MT のバージョンアップにしたがい、各バージョン用のサンプルテンプレートの修正を行った結果、本書「Movable Type デザインカスタマイズブック MT4.2 対応 ～クリエイターのための Value Design～」141 ページに記載の内容と、現在 SBCr より配布中のサンプルテンプレートの内容に違いが発生しております。謹んでお詫びさせていただくとともに、公開しているサンプルテンプレートに準じる形で、下記のように訂正情報を追加いたします。

Movable Type デザインカスタマイズブック MT4.2 対応 ～クリエイターのための Value Design～

Chapter3 テンプレートによるカスタマイズテクニック
12 メニューを左から右に移動させる

テンプレート C
136 ページ～141ページ

当該の項目は、「メニューを左から右に移動させる」というタイトルで、スタイルシートの仕組みを理解し、テンプレートの一部を編集することによって、ページのカラム構造を変更する、というものです。

本書 136 ページから 140 ページまでの間で、サンプルテンプレート C におけるカラム構造の仕組みと、スタイルシートでのレイアウト制御の方法を解説しています。まずはおさらいを兼ねて、整理していきましょう。

サンプルテンプレート C のメインページ(トップページ)は、4 カラムのレイアウト構造を持っていて、デフォルトの状態では、左から「beta エリア」、「alpha エリア」、「gamma エリア」、「delta エリア」の順に表示されています。表示順は、スタイルシートの「.layout-twtt」というセレクタ群の内容によって制御されています。

このデフォルトの 4 カラム構造を変更するためには、まず、左から、「alpha エリア」、「beta エリア」、「gamma エリア」、「delta エリア」の順に表示されるように、スタイルシートを記述する必要があります。本書が附録として配布しているサンプルテンプレートでは、変更用のスタイルシートの記述は前もって記述してあります。インデックステンプレートの「style.css」を確認してください。

<サンプルテンプレート C のインデックステンプレート「style.css」の当該部分>

```
/*===== W-T-T-T =====*/
.layout-wttt #content {
	background: url(../images/bg-content-wttt.jpg) top left repeat-y;
}
.layout-wttt #content-inner {
	background: url(../images/bg-content-top-wttt.jpg) top left no-repeat;
}
.layout-wttt #alpha {
	float: left;
	left: 25px;
	width: 360px;
}
.layout-wttt #beta {
	float: left;
	left: 55px;
	width: 155px;
}
.layout-wttt #gamma {
	float: right;
	right: 190px;
	width: 155px;
}
.layout-wttt #delta {
	float: right;
	right: -140px;
	width: 155px;
}
```

ここまでは、138 ページから 140 ページまで内容の補足説明です。以上の内容を確認していただいた上で、以後は 141 ページの内容の訂正となります。

スタイルシート側の準備はすでに完了していますので、あとは**テンプレートを適切に編集して、意図したとおりに「layout-wttt」クラスを適用させるだけです。**修正するテンプレートは、インデックステンプレートの「メインページ」です。インデックステンプレート「メインページ」の内容を確認してみましょう。

<サンプルテンプレート C のインデックステンプレート「メインページ」>

```
<MTSetVar name="body_class" value="mt-main-index">
<MTSetVar name="main_template" value="1">
<MTSetVar name="main_index" value="1">
<MTSetVar name="page_layout" value="layout-twtt">
<MTSetVar name="sidebar" value="1">
<MTSetVarBlock name="title"><$MTBlogName encode_html="1"$></MTSetVarBlock>

<$MTInclude module="ヘッダー"$>

<MTEntries category="features" lastn="1">
    <$MTEntryTrackbackData$>
    <$MTInclude module="オススメ記事の概要"$>
</MTEntries>

<!--おすすめ始まり-->
<div id="recommend" class="clearFix">
<h2><img src="images/sub-recommend.gif" alt="おすすめ商品 " /></h2>
<MTEntries category="recommend" lastn="3">
<$mt:Include module="商品の概要"$>
</MTEntries>
</MTIf>
</div>
<!--おすすめ終わり-->


<$MTInclude module="フッター"$>
```

注目するのは、4 行目の「<MTSetVar name="page_layout" value="layout-twtt">」という記述です。この**<MTSetVar>は変数を宣言してその中身を代入しています。**

具体的に言うと、変数名「page_layout」の中に、「layout-twtt」の値を入れています。「メインページ」は、いわゆるトップページを生成するのに使われるテンプレートです。トップページが生成される際、まず、ここで変数「page_layout」に値「layout-twtt」が入れられることを確認してください。

結論だけをいうと、**この変数に入れられる値を「layout-twtt」から「layout-wttt」に変更すれば、トップページ(メインページ)のカラムレイアウトは変更されます。**つまり、サンプルテンプレートCのインデックステンプレート「メインページ」の4行目の記述を、
<MTSetVar name="page_layout" value="layout-twtt">から
<MTSetVar name="page_layout" value="layout-wttt">に変更すればいいだけなのです。

せっかくですから、この変数「page_layout」が、実際に再構築が行われる際にどのように利用されるのかも併せて確認しましょう。変数「page_layout」の中身が HTML として生成される際の記述は、テンプレートモジュール「ヘッダー」の中にあります。関係のある部分だけを抜き出してみましょう。

<サンプルテンプレートCのテンプレートモジュール「ヘッダー」>

```
<MTUnless name="main_index"><MTSetVar name="page_layout"
value="layout-twt"></MTUnless>
<body class="<$MTGetVar name="page_layout"$><MTIfArchiveType
archive_type="Category"> item-list</MTIfArchiveType><MTIf name="datebased_archive">
item-list</MTIf><MTIf name="system_template"> item-list</MTIf>"<MTIf
name="body_onload"> onload="<$MTGetVar name="body_onload"$>"</MTIf>>

(中略)

<!--Content-->
<div id="content" class="<$MTGetVar name="page_layout" default="layout-twt"$>
clearFix">
```

変数「page_layout」が判定されるのは 2 か所です。まずは、<body>タグ部分から見ていきましょう。<body>タグ開始直前に実行されている<MTUnless>は**「もしこのページがトップページ(メインページ)でなければ、変数「page_layout」に値「layout-twt」を入れなさい」**という意味です。

ここでは、トップページ(メインページ)が生成される場合を想定して考えてみましょう。当然のことながら、トップページ(メインページ)が生成されようとしているので、<MTUnless>の中身は無視されます。結果として、先ほどインデックステンプレートで変数「page_layout」に代入された値「layout-wttt」がそのまま保持されます。

ちなみに、トップページ(メインページ)以外のページが生成される際には、ここで変数「page_layout」に代入された値「layout-twt」が代入されます。その結果、3カラムのページが表示されることになります。<MTUnless>の判定の後、<body>タグにクラスを指定する記述が続きます。「class="<$MTGetVar name="page_layout"$>"」として、先ほどの代入した変数「page_layout」の値を呼び出すように記述してある、というわけです。
(「<$MTGetVar name="page_layout"$>」の後に、<MTIf>で記述された他の条件判定が続きますが、カラムレイアウトの変更というテーマから離れますので、ここでは説明しません)。

この結果、トップページ(メインページ)が生成される際には、変数「page_layout」の値は「layout-wttt」が入っているはずですから、テンプレート編集後のサンプルテンプレート C の<body>タグ部分は「<body class="layout-wttt">」となり、事前に準備しておいたスタイルシートが適用され、カラムレイアウトの変更が行われます。

もう1ヶ所の、「<div id="content" class="<$MTGetVar name="page_layout" default="layout-twt"$> clearFix">」の部分についても、説明しておきましょう。
id名に「content」の タグを生成する際に、クラス名を同時に指定しようとしています。変数「page_layout」が関係する「<$MTGetVar name="page_layout" default="layout-twt"$>」の部分が、そのままクラス名となります。

これは、**「変数「page_layout」の値を呼び出しなさい。もし、何も設定されていない場合は値に「layout-twt」が入っているものとして呼び出しなさい」**という意味です。先ほどから繰り返しになりますが、トップページ(メインページ)が生成される際には、変数「page_layout」に値「layout-wttt」が設定されていますから、「layout-wttt」が呼び出されます。
結果、当該部分のHTMLのコードは、 となるはずです。これもテンプレートを編集して再構築した後に確認してみてください。

スタイルシートを理解すれば、カラムレイアウトの構造は理解できますし、準備しておいたスタイルシートをテンプレートから適切に呼び出して適用させるのも、何度かやってみれば身につきます。まずは、サンプルテンプレートをいろいろいじってみて実験してみましょう。試行錯誤を繰り返すうちに、スタイルシートとテンプレートの関係性に対して、理解は深まるはずです。