

目次

1. この資料について.....	3
2. CMS DESIGNER の基本的な仕組み.....	4
2. 1 概要	4
2. 2 主なフォルダ構成.....	6
2. 3 作業の流れ	7
2. 4 XML と文字コードについて	8
3. スキーマ定義	9
3. 1 スキーマとは何か.....	9
3. 2 スキーマファイルの作成	9
3. 3 スキーマファイルの編集	10
3. 4 スキーマファイルの設置	10
3. 5 スキーマ リファレンス.....	11
3. 5. 1 <i>schema</i> タグの設定	11
3. 5. 2 <i>data</i> タグの種類	11
3. 5. 3 <i>data</i> タグ - <i>text</i>	12
3. 5. 4 <i>data</i> タグ - <i>textarea</i>	13
3. 5. 5 <i>data</i> タグ - <i>int</i>	13
3. 5. 6 <i>data</i> タグ - <i>menu</i>	14
3. 5. 7 <i>data</i> タグ - <i>img</i>	15
3. 5. 8 <i>data</i> タグ - <i>file</i>	15
3. 5. 9 <i>data</i> タグ - <i>list</i>	16
3. 5. 10 ソート(並べ替え)指定.....	17
3. 5. 11 グループ(絞込み)指定	18
4. エントリ定義	19
4. 1 エントリ定義とは何か	19
4. 2 エントリ定義手順.....	19
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	19
4. 4 SITE.CONFIG.XML へのエントリ定義の追加.....	19

5. デザイン定義	21
5. 1 デザイン定義とは何か。	21
5. 2 XSLT (XML STYLESHEET LANGUAGE TRANSFORMATIONS) について	22
5. 3 エントリ1件用のデザイン定義	23
5. 4 エントリー一覧用のデザイン定義	24
5. 5 デザイン リファレンス	25
5. 5. 1 指定の箇所へデータ項目を埋め込む。	25
5. 5. 2 画像項目 (img 項目) を出力する。	26
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)。	27
5. 5. 4 繰り返し項目 (list 項目) を出力する。	28
5. 5. 5 データ値の内容によって処理を変える。	29
5. 5. 6 メニュー項目を表示する。	31
5. 5. 7 エントリー一覧から個別のエントリへリンクを張る。	32
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	33
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	35
6. ウェブサイトへの埋め込み	36
6. 1 デザイン定義と HTML 画面の関係	36
6. 2 埋め込み先の画面の作成	37
6. 3 埋め込み命令	38
6. 3. 1 cmsview::entry 命令 - エントリ1件分の埋め込み	38
6. 3. 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)	39
6. 3. 3 cmsview::listtop 命令 - エントリー一覧の埋め込み	40
6. 3. 4 cmsview::listpage 命令 - エントリー一覧の埋め込み (ナビゲーション付き)	40
6. 3. 5 絞込みの指定	41

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 サイトをご覧ください。

<http://cms.al-design.jp/>

※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、当ツールがまさにデザイナーが、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。

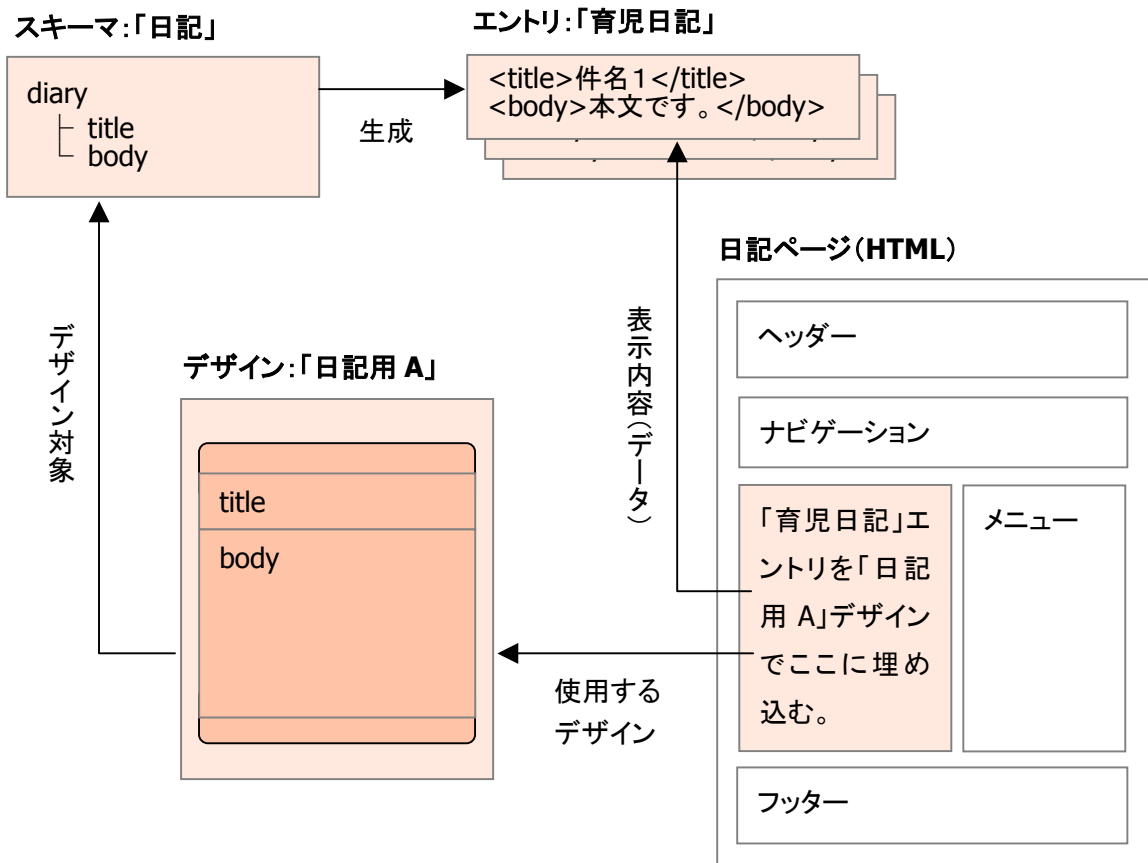
「デザイナー」以外に登場する役割としては「サイト管理者」があります。サイト管理者は、完成したウェブサイトを実際に更新していく人のことを指します。

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。



(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。

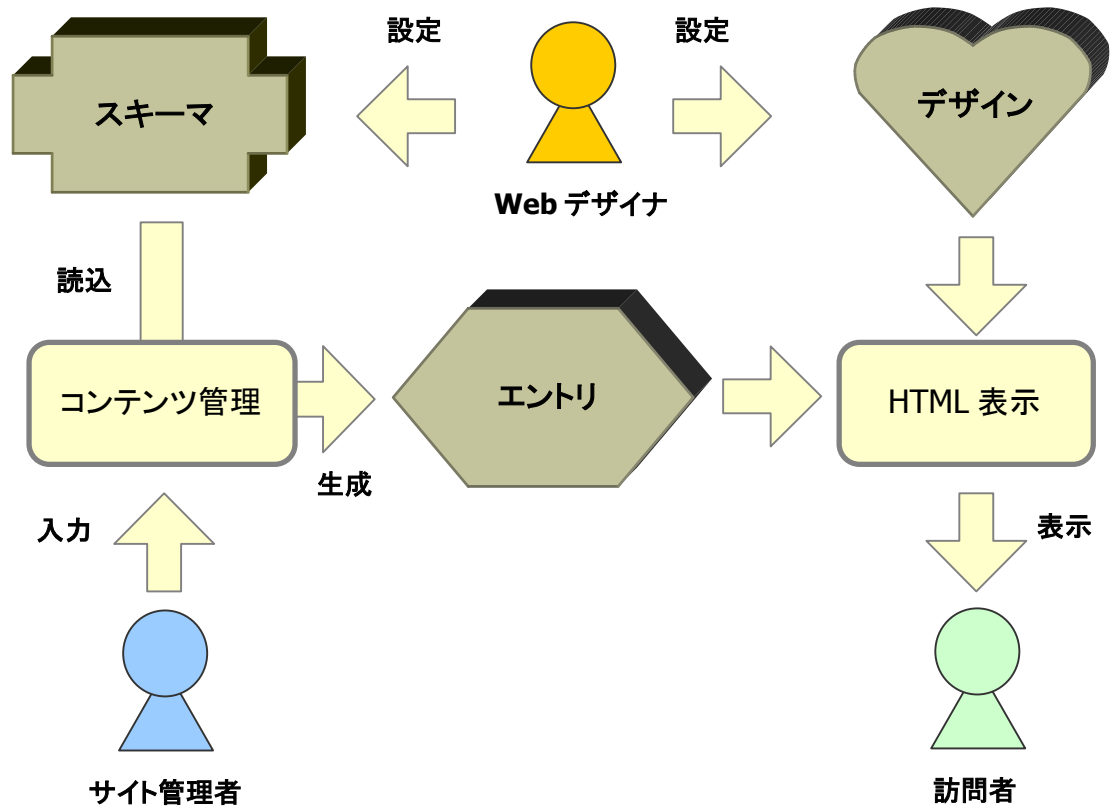


図: CMS Designer とその関係者

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

ユーザー名やパスワード(暗号化)、およびエントリ定義をここに設定します。

ユーザー名とパスワードは自動的に書き込まれますが、エントリ定義はデザイナーが直接編集します。

/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.xml

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.xml

エントリ一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

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。

スキーマの定義

スキーマ種類の洗い出し、各スキーマの項目の洗い出し、定義ファイルの作成、など。



エントリの定義

たいした作業ではありませんが、site.config.xml にエントリ名を追加し、エントリ用のフォルダを作成する作業です。



デザインの定義

小さなテンプレートを作っていく作業になります。1件用のデザイン、一覧用のデザインなど、1つのスキーマに対して複数のデザイン定義を行うこともあります。



ウェブサイトへの埋め込み

どのエントリを、どのデザインを使って、画面上のどこに埋め込むかを指定します。あなたの HTML ページにたった 2 行加えるだけの作業です。

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:)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

日記帳	
件名	<input type="text"/>
本文	<input type="text"/>

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、

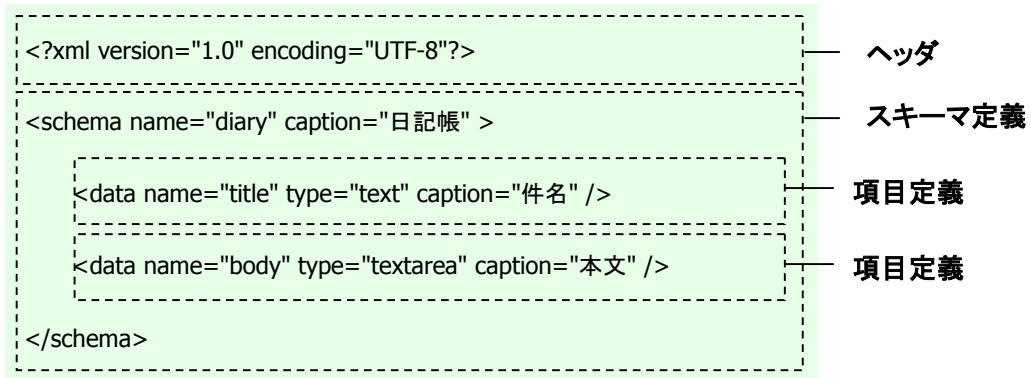
「スキーマ名」+「.schema.xml」

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。



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「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(全角不可)。
caption	不可	日本語名称。

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。
list	リスト項目。

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(全角不可)。 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。
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。
output	可	入力されたテキストデータをどのように HTML に出力するかを指定します。 - "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(自動開業)。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。

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(全角不可)。 同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。
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。
output	可	※text 項目と同じです。

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(全角不可)。 同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。

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(全角不可)。 同ー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。

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(全角不可)。 同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。

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(全角不可)。 同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。

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

3. 5. 9 data タグ - list

list 項目は、繰り返し項目を入力する際に使用します。

例えば、画像データを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="写真の説明文 1" />
  <data name="photo2" type="img" caption="写真の説明文 2" />
  <data name="photo3" type="img" caption="写真の説明文 3" />
  <data name="photo4" type="img" caption="写真の説明文 4" />
  <data name="photo5" type="img" caption="写真の説明文 5" />
  <data name="photo6" type="img" caption="写真の説明文 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="写真の説明文リスト" >
    <listitem caption="写真の説明文" >
      <data name="photo" type="img" 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(全角不可)。 同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 タグの中に指定する「繰り返したい入力項目(data タグ)」は、1 つだけでなく複数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。

3. 5. 10 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來ます。

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、「title」の項目定義に sort 属性を追加します。

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

sort 属性は、昇順か降順かで以下のように設定します。

sort 属性	説明
"asc"	昇順でソートする。
"desc"	降順でソートする。

尚、sort 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。また、複数指定することはできません。

例えば、次のように繰り返し項目内での指定はできませんのでご注意ください。

```
<?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="sectionlist" type="list" caption="各章 " >
    <listitem caption="各章の内容" >
      <data name="sectiontitle" type="text" caption="章タイトル" sort="desc" /> ←不可
      <data name="section" type="textarea" caption="章本文" />
    </listitem>
  </data>
</schema>
```

[!] 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。

3. 5. 11 グループ(絞込み)指定

エントリー一覧に全てのエントリーを表示するのではなく、全エントリーの中から「この種類のエントリーだけ」を一覧したい場合があります。

例えば「商品」エントリーを、「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ーの中から「お店種別」が「中華」や「ラーメン」などのエントリー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。

※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」のエントリー、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。

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名です。

`/cmsdesigner/data/entry`

例えば“mydiary”というエントリ名ならば、

`/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>
  <manager>
    <user name="*****" password="*****" />
  </manager>
  <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	エントリ名。半角英数字で設定します(全角不可)。 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。

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、スキーマを再利用することができます。

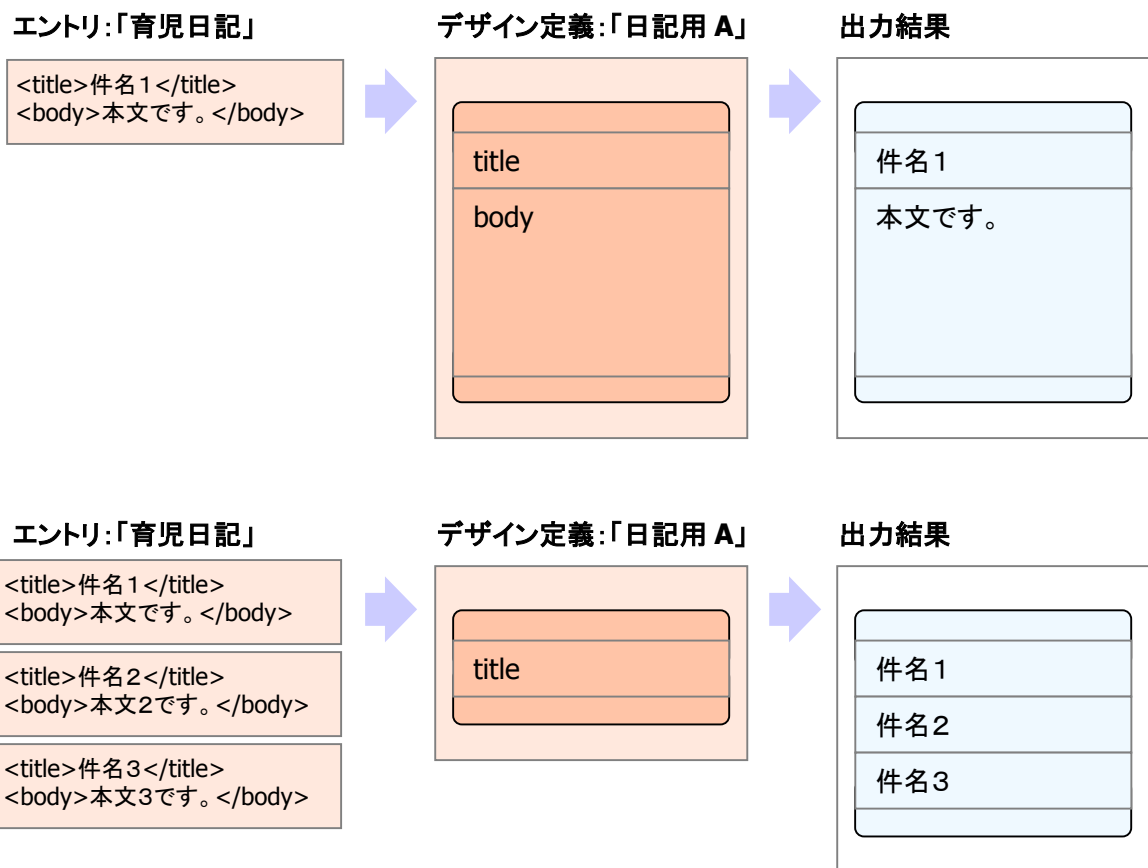
5 デザイン定義

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。



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="EUC-JP" 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。

5.3 エントリ1件用のデザイン定義

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.xml

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.xml
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。

<pre><?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="EUC-JP" 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></pre>	<p>ヘッダ</p> <p>デザイン</p> <p>フッタ</p>
---	-----------------------------------

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。

5. 4 エントリー一覧用のデザイン定義

エントリー一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.xml

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.xml
というファイル名になります。

エントリー一覧用のデザイン定義は以下のようになります。

<pre><?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="EUC-JP" omit-xml-declaration="yes" /> <xsl:template match="/entrylist"> <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></pre>	<p>ヘッダ</p> <p>デザイン</p> <p>フッタ</p>
---	-----------------------------------

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。

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트리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。

```

```

例) ``

ここで、次のように書くことはできません。

```
" />
```

属性の中に画像の URL を入れればよ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。

```

```

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。

```

```

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。

```

```

又は

```

```

例) ``

「&w;」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"></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、

```

```

ではなく、

```

```

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、

```

```

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はいけません。

5. 5. 4 繰り返し項目 (list 項目) を出力する。

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/lisitem">
    <tr>
      <td><xsl:value-of select="phototitle" /></td>
      <td></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" caption="画像のタイトル" />
    <data name="photo" 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></td>
  </tr>
  <tr>
    <td>画像2</td>
    <td></td>
  </tr>
  <tr>
    <td>画像3</td>
    <td></td>
  </tr>
</table>
```

5. 5. 5 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る場合」という書き方は、言い換えれば「画像データの中身が入っていたら」という意味になります。

「データに中身が入っていたら」と指定したい場合は、

test="データ名/text()"

と書きます。"/text()"というのは「～の中身」という意味です。

```
例)
<xsl:if test="photo/text()">
  
</xsl:if>
```

逆に、「データの中身が入っていなかったら」と書きたい場合は、

test="not データ名/text()"

と書きます。先頭に"not"を付けると、「～じゃなかったら」という意味になります。

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="photo/text()">
  
</xsl:if>
<xsl:if test="not photo/text()">
  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"

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<60">
  不合格
</xsl:if>
<xsl:if test="score>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>
```

5.5.6 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>
  <menuitem id="5">スイーツ</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>
<xsl:if test="shopkind=5">スイーツ</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(当たり前ですが)。

```
<xsl:if test="shopkind=1"></xsl:if>
<xsl:if test="shopkind=2"></xsl:if>
<xsl:if test="shopkind=3"></xsl:if>
<xsl:if test="shopkind=4"></xsl:if>
<xsl:if test="shopkind=5"></xsl:if>
```

5. 5. 7 エントリー一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、エントリー一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>
```


5. 5. 8 「次のエントリへ」「前のエントリへ」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>>](#)

「6. ウェブサイトへの埋め込み」でも説明しますが、CMS Designer でも同様のことが可能です。

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="EUC-JP" omit-xml-declaration="yes" />
  <xsl:template match="/entry">

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

```
<xsl:if="prev">
  <a href="?eid={prev/@id}">&lt;&lt;前へ</a> |
</xsl:if>
```

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。

“prev/@id”という名前のデータで次のエントリのエントリ ID を取得できるので、そのデータを使って
リンクを生成しています。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、“<”を表す記号です。

a タグの href 属性がいきなり“?eid=”で始まっているのは、同一の URL へジャンプするからです。
eid パラメータが変わるだけなので、ページの名前を省略しています。

```
<xsl:if test="next">
  <a href="?eid={next/@id}">次へ<img alt="next icon" data-bbox="585 122 605 135"/></a>
</xsl:if>
```

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。

“next/@id”という名前のデータで次のエントリのエントリ ID を取得できるので、そのデータを使ってリンクを生成しています。“>”というのは、“>”という文字を現す記号です。

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。

上記の「前へ」「次へ」のリンクが<xsl:if test="prev"></xsl:if>で囲ってあるのはその為です。

(test="prev"とは、「prev データが存在するならば」という意味になります。)

<xsl:if test="prev"></xsl:if>で囲ってある為、「前の記事」や「次の記事」が存在しない場合は、リンク自体が出力されません。

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、“not”を使って次のようにします(“not”の意味については「5. 5. 5 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:if test="prev">
    <a href="?eid={prev/@id}">&lt;&lt;前へ</a> |
  </xsl:if>
  <xsl:if test="not prev">
    &lt;&lt;前へ |
  </xsl:if>
  <xsl:if test="next">
    <a href="?eid={next/@id}">次へ<img alt="next icon" data-bbox="585 578 605 591"/></a>
  </xsl:if>
  <xsl:if test="not next">
    次へ<img alt="next icon" data-bbox="585 618 605 631"/>
  </xsl:if>
</xsl:for-each>
```

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

5. 5. 8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、エントリ一覧表示の場合にも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章をご覧ください)。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="EUC-JP" omit-xml-declaration="yes" />
  <xsl:template match="/entrylist">

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

細かい部分を除いて「5. 5. 8」と同じですので、詳しくはそちらを参考にしてください(eid パラメータが pageno パラメータに変わるだけです)。

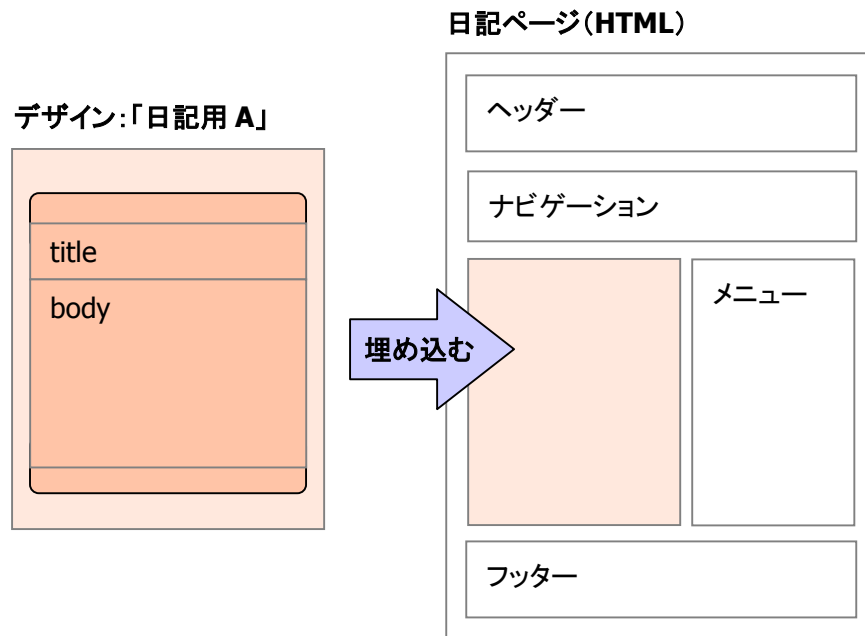
6 ウェブサイトへの埋め込み

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」、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」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。

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 euc-jp 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 euc-jp 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" ); ?>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=euc-jp">
  <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>
      <?php cmsview::entry( "public_diary", "default" ) ?>
    </td>
  </tr>
</table>
</body>
</html>
```

1行目の

```
<?php require( "cmsdesigner/include/view.php.inc" ); ?>
```

は、定型文として入れてください。

```
<?php cmsview::entry( "public_diary", "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 埋め込み命令」を参照してください。

6.3 埋め込み命令

6.3.1 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( "エントリ名", "デザイン名" ) ?>
```

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

指定したエントリ名を、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。

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。

「5.5.7 エントリー一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一覧から確認できます。

6. 3. 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::navi_entry 命令ではなく、この cmsview::navi_entry 命令を使います。

```
<?php cmsview::navi_entry( "エントリ名", "デザイン名" ) ?>
```

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

使い方は cmsview::navi_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. 8 「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. 3 cmsview::listtop 命令 - エントリー一覧の埋め込み

エントリー一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( "エントリー名", "デザイン名" ) ?>
```

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

指定したエントリー名のエントリー一覧を、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、エントリー一覧の表示件数は 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( "エントリー名", "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. 3. 4 cmsview::listpage 命令 - エントリー一覧の埋め込み(ナビゲーション付き)

エントリー一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( "エントリー名", "デザイン名" ) ?>
```

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

指定したエントリー名のエントリー一覧を、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、エントリー一覧の表示件数は 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( "エントリー名", "デザイン名", 表示件数 ) ?>
```

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

6. 3. 5 絞込みの指定

『3. 5. 11 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( "エントリー名", "デザイン名", 表示件数, 絞込み条件 ) ?>
```

```
<?php cmsview::listtop( "エントリー名", "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"のエントリー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"のエントリー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。

エントリー1件表示の場合でも、記事のナビゲーション(章「5. 5. 8 」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( "エントリー名", "デザイン名", 絞込み条件 ) ?>
```

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