

基本からしっかり学ぶ

 blog cms の
カスタマイズ

blogテーマ編



appleple

Web System&Design

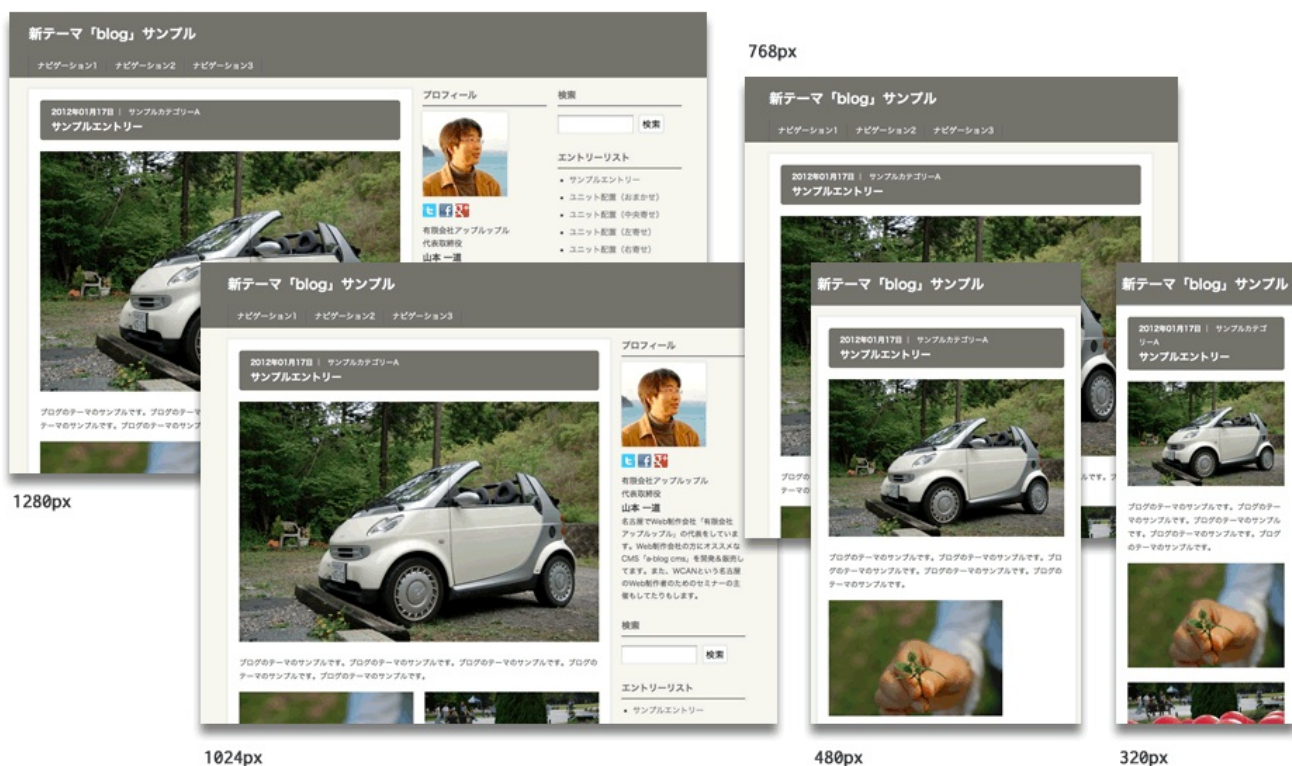
1.blog テーマの特徴

a-blog cmsをブログとしてお使いいただく際にベースとなるテーマです。

レスポンシブwebデザインを採用しており、PCの画面の幅や、スマートフォン、タブレットPCなど、画面の幅が異なるデバイスで閲覧をしても、そのデバイスに最適な大きさにレイアウトが変更されます。管理画面から「カラー設定」をすると、サイト全体の色を一括で変更する事が可能です。テンプレートファイルの数も1枚とシンプルな構成で、少しのカスタマイズをするだけで、様々なデバイスに適応したオリジナルのブログデザインに変更可能です。

レスポンシブwebデザイン

レスポンシブwebデザインを採用しており、PCの画面の幅や、スマートフォン、タブレットPCなど、画面の幅が異なるデバイスで閲覧をしても、そのデバイスに最適な大きさにレイアウトが変更されます。管理画面から「カラー設定」をすると、サイト全体の色を一括で変更する事が可能です。テンプレートファイルの数も1枚とシンプルな構成で、少しのカスタマイズをするだけで、様々なデバイスに適応したオリジナルのブログデザインに変更可能です。



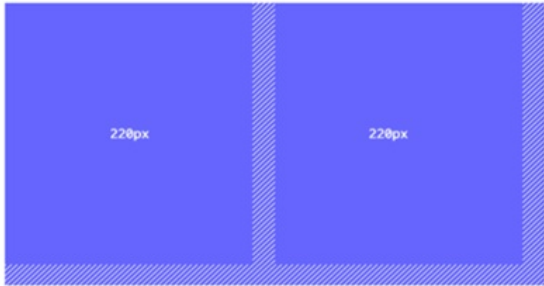
今回のブログのテーマについては、320px,480px,768px,1024px,それ以上の5つのサイズについて対応しております。

ユニットのサイズとCSS

これまでの a-blog cms の標準のテーマでは、画像やユニットグループのサイズについては、200px,300px,400pxという画像サイズで、ユニットグループについては、設定されていない状態でした。

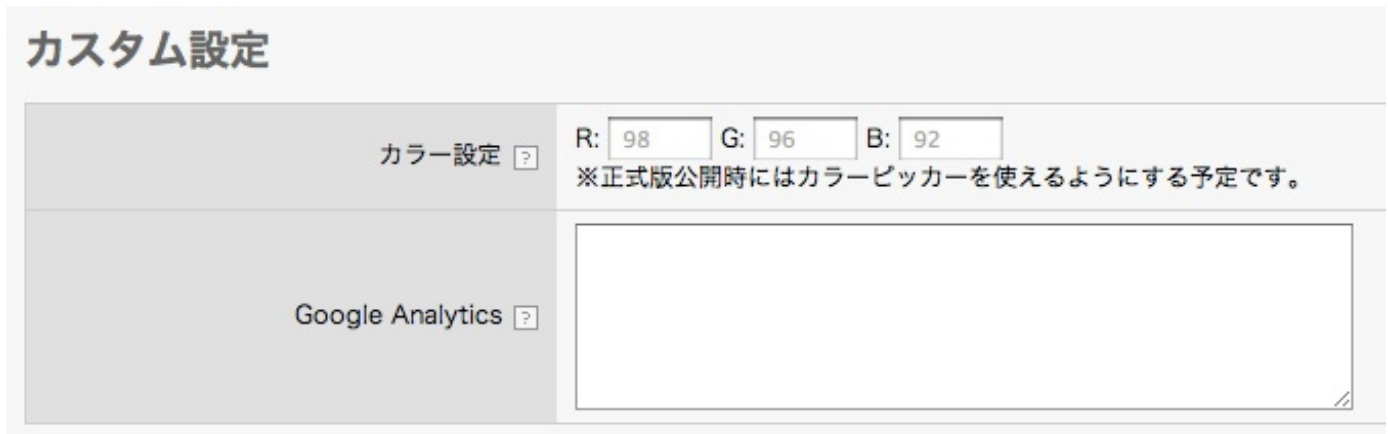
今回は、メインエリアの画像の横幅とユニットグループの横幅のサイズを、メインエリアのサイズいっぱいの幅を一番大きなサイズとし、中くらいのサイズには20pxのマージンを取った残りのサイズを1/2したサイズに、小のサイズについては同様に1/3にしたサイズを初期値として設定しました。

またメインのエリアには右側には Padding を用意せず、画像やユニットのDIVの右マージンがあったとしても大丈夫なようなCSSレイアウトの設計になっていて、中に入ってくる各ユニットのブロック要素の右側に20pxのマージンがついています。そのままのCSSを利用する際には、この部分に注意する必要があります。



2. ブログのカスタムフィールド

まず、ブログのカスタムフィールドで設定ができる簡単なところから紹介していきます。
サイト管理 > ブログ管理 > ブログ詳細 から（変更）ボタンをクリックすると、ブログの設定画面が表示されます。そこにカスタム設定というエリアがあります。



ここでは、ブログのカラーと Google Analytics のコードを指定する事ができるようになっています。

```
/themes/blog/admin/blog/field.html
```

カラー設定

a-blog cms では、HTMLファイルをテンプレートにする事以外にもテキストファイルであれば、どのようなファイルでもテンプレート化する事が可能です。今回の blog2012 では、CSSファイルにモジュールを埋め込み、カスタムフィールドで色の設定を変更できるようになっています。

CSS3のRGBAを利用し1色設定する事で、その色のアルファ値を変更し、指定カラーの薄い色を表現しています。インストールした初期値のままだと他の人と同じになってしまう可能性も高いので、好きな色に変更してみましょう！

また、ブログのカスタムフィールドの値を表示させたいという事になりますので、利用しているモジュールは Blog_Field モジュールになります。一度、以下のファイルをご覧ください。

```
/themes/blog/css/blog.css
```

Google Analytics

アクセス解析の定番である Google Analytics のコードを貼る欄も用意しておきました。このようにサイトのどこにでも利用するようなもので、ユーザー自身が書き換える事ができる部分はブログのカスタムフィールドとして設定します。

```
/themes/blog/index.html
```

上記の head の最後のあたりに以下のように記述してあります。

```
<!--BEGIN_MODULE Touch_Unlogin -->
  <!-- BEGIN_MODULE Blog_Field -->
    {googleAnalytics}[raw]
  <!-- END_MODULE Blog_Field -->
<!-- END_MODULE Touch_Unlogin -->
```

こちらでは、Blog_Field で {googleAnalytics} を校正オプションの [raw] を利用して、そのまま表示するという指示をしてあります。（rawを利用しない場合にはデフォルトでescapeとなり<>タグ等を実態参照に変換されてしまいます）また、Touch_Unlogin というログインしていない時のみ表示というモジュールで囲んでいる事から、ログイン時にはアクセス解析のタグを表示させないようにしています。

3.ユーザーのカスタムフィールド

プロフィール



有限会社アップルアップル

山本 一道

名古屋のホームページ制作会社「有限会社アップルアップル」の代表をしています。弊社で開発しているCMS「a-blog cms」や、名古屋の「WCAN」も、よろしくお願ひします。

上記のプロフィールの情報は、プロフィール（管理ページ > ユーザー一覧 > ユーザー詳細）から編集できるようになっています。今回の blog テーマは標準の管理ページから Twitter, Facebook, Google+ の3つの入力欄が拡張されており、入力する事で、個人のページへのリンクのついたアイコンが表示されるようになります。

プロフィール

所属 ?	<input type="text"/>
肩書き ?	<input type="text"/>
自己紹介文 ?	<input type="text"/>
画像 ?	<input type="button" value="ファイルを選択"/> ファイルが未選択です
代替テキスト ?	<input type="text"/>
Twitter ?	<input type="text"/>
Facebook ?	<input type="text"/>
Google+ ?	<input type="text"/>

この管理ページをカスタマイズしたい時には `/themes/blog/admin/entry/field.html` をご覧になってみて下さい。プロフィール以降のHTMLファイルが見つかります。

例えば、Flickr のURLを追加する場合には以下のような記述を追加します。

```
<tr>
  <th>Flickr</th>
  <td>
    <input type="text" name="flickr" value="{flickr}" size="50" />
    <input type="hidden" name="field[]" value="flickr" />
  </td>
</tr>
```

このようにユーザーの情報を追加する事は、ユーザーのカスタムフィールドのカスタマイズで可能です。

また、表示については `/themes/blog/index.html` のプロフィールには **User_Profile** モジュールが利用されています。このモジュールの中ではユーザーのカスタムフィールドは変数を書くだけで利用できるようになっています。

```
<!-- BEGIN iconList:veil -->
<ul class="iconList">
  <!-- BEGIN twitter:veil --><li><a href="{twitter}"></a></li><!-- END twitter:veil -->
  <!-- BEGIN facebook:veil --><li><a href="{facebook}"></a></li><!-- END facebook:veil -->
```

```
<!-- BEGIN googlePlus:veil --><li><a href="{googlePlus}"></a></li><!-- END googlePlus:veil -->  
</ul>  
<!-- END iconList:veil -->
```

a-blog cms では、<!-- BEGIN 名前 --> から <!-- END 名前 --> までの間に何も変数が編集されない場合には、そのエリアは無かった事になって何も表示しないという機能が用意されています。ですから、ここでは {twitter}, {facebook}, {googlePlus} について何も編集されない時にはタグが表示されない事になります。そして、その機能のために書かれている<!-- BEGIN 名前 -->の名前には :veil とする命名規則になっています。この消すための :veil 付きのBEGINとENDのコメントについては消してしまっても問題ありません。

ここに新しいリンクを増やしたい場合には、3つの書き方を真似て増やせばいい事になります。

4.ナビゲーションを設定する

初期状態では何も設定されていませんが、タイトルの下にナビゲーションが設定されています。追加すると以下のような感じに表示する事ができます。



また、画面がスマートフォンサイズの時には、横幅が足りずに表示する事ができない事が想定されますので、上部のナビゲーションは表示させずに、サブカラム側に非表示にしてある同様のモジュールを表示させるような準備がされています。

5. リンク集やバナーを設定する

こちらでも初期設定では何も設定されていませんが、サブカラムに外部サイトへのテキストリンクやバナーを追加する事が可能になっています。

リンク集

- リンク1
- リンク2
- リンク3



6.ユニットグループや画像のサイズに2/3を追加する

1.5 の blog のテーマからは、ユニットグループの設定が標準で設定されています。

blog のテーマのメインのエリアサイズは 640px です。1/1サイズ、1/2サイズ、1/3サイズが標準の設定となっています。

- 1/1サイズ 画像 640px
- 1/2サイズ 画像 310px + マージン 20px + 画像 310px = 全体 640px
- 1/3サイズ 画像 200px + マージン 20px + 画像 200px + マージン 20px + 200px = 全体 640px

2/3サイズを設定する

画像のサイズの設定や、ユニットグループの設定を学ぶために新しい設定を追加してみます。カスタマイズ管理 > コンフィグ > エントリー > 編集設定 > ユニットグループに 2/3カラム (column2_3) を追加設定します。

ユニットグループ

↑ ↓	クラス ?	ラベル ?	
↑ ↓	<input type="text"/>	<input type="text" value="---"/>	<input type="button" value="削除"/>
↑ ↓	<input type="text" value="column1"/>	<input type="text" value="1カラム"/>	<input type="button" value="削除"/>
↑ ↓	<input type="text" value="column2"/>	<input type="text" value="2カラム"/>	<input type="button" value="削除"/>
↑ ↓	<input type="text" value="column3"/>	<input type="text" value="3カラム"/>	<input type="button" value="削除"/>
↑ ↓	<input type="text" value="column2_3"/>	<input type="text" value="2/3カラム"/>	<input type="button" value="削除"/>
<input type="button" value="追加"/>			

このユニットグループに丁度いいサイズの画像の幅の設定も追加しておきましょう。

- 画像 200px + マージン 20px + 画像 200px = 画像 420px

	基準	サイズ	ラベル	
↑ ↓ イメージサイズ選択肢 ?	<input type="button" value="長辺 ↓"/>	<input type="text"/>	<input type="text" value="そのまま"/>	<input type="button" value="削除"/>
	<input type="button" value="横 ↓"/>	<input type="text" value="640"/>	<input type="text" value="横幅 1/1 (640px)"/>	<input type="button" value="削除"/>
	<input type="button" value="縦 ↓"/>	<input type="text" value="420"/>	<input type="text" value="横幅 2/3 (420px)"/>	<input type="button" value="削除"/>
	<input type="button" value="横 ↓"/>	<input type="text" value="310"/>	<input type="text" value="横幅 1/2 (310px)"/>	<input type="button" value="削除"/>
	<input type="button" value="横 ↓"/>	<input type="text" value="200"/>	<input type="text" value="横幅 1/3 (200px)"/>	<input type="button" value="削除"/>
<input type="button" value="追加"/>				

blog.css

```
/* -----  
2/3カラム用  
----- */  
div.column2_3 {  
    float: left;  
    width: 420px;  
    margin: 0 20px 20px 0;  
}  
  
/* 画像、Youtube、GoogleMaps */  
div.column2_3 .column-image-center,
```

```
div.column2_3 .column-image-left,  
div.column2_3 .column-image-right,  
div.column2_3 .column-image-auto,  
div.column2_3 .column-youtube-center,  
div.column2_3 .column-youtube-left,  
div.column2_3 .column-youtube-right,  
div.column2_3 .column-youtube-auto,  
div.column2_3 .column-eximage-center,  
div.column2_3 .column-eximage-left,  
div.column2_3 .column-eximage-right,  
div.column2_3 .column-eximage-auto{  
    float: none;  
    width: 420px;  
    margin: 0 0 20px;  
    padding-right: 0;  
}  
  
.entry div.column2_3 h2,  
.entry div.column2_3 h3,  
.entry div.column2_3 h4,  
.entry div.column2_3 h5,  
.entry div.column2_3 p,  
.entry div.column2_3 ul,  
.entry div.column2_3 ol,  
.entry div.column2_3 pre,  
.entry div.column2_3 table,  
.entry div.column2_3 blockquote{  
    margin-right: 0;  
}
```

responsive.css

blog のテーマはレスポンシブ Web デザイン対応している事から blog.css と同様な感じでCSSの設定追加が必要になります。div.column2 の設定がある部分をコピーし、div.column2_3 の設定を追加しましょう。

今回の blog のテーマでは、メインエリアの右側のPaddingが段組や画像のフロートしている右側のマージンを納めるために設定されておりません。ユニットグループを利用した段組をする際には、この部分も考慮してCSSを設定する必要があります。

7.Facebook に対応する

今、ソーシャル対応は必須なカスタマイズです。ここではエントリーに「いいね」ボタンと、そのボタンがクリックされ外部サイトからリンクされる際に表示される情報をコントロールするOGP(Open Graph Protocol)の設定をします。

1点、注意するところとしては、Facebook は変化の激しいところなので、ここに書いてある事がいつまで通用するか分かりません。場合によっては、期間指定で検索して情報を収集する必要があります。

いいねボタンの設置

まずは、<http://developers.facebook.com/docs/reference/plugins/like/> にアクセスして、自分のブログのURLを入力し、(Get Code)ボタンを押してみましょう。

Like Buttonのプラグインコード:

HTML5 XFBML IFRAME

1. ページにJavaScript SDKを含めます(理想的には、<body>のすぐ後に配置します)。

This script uses the app ID of your app: **kazumich_log** ▼

```
<div id="fb-root"></div>
<script>(function(d, s, id) {
  var js, fjs = d.getElementsByTagName(s)[0];
  if (d.getElementById(id)) return;
  js = d.createElement(s); js.id = id;
  js.src = "//connect.facebook.net/ja_JP/all.js#xfbml=1&appId=123934531024884";
  fjs.parentNode.insertBefore(js, fjs);
})(document, 'script', 'facebook-jssdk');
```

2. プラグインを表示したい場所にプラグインのコードを配置します。

```
<div class="fb-like" data-href="http://kazumich.com" data-send="true" data-width="450" data-show-faces="true"></div>
```

OK

ここに書かれているように<body>のすぐ後に、指定のコードをコピーし、いいねボタンの設置したいところにも、コードを追加します。今回はHTML5版のを指定する事とします。そうする事でいいね！ボタンを押した際にコメントの入力画面が表示されますのでお勧めです。



そのままのコードを貼っていると、全部のエントリーが同じになってしまいます。そこで data-href="自分のブログのURL" を data-href="{permalink}" のように書き換えます。グローバル変数 `{PERMALINK}` を利用しようとする方もいらっしゃるかと思いますが、グローバル変数のパーマリンクを利用すると、一覧表示時に一覧のURLが複数のエントリーで使われてしまいます。また、この場合はEntry_Bodyのentry:loop内である必要があります。

```
<div class="fb-like" data-href="{permalink}" data-send="true" data-width="600" data-show-faces="true"></div>
```

OGP(Open Graph Protocol)の設定

OGP(Open Graph Protocol)とは、ソーシャルグラフ内で任意のウェブページを表現するための情報を提供する技術でMETAタグを利用して設定します。

今回、設定するタグを長いですが、まずはご紹介しておきます。ここでお伝えしたい点は2つで、1つはタッチモジュールの使い方、もう1つはEntry_SummaryのモジュールIDの設定という事になります。

```
<meta property="og:type" content="blog" />
<meta property="og:url" content="{INHERLINK}" />
<meta property="og:site_name" content="{BLOG_NAME}" />
<meta property="fb:admins" content="*FacebookのID*" />

<!-- BEGIN_MODULE Touch_Top -->
<meta property="og:title" content="{BLOG_NAME}" />
<meta property="og:image" content="*トップページ用の画像*" />
<meta property="og:description" content="{META_DESCRIPTION}" />
<meta name="description" content="{META_DESCRIPTION}">
<meta name="keywords" content="{META_KEYWORDS}">
<!-- END_MODULE Touch_Top -->

<!-- BEGIN_MODULE Touch_Index -->

<meta property="og:image" content="*一覧ページ用の画像*" />
```

```

<meta name="keywords" content="%{META_KEYWORDS}">

<!-- BEGIN_MODULE Touch_Category -->
<meta property="og:title" content="カテゴリ : %{CATEGORY_NAME} | %{BLOG_NAME}" />
<meta property="og:description" content="%{BLOG_NAME} の カテゴリ 「%{CATEGORY_NAME}」 の一覧を表示して
います。" />
<meta name="description" content="%{BLOG_NAME} の カテゴリ 「%{CATEGORY_NAME}」 の一覧を表示しています。">
<!-- END_MODULE Touch_Category -->

<!-- BEGIN_MODULE Touch_Keyword -->
<meta property="og:title" content="キーワード : %{KEYWORD} | %{BLOG_NAME}" />
<meta property="og:description" content="%{BLOG_NAME} のキーワード 「%{KEYWORD}」 の検索結果一覧を表示して
います。" />
<meta name="description" content="%{BLOG_NAME} のキーワード 「%{KEYWORD}」 の検索結果一覧を表示しています。
">
<!-- END_MODULE Touch_Keyword -->

<!-- BEGIN_MODULE Touch_Tag -->
<meta property="og:title" content="タグ : %{TAG} | %{BLOG_NAME}" />
<meta property="og:description" content="%{BLOG_NAME} の タグ 「%{TAG}」 を含む記事の一覧を表示しています。
" />
<meta name="description" content="%{BLOG_NAME} の タグ 「%{TAG}」 を含む記事の一覧を表示しています。">
<!-- END_MODULE Touch_Tag -->

<!-- END_MODULE Touch_Index -->

<!-- BEGIN_MODULE Touch_Entry -->
<meta property="og:title" content="%{ENTRY_TITLE} | %{BLOG_NAME}" />

<!-- BEGIN_MODULE Entry_Summary id="ogpSummary" -->
<!-- BEGIN unit:loop --><!-- BEGIN entry:loop -->

<!-- BEGIN image:veil --><meta property="og:image" content="%{BLOG_URL}{path}" /><!-- END
image:veil -->
<!-- BEGIN noimage --><meta property="og:image" content="*画像が無い時のエントリーページ用の画像*"
/><!-- END noimage -->
<meta property="og:description" content="{summary}[trim(200, '...')]" />
<meta name="description" content="{summary}[trim(200, '...')]">
<!-- END entry:loop --><!-- END unit:loop -->
<!-- END_MODULE Entry_Summary -->

<!-- BEGIN_MODULE Entry_Body -->
<!-- BEGIN entry:loop -->
<meta name="keywords" content="<!-- BEGIN tag:loop -->{name}<!--BEGIN glue -->,<!--END glue --
><!-- END tag:loop -->">
<!-- END entry:loop -->
<!-- END_MODULE Entry_Body -->

<!-- END_MODULE Touch_Entry -->

```

タッチモジュールについて

ここで利用されているタッチモジュールは以下のものがあります。

Touch_Top トップページを表示している時に表示させるモジュール

Touch_Index 一覧を表示している時に表示させるモジュール

Touch_Category カテゴリーが検索条件に含まれている時に表示させるモジュール

Touch_Keyword キーワードが検索条件に含まれている時に表示させるモジュール

Touch_Tag タグが検索条件に含まれている時に表示させるモジュール

Touch_Entry 1つのエントリーを表示している時に表示させるモジュール

また、Touch_Index の中に、Touch_Category ・ Touch_Keyword ・ Touch_Tag が設定されています。例えば、カテゴリーの一覧の時には、Touch_Category と Touch_Index という部分が表示され、それ以外の部分は表示されない事になります。このように状態(トップ・一覧・詳細)によって、表示内容を制御するためにタッチモジュールは利用します。

モジュールは入れ子に記述する事ができ、実行順としては内側から外側の順番で実行されます。ですので基本的には全てのモジュールが動作している事になります。

モジュールIDの指定

今回の OGP 設定については、Touch_Entry の中に Entry_Summary ・ Entry_Body が設定されています。ここでの Entry_Summary は、OGPの画像を表示させる事と、本文を指定文字数表示させるのに利用し、Entry_Summary で表示させられないタグの情報をMETAのKEYWORDに編集するために Entry_Body を利用しています。

Entry_Body については、URLコンテキストの初期スコープ設定の関係で、モジュールID を設定する必要ありませんが、Entry_Summary については、eid で検索できるようにモジュールIDを設定します。今回は id="ogpSummary" とIDが設定されています。

ID情報

モジュール <small>?</small>	サマリー (Entry_Summary) <small>↓</small>
id <small>?</small>	ogpSummary
名前 <small>?</small>	OGPで利用するエントリーサマリーモジュール用
説明 <small>?</small>	
グローバル <small>?</small>	<input type="checkbox"/> 下の階層のブログが利用することを許可する

引数 (チェックするとURLコンテキストが優先されます)

ブログID (bid) <small>?</small>	<input type="checkbox"/>	<input type="text"/>	(ID参照)
ユーザーID (uid) <small>?</small>	<input type="checkbox"/>	<input type="text"/>	(ID参照)
カテゴリID (cid) <small>?</small>	<input type="checkbox"/>	<input type="text"/>	(ID参照)
エントリーID (eid) <small>?</small>	<input checked="" type="checkbox"/>	<input type="text"/>	
キーワード (keyword) <small>?</small>	<input type="checkbox"/>	<input type="text"/>	
タグ (tag) <small>?</small>	<input type="checkbox"/>	<input type="text"/>	
フィールド (field) <small>?</small>	<input type="checkbox"/>	<input type="text"/>	
開始日時 (start) <small>?</small>	<input type="checkbox"/>	日付: <input type="text"/>	時刻: <input type="text"/>
終了日時 (end) <small>?</small>	<input type="checkbox"/>	日付: <input type="text"/>	時刻: <input type="text"/>
ページ (page) <small>?</small>	<input type="checkbox"/>	<input type="text"/>	
表示順 (order) <small>?</small>	<input type="checkbox"/>	<input type="text"/>	

ここで eid にチェックをつける事で、モジュールがエントリーIDで検索できるようになります。

8.レスポンシブ Web デザインの問題点を改善

レスポンシブ Web デザイン 対応をする際には、CSSのメディアクエリを利用して、画面のサイズに合わせてレイアウトをいくつか設定します。ウィンドウが狭い時には、画像がメインの カラムから飛び出さないようにサイズを小さくして表示させる事でレイアウトを崩す事なく表示ができるようになります。

しかし、そのままのテンプレートの書き方では、画像自体は大きなまま転送される事になりスマートフォンでの3G回線では、遅くて表示に時間がかかってしまうという問題があります。そこで1.5.0 から追加された新しい機能を利用する事で、ウィンドウサイズの小さなスマートフォンの時には小さな画像を利用するようにします。

a-blog cms の画像生成について

画像ユニットでは、1枚の画像をアップロードすると画像ユニット上に設定されているサイズで画像が生成され、初期設定時には拡大表示用の画像と小さな画像の3枚を生成します。また、1.5.0からは正方形にトリミングした画像も生成できるようになりました。ランダム文字列.jpg（通常画像）、tiny-ランダム文字列.jpg（小さな画像）、large-ランダム文字列.jpg（大きな画像）、square-ランダム文字列.jpg（正方形の画像）というファイル名になります。

管理ページ > カスタマイズ管理 > コンフィグ > エントリー 編集設定 からサイズの設定が可能です。

イメージ編集

拡大表示時の最大サイズ ?	横	1000	px
モバイル用画像の最大サイズ ?	横	280	px
正方形画像の一边サイズ ?		0	px
JPEG画像の圧縮率 ?		95	%

js-adaptive_image

タグに class="js-adaptive_image" を設定する事で、tiny-ランダム文字列.jpg の tiny- の無いファイル名に変換する事ができるようになります。切り換えするウィンドウのサイズについては /js/config.js にある adaptiveImageSize の設定値で指定します。

```
//-----  
// adaptive image sizing  
adaptiveImageMark : 'img.js-adaptive_image',  
adaptiveImageSize : 500,
```

themes/system/include/column.html のファイルをベースに、自分のカスタマイズしているフォルダ内に include/column.html を作成し、<!-- 画像 -->のあたりを以下のように変更します。

```
<!-- BEGIN column#image -->  
<!-- 画像 -->  
<div class="column-image-{align}" ><!-- BEGIN link#front -->  
  <a href="{url}" {viewer} [raw]><!-- END link#front -->  
    <!-- BEGIN link#rear -->  
  </a><!-- END link#rear --><!-- BEGIN caption:veil -->  
  <p class="caption">{caption}</p><!-- END caption:veil -->  
</div>  
<!-- END column#image -->
```


9. Twitter をイロイロ使えるようにする

a-blog cms は Twitter の情報を表示する事ができるようなGETモジュールが5つと、Tweet するためのPOSTモジュールが用意されています。

- 自分のタイムラインを表示 (Api_Twitter_Statues_HomeTimeline)
- 自分のツイートを表示 (Api_Twitter_Statues_UserTimeline)
- リストのメンバーを表示 (Api_Twitter_List_Members)
- リストのタイムラインを表示 (Api_Twitter_List_Statues)
- Twitter上の検索結果を表示 (Api_Twitter_Search)

上記のタイムラインを表示させるようなモジュールを利用する際には、キャッシュを利用するといつまでも古い情報が表示されてしまう事になりますので、ご注意下さい。ここでは、TweetするためのPOSTモジュールの利用について紹介します。

Twitter と連携する際の事前設定

まずは、<https://dev.twitter.com/apps> で、Twitter 側の設定を行います。以下の設定を行うと **Consumer key** と **Consumer secret** を取得できます。

Application Name

今回登録するtwitterのアプリ名。任意の名前で構いませんが、同じ名前のを複数登録はできません。サイト（ブログ）名など、固有のものを使う事をおすすめします。

Description

このアプリケーションの説明文です。どのサイトに使っているかなどの説明を残しておく、アプリケーション一覧で確認できます。最低10字以上の説明文が必要です。

Application Website

a-blog cms をインストールしてるサイトのURLです。基本的には子ブログと共用はできませんので、必要に応じて実際にTwitter 連携を使用する場所のURLを入力してください。

Organization

アプリケーション制作者の所属欄です。この項目は任意入力のため、入力しなくても構いません。

Application Type

アプリケーションの形式です。「Browser」を選択してください。

Callback URL

<http://設置先URL/callback/twitter.html> のように指定して下さい。

Default Access type

基本のアクセス形式です。「Read & Write」を選択してください。

コンフィグの設定

Twitter 側で用意された **Consumer key** と **Consumer secret** を [管理ページ](#) > [カスタマイズ管理](#) > [コンフィグ](#) > [ブログ](#) > [プロパティ設定](#) > [ウェブサービス](#) に設定をする項目が用意されています。

ウェブサービス

Google Maps API key ?	<input type="text"/>
YahooアプリケーションID ?	<input type="text"/>
Twitter アプリケーション ?	Consumer Key <input type="text"/>
	Consumer Secret <input type="text"/>

次に、管理ページ > カスタマイズ管理 > コンフィグ > ブログ > 外部認証設定 より、(OAuth 認証を行う) ボタンを押して認証処理を行います。

外部認証設定 ショートカットに追加 +

Twitter OAuth

認証 [?](#) OAuth認証を行う 認証キーの直接入力
※ この設定は、ブログ単位で行われます。ルールを利用した個別の設定はできません。

最終的には、以下のような画面が表示されれば Twitter 関連の下準備が完了した事になります。

外部認証設定 ショートカットに追加 +

Twitter OAuth

認証 [?](#) 山本一道@アップルuppul : kazumich で認証済み (API 残回数 281 / 350) 認証情報を破棄

ログインしているだけで Tweet できるようにする

Twitter と連携する際の事前設定 を完了している必要があります。

以下のようなフォームを追加する事で、a-blog cms にログインする事で、Twitter 側のログイン情報を知らない人でも a-blog cms のユーザーとしてログインできれば Tweet する事が可能になります。

```
<form action="" method="post">
<input type="text" name="tweet" value="" />
<input type="hidden" name="twitter[]" value="tweet" />
<input type="submit" name="ACMS_POST_Api_Twitter_Statuses_Update" value="ツイート" />
</form>
```

設置先としては、/themes/system/admin/action.html を /themes/blog/admin/action.html にコピーして下さい。そうする事で (エントリー作成) や (Logout) ボタンのあたりにフォームを追加する事ができます。

公開時に Tweet する

Twitter と連携する際の事前設定 を完了している必要があります。

多くの場合には、RSS情報と連携して Twitter 上に新規のエントリーが追加された事を Tweet するようなサービスを使いますが、a-blog cms では以下のようにカスタマイズする事で直接 Tweet できるようになります。

```
<form action="" method="post" class="adminBtn">
<input type="submit" name="ACMS_POST_Api_Twitter_Entry_Open" value="ツイートして公開" />
```

```
<input type="text" name="tweet" value="更新しました: {status.title} ( {status.url} )" style="width:300px;">
<input type="hidden" name="bid" value="{bid}" />
<input type="hidden" name="cid" value="{cid}" />
<input type="hidden" name="eid" value="{eid}" />
</form>
```

設置先としては、 /themes/system/admin/entry/action.html を /themes/blog/admin/entry/action.html にコピーして下さい。そうする事で（公開）ボタンのあたりにフォームを追加する事ができます。また、 /themes/sample@blog/admin /entry/action.html にカスタマイズ済みのファイルがありますので、それをコピーするだけでも大丈夫です。

この機能を利用する事で、Twitter 上で利用しているアプリケーション名や、そのURLを表示する事ができる場所では、ブログ名とそのURLが利用されるようになります。

10.Instagram の写真を表示する



スマートフォンで人気の写真アプリ「instagram」で撮影した画像を a-blog cms で利用する事ができるモジュールが 1.5.0 から用意されています。

- Instagram最近の投稿 (Api_Instagram_Users_Media_Recent)
- Instagram最近のLiked (Api_Instagram_Users_Media_Liked)

自分が投稿した写真と、自分がいいね！とした写真を表示させるモジュールがあります。

Instagram と連携する際の事前設定

まずは、<http://instagr.am/developer/manage/> で、Instagram 側の設定を行います。以下の設定を行うと CLIENT ID と CLIENT SECRET を取得できます。

Application Name

今回登録する Instagram のアプリ名を任意の名前で設定します。

Description

このアプリケーションの説明文です。どのサイトに使っているかなどの説明を残しておく、アプリケーション一覧で確認できます。

Website

a-blog cms をインストールしてるサイトの URL です。基本的には子ブログと共用はできませんので、必要に応じて実際に Instagram 連携を使用する場所の URL を入力してください。

OAuth redirect_uri

<http://設置先URL/callback/twitter.html> のように指定して下さい。以下の **カスタマイズ管理 > コンフィグ > ブログ 外部認証設定** に書かれているアドレスを書きます。

コンフィグの設定

Instagram 側で用意された CLIENT ID と CLIENT SECRET を **管理ページ > カスタマイズ管理 > コンフィグ > ブログ > プロパティ設定 > ウェブサービス** に設定をする項目が用意されています。

。

Instagram アプリケーション ?	Client Id <input type="text"/>
	Client Secret <input type="text"/>
	Client Redirect <input type="text"/>

次に、管理ページ > カスタマイズ管理 > コンフィグ > ブログ > 外部認証設定 より、(OAuth 認証を行う) ボタンを押して認証処理を行います。

Instagram OAuth

認証 ?	<p>認証には、Instagramアプリケーションの登録が必要です。 登録後にプロパティ設定で、取得したコンシューマーキーとシークレットを入力してください。</p> <p>Instagramアプリケーションを登録するときに、下記のURLをコールバックURLに指定してください。 このブログのコールバックURL: http://localhost/callback/instagram.html</p>
------	---

最終的には、以下のような画面が表示されれば Instagram 関連の下準備が完了した事になります。

Instagram OAuth

認証 ?	Kazumich Yamamoto : kazumich で認証済み 認証情報を破棄
------	--

自分の撮影した Instagram の写真を表示する

パソコンやデジカメのデータを加工したりとかが苦手な方でも、iPhone だけでカッコいい感じのフォトギャラリーを用意する事ができます。

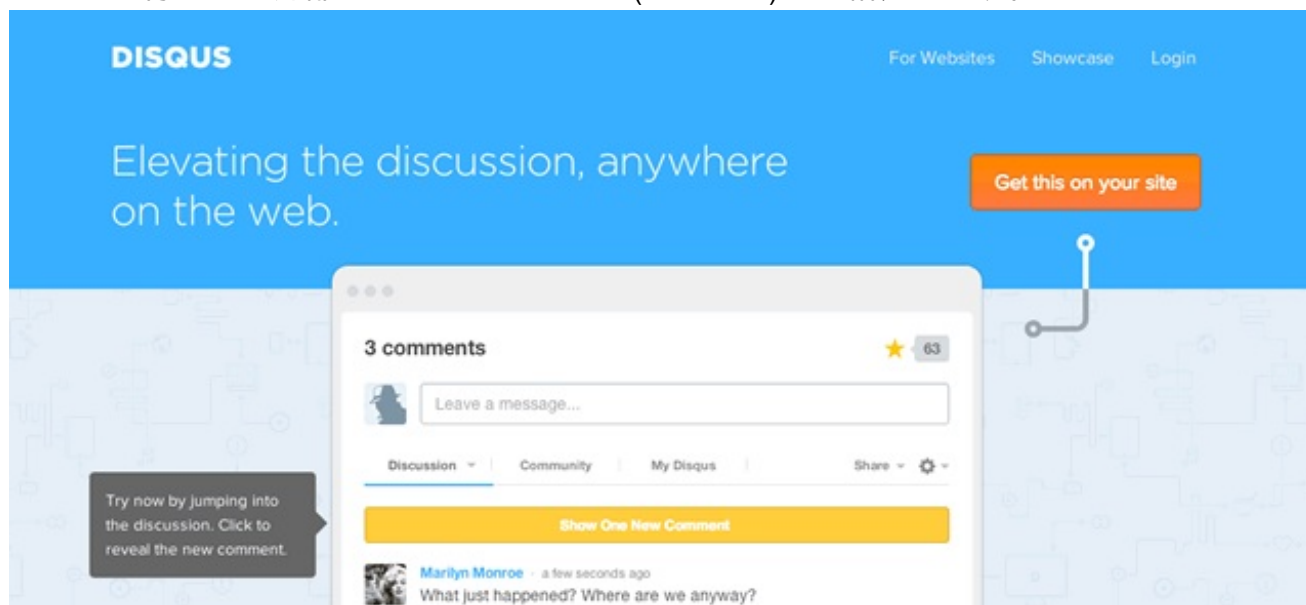
Instagram最近の投稿 (Api_Instagram_Users_Media_Recent) というモジュールがありますので、そちらのスニペットをテンプレートに記述する事で、簡単にフォトギャラリーを用意する事ができるようになります。

```
<!-- BEGIN_MODULE Api_Instagram_Users_Media_Recent -->
<!-- BEGIN photo:loop -->
<div>

<p>{caption}</p>
</div>
<!-- END photo:loop -->
<!-- END_MODULE Api_Instagram_Users_Media_Recent -->
```

11.DISQUS 外部コメントサービスを利用する

a-blog cms 内部のコメントを利用する事で便利な事もありますが、スパム対策等も無い事からコメントに特化した外部のコメントサービス(DISQUS)をご紹介します。



スパム対策はもちろん、Facebook や Twitter でログインしたり、ログインしないでコメントを残す事も自由に設定する事ができます。サイト上でのコメントはもちろんですが、Twitter上でTweetされた事についても BackType という機能で用意されており、エントリーの事がどこかで書かれている際には分かるような仕組みも利用できます。

記述するコード

Entry_Body の entry:loop 内であれば、以下のように記述します。一覧表示時には、表示させないように Touch_Entry というタッチモジュールを利用します。

```
<!-- BEGIN_MODULE Touch_Entry -->
<div id="disqus_thread" style="margin-right:20px"></div>
<script type="text/javascript">
  /* * * CONFIGURATION VARIABLES: EDIT BEFORE PASTING INTO YOUR WEBPAGE * * */
  var disqus_shortname = 'kazumichcom'; // required: replace example with your forum shortname

  // The following are highly recommended additional parameters. Remove the slashes in front
  to use.
  var disqus_identifier = 'entry-{entry:loop.eid}';
  var disqus_url = '{permalink}';

  /* * * DON'T EDIT BELOW THIS LINE * * */
  (function() {
    var dsq = document.createElement('script'); dsq.type = 'text/javascript'; dsq.async =
true;
    dsq.src = 'http://' + disqus_shortname + '.disqus.com/embed.js';
    (document.getElementsByTagName('head')[0] ||
document.getElementsByTagName('body')[0]).appendChild(dsq);
  })();
```

```
</script>
<noscript>Please enable JavaScript to view the <a href="http://disqus.com/?
ref_noscript">comments powered by Disqus.</a></noscript>
<a href="http://disqus.com" class="dsq-brlink">blog comments powered by <span class="logo-
disqus">Disqus</span></a>
<!-- END_MODULE Touch_Entry -->
```

13. 検索結果を Ajax で表示

a-blog cms には Post_include という機能があります。これは、JavaScript や PHP、MySQL の知識も無く、ページのリロード無しにコンテンツを切り換える事ができるようにする仕組みです。まず、検索フォーム部分は以下のように記述します。

```
<form action="" method="post" name="searchForm" id="searchForm" class="js-post_include"
target="#main">
  <input type="text" name="keyword" class="searchText" value="{KEYWORD}" size="15" />
  <input type="text" name="iebug" value="" style="display:none;" class="iebug" />
  <input type="submit" name="ACMS_POST_2GET" class="searchBtn" value="検索" />
  <input type="hidden" name="tpl" value="search.html" />
  <input type="hidden" name="bid" value="{BID}" />
</form>
```

class="js-post_include" が今回の大事な部分で、このクラスをつける事で、このフォームはページ遷移無しに検索結果を表示できるようになります。次に、**target="#main"** です。このターゲット属性は、どの部分を書き換えるのかを指示します。もし、これが無い場合には、<form>タグ全体が置き換えられる事になります。

検索しようとする、ページの遷移無しに次の画面の情報を要求できるようになりました。この次は、どのようなものを表示させるのかテンプレートを指定します。

```
<input type="hidden" name="tpl" value="search.html" />
```

この **tpl** のタグが、次に表示させるテンプレートの指定になります。部分的に表示させるものなので、<html>や<head>タグのようなものは必要無く、検索後に表示したいモジュールだけが書かれているテンプレートファイルを準備して下さい。

```
<div id="main">
  <!-- BEGIN_MODULE Entry_List -->
  <ul>
    <!-- BEGIN entry:loop -->
    <li><a href="{url}">{title}</a></li>
    <!-- END entry:loop -->
  </ul><!-- END_MODULE Entry_List -->
</div>
```

<div id="main">自体が消えてしまうとレイアウトが崩れてしまう事もあると思いますので、search.html には id="main" の DIV も用意しましょう。上記は、簡単なモジュールという事で Entry_List が例に書かれていますが、Entry_Body や Entry_Summary 等を指定してもいいと思います。