">
                <xsl:value-of select="." disable-output-escaping="yes" />
              </a>
          </li>
        </xsl:for-each>
      </ul>
    </xsl:for-each>

    <!-- グループ絞り込みで現在絞り込まれている shopkind の取得と表示 -->
    <xsl:for-each select="group[@key='shopkind']">
      <xsl:variable name="sk" select="@value" />
      絞り込み：<xsl:value-of select="/entrylist/menuinfo[@name='shopkind']/menuitem[@id=$sk]" />
    </xsl:for-each>

    <!-- エントリー覧の表示 -->
    <xsl:for-each select="entry">
      (省略)

    </xsl:for-each>

  </xsl:template>
</xsl:stylesheet>
```

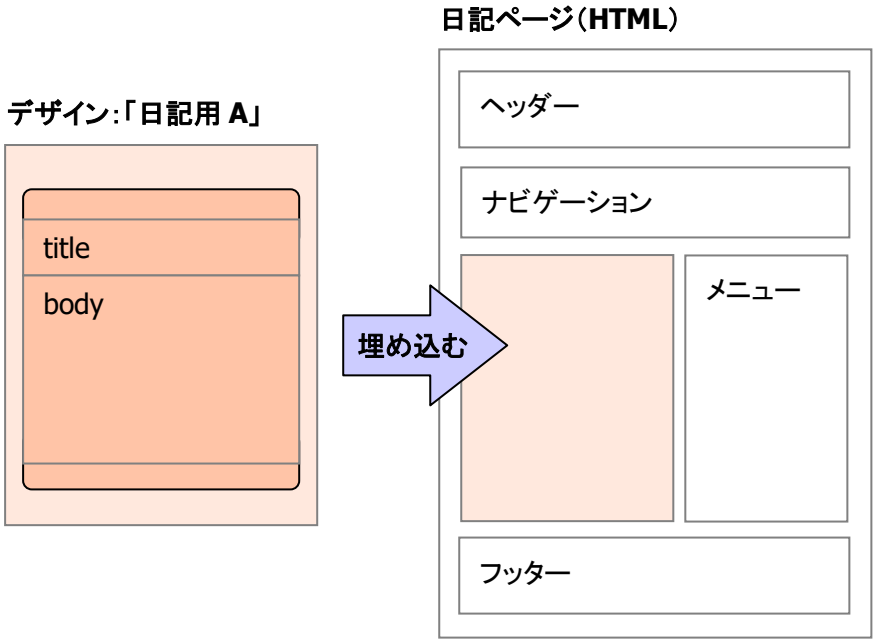
出力例：**shopkind にグループ絞り込み指定をし、shopkind=1 で絞り込み表示された場合**

```
<ul>
  <li><a href=”articlelist.php?shopkind=1” class=”selected”>中華</a></li>
  <li><a href=”articlelist.php?shopkind=2”>ラーメン</a></li>
  <li><a href=”articlelist.php?shopkind=3”>和食</a></li>
  <li><a href=”articlelist.php?shopkind=4”>洋食</a></li>
</ul>
絞り込み：中華
(省略)
(省略)
(省略)
  :
```

6 ウェブサイトへの埋め込み

6．1 デザイン定義と HTML 画面の関係

CMS Designer では、デザイン定義で「画面全体」を作成するような事はしません。
あくまでデザイン定義は「画面の一部分」、エントリに関係ある部分だけをデザインします。
その「画面の一部分」を、画面全体を記述した HTML の中に「埋め込み」ます。



この仕組みにより、デザインを「部品」のように扱うことができるようになり、異なるページで使いまわしたり、スキーマとデザインを１セットにして「〇〇セット」のように配布して広く使ってもらったり、ということが可能になります。

また、画面全体の HTML はこれまで通り dreamweaver などの既存のオーサリングツールを使って編集できますし、逆に既に作成済のページにデザインを埋め込む際にも、更新部分にのみ手を加えれば良いだけなので、非常に簡単に CMS 化することができます。

同一ページに複数のコンテンツを埋め込む事も可能です。例えば、トップページに「新着情報一覧」と「最新の記事」と「お勧め商品」を表示する、というような事が可能です。

６．２ 埋め込み先の画面の作成

埋め込み先画面の作成は、基本的に通常の HTML として作成すれば OK です。
但し、以下の注意事項があります。

① 文字コードを UTF-8 で作成する。
② 拡張子を.php にする。
③ サイトのルート直下に作成する。

新規作成する場合は上記の①②③の通り作成してください。既に存在するページにコンテンツを埋め込む場合は、ツールの文字コード変換機能などを使って UTF-8 に変換し、拡張子を php に変更してください。③については回避方法もありますが、少々ややこしくなる為ここでは説明しません。

埋め込み先 HTML にコンテンツを埋め込むには、例えば次のようにします。

```
<?php require( "cmsdesigner/include/view.php.inc" ); // encoding="UTF-8" ?>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>テストページです。</title>
</head>
<body>
テストページ<br>
<table border="1">
  <tr>
    <td>
      メニュー部分
    </td>
    <td>
      <cmsd:entry name="public_diary" design="default" />
    </td>
  </tr>
</table>
</body>
</html>
<?php cmsd_end_template(); ?>
```

1行目と最終行の

```
<?php require( "cmsdesigner/include/view.php.inc" ); // encoding="UTF-8" ?>
```

```
<?php cmsd_end_template(); ?>
```

は、定型文として入れてください。（「// encoding="UTF-8"」は Dreamweaver の文字化け（不具合？）回避の為のおまじないです。）

```
      <cmsd:entry name="public_diary" design="default" />
```

この箇所が、コンテンツを埋め込んでいる部分です。この部分を「埋め込み命令」と呼びます。
埋め込み命令の詳細は「６．３ 埋め込み」を参照してください。

６．３ 埋め込み命令タグ

６．３．１ 埋め込み命令タグ一覧

埋め込み命令タグの仕様です。タグの詳しい使い方は６．３．２以降を参照して下さい。

（1）cmsd:entry タグ - エントリ 1 件出力

属性値	省略	内容
name	不可	エントリフォルダ名。
design	可	デザイン定義名。output 属性を"xml"にした場合は省略可能。又は dataname 属性を指定した場合は同時に指定できない。
eid	可	表示するエントリを固定したい場合にエントリ ID を指定する。 "top"を指定すると、常に先頭エントリを表示する。 "random"を指定すると、ランダムなエントリを表示する。 ※この属性を指定すると、エントリ切替ナビゲーションが出力されません。 ※eid に"top"又は"random"を指定すると、多少処理速度が落ちます。
tagremoving	可	省略値は"off"。"on"を指定すると、エントリデータ中の HTML タグが除去されて表示される。データそのものから削除される訳ではないので注意。
navigation	可	「前のエントリ」「次のエントリ」等のナビゲーションをつけたい時に"on"を指定する。
output	可	出力形式を指定。"xml"又は"rss"を指定する。省略すると html 形式での出力となる。"xml"を指定した場合、design 属性を省略すると、生のエントリデータが XML 形式で出力される。
dataname	可	エントリ内の特定の１項目のみを直接出力したい場合にその項目名を指定する。この属性を指定した場合、同時に design 属性を指定することはできない。
interlock	可	エントリー覧・詳細連動の指定。"no"又は"yes"を指定する。省略値は"no"。"yes"を指定すると、同一画面上に同じエントリフォルダに対する cmsd:entrylist タグを埋め込む場合に、現在表示されているエントリー覧ページ（pageno）の先頭エントリID（eid）を自動的に選択する。詳しくは６．３．１３ 参照。

例）最も単純な例。
```
<cmsd:entry name="mydiary" design="default" />
```

例）エントリ ID=00001 を固定で表示。
```
<cmsd:entry name="mydiary" design="default" eid="00001" />
```

例）shopkind が 2 のエントリの先頭エントリを表示。
```
<cmsd:entry name="mydiary" design="default" eid="top" >
  <cmsd:group key="shopkind" value="2" />
</cmsd:entry>
```

例）エントリを XML 形式（加工前）で表示。
```
<cmsd:entry name="mydiary" output="xml" />
```

例）エントリの title 項目のみを表示。
```
<cmsd:entry name="mydiary" dataname="title" />
```

（2）cmsd:entrylist タグ - エントリー覧出力

属性値	省略	内容
name	不可	エントリフォルダ名。
design	可	デザイン定義名。output 属性を"xml"にした場合は省略可能。
rows	可	1ページに出力するエントリ数。省略値は 10。
pageno	可	表示するページ番号を固定したい場合に指定する。通常は省略する。"top"を指定すると、常に先頭ページを表示する。 ※この属性を指定すると、ページ切替ナビゲーションが出力されない為、処理速度が向上する。「最新 5 件のみ表示」等の場合には、pageno="top"と指定しておいた方が良い。
tagremoving	可	省略値は"off"。"on"を指定すると、エントリデータ中の HTML タグが除去されて表示される。データそのものから削除される訳ではないので注意。
cols	可	エントリー覧をテーブル状に表示したい時に指定する。列数を指定する。
output	可	出力形式を指定。"xml"又は"rss"を指定する。省略すると html 形式での出力となる。"xml"を指定した場合、design 属性を省略すると、生のエントリデータが XML 形式で出力される。
interlock	可	エントリー覧・詳細連動の指定。"no"又は"yes"を指定する。省略値は"no"。"yes"を指定すると、同一画面上に同じエントリフォルダに対する cmsd:entry タグを埋め込む場合に、現在表示されているエントリ（eid）のを含むエントリー覧のページ番号(pageno）を自動的に選択する。詳しくは6．３．１３ 参照。

例）1 ページ 20 件で出力する（ナビゲーション切替付き）。
```
<cmsd:entrylist name="mydiary" design="default" rows="20" />
```

例）先頭 5 件のみ出力する。
```
<cmsd:entrylist name="mydiary" design="default" rows="5" pageno="top" />
```

例：shopkind=2 のエントリー覧のみを出力する。
```
<cmsd:entrylist name="mydiary" design="default">
  <cmsd:group key="shopkind" value="2" />
</cmsd:entrylist>
```

（3）cmsd:searchresult タグ – 検索結果出力

属性値	省略	内容
design	不可／可	デザイン定義名。デザイン定義ファイルは config/schema/cmsd_system フォルダに固定で格納する。ファイル名は「cmsd_system.list.デザイン定義名.design.xml」となる。 cmsd:design 要素を子要素として指定し、そこにインライン XSLT を記述している場合には、この属性値は指定しない。
rows	可	検索結果を表示する最大エントリ数。ここで指定した件数に達したらそれ以上の検索は行われない。省略値は 100。 ※cmsd:searchresult の出力結果はページ切替に対応**しません**。検索結果は 1 ページに全て出力されます。ページ切替が必要な場合は JavaScript 等を使って対応してください。
以下は子要素	cmsd:searchtarget の属性値。cmsd:searchtarget は複数指定可能。	
name	不可	検索対象のエントリフォルダ名。
title	不可	検索結果のタイトルとして使用したい、対象エントリ（name 属性で指定したエントリ）のスキーマ中の項目名。 例）"shopname"がスキーマ中でタイトルとして使用したい項目の場合、title="shopname"と指定する。
pageurl	不可	検索結果画面から個別の記事へジャンプする際に使用する、ジャンプ先の個別記事表示用ページの URL。 例）"shop.php"が、対象エントリ 1 件分の記事を表示する為のページとして用意されている場合、pageurl="shop.php"と指定する。

例） 検索対象エントリフォルダは1つ。検索結果は 10 件まで表示。

```
<cmsd:searchresult rows="10" design="simple">
    <cmsd:searchtarget name="news1" title="title" pageurl="news.php" />
</cmsd:searchresult>
```

例： 検索対象エントリフォルダを 2 つ指定した場合。検索結果表示件数（rows）は省略（100）。

```
<cmsd:searchresult design="simple">
    <cmsd:searchtarget name="news1" title="title" pageurl="news.php" />
    <cmsd:searchtarget name="tabearuki1" title="shopname" pageurl="shop.php" />
</cmsd:searchresult>
```

６．３．２ エントリ1件分の埋め込み- cmsd:entry 命令タグ(1)

エントリ1件分のコンテンツを埋め込む際は、cmsd:entry 命令タグを使います。
この命令の最も基本的な使い方は次の通りです。

```
<cmsd:entry name="エントリフォルダ名" design="デザイン名" />
```

```
例：<cmsd:entry name="mydiary" design="default" />
```

指定したエントリフォルダのエントリを、指定したデザイン定義を使ってこの場所へ埋め込みます。
エントリフォルダ名は、エントリ定義で定義した名前を、デザイン名には、デザイン定義で決めたデザイン名を指定します（デザインファイル名ではなく、デザイン名です）。

実際に表示するエントリのエントリ ID は、URL から指定します。
例えば mypage.php という名前の埋め込み先画面を作成したとすると、

```
http://あなたのサイトの URL/mypage.php?eid=エントリ ID
```

```
例：http://www.hogehoge.com/mypage.php?eid=00001
```

のように指定します。

実際には、ユーザーが直接このような URL を入力する必要はありません。
「５．５．８ エントリー覧から個別のエントリへリンクを張る」に記述されているように、一覧表示用デザイン定義と組み合わせて使用します。

又、あまりないとは思いますが、場合によってはエントリ ID を外部から受け取らず、固定で表示したい場合があります。例えば定番商品をトップページに恒久的に表示したい場合などです。この場合は、以下のように直接エントリ ID を埋め込みます。

```
<cmsd:entry name="エントリフォルダ名" design="デザイン名" eid="エントリ ID" />
```

```
例：<cmsd:entry name="mydiary" design="default" eid="00001" />
```

こうすると、URL から eid パラメータを指定してもしなくても、埋め込んだエントリ ID のエントリが常に表示されるようになります。

エントリ ID は、コンテンツ管理画面のエントリー覧から確認できます。

６．３．３ エントリー件分の埋め込み（ナビゲーション付き） - cmsd:entry 命令タグ(2)

『５．５．８ 「次のエントリへ」「前のエントリ」へのリンクをつける』で解説している「記事のナビゲーション」付きデザインを有効にするには、cmsd:entry 命令タグに次のように指定します。

```
<cmsd:entry name="エントリフォルダ名" design="デザイン名" navigation="on" />
```

```
例：<cmsd:entry name="mydiary" design="default" navigation="on" />
```

navigation="on"を追加するだけです。

以下のような記事のナビゲーションを出力することができます。

<<前の記事へ | 次の記事へ>>

この指定は「ナビゲーションを表示して下さい」と指令を送っているだけで、実際にナビゲーションを表示するには、ナビゲーション表示用のデザイン定義を行う必要があります。記事のナビゲーションデザインを定義する方法は、『５．５．９ 「次のエントリへ」「前のエントリ」へのリンクをつける』を参照してください。

６．３．４ エントリー覧の埋め込み - cmsd:entrylist 命令タグ

エントリー覧のコンテンツを埋め込む際は、cmsd:entrylist 命令タグを使います。

この命令の最も基本的な使い方は次の通りです。

```
<cmsd:entrylist name="エントリフォルダ名" design="デザイン名" />
```

```
例：<cmsd:entrylist name="mydiary" design="default" />
```

指定したエントリフォルダのエントリー覧を、指定したデザイン定義を使ってこの場所へ埋め込みます。規定では、エントリー覧の表示件数は 10 件で、それ以降はページ切替ナビゲーションを使ってページを切り替えて表示します。ページ切替ナビゲーションのデザイン定義については、『５．５．９　一覧表示で「次のページへ」「前のページへ」のリンクをつける』を参照してください。

１ページ毎の表示件数を指定する場合、次のようにします。

```
<cmsd:entrylist name="エントリフォルダ名" design="デザイン名" rows="表示件数" />
```

```
例：<cmsd:entrylist name="mydiary" design="default" rows="20" />
```

通常、ページの切替はページ切替ナビゲーションのデザイン定義側で自動的に（URL の pageno パラメータを介して）行われますが、「５ページ目」や「１ページ目」のように、ページ番号を固定で表示させることもできます。

```
<cmsd:entrylist name="エントリフォルダ名" design="デザイン名" pageno="ページ番号" />
```

```
例：<cmsd:entrylist name="mydiary" design="default" pageno="5" />
```

pageno="top"と指定すると、必ず１ページ目が表示されるようになります。

```
例：<cmsd:entrylist name="mydiary" design="default" pageno="top" />
```

※タグ内で直接 pageno を指定すると、ナビゲーションが出力されなくなります。

例えばサイトのトップページ等で「最新エントリ５件表示」を実現したい場合は次のようにします。

```
例：<cmsd:entrylist name="mydiary" design="default" pageno="top" rows="5" />
```

６．３．５ 絞込みの指定（固定）

『３．５．１４ グループ（絞込み）指定』にて、グループを指定したスキーマを作成した場合、グループによる絞込み表示を行うことができます。

グループによる絞込み表示とは、例えば「都道府県」という項目をグループ指定した場合に、「都道府県＝福井県」等で絞込んだ結果を表示することができます。

一覧表示の場合、以下のようにグループ項目名と絞込みたい値を指定します。

```
<cmsd:entrylist name="エントリフォルダ名" design="デザイン名">
  <cmsd:group key="項目名" value="絞り込む値" />
</cmsd:entrylist>
```

```
例：<cmsd:entrylist name="mydiary" design="default">
      <cmsd:group key="shopkind" value="2" />
    </cmsd:entrylist>
```

上記の例だと、グループ項目「shopkind」の値が"2"のエントリだけを一覧表示します。

グループ項目を複数設定している場合は、cmsd:group タグを増やします。

```
例：<cmsd:entrylist name="mydiary" design="default">
      <cmsd:group key="shopkind" value="2" />
      <cmsd:group key="review" value="5" />
    </cmsd:entrylist>
```

上記の例では、"review"というグループ項目が"5"で、且つ、"shopkind"というグループ項目が"2"のエントリだけを抽出して一覧表示します。

エントリ１件表示の場合でも、記事のナビゲーション（章「５．５．９ 」を参照）をつける場合、指定したグループ内で記事を切り替えていくことができます。

```
<cmsd:entry name="エントリフォルダ名" design="デザイン名">
  <cmsd:group key="項目名" value="絞り込む値" />
</cmsd:entry>
```

```
例：<cmsd:entry name="mydiary" design="default">
      <cmsd:group key="shopkind" value="2" />
    </cmsd:entry>
```

６．３．６ URL パラメータからの動的な絞込みの指定

『６．３．５ 絞込みの指定』の方法だと、固定の絞込み条件しか指定できません。
例えば「種別」ごとに絞込み表示を行うとして、「種別」が１０種類あったとしたら、

list_kind01.php
list_kind02.php
list_kind03.php
　　：
list_kind10.php

のように、それぞれの種別毎に埋め込みページを作らなくてはならなくなります。

しかし、これらの埋め込みページは絞込み条件部分以外はほとんど同じであることが多く、できれば共通化して、絞込み条件だけを外部から与えるようにしたいと思う事でしょう。

これを実現するには、cmsd:group タグの value 属性を指定しないようにします。

```
例：<cmsd:entrylist name="mydiary" design="default">
      <cmsd:group key="shopkind" />
      <cmsd:group key="review" />
    </cmsd:entrylist>
```

そして、これを埋め込んだページの URL に、次のように絞込み条件を与えて呼び出します。

```
http://xxx.xxx.xxx/list_kind.php?review=1&shopkind=5
```

これで、「review が 1 で且つ、shopkind が 5 のエントリー覧」を表示させることができます。

この仕組みを利用して、簡単な検索システムのようなものを作ることも出来ます。
上記の shopkind と review での絞り込みであれば、次のようなフォームから php ページに飛ばすようにすれば、自動的に URL にパラメータが与えられる為、利用者が検索した結果を表示することができます。

```
<form method="get" action="list_kind.php">

お店の種別：<select name="shopkind"><option selected value="">指定しない
</option><option value="1">中華</option><option value="2">和食</option><option
value="3">洋食</option><option value="4">デザート</option><option value="5">その
他</option></select>

評価：<select name="review">
<option selected value="">指定しない</option><option value="1">★</option><option
value="2">★★</option><option value="3">★★★</option><option value="4">★★★
★</option><option value="5">★★★★★</option></select>

</form>
```

６．３．７ text/textarea 項目のタグ除去

text/textarea 項目のテキストを出力する際に、タグが入っていたら除去してから表示する機能です。

cmsd:entry 命令タグ、cmsd:entrylist 命令タグに、tagremoving="on"属性を指定します。

```
<cmsd:entrylist name="エントリフォルダ名" design="デザイン名" tagremoving="on" />
```

```
<cmsd:entry name="エントリフォルダ名" design="デザイン名" tagremoving="on" />
```

この機能は、次のような場合に利用します。

例えば、デザイン定義にて次のように書くことで、「先頭 20 文字分だけ出力」ということが出来ました。ニュース系のサイトで、本文の要約を表示したい時などに利用します。

```
<xsl:value-of select="substring(description/text(), 0, 100)" />
```

これは、XSLT の「substring()」命令を使い、description 項目の 0 文字目から 100 文字分を出力しています。

しかし、もし description 項目に HTML タグが含まれる場合、HTML タグも 1 文字として数える為、HTML タグの途中で文章が切れてしまう可能性があります。その場合、HTML の整合性が取れなくなり、画面表示が崩れてしまいます。

この際に、このタグ除去機能を使えば問題が解決します。

６．３．８　RSS の出力

エントリー覧の情報を、RSS で出力することができます。

但し、この機能を使うには RSS 出力専用のデザイン定義ファイルを作成する必要があり、少々作業が必要になります。RSS 出力用のデザイン定義ファイルのサンプルは公式サイトよりダウンロード可能です。

まず、rss 配信用の PHP ファイルを作成します。

仮に rss.php とします。

```
<?php require_once( "cmsdesigner/include/view.php.inc" ); // encoding="UTF-8" ?>
<cmsd:entrylist name="mydiary" design="rss" output="rss" />
<?php cmsd_end_template(); ?>
```

非常に短い記述ですが、余計な HTML タグが無いだけで実際には通常の埋め込みページと構造は変わりません。

次に、cmsd:entrylist 命令タグの design 属性で指定している"rss"という名前のデザイン定義を作成します。これは、エントリー覧データを RSS に変換する為のデザイン定義で、次のページのデザイン定義サンプルを元に作成して下さい。

ここでのファイル名は、mydiary エントリのスキーマ名が diary と仮定すると、RSS 出力用のデザイン定義ファイル名は「diary.list.rss.design.xsl」になります。

（↓次のページへ）

■RSS 出力用デザイン定義ファイル雛形サンプル

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet
  version="1.0"
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:sy  = "http://purl.org/rss/1.0/modules/syndication/"
  xmlns:dc  = "http://purl.org/dc/elements/1.1/"
  xmlns:xh  = "http://www.w3.org/1999/xhtml"
  xmlns     = "http://purl.org/rss/1.0/" >
  <xsl:output method="xml" encoding="UTF-8" omit-xml-declaration="no" indent="no" />
  <xsl:variable name="baseUrl"     select="'あなたのサイトのルート URL'" />
  <xsl:variable name="index"       select="concat($baseUrl, 'サイトのトップページ名')" />
  <xsl:variable name="rdfphp"      select="concat($baseUrl, 'RSS 配信ページ名')" />
  <xsl:variable name="entryphp"    select="concat($baseUrl, '1件分のエントリ表示ページ名')" />
  <xsl:variable name="title"        select="'サイト名'" />
  <xsl:variable name="description" select="'サイトの説明'" />
  <xsl:variable name="subject"      select="'カテゴリ名'" />

  <xsl:template match="/entrylist">
    <rdf:RDF xmlns="http://purl.org/rss/1.0/"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xml:lang="ja">
      <channel rdf:about="{$index}">
        <title><xsl:value-of select="$title" /></title>
        <link><xsl:value-of select="$index" /></link>
        <description><xsl:value-of select="$description" /></description>
        <items>
          <rdf:Seq>

            <xsl:for-each select="entry">

              <rdf:li rdf:resource="{$entryphp}{@href}"/>

            </xsl:for-each>

          </rdf:Seq>
        </items>
      </channel>
      <xsl:for-each select="entry">
        <item rdf:about="{$entryphp}{@href}">
          <title><xsl:value-of select="タイトル用項目名" /></title>
          <link><xsl:value-of select="concat($entryphp,@href)" /></link>
          <description><xsl:value-of select="説明用項目名" /></description>
          <dc:date><xsl:value-of select="@year" />-<xsl:value-of select="@month"
/>-<xsl:value-of select="@day" />T<xsl:value-of select="@hour" />:<xsl:value-of
select="@minute" />:<xsl:value-of select="@second" />+09:00</dc:date>
          <dc:subject><xsl:value-of select="$subject" /></dc:subject>
        </item>
      </xsl:for-each>
    </rdf:RDF>
  </xsl:template>
</xsl:stylesheet>
```

例えば、CMS Designer のお知らせ配信に使っている RSS 配信用デザイン定義は次のようになっています。

■「CMS Designer ウェブサイトお知らせ配信」RSS 出力用デザイン定義ファイルサンプル

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet
  version="1.0"
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:sy  = "http://purl.org/rss/1.0/modules/syndication/"
  xmlns:dc  = "http://purl.org/dc/elements/1.1/"
  xmlns:xh  = "http://www.w3.org/1999/xhtml"
  xmlns     = "http://purl.org/rss/1.0/" >
  <xsl:output method="xml" encoding="UTF-8" omit-xml-declaration="no" indent="no" />
  <xsl:variable name="baseUrl"     select="'http://cms.al-design.jp/'" />
  <xsl:variable name="index"       select="concat($baseUrl, 'index.php')" />
  <xsl:variable name="rdfphp"      select="concat($baseUrl, 'topnewsrss.php')" />
  <xsl:variable name="entryphp"    select="concat($baseUrl, 'topnewsentry.php')" />
  <xsl:variable name="title"       select="'CMS Designer からのお知らせ'" />
  <xsl:variable name="description" select="'CMS Designer からのお知らせです。'" />
  <xsl:variable name="subject"     select="'お知らせ'" />

  <xsl:template match="/entrylist">
    <rdf:RDF xmlns="http://purl.org/rss/1.0/"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xml:lang="ja">
      <channel rdf:about="{$index}">
        <title><xsl:value-of select="$title" /></title>
        <link><xsl:value-of select="$index" /></link>
        <description><xsl:value-of select="$description" /></description>
        <items>
          <rdf:Seq>
            <xsl:for-each select="entry">
              <rdf:li rdf:resource="{$entryphp}{@href}"/>
            </xsl:for-each>
          </rdf:Seq>
        </items>
      </channel>

      <xsl:for-each select="entry">
        <item rdf:about="{$entryphp}{@href}">
          <title><xsl:value-of select="title" /></title>
          <link><xsl:value-of select="concat($entryphp,@href)" /></link>
          <description><xsl:value-of select="description" /></description>
          <dc:date><xsl:value-of select="@year" />-<xsl:value-of select="@month"
/>-<xsl:value-of select="@day" />T<xsl:value-of select="@hour" />:<xsl:value-of
select="@minute" />:<xsl:value-of select="@second" />+09:00</dc:date>
          <dc:subject><xsl:value-of select="$subject" /></dc:subject>
        </item>
      </xsl:for-each>
    </rdf:RDF>
  </xsl:template>
</xsl:stylesheet>
```

Full 版に同梱されている cmsd_doc_rss.zip も参考にして下さい。

６．３．９ XML 形式での出力（Flash や Ajax との連携）

エントリデータを XML 形式で出力する方法です。上級者向けです。

場合によってはコンテンツを、HTML/XHTML ではなく XML データとして出力したい場合があります。代表的なものは、Flash との連携や、Ajax での利用です。

又、特に用途が無くても、XSLT 変換する前の XML データの状態を確認しておくと、デザイン定義をする上で大いに役立つでしょう。

エントリー覧の XML を表示するには、次のようにページを作成します。

```
<?php require_once( "cmsdesigner/include/view.php.inc" ); // encoding="UTF-8" ?>
<cmsd:entrylist name="mydiary" output="xml" />
<?php cmsd_end_template(); ?>
```

上記の内容「だけ」が記述された php ファイルを作成し、そこへアクセスすると、HTML に変換する前の XML データがそのまま取得できます。IE 以外のブラウザで表示しても文字化けしたりおかしく表示されたりするので、「ソースを表示」でソースを確認してください。

また、XSLT に詳しい方は、エントリデータを好きな形の XML データに加工するデザイン定義ファイルを作成して、次のようにデザイン定義ファイルを指定することができます。

```
<?php require_once( "cmsdesigner/include/view.php.inc" ); // encoding="UTF-8" ?>
<cmsd:entrylist name="mydiary" design="myxml1" output="xml" />
<?php cmsd_end_template(); ?>
```

エントリー件分のエントリデータも、同様に output 属性を指定することで XML として取得できます。

```
<?php require( "cmsdesigner/include/view.php.inc" ); // encoding="UTF-8" ?>
<cmsd:entry name="mydiary" output="xml" />
<?php cmsd_end_template(); ?>
```

この場合は URL パラメータから「http://xxxx/xxxx.php?eid=エントリ ID」のように、エントリ ID を与えて下さい。

Flash や Ajax での XML の扱いに慣れている方であれば、エントリー覧の要約のみ XML データとして取得し、必要なエントリだけを再度 XML データとして個別に取得して表示する……というような高度な使い方も可能です。

６．３．１０ 実行時ソート

おまけ的な機能です。パフォーマンスに関してまったく考慮されていない為、必要なパフォーマンスが得られない場合があります。注意して使用して下さい。

CMS Designer では基本的に「エントリ投稿時に、事前に決められた順番でソートしておき、その順序で表示を行う」ようになっています。これはパフォーマンスを考慮しての仕様です。しかし、場合によっては実行時（表示時）に好きな順序でソートしなおして表示したい事もあると思います。
以下はそれを実現する手順です。

まず、ソートに使用したい項目に対して、スキーマ側でグループ項目の指定を行います。詳しくは「３．５．１４ グループ（絞込み）指定」をご覧下さい。
次に、埋め込み命令タグに以下のタグを追加します。

```
<cmsd:entrylist name="エントリフォルダ名" design="デザイン名">
  <cmsd:sort key="項目名" order="asc|desc" type="text|number" />
</cmsd:entrylist>
```

```
例 <cmsd:entrylist name="shop1" design="default">
      <cmsd:sort key="review" order="desc" />
   </cmsd:entrylist>
```

尚、order、type は省略できます。
order には asc（昇順）又は desc（降順）を指定します。指定しないと降順になります。
type には key に指定した項目を文字列として比較するか、数値として比較するかを指定します。通常は省略すれば OK です。文字列の場合は text、数値の場合は number と指定してください。
type が必要な場合とは、例えば 1000 と 200 を比較すると、type="text"の場合は 200 の方を「大きい値」と判断します。文字列の比較は先頭の文字から順に比べてゆくので、1 と 2 を比較するのです。type="number"を指定すれば、正しく 1000 を「大きい値」と判断します。

ソート項目にエントリ ID を指定することも可能です。その場合は、key="@id"と指定します。
尚、ソート項目にエントリ更新日付を指定することは現在出来ません。

ソート項目は複数指定可能です。

実行時ソートの高度な使い方として、URL パラメータから動的にソート条件を与える事ができます。

埋め込み命令タグを以下のように指定します。

```
<cmsd:entrylist name="エントリフォルダ名" design="デザイン名">
  <cmsd:sort name="任意の URL パラメータ名" />
</cmsd:entrylist>
```

「任意の URL パラメータ名」とは、ソート条件を与える為の URL パラメータの名前です。例えば仮に「sortparam1」とします。

```
<cmsd:entrylist name="shop1" design="default">
  <cmsd:sort name="sortparam1" />
</cmsd:entrylist>
```

次に、その PHP ページをブラウザから呼び出す際に、URL パラメータとして次のようなパラメータを指定します。

```
http://ドメイン名/testxxx.php?sortparam1=review,desc
```

パラメータの形式は次の通りです。

```
http://ドメイン名/xxx.php?URL パラメータ名=ソート項目名,順序,型
```

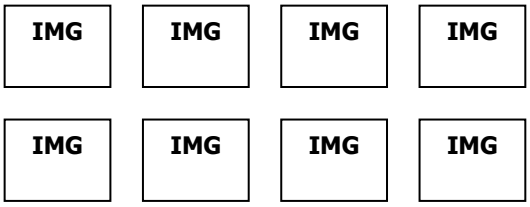
ソート項目名、順序、型をそれぞれカンマで区切って指定します。順序、型は省略可能です。

さらに高度な応用として、HTML のフォームから呼び出す事で利用者に任意のソートを実行させる事ができます。

```
<form method="get" action="testxxx.php">
ソート方法：
<select name="sortparam1">
  <option selected value="">指定しない</option>
  <option value="review,asc">評価（昇順）でソート</option>
  <option value="review,desc">評価（降順）でソート</option>
  <option value="shopkind,asc">お店種別でソート</option>
</select>
</form>
```

６．３．１１ エントリー覧のテーブル表示機能

例えば写真集をCMS Designerで管理していて、1写真＝1エントリだったとします。この場合、エントリー覧表示画面で、縦に写真を並べるのではなく、横4列ぐらいのTABLEにサムネイルを収めて表示したい時があります。商品一覧なども同様の要求があるでしょう。



しかし、普通にエントリー覧用のデザイン定義でそれを行おうとすると、途方にくれることになります。なぜならリストデータを「n個ごとにひとまとめにする」（この場合は<tr></tr>の中に入れる）という処理は、XSLT では非常に苦手な処理だからです。ちなみに、n個ごとに何か（
等）を挿入する、というだけなら簡単にできます。

そこで CMS Designer 側で事前にエントリー覧をn個ごとにブロック化しておくことで、デザイン（XSLT）側ではそのブロック単位にTRタグを出力するだけで良いようにしました。具体的には次のような手順になります。

【使用方法】

まず、埋め込み命令タグに以下のように cols 属性を指定します。

```
<cmsd:entrylist name="エントリフォルダ名" design="デザイン名" cols="列数" />
```

```
例 <cmsd:entrylist name="shop1" design="default" cols="4" />
```

cols="4"と指定すると、4列のテーブルを出力する事ができます。

しかし、これだけではまだ準備が整っただけです。実際にTABLEタグを出力するにはデザイン定義側もあわせて変更する必要があります。

（変更前のデザイン定義）

```
<xsl:template match="/entrylist">
  <xsl:for-each select="entry">
    ～
  </xsl:for-each>
</xsl:template>
```

（変更後のデザイン定義）

```
<xsl:template match="/entrylist">
  <xsl:for-each select="row">
    <xsl:for-each select="entry">
      ～
    </xsl:for-each>
  </xsl:for-each>
</xsl:template>
```

つまり、<xsl:for-each select="entry">の部分を<xsl:for-each select="row">で囲ってください。

TABLE タグを用いて出力を行うデザイン定義の例を示します。

```
<xsl:template match="/entrylist">
  <xsl:for-each select="navi">
    ～
  </xsl:for-each>

  <table border="1">
    <xsl:for-each select="row">
      <tr>
        <xsl:for-each select="entry">
          <td>
          ～
          </td>
        </xsl:for-each>
      </tr>
    </xsl:for-each>
  </table>

</xsl:template>
```

これで、以下のような出力が得られます。

```
<table border="1">
  <tr>
    <td>～</td>
    <td>～</td>
    <td>～</td>
  </tr>
  <tr>
    <td>～</td>
    <td>～</td>
    <td>～</td>
  </tr>
</table>
```

デザイン定義を変更せずに埋め込み命令タグ側で cols="x"の指定のみを行った場合、画面に何も出力されなくなるのでご注意下さい。

６．３．１２ エントリ項目の直接出力

例えば、HTML の TITLE タグにエントリの特定の項目を出力したい時があります。
これを実現するには、デザイン定義にて<html>～</html>の全ての HTML を定義すれば可能です。しかし、CMSD のコンセプトとしてはできる限りデザイン定義はシンプルに局所的に留めておきたい為、できれば回避したい方法です。

他の方法として、タイトルだけを出力するシンプルなデザイン定義を別に作成し、title タグの中にその出力結果を埋め込むという方法もあります。この方法は CMSD 的には正しいのですが、たかが1項目を出力するのにデザイン定義を作成するのは結構面倒です。

第 3 の方法が、この「エントリ項目の直接出力」です。この方法では、項目名を指定してデザイン定義なしに、その項目だけを出力することができます。

```
<cmsd:entry name="エントリフォルダ名" dataname="項目名" />
```

```
例：<cmsd:entry name="mydiary" dataname="shopname" />
```

design 属性が指定されていない事に注意して下さい。dataname 属性と design 属性を同時に指定することはできません。

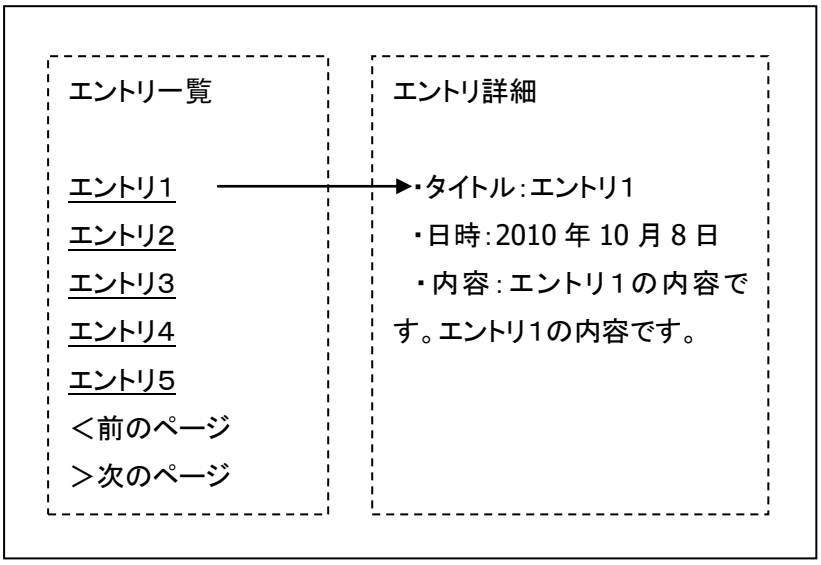
実際に TITLE タグに埋め込むには次のようにします。

```
<?php require_once( "cmsdesigner/include/view.php.inc" ); // encoding="UTF-8" ?>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title><cmsd:entry name="myshop" dataname="shopname" /></title>
</head>
<body>
  :
```

６．３．１３ 一覧ー詳細連動機能

同じコンテンツのエントリー覧とエントリ詳細が同一ページ上に同居している場合に、それらのナビゲーションを連動させます。

例えば次のようなレイアウトになっているような場面です。



エントリー覧からタイトルのリンクをクリックすると、ページ右側のエントリ詳細にそのエントリの詳細が表示されるようなレイアウトです。
エントリー覧側に「次のページ」「前のページ」のようなリンクがなく、ページ切替をしない場合にはこれで問題ないのですが、この例のようにページ切替がある場合に問題が生じます。

例えばエントリー覧の「次のページ」を何度か押して一覧のページを切り替えた場合に、エントリー覧には「エントリ１１～エントリ１５」までが並んでいても、エントリ詳細には「エントリ１」の内容が表示されつづけてしまいます。これは、エントリー覧側のリンクにエントリ ID 情報が付与されていない為です。
同様に、ページ切替して3ページ目を表示している状態で「エントリ１３」のリンクを押すと、エントリ詳細の内容は確かにエントリ１３に切り替わりますが、ページ番号情報が失われる為、エントリー覧側が1ページ目に戻ってしまいます。

連動機能ではこれらの問題を解決する為に、以下の機能を有効にすることができるようになりました。
・詳細画面では、エントリ ID が与えられなかった場合に、与えられたページ番号から該当ページの先頭エントリを表示する。
・一覧画面では、ページ番号が与えられなかった場合に、与えられたエントリ ID からそのエントリを含むページを表示する。

cmsd:entry タグに interlock="yes"という指定を追加すると、現在表示されているページの先頭エントリが自動的に表示されるようになります。また、cmsd:entrylist タグにも同様に interlock="yes"指定をすると、現在表示されているエントリを含むエントリー覧ページが自動的に選択されます。※連動表示は、実行時ソート機能を使っている場合は正常に動作しません。

例：<cmsd:entry name="mydiary" dataname="shopname" interlock="yes" />

例：<cmsd:entrylist name="mydiary" dataname="shopname" interlock="yes" />

また、エントリー覧のデザイン定義（*.xsl）側で、「そのエントリが現在表示されているか」を取得することができます。値は「@current」で取得可能で、表示されている（カレントである）ならば@current 属性が存在します。カレントでない場合はこの属性自体が存在しません。

例として、エントリー覧のタイトルに詳細表示用のリンクをつけ、現在表示されている場合にはリンクを外すデザイン定義を示します。

```
<xsl:for-each select="entry">
  <xsl:if test="@current"><xsl:value-of select="title" /></xsl:if>
  <xsl:if test="not(@current)">
    <a href="{@href}"><xsl:value-of select="title" /></a>
  </xsl:if>
</xsl:for-each>
```

リンクを外すのではなく、強調表示などにしたい場合の例も以下にご紹介します。xsl:attribute 命令を使って次のようにします。

```
<xsl:for-each select="entry">
  <a href="{@href}">
    <xsl:if test="@current">
      <xsl:attribute name="style" >font-weight:bold;</xsl:attribute>
    </xsl:if>
    <xsl:value-of select="title" />
  </a>
</xsl:for-each>
```

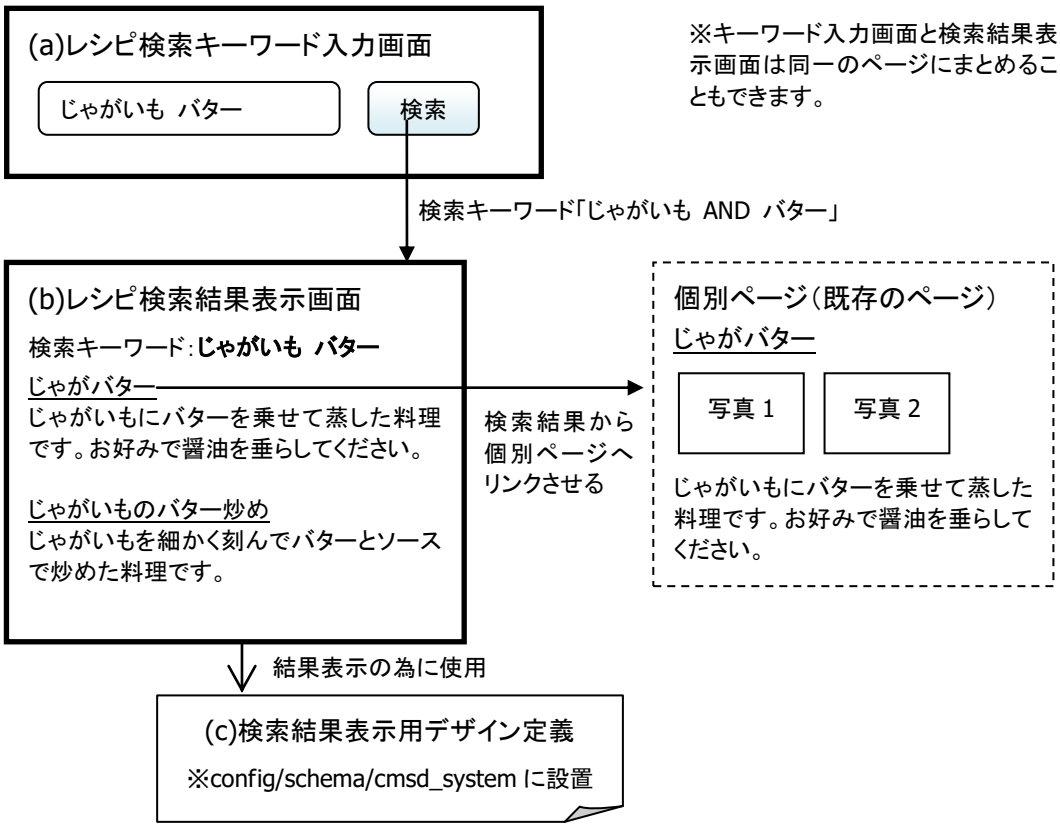
６．３．１４ フリーワード検索機能（複数キーワードによる AND 検索）

例えば料理レシピのようなエントリが増えて数百件を超えてくると、カテゴリ分けだけでは目的の商品を訪問者に案内するのが困難になります。フリーワード検索機能により、複数のエントリフォルダから、指定した複数のキーワードを全て含むエントリのみを全文検索し、タイトルと本文（エントリ中の全テキストデータからタグ情報を除去したもの）を表示させてそこから個別記事へとジャンプさせることが出来ます。

CMS Designer はデータベースを使用しておらず、データをファイルベースで管理している為、全文検索処理は高速ではなく、サーバへの負荷も低くはありません。可能であれば Google などのクラウドが提供するサイト内検索機能などで代用し、どうしても必要な場合のみ、性能テストを十分に行ったうえでご利用ください。

（1）フリーワード検索の構成

フリーワード検索機能を使うには、「(a)キーワード入力画面」「(b)検索結果表示画面」「(c)検索結果表示用のデザイン定義ファイル」が必要です。それぞれ次のような関係になります。



(a) 検索キーワード入力画面

例えば「search.html」のようなファイルを作り、次のようなフォームを設置します。このページが検索キーワードを入力する画面になります。

このフォームに入力された「words」パラメータ他が、(b)の検索結果表示画面の埋め込みタグに渡されます。

```
<form method="GET" action="検索結果表示画面の URL">
    <input type="text" name="words" />
    <inpyt type="hidden" name="encoding" value="このページの文字エンコーディング" />
    <input type="submit" value="検索" />
</form>
```

例：
```
<form method="GET" action="searchresult.php">
    <input type="text" name="words" />
    <inpyt type="hidden" name="encoding" value="UTF-8" />
    <input type="submit" value="検索" />
</form>
```

CMS Designer のページは通常 UTF-8 で作られますので、encoding パラメータには UTF-8 を指定して下さい。

(b) 検索結果表示画面

例えば「searchresult.php」のようなファイルを作り、次のような埋め込みタグを記述します。このページが検索結果を表示する画面になります。

埋め込みタグには、検索結果をデザインする為のデザイン定義や、検索対象にするエントリフォルダ名（複数指定可）などを指定します。

cmsd:searchresult タグ、及び cmsd:searchtarget タグの詳細については「フリーワード検索機能（複数キーワードによる AND 検索）」を参照してください。

```
<cmsd:searchresult design="デザイン定義名" rows="検索結果を表示する最大エントリ数">
    <cmsd:searchtarget name="検索対象エントリフォルダ名"
                       title="タイトルとして使用する項目名"
                       pageurl="個別記事表示用URL" />
</cmsd:searchresult>
```

例：エントリフォルダ「recipe1」の内容を検索対象とする。タイトルは「title」を使用。検索結果から個別記事へジャンプする先は recipe.php。

```
<cmsd:searchresult rows="10" design="simple">
    <cmsd:searchtarget name="recipe1" title="title" pageurl="recipe.php" />
</cmsd:searchresult>
```

例：エントリフォルダを複数指定。レシピと販売商品をまとめて検索する。

```
<cmsd:searchresult rows="10" design="simple">
    <cmsd:searchtarget name="recipe1" title="title" pageurl="recipe.php" />
    <cmsd:searchtarget name="item1" title="name" pageurl="shopitem.php" />
</cmsd:searchresult>
```

(c) 検索結果表示用デザイン定義

(b)の design 属性で指定したデザイン定義に対応するデザイン定義ファイルを作成し、config/schema/cmsd_system フォルダに設置します。cmsd_system フォルダが存在しない場合は新しく作成して下さい。下記はデザイン定義の一例です。

```
例：
<xsl:template match="entrylist">
        キーワード：<strong><xsl:value-of select="searchword" /></strong>
        <dl>
        <xsl:for-each select="entry">
                <dt><a href="{@url}"><xsl:value-of select="title" /></a></dt>
                <dd><xsl:value-of select="body" /></dd>
        </xsl:for-each>
        </dl>
</xsl:template>
```

検索キーワードは //entrylist/searchword に空白区切りで渡されます。

各検索結果は、通常のエントリー覧のデザイン定義と同様に、//entrylist/entry にそれぞれ1件ずつ渡されますので、<xsl:for-each select="entry">を使ってデザインして下さい。

entry/@url には個別記事へのリンク用 URL、entry/title には検索結果ごとのタイトル、entry/body には該当エントリの全テキストデータからタグ情報を除去したデータが渡されます。

（2）高度な利用方法： 検索結果表示最大件数を可変にする

検索キーワード入力用フォームから、「10 件／20 件／50 件／100 件」のようなドロップダウンリストを作って検索結果表示最大件数を指定させたい事があるかもしれません。

この場合には、(b)の埋め込みタグ（cmsd:searchresult）の rows 属性を省略し、代わりに(a)のフォームに以下のような形で rows パラメータを追加してください。

```
例：XHTML の例
検索結果の表示最大件数 <select name="rows">
    <option value="10" selected="selected">最大 10 件</option>
    <option value="20">最大 20 件</option>
    <option value="50" >最大 50 件</option>
    <option value="100" >最大 100 件</option>
</select>
```

６．３．１５ デザインパラメータ

デザインパラメータを使って、URL から任意のパラメータをデザイン定義ファイルへ与える事が可能です。これにより、パラメータの内容によってコンテンツの表示方法を変える事ができます。

例えば、URL に「details=on」というパラメータを付けた時にだけ、detail というテキスト項目の内容を表示する、というような事が可能です。

この機能は高度な知識を必要とする為、使用の際には十分注意してください。

（1）デザインパラメータの利用方法

デザインパラメータを利用するには、埋め込みタグ側に以下のようにデザインパラメータ名を指定します。

```
<cmsd:entry name=”xxx” design="xxx" >
    <cmsd:param key=”デザインパラメータ名” />
</cmsd:entry>
```

例えば details というパラメータを作るには、以下のように記述します。

```
<cmsd:entry name=”xxx” design="xxx" >
    <cmsd:param key=”details” />
</cmsd:entry>
```

そして、URL からは次のようにそのパラメータを与えます。

```
http://yourdomain.jp/xxx.php?eid=12345&デザインパラメータ名＝値
```

例えば次のようにします。

```
http://yourdomain.jp/xxx.php?eid=12345&details=on
```

デザイン定義側には、パラメータ名と値が次のような形で渡されます（概略）。

```
<entrylist>
    <param key="デザインパラメータ名 1" value="値" />
    <param key="デザインパラメータ名 2" value="値" />
    :
または、
<entry>
    <param key="デザインパラメータ名 1" value="値" />
    <param key="デザインパラメータ名 2" value="値" />
    :
```

よって、デザイン定義側でこの param/@value を取得して値をチェックするには、xsl:if を使って次のように記述します。

```
<xsl:if test=”//param[@key='デザインパラメータ名']/@value='値'”>
```

```
<xsl:if test=”//param[@key='details']/@value='on'”>
    details パラメータが'on'の場合のみ、詳細情報を表示する。
</xsl:if>
```

もし、上述の xsl:if やまたは xsl:choose 等をたくさん使う場合、何度も記述するのは面倒なので、xsl:variable 命令を使って変数にパラメータ値をコピーしておくと良いでしょう。

```
<xsl:variable name="任意の変数名"　select="//param[@key='デザインパラメータ名']/@value" />
```

xsl:variable で定義した変数は、「$+任意の変数名」という形式でアクセスすることができますので、例えば次のように使用できます。

```
<xsl:variable name="paramdetails" select="//param[@key='details']/@value" />

<xsl:if test="$paramdetails='on'">
    details パラメータが'on'の場合、詳細情報を表示する。
</xsl:if>
<xsl:if test="$paramdetails='all'">
    details パラメータが'all'の場合、詳細情報に加えて住所や写真などもすべて表示する。
</xsl:if>
```

【注意！】
URL パラメータには何を指定されるか分からないので、URL パラメータで受け取った値をデザイン定義中で画面に「表示」する為に使用しては**絶対にいけません。**これをすると、そのページは**セキュリティ上重大な脆弱性**（XSS：クロスサイトスクリプティング）を持つことになり、サーバを攻撃されたり、訪問者の個人情報を盗まれたり、第三者のサーバを攻撃するための踏み台として利用されたりする可能性があります。これは、CMS Designer に限らず全てのサーバサイドスクリプトが守るべきマナーです。

具体的には、次のように、パラメータ値を **xsl:value-of** を使って出力するコードを絶対に記述しないでください。
例： <xsl:value-of select="//param[@key='details']/@value" disable-output-escaping="yes" />
（このコードは絶対にダメ）

例えば、URL パラメータ経由で「あなたの名前」を入力させ、それを画面に表示して「○○さんの運勢は・・・」のような占い機能を作る、というのも NG です。
URL パラメータで受け取った値は、デザイン定義中で xsl:if 等の条件分岐にのみ使用するようにして下さい。

【注意！】
デザインパラメータは非常に強力な機能であり、プログラミング経験のある人であれば、かなり高度な仕組みを構築することもできます。しかし、もともと CMS Designer はノンプログラマー向けのツールであり、あまりに複雑な処理を構築してしまうと、メンテナンスが非常に大変になり、「担当者がいなくなったら誰もメンテナンスできなくなってしまった」という事にもなりかねません。
デザインパラメータは同時に１個か２個程度に抑え、それ以上同時に使う必要が出てきた場合、そもそもの仕様が CMS Designer に向いていない可能性を考慮し、仕様をもっとシンプルなものに変更するか、または適切な別のツールを使うことも検討してみてください。

（2）デザインパラメータの利用例

参考として、デザインパラメータの利用例をいくつかご紹介します。

(a) ある項目の表示の ON/OFF 切替

例えば状況によって詳細情報の表示を ON/OFF したい事があるかもしれません。

このような時に、ページの ON/OFF を切り替える指定をデザインパラメータとしてデザイン定義ファイルに渡すことができます。

例えば、

```
http://yourdomain.jp/xxx.php?eid=12345&viewdetail=off
```

とした場合に写真を非表示にしたいとします。そのためには、まず埋め込みタグに「viewdetail」というデザインパラメータを使用することを伝えます。

デザインパラメータの指定は「cmsd:param」タグで行います。

```
<cmsd:entry name=”xxx” design="xxx" >
    <cmsd:param key=”viewdetail” />
</cmsd:entry>
```

上記の指定によって、「viewdetail」というデザインパラメータを利用できる状態になりました。

次に、デザイン定義ファイルを変更します。デザイン定義側で、現在以下のようになっている箇所があるとします。

```
<div clsss="detail">
    <xsl:value-of select="detail" disable-output-escaping="yes" />
</div>
```

これを、viewdetail=off の場合に非表示にするには、この箇所を以下のように変更します。

```
<!-- viewdetail パラメータを$viewdetail 変数に格納 -->
<xsl:variable name="viewdetail" select="//param[@key='viewdetail']/@value" />

<xsl:if test="$viewdetail!='off'">
<div clsss="detail">
    <xsl:value-of select="detail" disable-output-escaping="yes" />
</div>
</xsl:if>
```

ちょっとソースが見にくいですが、「!=」を使って「off じゃない場合に表示する」ようにしています。

(b) リスト項目の各項目を個別ページとして表示する

例えばあるスキーマに、「この物件の写真」というリスト項目があったとして、それぞれの写真を独立したページとして表示したい事があるかもしれません。しかし、ページに埋め込めるコンテンツの最小単位は「エントリ」単位（cmsd:entry）である為、「リスト項目の 5 番目だけ表示せよ」といった指定はできません。

このような時に、表示したいリスト番号をデザインパラメータとしてデザイン定義ファイルに渡すことができます。

例えば、

```
http://yourdomain.jp/xxx.php?eid=12345&lid=00001
```

のように、リスト番号として「lid」というデザインパラメータを与え、デザイン定義ファイルではそのリスト番号の項目のみを表示する、というやり方です。

具体的には以下のようになります。

■リスト項目１件分表示用ページの埋め込みタグ

```
<cmsd:entry name="xxx" design="xxx">
  <cmsd:param key="lid" />
</cmsd:entry>
```

■リスト項目１件分表示用ページのデザイン定義

```
<xsl:template match="entry">

<!-- lid パラメータを$lid 変数に格納 -->
<xsl:variable name="lid" select="param[@key='lid']/@value" />

<xsl:for-each select="photolist/listitem[@id=$lid]">
リスト項目１件分の表示
</xsl:for-each>
```

■エントリ１件分表示用ページのデザイン定義

```
<xsl:template match="entry">

<!-- lid パラメータを$lid 変数に格納 -->
<xsl:variable name="lid" select="param[@key='lid']/@value" />

<xsl:for-each select="xxxlist/listitem">
  <a href="xxx.php{../../@href}&amp;lid={@id}"><xsl:value-of select="caption1" /></a>
</xsl:for-each>
```

※処理のイメージ

i) 一覧画面で次のような出力がされる

```
<a href="xxx.php?eid=12345&lid=00001">キャプション 1</a>
<a href="xxx.php?eid=12345&lid=00002">キャプション 2</a>
<a href="xxx.php?eid=12345&lid=00003">キャプション 3</a>
```

ii) キャプション 2 をクリックすると、listdetail.php に、lid=00002 が渡される。

iii) listdetail.php で、<xsl:for-each select=”photolist/listitem[@id=$lid]”>としているので、リスト項目の id が 00002 のものだけ表示される。

(c) エントリ毎にパスワードをかける

少し特殊な事例です。CMS Designer には「エントリにパスワードをかける」という機能はありませんが、デザインパラメータを使って擬似的に実現してみましょう。

まず、スキーマに「password」というテキスト項目を1つ追加します。ここにパスワードを入力しておきます。

デザイン定義側では、「パラメータ pass の値と password の内容が同じ場合にのみ、エントリ内容を表示する」ようにしておきます。
具体的には次のようになります。

```
<!-- pass パラメータを$pass 変数に格納 -->
<xsl:variable name="pass" select="//param[@key='pass']/@value" />

<xsl:if test="password=$pass">
  エントリ内容を表示。
</xsl:if>
<xsl:if test="password!=$pass">
  パスワードが違います。
</xsl:if>
```

尚、上記の方法でエントリにパスワードをかけたとしても、一度パスワードでエントリを表示した際に画像の URL をコピーされてしまえば、画像に関しては直接リンクでパスワード無しで表示することができてしまいますので、厳密な機密管理には使用しないでください。

(d) クイズ機能を作る（アイデアのみ）

アイデアのみの紹介です。興味がある方はぜひ、この機能の構築に挑戦してみてください。

例えば、「クイズ」というスキーマとエントリフォルダを作るとします。このスキーマには「question」というテキスト項目が1つ、「answer」という数値項目が1つ、option1、option2、option3、というテキスト項目が3つあるとします。question が「日本一高い山は？」というようなクイズで、option1～3 は回答選択肢、そして実際に 1～3 のどれが答えなのかを answer に記入するというものです。

もちろん、answer をそのまま画面に表示してしまっては面白くありません。answer は秘密にしておき、ユーザーに 1～3 のリンクを押させます。リンクにはそれぞれ、ans=1、ans=2、ans=3 のようなパラメータを URL に付与しておき、それをデザインパラメータ側で受け取って、ans が answer と同じならば「正解！」と表示するような事が可能になります。

６．４ 埋め込み命令（旧コマンド）

※非推奨の機能です。「６．３ 埋め込み命令タグ」をご利用下さい。

６．４．１ cmsview::entry 命令 - エントリ1件分の埋め込み

エントリ1件分のコンテンツを埋め込む際は、cmsview::entry 命令を使います。
この命令の最も基本的な使い方は次の通りです。

```
<?php cmsview::entry( "エントリフォルダ名", "デザイン名" ) ?>
```

```
例：<?php cmsview::entry( "mydiary", "default" ) ?>
```

指定したエントリフォルダのエントリを、指定したデザイン定義を使ってこの場所へ埋め込みます。
エントリフォルダ名は、エントリ定義で定義した名前を、デザイン名には、デザイン定義で決めたデザイン名を指定します（デザインファイル名ではなく、デザイン名です）。

実際に表示するエントリのエントリ ID は、URL から指定します。
例えば mypage.php という名前の埋め込み先画面を作成したとすると、

```
http://あなたのサイトの URL/mypage.php?eid=エントリ ID
```

```
例：http://www.hogehoge.com/mypage.php?eid=00001
```

のように指定します。

実際には、ユーザーが直接このような URL を入力することはありません。
「５．５．８ エントリー覧から個別のエントリへリンクを張る」のように、一覧表示用デザイン定義と組み合わせて使用します。

又、あまりないとは思いますが、場合によってはエントリ ID を外部から受け取らず、固定で表示したい場合があります。例えば定番商品をトップページに恒久的に表示したい場合などです。この場合は、以下のように直接エントリ ID を埋め込みます。

```
<?php cmsview::entry( "エントリフォルダ名", "デザイン名", "エントリ ID" ) ?>
```

```
例：<?php cmsview::entry( "mydiary", "default", "00001" ) ?>
```

こうすると、URL から eid パラメータを指定してもしなくても、埋め込んだエントリ ID のエントリが常に表示されるようになります。

エントリ ID は、コンテンツ管理画面のエントリー覧から確認できます。

６．４．２ cmsview::navi_entry 命令 - エントリー件分の埋め込み（ナビゲーション付き）

『５．５．８「次のエントリへ」「前のエントリ」へのリンクをつける』で解説している「記事のナビゲーション」付きデザインを有効にするには、cmsview::entry 命令ではなく、このcmsview::navi_entry 命令を使います。

```
<?php cmsview::navi_entry( "エントリフォルダ名", "デザイン名" ) ?>
```

```
例：<?php cmsview::navi_entry( "mydiary", "default" ) ?>
```

使い方は cmsview::entry 命令と全く同じです。ただ、記事のナビゲーション情報が出力される点だけが違います。

以下のような記事のナビゲーションを出力することができます。

<<前の記事へ | 次の記事へ>>

記事のナビゲーションデザインを定義する方法は、『５．５．９「次のエントリへ」「前のエントリ」へのリンクをつける』を参照してください。

６．４．３ cmsview::listtop 命令 - エントリー覧の埋め込み

エントリー覧のコンテンツを埋め込む際は、cmsview::listtop 命令を使います。

この命令の最も基本的な使い方は次の通りです。

```
<?php cmsview::listtop( "エントリフォルダ名", "デザイン名" ) ?>
```

```
例：<?php cmsview::listtop( "mydiary", "default" ) ?>
```

指定したエントリフォルダのエントリー覧を、指定したデザイン定義を使ってこの場所へ埋め込みます。規定では、エントリー覧の表示件数は 10 件で、それ以降は表示しません。

表示件数を指定する場合、次のようにします。

```
<?php cmsview::listtop( "エントリフォルダ名", "デザイン名", 表示件数 ) ?>
```

```
例：<?php cmsview::listtop( "mydiary", "default", 20 ) ?>
```

尚、1ページに表示する件数を〇件にして、ページ切替のナビゲーションを使ってページを切り替えていくには、次の cmsview::listpage 命令を使います。

６．４．４ cmsview::listpage 命令 - エントリー覧の埋め込み（ナビゲーション付き）

エントリー覧のコンテンツをページ切替のナビゲーション付きで埋め込む際は、cmsview::listpage 命令を使います。

この命令の最も基本的な使い方は次の通りです。

```
<?php cmsview::listpage( "エントリフォルダ名", "デザイン名" ) ?>
```

```
例：<?php cmsview::listpage( "mydiary", "default" ) ?>
```

指定したエントリフォルダのエントリー覧を、指定したデザイン定義を使ってこの場所へ埋め込みます。規定では、エントリー覧の表示件数は 10 件で、それ以降はページ切替ナビゲーションを使ってページを切り替えて表示します。ページ切替ナビゲーションのデザイン定義については、『５．５．９　一覧表示で「次のページへ」「前のページへ」のリンクをつける』を参照してください。

尚、10 件以外の表示件数を指定する場合、次のようにします。

```
<?php cmsview::listpage( "エントリフォルダ名", "デザイン名", 表示件数 ) ?>
```

```
例：<?php cmsview::listpage( "mydiary", "default", 20 ) ?>
```

６．４．５ 絞込みの指定

『３．５．１４ グループ（絞込み）指定』にて、グループを指定したスキーマを作成した場合、グループによる絞込み表示を行うことができます。

一覧表示の場合、以下のようにグループ項目名と絞込みたい検索値を指定します。

```
<?php cmsview::listpage( "エントリフォルダ名", "デザイン名", 表示件数, 絞込み条件 ) ?>
```

```
<?php cmsview::listtop( "エントリフォルダ名", "デザイン名", 表示件数, 絞込み条件 ) ?>
```

```
例：<?php cmsview::listpage( "mydiary", "default", 10, "5" ) ?>
```

上記の例だと、グループ項目の値が"5"のエントリだけを一覧表示します。

グループ項目を複数設定している場合は、項目名と絞込み条件を対で指定する必要がある為、以下のように記述します。

```
例：<?php cmsview::listpage( "mydiary", "default", 10,
        array( "review"=>"5", "shopkind"=>"2" ) ) ?>
```

上記の例では、"review"というグループ項目が"5"で、且つ、"shopkind"というグループ項目が"2"のエントリだけを抽出して一覧表示します。

このように、複数のグループ項目を設定している場合は、array( )で囲って"=>"の左側にグループ項目名、右側に絞り込む値を指定します。

```
array( "グループ項目名 1"=>"絞り込む値", "グループ項目名 2"=>"絞り込む値" )
```

上記例では2つまでですが、","で区切っていくつでも指定できます。

エントリ1件表示の場合でも、記事のナビゲーション（章「５．５．９ 」を参照）をつける場合、指定したグループ内で記事を切り替えていくことができます。

```
<?php cmsview::navi_entry( "エントリフォルダ名", "デザイン名", 絞込み条件 ) ?>
```

```
例：<?php cmsview::navi_entry( "mydiary", "default", "5" ) ?>
```

６．４．６ URL パラメータからの絞込みの指定

『６．３．５ 絞込みの指定』の方法だと、固定の絞込み条件しか指定できません。
例えば「種別」ごとに絞込み表示を行うとして、「種別」が１０種類あったとしたら、

list_kind01.php
list_kind02.php
list_kind03.php
　　：
list_kind10.php

のように、それぞれの種別毎に埋め込みページを作らなくてはならなくなります。

しかし、これらの埋め込みページは絞込み条件部分以外はほとんど同じであることが多く、できれば共通化して、絞込み条件だけを外部から与えるようにしたいと思う事でしょう。

少々難易度が高いかもしれませんが、次の方法で絞込み条件を外部から与える事ができるようになります。

```
例：<?php cmsview::listpage( "mydiary", "default", 10,
        array( "review"=>null, "shopkind"=>null ) ) ?>
```

つまり、これまで"=>"の後につけていた絞込み条件部分を"null"という文字に置き換えます。このとき、「"null"」ではなく「null」とだけ書いてください。

そして、これを埋め込んだページの URL に、次のように絞込み条件を与えて呼び出します。

```
http://xxx.xxx.xxx/list_kind.php?review=1&shopkind=5
```

これで、「review が 1 で且つ、shopkind が 5 のエントリー覧」を表示させることができます。

６．４．７ cmsview::rss 命令 - RSS の出力

エントリー覧の情報を、RSS で出力することができます。

但し、この機能を使うにはRSS出力専用のデザイン定義ファイルを作成する必要があり、少々作業が必要になります。

詳しくは、Full 版に同梱されている cmsd_doc_rss.zip の中をご覧ください。

６．４．８ cmsview::xml_entry 命令 - エントリ1件分の XML 出力

上級者向けの命令です。場合によってはコンテンツを、HTML/XHTML ではなく XML データとして出力したい場合があります。代表的なものは、Flash との連携です。

特に用途が無くても、XSLT 変換する前の XML データの状態を確認しておくと、デザイン定義をする上で大いに役立つでしょう。

エントリ 1 件分の XML を表示するには、次のようにページを作成します。

```
<?php
        require_once( "cmsdesigner/include/view.php.inc" );
        cmsview::xml_entry( "エントリフォルダ名", null );
?>
```

上記の内容**だけ**が記述された php ファイルを作成し、そこへアクセスすると、HTML に変換する前の XML データがそのまま取得できます。ブラウザで表示しても文字化けしたりおかしく表示されたりするので、「ソースを表示」でソースを確認してください。

"<?php" の行の前に空白や改行があってはいけません。また、"?>"の後にも空白や改行を入れてはいけません。

尚、cmsview::xml_entry 命令は、cmsview::entry 命令とまったく同じ使い方ができます。

よって、絞込み条件を指定したり、エントリ ID を直接指定したり、URL パラメータから受け取ったりという使い方が可能です。

上記の例ではデザイン名として null を指定していますが、HTML ではなく「XML に変換する XSLT」を作ってデザイン定義ファイルとして保存し、そのデザイン名を指定する事で、お好きな形式の XML データに加工した上で取得することも可能です（上級者向けです）。

６．４．９ cmsview::xml_navi_entry 命令 - エントリー件分の XML 出力（ナビゲーション付き）

『６．３．９ XML 形式での出力』と同様に、cmsview::navi_entry で出力される内容を HTML/XHTML ではなく XML データとして取得する命令です。

エントリ１件分のナビゲーション情報付き XML を表示するには、次のようにページを作成します。

```
<?php
        require_once( "cmsdesigner/include/view.php.inc" );
        cmsview::xml_navi_entry( "エントリフォルダ名", null );
?>
```

その他、詳細は『6．３．９ XML 形式での出力』と同様ですので、そちらを参照してください。

６．４．10 cmsview::xml_listtop 命令 - エントリー覧の XML の出力

『6．３．９ XML 形式での出力』と同様に、cmsview::listtop で出力される内容を HTML/XHTML

ではなくXMLデータとして取得する命令です。

この XML データを表示するには、次のようにページを作成します。

```
<?php
        require_once( "cmsdesigner/include/view.php.inc" );
        cmsview::xml_listtop( "エントリフォルダ名", null, 10 );
?>
```

その他、詳細は『6．3．9 XML 形式での出力』と同様ですので、そちらを参照してください。

6．4．11 cmsview::listpage 命令 - エントリー覧の XML の出力（ナビゲーション付き）

『6．3．9 XML 形式での出力』と同様に、cmsview::listpage で出力される内容を HTML/XHTML ではなくXMLデータとして取得する命令です。

この XML データを表示するには、次のようにページを作成します。

```
<?php
        require_once( "cmsdesigner/include/view.php.inc" );
        cmsview::xml_listtop( "エントリフォルダ名", null, 10 );
?>
```

その他、詳細は『6．3．9 XML 形式での出力』と同様ですので、そちらを参照してください。

7 その他

7．1 出力文字コードの変換

CMS Designer の規定の出力文字コードは UTF-8 ですが、これを任意の文字コード（Shift_JIS 等）に変換することが可能です。

出力文字コードの指定方法は三通りあります。

（1）サイト全体の出力文字コードを site.config.xml で指定する。
（2）ページ個別の出力文字コードを各ページのファイル（*.php）で指定する。
（3）XSLT の文字コード変換を用いる。

このうち（3）の方法については CMS Designer の出力文字エンコーディングについてよく理解されている方のみ使用して下さい。

7．1．1 出力文字コードの変換機能を使う際の注意事項（必ずお読み下さい）

CMS Designer では、最終的な出力文字コードがなんであっても、内部的には全て UTF-8 で処理しています。UTF-8 で処理を行った後、最後に指定された文字コードへ変換してブラウザへ送っています。

この為、例え出力文字コードとして Shift_JIS を指定したとしても、埋め込み先の PHP ファイルは常に UTF-8 で作成しなければなりません。また同様に、XSL ファイルの xsl:output タグで指定する output 属性も、常に UTF-8 を指定する必要があります。

逆に考えれば、最終的な出力文字コードに関係なく、常に UTF-8 でサイトを構築し、最後に出力文字コードを指定すれば良いということです。

[！] meta タグの charset について

埋め込み先の PHP ファイルの文字コードは必ず UTF-8 で作成しますが、もし出力文字コードを UTF-8 以外にしたい場合、meta タグの charset はどのように指定すれば良いのでしょうか？

答えは、**「それも UTF-8 を指定してください」**です。CMS Designer では、出力データの文字コード変換時に meta タグの charset を自動的に適切な文字列に置き換えます。

もしこの仕様が問題を起こす場合には、meta タグの charset 自動変換機能を OFF にすることもできます。詳しくはこの章をご覧下さい。

７．１．２ サイト全体の出力文字コードを指定する

site.config.xml でサイト全体の出力文字コードを指定することができます。省略値は UTF-8 です。

site.config.xml に次のように output タグを記述します。

site.config.xml の例（Shift_JIS にする場合）

```
<?xml version="1.0" encoding="UTF-8"?>
<site>
  <output encoding="shift_jis" />
  <entries>
      <entry name="news1" schema="news" caption="新着情報" />
      <entry name="news2" schema="news" caption="お知らせ" />
  </entries>
</site>
```

output タグの設定内容

属性値	省略	内容
encoding	可能	"shift_jis"や"utf-8"など、出力文字コードを指定する。 省略値は"UTF-8"。"pass"を指定すると、出力文字コード変換を行わない（通常は指定しなくてよい）。
meta-tag-replacement	可能	"True"又は"False"を指定する。省略値は"True"。 "True"の場合、CMSD は出力画面中の meta タグの「charset=UTF-8」を自動的に encoding 属性で指定したもの（例：charset=shift_jis）に置き換える。 通常、この属性は指定しなくてよい。自動置き換えが問題を起こした場合や、少しでも処理速度向上を狙いたい場合に"False"を指定する。
xslt-libno	可能	使用する XSLT 関数を選択する。通常は指定しなくてよい。表示に問題があった場合等に、0 以外を指定して調整する。 "0"：自動選択（規定値）。 "10"：Sablotron 関数 "20"：domxml 関数（旧） "21"：domxml 関数（新）
disable-xslt-encoding	可能	環境によっては XSLT ライブラリによる文字コード変換が無効になっており、出力が文字化けすることがある。xslt-libno を調整しても問題が解決しない場合に、この設定で強制的に XSLT ライブラリによる文字コード変換を無効にし、代わりに CMSD 側で文字コード変換を行うようにできる（パフォーマンスは若干落ちる）。 "True"：XSLT ライブラリによる文字コード変換を使用しない "False：XSLT ライブラリによる文字コード変換を使用する ※省略値は"False"。

７．１．３ ページ個別の出力文字コードを指定する

埋め込み先 PHP に cmsd:output タグを記述することで、ページ個別の出力文字コードを指定することができます。省略値は UTF-8 です。埋め込み先 PHP のなるべく上の方に次のように cmsd:output タグを記述します。

※注意：cmsd:output タグの指定内容に関わらず、PHP ファイル自体の文字コードは常に UTF-8 で記述して下さい。これは PHP の仕様による制限です。

埋め込み先 PHP の例（Shift_JIS にする場合）

```
<?php require( "cmsdesigner/include/view.php.inc" ); // encoding="UTF-8" ?>
<cmsd:output encoding="shift_jis" />
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>テストページです。</title>
</head>
<body>
テストページ<br>
<table border="1">
  <tr>
    <td>
      メニュー部分
    </td>
    <td>
      <cmsd:entry name="public_diary" design="default" />
    </td>
  </tr>
</table>
</body>
</html>
<?php cmsd_end_template(); ?>
```

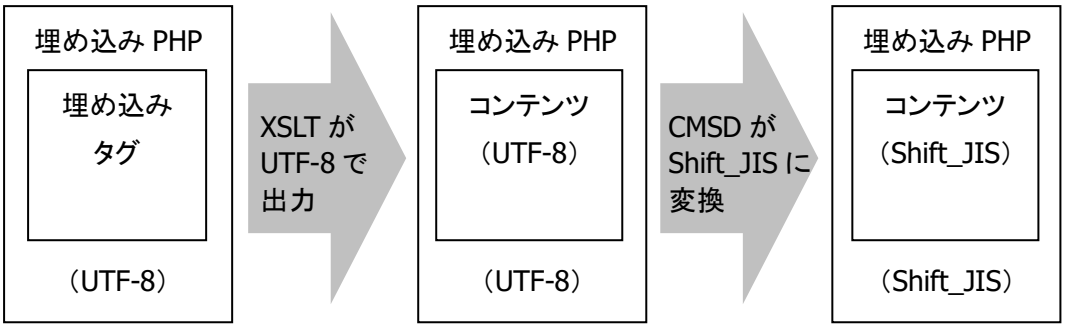
※ファイルの文字コードは UTF-8 で作成すること。

尚、cmsd:output タグの設定内容は、site.config.xml の output タグと同じです。
「７．１．２ サイト全体の出力文字コードを指定する」を参照して下さい。

余談ですが、上記の例で、meta タグの charset が「charset=UTF-8」になっているのに違和感を感じる方もいらっしゃると思います。ブラウザに表示される時点ではデータは shift_jis になっている訳ですから、ここは「charset=shift_jis」にするべきではないか？と思われるでしょう。実は、CMSD は文字コードの変換の際に、meta タグの charset 指定を検出し、自動的にこれを「charset=shift_jis」等の適切な指定に置換します。この為、問題は起こりません。「そんな面倒なことをしなくても、最初から charset=shift_jis と書いておけばいいのではないか？」というご意見もあるでしょう。実際それでも正しく動作するのですが、一部のウェブサイト編集ソフト（Dreamweaver 等）は、ファイルの文字コードと meta タグの charset 指定が合致していないと編集時に文字化けを起こしてしまうのです。ファイルの文字コードは UTF-8 なので、この場合は charset=UTF-8 にしておく必要があるのです。

7．1．4 XSLT の文字コード変換を利用する

CMS Designer の文字コード変換の流れは次のようになっています。



つまり、内部的に一旦全体を UTF-8 で統一し、最後に指定された文字コード（上記例では Shift_JIS）に変換しています。出力文字コードが UTF-8 の場合は、最後の変換を行いません。

なぜ直接 XSLT で Shift_JIS にしてしまわないかというと、埋め込み PHP 側に書かれた日本語が UTF-8 になっている為です。埋め込み PHP を UTF-8 にしなければならないのは、PHP の仕様による制限です。

よって、埋め込み PHP 側に日本語が存在しない場合に限り、XSLT 側で直接最終的な文字コードに変換してしまい、パフォーマンスを向上させることができます。

単に XSL 側で<xsl:output encoding="shift_jis" />のように指定しするだけだと、CMSD 側で再度変換処理を行い文字化けしてしまうので、cmsd:output タグにて encoding="pass"と記述して下さい。これで、最後の文字コード変換処理が行われず、目的の出力を得る事ができます。

尚、cmsd:entry タグや cmsd:entrylist タグで output="xml"や output="rss"などを指定した場合は、自動的に出力文字コードが"pass"扱いとなります。これは出力文字コードを任意に指定できなかった旧仕様との互換性の為です。

７．２ XSLT ライブラリが原因のトラブル対処方法

現在、PHP4 から利用できる XSLT 技術は複数存在し、大別して Sablotron の関数を使っているものと、libxml2 の関数を使ったものがあります。

これらは排他的ではありませんが、環境によってはどちらか一方しか入っていない時があります（もちろん、どちらも入っていない場合もあります）。

また、Sablotron は XSLT の文字コード変換処理に iconv を利用している為、システムに iconv が存在しない場合、XSLT の文字コード変換処理が無効になるという場合もあります。

これらの状況に対処する為、CMSD では（1）使用する XSLT 関数を切り替える機能、（2）XSLT の文字コード変換処理を CMSD で肩代わりする機能、を用意しています。

設定は site.config.xml の output タグか、埋め込み PHP 側の cmsd:output タグで行う事ができます。詳しい設定値は「７．１．２ サイト全体の出力文字コードを指定する」を参照して下さい。

７．３ ページキャッシュによるページ表示のパフォーマンス改善

もし CMS Designer のコンテンツを埋め込んだページの表示が重いと思われた場合には、この「ページキャッシュ機能」をお試し下さい。（Ver.1.1.8a 以降）

７．３．１ ページキャッシュ機能について

CMS Designer のコンテンツが埋め込まれた php ページは動的生成の為、常に最新の情報を表示できます。また、事前生成ではない為、エントリを更新する度にサイト全体を「再構築」したりといった手間がありません。しかし、動的生成の常として、アクセスがある度に該当ページのコンテンツを生成する為に、ファイル読み込み、加工、出力といった各種処理によるサーバへの負荷が発生し、サーバ環境やアクセス数によってはこれが無視できないケースもあります。

この負荷を軽減する為に、CMS Designer には「ページキャッシュ機能」があります。これは、一旦生成したページへの表示内容をサーバ側でキャッシュとして保存しておき、次回のアクセス時にはそのキャッシュをブラウザへ返すようにすることで、ファイル処理と加工にかかるサーバ負荷を大幅に軽減するというものです。このキャッシュはコンテンツの更新時に破棄される為、常に最新の情報をキャッシュできるようになっています。

多くのケースにおいて、ページキャッシュの利用は良い結果を上げるでしょう。
但し、このように便利なページキャッシュですが、公開日時指定、外部 XML 読み込み、ランダム表示機能などを利用している場合には注意が必要です。この件については７．３．４ を参照して下さい。

７．３．２ ページキャッシュの利用方法

ページキャッシュはデフォルトでは無効になっています。

(1) 準備

cmsdesigner/cache/pagecache フォルダを作成し、パーミッションを 707（書き込み可能）にします。

(2) 指定のページのページキャッシュを有効にする。

ページキャッシュを有効にしたいページの末尾を以下のように修正します。
現在、次のようになっている部分を、

```
<?php cmsd_end_template(); ?>
```

以下のように修正します。

```
<?php cmsd_pagecache("on"); cmsd_end_template(); ?>
```

(3) 確認方法

該当のページへブラウザから 1 度以上アクセスすると、ページキャッシュファイルがサーバ上で生成されます。そのページキャッシュファイルが生成されたことを確認して下さい。FTP ソフトで cmsdesigner/cache/pagecache フォルダ内を開き（最新の状態に更新）、そこに、「pg～」から始まる長い名前のファイル（ファイル名は URL から生成しています）が存在していれば、それがキャッシュデータです。

もしキャッシュデータが生成されていなければ、(1)や(2)の手順を再度ご確認下さい。

７．３．３ 全てのページのページキャッシュをまとめて有効にする

多くのケースでは、サイト全体の CMS Designer ページに対してまとめてページキャッシュを有効にしたいことが多いと思います。

その場合には、7.3.2 のように個別ページに手を加えるのではなく、以下のように設定ファイルを一箇所修正するだけで、サイト全体に対してページキャッシュを有効にすることができます。

手順は、cmsdesigner/include/cache.php.inc をエディタで開き、19 行目付近の

```
define( 'AD_PAGECACHE_ENABLED', false ); // false： キャッシュしない,　true： キャッシュする
```

の部分を、

```
define( 'AD_PAGECACHE_ENABLED', true ); // false： キャッシュしない,　true： キャッシュする
```

にします。

これで、全てのページのページキャッシュが有効になります。但し、この方法を使った場合、CMS Designer 本体のバージョンアップ時に cache.php.inc を上書きすると、また false に戻ってしまいますので、バージョンアップの際に毎回この設定を行うようにして下さい。

尚、この状態で、指定した一部のページのみページキャッシュを無効にするには、各ページの末尾を以下のように「off」に指定します。

```
<?php cmsd_pagecache("off"); cmsd_end_template(); ?>
```

７．３．４ ページキャッシュの日次更新

(1) ページキャッシュ機能の副作用

ページキャッシュを用いることでサーバの負荷を下げ、レスポンスを高めることが可能ですが、その特性上、CMS Designer の一部の機能と相性が良くありません。

- **公開日時指定機能が機能しない。**
  →最初にページにアクセスがあった時の状態がキャッシュとして保存される為、公開日が来ても保存されたキャッシュ（公開されていない状態）が表示されます。※単純な「公開／非公開」はページキャッシュ利用時も有効です。
- **ランダム表示機能を使うとリロードしても毎回同じ結果になる。**
  →同様に最初のアクセス時の状態がキャッシュとして保存される為、その時点でのランダム結果がその後もずっと表示されます。
- **外部 XML 表示機能を使うと外部 XML の更新に追従しない。**
  →同様に最初のアクセス時の状態が保存される為、その時点での外部 XML 状態がその後もずっと表示されます。

これらの問題を**完全に**避けるには、該当ページについてはページキャッシュを off にして頂く必要がありますが、次に説明するページキャッシュの日次更新指定を行うことで、ある程度この問題を回避しつつ、ページキャッシュの負荷低減効果を最大限得ることができます。

(2) ページキャッシュの日次更新指定方法

(1)で示した副作用を可能な限り回避する為に、「毎日定時を超えたらページキャッシュを破棄し、最新の表示内容と置き換える」という機能を用意しています。

これを利用するには、該当ページの最後尾に次のように指定します。

現在、次のようになっている部分を、

```
<?php cmsd_end_template(); ?>
```

以下のように追記します。

```
<?php cmsd_pagecache("5:00"); cmsd_end_template(); ?>
```

これで、毎日 5:00 を過ぎるとページキャッシュを破棄し、最新の表示内容で置き換えるようになります。 公開日時指定なども、「〇月〇日の 0:00 に公開」のように指定しておけば、5:00 にはちゃんと公開済みになります。また、ランダム表示や外部 XML 表示機能は、一日単位で変更されるようになります。

このような制限はありますが、それで問題ない場合には、このページキャッシュの日次更新機能をご利用下さい。

(3) 全ページへのページキャッシュの日次更新指定方法

(2)の日次更新指定は個別ページに対してですが、場合によっては全ページまとめて日次更新指定を行いたい場合があるかもしれません。

その場合には、cmsdesigner/include/cache.php.inc をエディタで開き、20 行目の

```
define( 'AD_PAGECACHE_EXPIRES', '' ); // '' : キャッシュ有効期限なし,  '時:分' : 毎日その時刻を超えたらキャッシュを再作成する。例）'5:00'
```

の部分を、

```
define( 'AD_PAGECACHE_EXPIRES', '5:00' ); // '' : キャッシュ有効期限なし,  '時:分' : 毎日その時刻を超えたらキャッシュを再作成する。例）'5:00'
```

のようにします。

これで、全てのページに対して日次更新指定をしたことになります。

これだけではページキャッシュを有効にしたことにはなりませんので、併せて７．３．２ や７．３．３ の指定を行ってください。

７．３．５ 注意事項

ページデザインを変更した場合など、キャッシュ情報を一旦破棄したい場合には、FTP ソフトなどで cmsdesigner/cache/pagecache フォルダ内のファイルを全て削除して下さい。

７．４ HTML エディタの利用（WYSIWYG エディタ）

CMS Designer（Ver.1.1.8a 以降）上で FCKeditor を利用することができます。

FCKeditor はブラウザ上でワードプロセッサのようなリッチなテキスト編集を可能にする Javascript アプリケーションで、Frederico Caldeira Knabben 氏によってhttp://www.fckeditor.net/ で開発・配布されているオープンソースソフトウェア（OSS）です。オープンソースソフトウェアとは、誰でも無料で自由に使っていいソフトウェア**ではありません**。それぞれの OSS に、それぞれのライセンスが存在し、その制限の範囲内でのみ、利用できます。

CMS Designer は FCKeditor のライセンスの頒布に関する制限の影響を受けないようにする為、CMS Designer 本体に FCKeditor を同梱しておりません。

FCKeditor をご利用頂く際には、CMS Designer を利用するウェブサイト制作者が、ご自分で FCKeditor をダウンロードし、CMS Designer の指定箇所に設置して頂く必要があります。但し、極力簡単に設置できるようになっています。

FCKeditor のご利用に関しては、必ず FCKeditor のライセンス条項をご確認頂き、その制限の範囲内でご利用下さい。

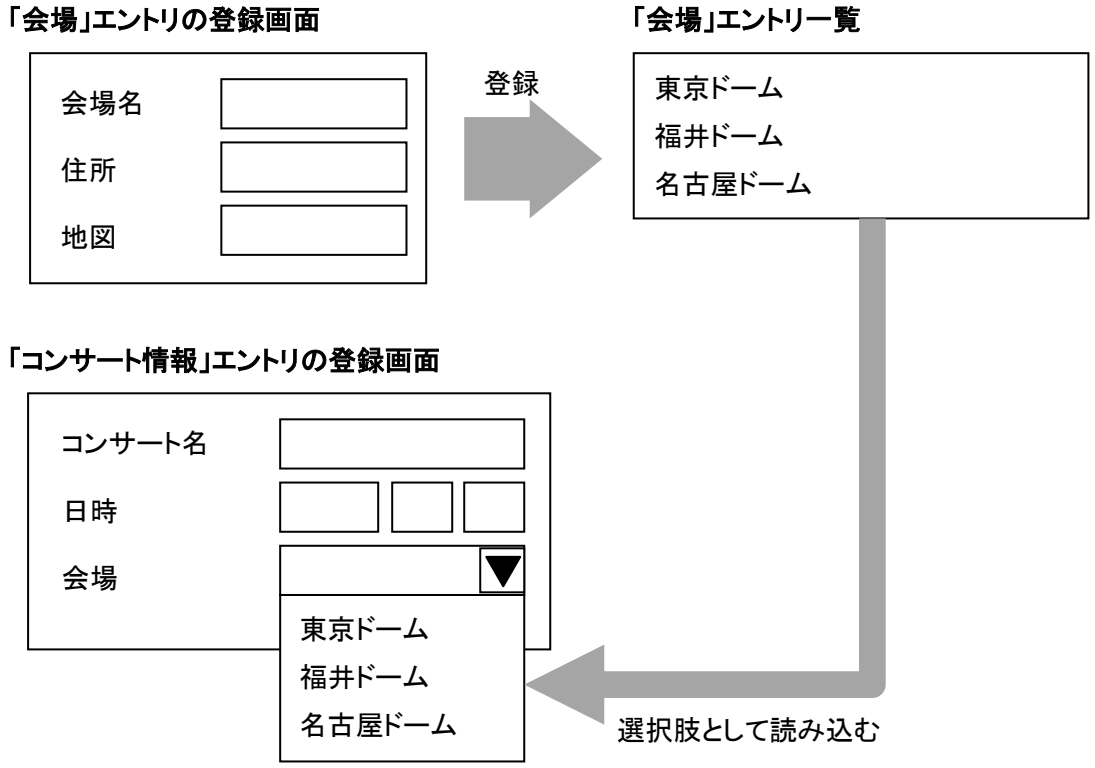
FCKeditor を CMS Designer から利用できるようにする方法は、/cmsdesigner/fckeditor/ フォルダの中にある、howtouse_fckeditor_for_cmsd.txt をご覧下さい。

【追記：ver.1.1.12a】

FCKeditor が開発終了となった為、Ver.1.1.12a 以降にて、FCKeditor の後継ソフトウェアである「CKeditor（http://ckeditor.com/）」に対応しました。詳細は /cmsdesigner/ckeditor/フォルダの中にある。Howtouse_ckeditor_for_cmsd.txt をご覧ください。

７．５ メニュー項目の選択肢として他のエントリー覧（外部データソース）を読み込む

Ver.1.1.18a 以降の機能として、あるメニュー項目の選択肢として、他のエントリー覧のタイトル等を読み込んできて使用する事ができるようになりました。



この例で説明すると、例えば「会場」エントリー覧に新しく「日本武道館」を追加すれば、自動的に「コンサート情報」の「会場」メニュー項目にも「日本武道館」が追加されます。運用中にサイト運営者がメニュー項目の内容を自由に追加・削除できるようになる上、後述するように「コンサート情報」を表示するページの中に、関連情報として「会場」エントリの内容を入れ子状にして表示することが出来るようになります。

この時、コンサート情報の「会場」メニュー項目から見た「会場エントリー覧」を「**外部データソース**」と呼びます。メニュー項目の選択肢データをスキーマから読み込むのではなく、外部から読み込んでいる為、このような呼び方となります。

このような、**メニュー項目への外部データソースの読み込み**を行いたい場合、次ページ以降の手順を実行してください。この「会場」と「コンサート情報」の関連付けを例に説明します。

７．５．１ 設定手順

(1) 設定手順 1：読み込まれる側のエントリのスキーマの設定

読み込まれる側のエントリ（この例では「会場」）のスキーマを設定します。この例では「会場スキーマ」とします。会場スキーマは例えば次のようになります。

重要な点は１点のみで、メニュー項目の選択肢ラベルとして使いたい項目（ここでは「会場名」）に、必ず group="True"指定をつけるようにして下さい（CMS Designer の仕様上必要となります）。

```
<?xml version="1.0" encoding="UTF-8"?>
<schema name="site" caption="会場" title="sitename">
  <data name="sitename" type="text" caption="会場名" group="True" />
  <data name="description" type="text" caption="説明" />
  <data name="photo" type="img" caption="写真" maxfilesize="10KB" />
  <data name="location" type="geolocation" caption="地図" />
</schema>
```

【注意】group 属性はサイトの運用中に変更しても保存されているエントリデータには反映されない為、既存のエントリを外部データソースとして利用したい場合には注意が必要です。もし既存のエントリを外部データソースとして利用したい場合には、該当の項目の group 属性を"True"に指定した後、このエントリフォルダの全てのエントリを管理画面上から保存しなおして下さい。

(2) 設定手順 2：読み込む側のエントリのスキーマの設定

読み込む側のエントリ（この例では「コンサート」）のスキーマを設定します。この例では「コンサートスキーマ」とします。コンサートスキーマは例えば次のようになります。

特に注意する点はありませんが、「会場」メニュー項目には menuitem 要素を指定する必要はありませんので、以下のように空要素にしても大丈夫です。menuitem 要素を指定しても、外部データソースを読み込む際に、ここで指定した内容は無視されます。

```
<?xml version="1.0" encoding="UTF-8"?>
<schema name="consert" caption="コンサート情報" title="title">
  <data name="title" type="text" caption="コンサート名" />
  <data name="description" type="text" caption="説明" />
  <data name="photo" type="img" caption="写真" maxfilesize="10KB" />
  <data name="site" type="menu" caption="会場" />
</schema>
```

(3) 設定手順 3：エントリ定義（site.config.xml）に datasource 要素を追加する

エントリ定義で、読み込む側のエントリと読み込まれる側のエントリを関連付けます。

site.config.xml をエディタで開き、読み込む側のエントリ（ここでは「コンサート」エントリ）のエントリ定義に以下のように datasource 要素を追加します。

```
<?xml version="1.0" encoding="UTF-8"?>
<site>
  <entry name="site1" schema="site" caption="会場" />
  <entry name="concert1" schema="concert" caption="コンサート情報" >
    <datasource for="site" type="entrylist" from="site1" label="sitename" />
  </entry>
</site>
```

datasource 要素に指定するのは、「どのメニュー項目に、どのエントリフォルダのどの項目を選択肢ラベルとして読み込むか」です。詳細は以下の通りです。

datasouce 要素

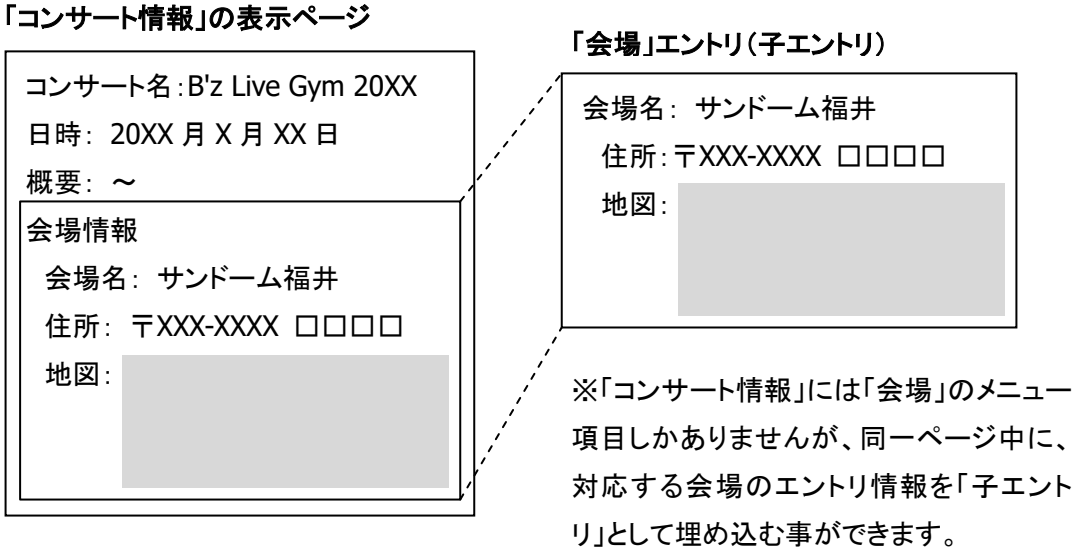
属性値	省略	内容
for	不可	読み込み先となるメニュー項目の項目名。
type	不可	"entrylist"と指定します。
from	不可	外部データソースとなるエントリフォルダ名を指定します。
label	不可	外部データソース側のエントリのスキーマから、どの項目を選択肢のラベルとして用いたいかを指定します。(1)設定手順 1 も併せて参照してください。
embedding	可	"False"を指定すると、子エントリ情報の埋め込みを行いません。詳細は「７．５．２ メニュー項目に外部データソースを使用した場合のデザイン定義(3)」を参照してください。省略値は"True"で、子エントリ情報の埋め込みを行います。

以上の手順で、メニュー項目の選択肢として外部のエントリフォルダの内容が読み込まれるようになります。

７．５．２ メニュー項目に外部データソースを使用した場合のデザイン定義

Ver.1.1.18a 以降、デザイン定義内でメニュー項目の選択肢を、ID（例：「1」「2」）だけでなくラベル（例：「中華」「和食」）で取得して利用することができるようになりました。また、選択肢ラベルの一覧を取得して表示することも可能になっています。これらの機能は、メニュー項目に外部データソースを使用した場合も同様に使用することができます。詳しくは「５．５．７ メニュー項目を表示する。」や「５．５．17 メニュー項目の選択肢リストを表示する。」をご覧ください。

この章では、メニュー項目に外部データソースを使用した場合に、外部データソース側のエントリ情報を丸ごと、読み込み先のエントリ情報に「**子エントリ**」として埋め込んで表示する方法を紹介します。これは例えば、「コンサート情報」と「会場」というエントリフォルダがそれぞれあって、コンサート情報のメニュー項目として「会場」エントリフォルダが外部データソースとして読み込まれている場合に、「コンサート情報」のページ中に、対応する「会場」の詳細情報も一緒に表示する、というようなケースです。同様に、「講座情報」と「担当講師」、「商品情報」と「メーカー情報」など、活用範囲は様々です。



具体的なデザイン定義を次のページに示します。

(1) 子エントリのデザイン定義

外部データソースとして他のエントリを読み込んでいるメニュー項目があった場合、その項目名に「**/entry**」を付ける事で、入れ子のように子エントリの情報を挿入することができます。

デザイン定義側では、以下のように xsl:for-each を使って挿入するのが便利です。

```
<xsl:for-each select="メニュー項目名/entry">
  ～
</xsl:for-each>
```

具体的には次のように記述します。

```
<xsl:for-each select="entry">
:
(コンサート情報のデザイン定義)
:
<h3>会場情報</h3>
<xsl:for-each select="site/entry">
<ul>
  <li>会場エントリ ID：<xsl:value-of select="@id" disable-output-escaping="yes" /></li>
  <li>会場名：<xsl:value-of select="sitename" disable-output-escaping="yes" /></li>
  <li>説明：<xsl:value-of select="description" disable-output-escaping="yes" /></li>
  <li><img src="{photo}" /></li>
</ul>
</xsl:for-each>
<xsl:for-each select="site/noentry">
該当の会場情報がありません。
</xsl:for-each>
:
```

<xsl:for-each select="メニュー項目名/entry"></xsl:for-each>の中身は、通常のエントリのデザイン定義のように、その子エントリのデザインを記述することができます。

"**メニュー項目名/noentry**"は、埋め込み対象のエントリ ID のエントリが存在しなかった場合に有効になります。「会場」エントリリストから該当のエントリを削除してしまった時など、通常運用で起こりえる事ですので、noentry を使った対処を行っておいた方が良いでしょう。

例えば、xsl:for-each を使わず、次のように直接パスを記述して xsl:value-of でデータを取得することも可能ですが、子エントリが存在しないケースを考えると避けた方が無難でしょう。

```
<xsl:for-each select="entry">
  :
  (コンサート情報のデザイン定義)
  :
  会場名：<xsl:value-of select="site/entry/sitename" disable-output-escaping="yes" />,
  説明：<xsl:value-of select="site/entry/description" disable-output-escaping="yes" />
</xsl:for-each>
```

(2) 子エントリの埋め込みが不要な場合のパフォーマンスアップ

外部データソースを「子エントリ」としてデザインに埋め込む必要がない場合、これをオフにしてサーバへの負荷を低減することができます。例えば「コンサート情報」ページでは会場名だけ分かれば良い、という場合は、子エントリの「会場」エントリへアクセスせずとも「site/@label」だけで十分ですので、子エントリの埋め込みをオフにした方が良いでしょう。方法は次の通りです。

方法

site.config.xm のエントリ定義内の datasource 要素に **embedding="False"** を指定します。

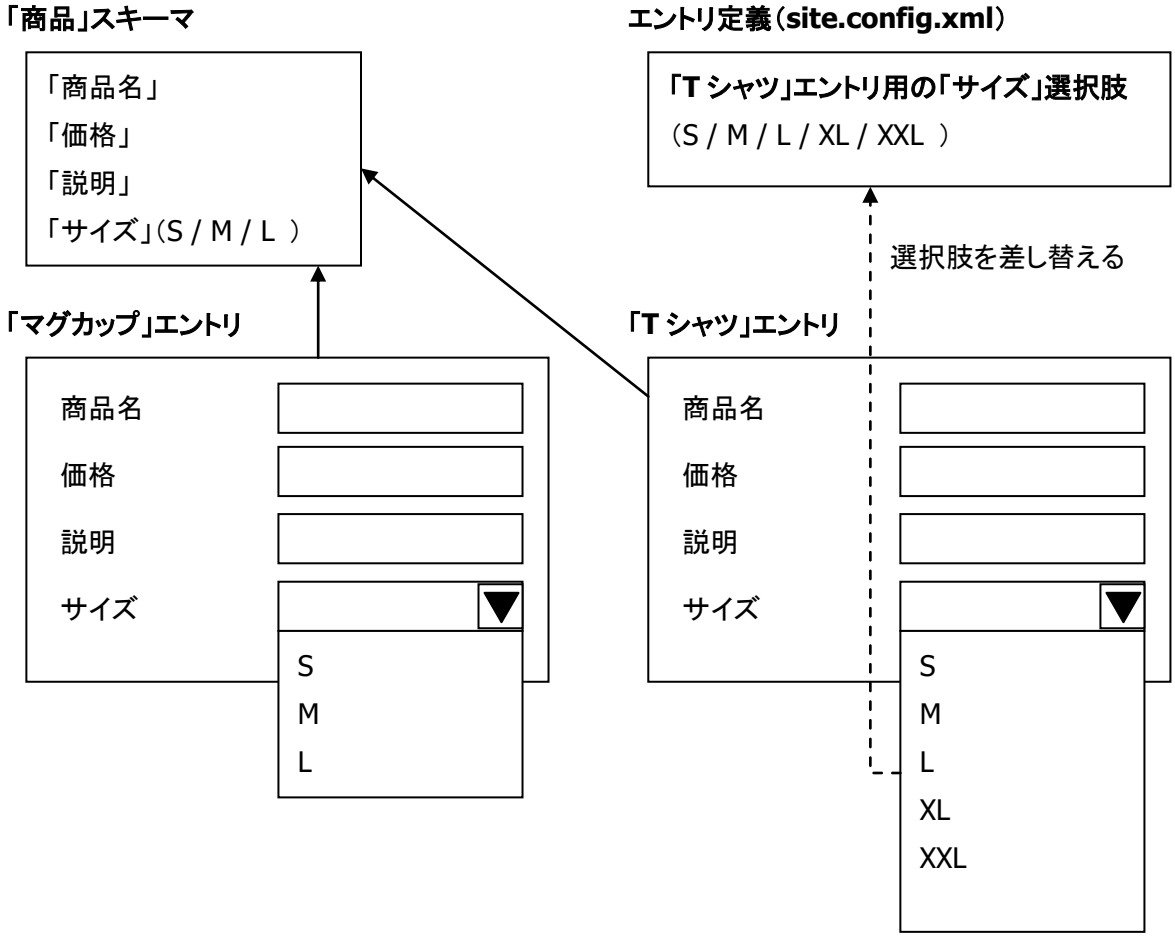
例

```
<?xml version="1.0" encoding="UTF-8"?>
<site>
  <entry name="concert1" schema="concert" caption="コンサート情報" >
    <datasource for="site" type="entrylist" from="site1" label="sitename"
      embedding="False" />
  </entry>
</site>
```

７．５．３ 外部データソースとしてエントリ定義（site.config.xml）を利用する

メニュー項目の外部データソースとしてエントリフォルダを指定するのではなく、エントリ定義に直接メニュー項目選択肢を指定することができます。利用シーンとしては、複数のエントリフォルダが同一のスキーマを参照している際に、エントリフォルダ毎に異なる選択肢を指定したい場合などに利用します。

例えば「商品」スキーマに「商品名」「説明」「価格」「サイズ」という項目があり、このスキーマを元に「T シャツ」というエントリフォルダと「マグカップ」というエントリフォルダを作って使用しているとします。この時、T シャツとマグカップで、「サイズ」の選択肢を、当初「S/M/L」で統一しており、使い回しができていたのですが、ある時、T シャツの方が「XL/XXL」を追加しなくてはならなくなった、というようなケースで、この機能を用いて「選択肢だけ別のものに差し替える」事ができます。



具体的な指定方法は次ページに示します。

(1) エントリ定義(site.config.xml)に datasource 要素を追加する

site.config.xml をエディタで開き、以下のように datasource 要素を追加します。重要な点として、datasource 要素の type 属性を"**inline**"に指定します。

```
<?xml version="1.0" encoding="UTF-8"?>
<site>
  <entry name="tshirt" schema="shopitem" caption="T シャツ" >
    <datasource for="size" type="inline" >
      <menuitem id="S">S サイズ</menuitem>
      <menuitem id="M">M サイズ</menuitem>
      <menuitem id="L">L サイズ</menuitem>
      <menuitem id="XL">XL サイズ</menuitem>
      <menuitem id="XXL">XXL サイズ</menuitem>
    </datasource>
  </entry>
</site>
```

datasource 要素に指定するのは、「どのメニュー項目に、どのエントリフォルダのどの項目を選択肢ラベルとして読み込むか」です。詳細は以下の通りです。

datasouce 要素

属性値	省略	内容
for	不可	読み込み先となるメニュー項目の項目名。
type	不可	"liline"と指定します。

menuitem 子要素

属性値	省略	内容
id	不可	選択肢の ID となる英数字を指定します。同じメニュー項目内に同じ選択肢 ID が複数存在しないようにしてください。
(本文)	不可	選択肢のラベルを指定します。

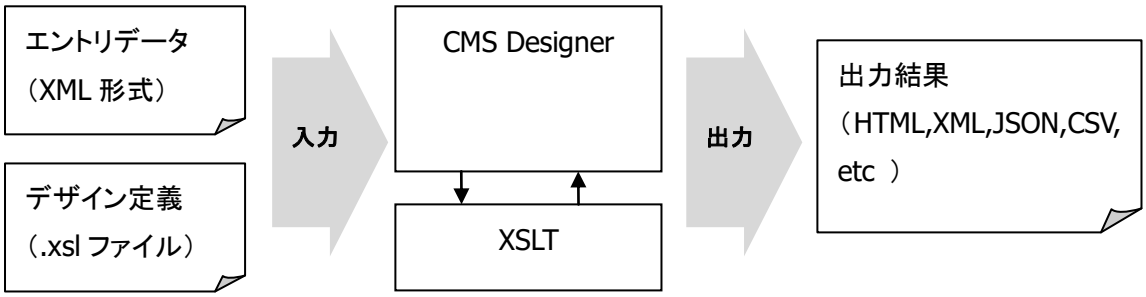
8 資料

8．1 デザイン定義（xsl ファイル）へ渡されるエントリデータの XML

CMS Designer では、テンプレート・エンジンとして XSLT という技術を利用しています。これは XSL というコンピュータ言語を用いて XML データを他のデータ表現へと変換する技術で、シンプルな表現で柔軟かつ強力な処理を行う事が可能です。

このリファレンスマニュアル中では、xsl:for-each や xsl:value-of、xsl:if などの基本的な XSL タグのみご紹介していますが、XSL について学べば、もっとさまざまな処理が可能となります。

この章では、CMS Designer がデザイン定義（xsl ファイル）へエントリデータを渡す際の内部 XML 表現を紹介します。ご自身で XSL の他の命令を使用する際に参考にしてみて下さい。



CMS Designer/XSLT に入力される「エントリデータ」は、エントリデータ保存フォルダに保存されている xsl ファイルをベースに、加工や実行時データなどの追加編集を行った内部データとなります。次のページに、その内部データの XML 形式のサンプルを紹介します。

尚、見やすくする為に改行等を入れていますが、実際のデータは改行の有無やデータの登場順序等が異なっています。

(1) エントリ 1 件分の XML データ（サンプル）

```
<?xml version="1.0" encoding="UTF-8"?>
<entry date="2006-06-20 18:02" year="2006" month="06" day="20"
        hour="18" minute="02" second="00" weekday="tue" time="1150794120"
        visible="True" enddate="999999999999" startdate="000000000000"
        id="01006" author="user01" schemaname="event"
        href="?eid=01006" daysago="3327">
```

エントリの基本情報。年月日、日時やエントリ ID、公開情報等。@time は年月日時の秒表現。@daysago は今日から見た日数差。

```
  <today year="2015" month="07" day="30"
         hour="11" minute="04" second="11"
         weekday="thu" time="1438221851"
         tzoffset="32400">2015-07-30 11:04</today>
```

現在日時情報。要素値は YYYY-MM-DD hh:mm 形式の文字列。@tzoffset はタイムゾーンのオフセット値（秒）。

```
  <navi position="0" allcount="1007">
    <next id="01014" href="?eid=01014" />
  </navi>
```

ナビゲーション情報。ON にした時のみ追加される。@position は現在ページ、@allcount は該当のエントリ総件数。next 要素は次ページの情報。

```
  <group key="level" value="5"/>
  <group key="site" value="00003"/>
```

グループ絞り込み情報。絞り込み条件が指定された時のみ追加される。@key は識別子であり、値は対応する絞り込み項目の項目名。

```
  <menuinfo name="site">
    <menuitem id="00001">東京ドーム</menuitem>
    <menuitem id="00003">サンドーム</menuitem>
    <menuitem id="00002">日本武道館</menuitem>
  </menuinfo>
  <menuinfo name="phototype">
    <menuitem id="0">指定なし</menuitem>
    <menuitem id="1">正面</menuitem>
    <menuitem id="2">全体</menuitem>
  </menuinfo>
```

menuinfo。メニュー項目の選択肢リストがスキーマからコピーされる。外部データソースを利用している場合は、該当の外部データソースからコピーされる。複数のメニュー項目が同一エントリ内にある場合は、各メニュー項目に対応する menuinfo がある為、識別子として@name がある。@name の中身は、対応するメニュー項目の項目名。

```
  <eventname output="text1">イベント 16</shopname>
  <url output="text1">http://test.test.test</url>
  <description output="html2">説明です。</description>
  <level>5</level>
```

エントリの各項目データ。要素名は項目名と同一となっている。

```
  <site label="サンドーム">
    00003
    <entry date="2005-08-06 16:29" year="2005" month="08" day="06" hour="16"
           minute="29" second="00" weekday="sat" time="1123313340"
           visible="True" enddate="999999999999" startdate="000000000000"
           author="user01" id="00003" schemaname="site">
      <sitetype label="ドーム">03</group>
      <sitename output="text1">サンドーム</singername>
      <description output="text1">サンドームです。</description>
      <photo filetype="image" alt="" org="" filesize="0" state="saved" fileid="00000001" />
      <menuinfo name="sitetype">
        <menuitem id="01">ビル</menuitem>
        <menuitem id="02">屋外</menuitem>
        <menuitem id="03">ドーム</menuitem>
      </menuinfo>
    </entry>
  </site>
```

外部データソースを読み込んでいるメニュー項目。このサンプルでは、「会場(site)」というメニュー項目に、外部の「会場」エントリフォルダを読み込んでいる。
@label に選択値（"サンドーム"）、テキスト情報として"00003"が入っており、その中に該当の「会場」エントリが丸ごと、entry 要素として格納されている。

```
  <photolist>
    <listitem id="1">
      <photocaption output="text1">写真 1</photocaption>
      <photo filetype="image" alt="" org="myphoto12345.JPG" filesize="1539"
             state="saved" fileid="00000001" width="43" height="50"
             src="tabearuki1.01006.00000001.JPG">
             cmsdesigner/viewimg.php?entryname=event1&amp;entryid=01006&amp;fileid=00000001&amp;/myphoto12345.JPG
      </photo>
    </listitem>
    <listitem id="2">
      <photocaption output="text1">写真 2</photocaption>
      <photo filetype="image" alt="" org="myphoto67890.JPG" filesize="1425"
             state="saved" fileid="00000002" width="43" height="50"
             src="tabearuki1.01006.00000002.JPG">
             cmsdesigner/viewimg.php?entryname=event1&amp;entryid=01006&amp;fileid=00000002&amp;/myphoto67890.JPG
      </photo>
    </listitem>
  </photolist>
```

リスト項目。リスト内の各要素が listitem に格納されている。

```
  <datetime1 year="2006" month="06" day="00"
             hour="00" minute="00" second="00" weekday="wed"
             time="1149001200">2006-06</datetime1>
  <file2 filetype="file" org="" filesize="0" fileid="00000003" state="saved" />
</entry>
```

「イベント情報」スキーマ（サンプル）

```
<?xml version="1.0" encoding="UTF-8"?>
<schema name="event" caption="イベント情報" sortkey="level"
            sortorder="asc" title="sitename,sitetype,description">
  <data name="eventname" type="text" caption="イベント名" maxlength="10" minlength="4"
                                                                    size="20" group="True"/>
  <data name="url" type="text" caption="URL" />
  <data name="description" type="textarea" caption="説明" output="html2"
                                                          maxlength="1000" autolink="True" />
  <data name="site" type="menu" caption="会場" group="True" />
  <data name="level" type="int" caption="レベル" min="1" max="5" size="20" group="True" />
  <data name="photolist" type="list" caption="写真リスト" >
    <listitem caption="写真" title="photo,photocaption,phototype">
      <data name="phototype" type="menu" caption="写真種別" group="True" >
        <menuitem id="0">指定なし</menuitem>
        <menuitem id="1">正面</menuitem>
        <menuitem id="2">全体</menuitem>
      </data>
      <data name="photocaption" type="text" caption="写真の説明文" />
      <data name="photo" type="img" alt="True" caption="写真" _maxfilesize="300KB"
                                                              width="1024" height="768" />
    </listitem>
  </data>
  <data name="datetime1" type="month" group="True" caption="開催日時" />
  <data name="file2" type="file" maxfilesize="300KB" caption="その他の添付ファイル" />
</schema>
```

「会場」スキーマ（サンプル）

```
<?xml version="1.0" encoding="UTF-8"?>
<schema name="site" caption="会場" title="sitename" >
  <data name="sitetype" type="menu" caption="会場種別">
    <menuitem id="01">ビル</menuitem>
    <menuitem id="02">屋外</menuitem>
    <menuitem id="03">ドーム</menuitem>
  </data>
  <data name="sitename" type="text" caption="建物名" group="True" />
  <data name="description" type="text" caption="説明" />
  <data name="photo" type="img" caption="写真" maxfilesize="10KB" />
</schema>
```

イベント情報と会場情報のエントリ定義（サンプル）

```
<entry name="event1" schema="event" caption="イベント情報">
  <datasource for="site" type="entrylist" from="site1" label="sitename" />
</entry>
<entry name="site1" schema="site" caption="会場" />
```

(2) エントリー覧の XML データ（サンプル）

エントリー覧の XML は、エントリー件分の XML データとほぼ同一です。ルート要素が entry ではなく entrylist となり、その下に複数の entry 要素が存在します。これに合わせ、menuinfo や navi 要素なども entrylist 要素の直下に移動します。

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

  <navi position="1" allcount="97" maxpage="5" maxpagerows="20">
    <page id="1" href="?pageno=1" />
    <page id="2" href="?pageno=2" />
    <page id="3" href="?pageno=3" />
    <page id="4" href="?pageno=4" />
    <page id="5" href="?pageno=5" />
    <next id="2" href="?pageno=2" />
  </navi>

  <today year="2015" month="07" day="30"
         hour="11" minute="04" second="11"
         weekday="thu" time="1438221851"
         tzoffset="32400">2015-07-30 11:04</today>

  <group key="level" value="5"/>
  <group key="site" value="00003"/>

  <menuinfo name="site">
    <menuitem id="00001">東京ドーム</menuitem>
    <menuitem id="00003">サンドーム</menuitem>
    <menuitem id="00002">日本武道館</menuitem>
  </menuinfo>
  <menuinfo name="phototype">
    <menuitem id="0">指定なし</menuitem>
    <menuitem id="1">正面</menuitem>
    <menuitem id="2">全体</menuitem>
  </menuinfo>

  <entry date="2006-06-20 18:02" year="2006" month="06" day="20"
         hour="18" minute="02" second="00" weekday="tue" time="1150794120"
         visible="True" enddate="999999999999" startdate="000000000000"
         author="user01" id="01006" schemaname="shop" href="?eid=01006"
         daysago="3327" current="True">
   （エントリ情報は省略）
  </entry>

  <entry date="2009-12-15 15:08" year="2009" month="12" day="15"
         hour="15" minute="08" second="00" weekday="tue" time="1260857280"
         visible="True" enddate="999999999999" startdate="000000000000"
         author="user01" id="01016" schemaname="shop" href="?eid=01016"
         daysago="2053">
   （エントリ情報は省略）
  </entry>

    :

</entrylist>
```

entrylist。ルート要素。属性値などは特にない。

ナビゲーション情報。エントリー覧の場合、@maxpage 属性や@maxpagerows 属性、そして各 page 要素がある。各 page 要素は全てのページの分存在し、ページ番号としての@id とこのページを表す URL パラメータが格納された@href がある。next 要素は「次のページ」を表す情報が格納されている。

today, group, menuinfo 等の各付随情報は、エントリ 1 件分と同様。但し、親要素は entry ではなく entrylist になる。

各エントリ情報を格納する enrty 要素は、エントリ基本情報を含めて基本的にエントリ 1 件分の情報と同一のものが含まれる。

「一覧-詳細連動機能」を使用している場合には、「現在のエントリ」を示す@current 属性が "True"として設定されるエントリが 1 件ある。