

独自テーマ

アプリをインストールする際に選択するオプション

Choose your environment > Customを選択する

:::::::::::::::::::

アプリのディレクトリを作成

app/public/wp-content/themes/new-app-name

:::::::::::::::::::

必須ファイル

```
index.php  
style.css  
functions.php
```

:::::::::::::::::::

テーマの認識

style.cssへ記述する。

```
@charset 'utf-8';
```

元になるSassに認識票みたいなものにつける。私の場合は、_general.sassの先頭行へ記述する。

```
/* Theme Name: general */
```

以上の設定後、ダッシュボード/外観にて独自テーマを確認・有効化をする。

:::::::::::::::::::

設定

- 日本語化
- プラグインの導入
 - WP Multibyte --Patch WordPress 日本語版パッケージのためのマルチバイト機能の拡張をする。
- 設定／検索エンジンでの表示
 - 検索エンジンでの表示 検索エンジンがサイトをインデックスしないようにするにチェックを入れるかどうかは、開発中か公開しているかの状態で決定する。
- 設定／パーマリンク設定

- 投稿ページのURLの付け方を設定する。ここは投稿名にする。
- 投稿を新規追加する毎に、投稿ページのサイドバーにあるURLから日本語になっている場合は適宜英語を割り当てて変更する。

外観／ウィジットの追加

独自テーマでは外観／ウィジットの項目がない。これがないと投稿の設定（右サイドバーの項目）でカテゴリー・タグ・アイキャチ画像の各項目が表示できない。

functions.phpに以下のコードを書いて適宜処理をする。

```
<?php
// ウィジェットの登録
function theme_slug_widgets_init() {
    register_sidebar( array(
        'name' => 'サイドバー', // ウィジェットの名前を入力
        'id' => 'sidebar', // ウィジェットに付けるid名を入力
    ) );
}
add_action( 'widgets_init', 'theme_slug_widgets_init' );
```

aside（サイドバー）の設定

function.phpの編集

functions.php

サイドバーの登録は、`theme_slug_widgets_init()`関数に`register_sidebar()`関数の引数へ配列に値として設定する。`'name'`にはウィジェットに登録される名称を入力する。`'id'`で任意に設定した名称の『ー（ハイフン）』より右側がCSSでのクラス名。HTMLに埋め込む際`get_sidebar("ハイフンより右側の名称")`関数の引数となる。

ウィジェットへの登録

```
function theme_slug_widgets_init() {
    register_sidebar(
        array(
            // ウィジェット上で認識される名前
            'name' => 'sidebarLeft',
            // ハイフンより右側が、
            // HTMLに埋め込む際に関数の引数
            // CSSでのクラス名
            'id' => 'sidebar-left',
        )
    );
}
必要な分だけ『register_sidebar()』関数で定義していく。
register_sidebar(
    array(
```

```
        'name' => 'sidebarRight',
        'id' => 'sidebar-right',
    )
);
}
```

コンポーネントとなるHTML (php) へphpを埋め込む

`dynamic_sidebar()`関数に`functions.php`で設定した`'name'`キーの値を与える。`sidebar-left.php`, `sidebar-right.php`

```
<aside class="sidebar left">
    left side menu
    <ul class="widget">
        // phpを埋め込んだ範囲でWPが適当に構造を作るので
        // その要素に合わせてスタイルをつけていく。
        <?php dynamic_sidebar("sidebarLeft") ?>
    </ul>
</aside>
```

レイアウトするため台紙となるHTML (php) へphpを挿し込む

`get_sidebar()`関数に`functions.php`で設定した`id`の『ー（ハイフン）』より右側を名称を引数として渡す。

front-page.php

```
<div class="wrapper">
    <?php get_sidebar("left"); ?>
    ...
    ...
    <?php get_sidebar("right"); ?>
</div>
```

セキュリティ

- プラグインの`SiteGuard WP Plugin`をインストールする。
- インストールされたら左サイドバーに`SiteGuard`のアイコンが現れるので、そこから設定に入る。
- 管理・制作の負担になる設定項目はチェックを外す。
 - ログインページ変更
 - 画像認証
 - ログインアラート

バックアップ

- プラグインUpdraftPlus WordPress Backup Pluginをインストールする。
- 定期的なバックアップの設定
 - 設定から期間を選択 * n個を設定する。
 - 週 * 4でれば、4週間分のバックアップを取るという意味。

独自テーマの作成

header, main, aside, footerという構成要素があったとして、それらの要素をWordPressの規則に則ってファイルで構成させる。

基礎となるサイトの作成

最低限のコード。

header

スタイルシート設置の宣言

```
<link rel="stylesheet" href="php echo get_stylesheet_uri(); ?&gt;"&gt;</pre
```

header.phpとしているが、要素としてのheadの区切りはここですと宣言するphpの埋め込み

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

全体

```
<!DOCTYPE html>
<html lang="ja">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <link rel="stylesheet" href="php echo get_stylesheet_uri(); ?&gt;"&gt;
  &lt;!-- WordPressから出力されるJavaScript。
       他のphp、プラグインなどを呼び込む機能がある。 --&gt;
  &lt;?php wp_head(); ?&gt;
&lt;/head&gt;
&lt;!-- テンプレートタグは、各ページの種類によって
     自動でclassを割り当ててくれる機能がある。 --&gt;
&lt;body &lt;?php body_class(); ?&gt;&gt;

&lt;header&gt;
  &lt;ul&gt;
    &lt;li&gt;menu1&lt;/li&gt;
    &lt;li&gt;menu2&lt;/li&gt;
    &lt;li&gt;menu3&lt;/li&gt;</pre
```

```
</ul>
</header>
```

main, aside

headerのコンポーネントを呼び出す

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

headerのコンポーネントを呼び出す

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

全体

```
<!-- headerのパーシャルを呼び出す -->
<?php get_header(); ?>

<main>
  <h1>main title</h1>
  <p>hello, word press!</p>
</main>

<aside>side menu</aside>

<!-- footerのパーシャルを呼び出す -->
<?php get_footer(); ?>
```

footer

footer.phpとしているが、要素としてのfooterの区切りはここですと宣言するphpの埋め込み

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

全体

```
<footer>
  <small>footer here!</small>
</footer>
<!-- WordPressから出力されるJavaScript、他のphp、プラグインなどを呼び込む機能がある。 -->
<?php wp_footer(); ?>
</body>
```

```
</html>
```

投稿を表示させる

固定ページへ投稿されている『全て』の概要（インデックス）を出力させる。

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

<main>
    <h1>最新の投稿</h1>
    <ul class="postsList">
        <!-- もし、投稿があれば、投稿が尽きるまでループする。 -->
        <!-- ループしている間は次々にポストを投げ続ける。 -->
        <?php if (have_posts()) : while (have_posts()) : the_post(); ?>
            <li>
                <!-- 記事の本体があるパスを出力する関数 -->
                <a href="<?php the_permalink(); ?>">
                    <!-- 投稿のタイトルを取得する。 -->
                    <h2><?php the_title(); ?></h2>
                    <!-- 投稿日を取得する。datetime属性にも応用してみる。 -->
                    <time datetime="<?php echo get_the_date('Y-m-d'); ?>">
                        <?php echo get_the_date('Y年m月d日'); ?>
                    </time>
                    <?php
                        // the_excerpt()関数で取り出した本文の文字数を変更する。
                        add_filter('excerpt_length', function ($length) {
                            return 180; //表示する文字数
                        }, 999);
                    ?>
                    <!-- 投稿の抜粋を取得する。 -->
                    <p><?php the_excerpt(); ?></p>
                    <!-- 全文を出力する場合はこちらの関数 -->
                    <!-- <p><?php the_content(); ?></p> -->
                </a>
                <!-- この投稿に付与されたカテゴリーを全て取得する -->
                <!-- the_permalink()関数で作るループに入れたらHTML構造を破壊する -->
                <?php the_category(); ?>
            </li>
        <!-- if, whileをクローズさせないといけない。 -->
        <?php endwhile; endif; ?>
    </ul>
</main>

<aside>side menu</aside>

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

WordPressではindex.phpにトップページの内容を書かない。トップページの内容は、front-page.phpという名称のファイルに記述する。

投稿の編集

アイキャッチ画像を編集できる状態にする

ウィジェットで投稿の編集時、デフォルトでは右サイドバーにアイキャッチ画像の項目は無いので追加する。functions.phpに以下のコードを書く。

```
// 追記する。  
// 投稿のサイドバーにアイキャッチ画像を付与。  
add_theme_support('post-thumbnails');
```

投稿にキャッチの画像を紐付ける

投稿のアイキャッチになる画像を挿入する。画像の関連付けは、ダッシュボード/投稿/投稿一覧から各投稿を開いて右サイドバーのアイキャッチ画像で編集する。

投稿のHTMLにimg要素としてレイアウトする

投稿をループしながら、該当する投稿の箇所にthe_post_thumbnail()関数をphpで埋め込む。埋め込まれたphpは、以下のようなクラス名が付与されたimg要素となる。

```
<li>  
  <a href="php the_permalink(); ?&gt;"&gt;<br/    <!-- php ----- -->  
    <?php the_post_thumbnail(); ?>  
    <!-- ----- -->  
    <h2><?php the_title(); ?></h2>  
    <time><?php echo get_the_date('Y年m月d日'); ?></time>  
    <p><?php the_excerpt(); ?></p>  
  </a>  
  <?php the_category(); ?>  
</li>
```

```
.attachment-post-thumbnail  
.size-post-thumbnail  
.wp-post-image"
```

クラス名または、要素名で指定する。

```
a
display: block
// img要素の場合
img
width: 100%
height: 35vw
object-fit: cover
border-radius: 5px
```

投稿のHTMLにbackground属性としてレイアウトする

WPでbackground-image属性を使う際は、HTMLでbackground-image属性を指定し、CSSでその他に必要な指定をする。

```
<li>
<a href=<?php the_permalink(); ?>>
<!-- background-image属性の指定 --&gt;
&lt;div
  class="thumbnail"
  style="background-image:
    url(&lt;? echo wp_get_attachment_url(get_post_thumbnail_id()); ?&gt;)"&gt;
&lt;/div&gt;
&lt;!--
--&gt;
&lt;h2&gt;&lt;?php the_title(); ?&gt;&lt;/h2&gt;
...
...
&lt;/a&gt;
&lt;?php the_category(); ?&gt;
&lt;/li&gt;</pre>
```

CSSへbackground-image属性を指定した際に、残り必須の指定を付け加える。

```
.thumbnail
width: 100%
height: 35vw
background-repeat: no-repeat
background-position: center
background-size: cover
border-radius: 5px
```

固定ページのリンクへ飛ぶ

```
front-page.php
<?php echo home_url('/') ?>
```

```
archive.php  
<?php echo home_url('/archive') ?>  
  
category.php  
<?php echo home_url('/category') ?>
```

HTMLから画像の呼び出し

- CSS, JSへのパスを通す。
- 画像・動画・音楽までのパスを通しHTMLにレイアウトする。

これらをするために、`get_template_directory_uri()`関数を使ってテーマフォルダまでのパスを取得する。

```
<script src="php echo get_template_directory_uri(); ?&gt;/to/js/path"&gt;<br/</script>  
  

```

固定ページと投稿ページ

固定ページ・投稿ページとも設定していない状態では、投稿ページのURL/パーマリンクにあるURLをクリックすると`index.php`へ飛んでしまう。

雛形となるページを編集する。

- 固定ページは => `page.php`
- 投稿ページは => `single.php`

HTML上で投稿記事の領域をクリックすると投稿内容の詳細ページへ飛ぶ仕様になっている。飛んだページ`single.php`のインスタンスが持っている関数を使ってフォーマットを編集する。

ページの全体レイアウトの整理

ページを新規で作る場合、ヘッダーとフッターのパーシャルを埋め込む。

```
<?php get_header(); ?>  
<!-- ここがコンテンツ -->  
<?php get_footer(); ?>
```

中身のレイアウトを作成

投稿のインスタンスにある変数や関数

- \$post
- get_the_post_thumbnail()
- the_title()
- the_content()
- the_category()
- get_the_date('Y年m月d日')
- the_modified_date("Y-m-d")

phpの変数を確認する方法は、

```
<?php var_dump(変数) ?>
```

投稿に紐づいた内容を定義された変数や関数を使ってレイアウトを整える。

```
<main>
  <section>
    <!-- 変数$postに何が入っているか調べる。 -->
    <?php var_dump($post) ?>
    <!-- サムネールを出力する。 -->
    <!-- 投稿された固定ページのID、出力するimg要素に付けるクラス名を引数にする。 -->
    <!-- そのクラス名にはプリフィックスが付与される。 -->
    <!-- この場合は、『.attachment-header-vusual』となる。 -->
    <?php echo get_the_post_thumbnail($post->ID, 'header-vusual'); ?>
    <!-- 投稿データから記事のタイトルを取得する。 -->
    <h1><?php the_title(); ?></h1>
    <div class="contents">
      <p>
        <!-- 投稿データから記事の本文を取得する。 -->
        <?php the_content(); ?>
      </p>
    </div>
  </section>
</main>
```

固定ページでは、.pageクラス、投稿ページでは、.singleクラスが付与される。定義する際には大元にそれらのクラスをつけてから展開していく。なお、固定ページと投稿ページの体裁の共通をスタイルリングさせて効率的にサイトを作成できる。

```
.page, .single
  main
    width: min(800px, 100%)
    margin: 50px auto
    background-color: #eee
    section
      padding: 45px
      .attachment-header-vusual
```

```
width: 100%
height: 60vw
object-fit: cover
h1
  font-size: 20px
  font-weight: 900
  margin: 20px 0
p
  font-size: 15px
  font-weight: 500
  line-height: 1.7
```

以下の要素を出力する関数でレイアウトしたサンプル

- 画像にクラスの付与
- タイトル
- カテゴリー
- 本文
- 最終更新日

```
<main>
  <section>
    <?php echo get_the_post_thumbnail($post->ID, 'header-vusual'); ?>
    <h1><?php the_title(); ?></h1>
    <!-- カテゴリーを出力する関数を挿入する。----- -->
    <?php the_category(); ?>
    <!-- ----- -->
    <div class="contents">
      <p>
        <?php the_content(); ?>
      </p>
      <p class="lastUpdate">
        <!-- 最終更新日を出力する関数を挿入する。----- -->
        最終更新日:<time datetime="<?php the_modified_date("Y-m-d"); ?>">
          <?php the_modified_date("Y/m/d"); ?>
        </time>
        <!-- ----- -->
      </p>
    </div>
  </section>
</main>
```

JavaScriptの読み込み

任意の階層にjsを配置して読み込ませるには、functions.phpに以下のコードを書く。

```
<?php
function my_script() {
```

```
wp_enqueue_script(
    // 重複しないよう呼び込むスクリプトの名称をつける
    'myscript',
    // 呼び出す関数までのパスを読み込む
    get_template_directory_uri().'/assets/js/behavior.js',
    // このjsを読み込む前に取り込んでおきたいjsがあればこの配列にパスを代入しておく
    array(),
    // true=> versionを出力する
    false,
    // true=>footerでjsを読み込む
    // false=>headerでjsを読み込む
    // HTMLが読まれてからjs読み込むのが吉なので、
    true
);
}

add_action('wp_enqueue_scripts', 'my_script');
```

独自テーマでカスタムメニューを設定・表示させる

メニューの登録

HTML

```
<header>
<?php
wp_nav_menu(array(
    'theme_location' => 'global'
))
?>
</header>
```

functions.php

```
register_nav_menus(array(
    'global' => 'グローバルメニュー',
    'header' => 'ヘッダーメニュー',
    'footer' => 'フッターメニュー',
));
```

ダッシュボード／外観／メニューから設定する。

固定ページを作る

WEBサイトではホーム（WPではfront-page.php）と複数コンテンツを有するページ（work, about, info, contact etc...）で構成される。WPは、これらのルーティングを統括する機能を持つ。編集は、ダッシュ

ュボード／固定ページで行こなう。なお、archive.php, category.php, header.php, footer.php, single.phpはデフォルトでルーティングは解決されている。

固定ページをpage.phpという名称で作成してみる。

- page.phpファイルを作る。
- <?php /* Template Name: page */ ?>を冒頭につける。
 - これによりWPダッシュボード／固定ページでページテンプレートとして登録される。
- <?php the_content(); ?>としておく。ここが重要。
- 例えば、新たにinfo.phpという名称で固定ページを作成して、ページテンプレートでpageを選ぶと、ヘッダーとフッターはpage.phpを反映し中身はinfo.phpで記述した内容が反映されたページが出来上がる。

```
<?php /* Template Name: page */ ?>
// ヘッダー
<?php get_header(); ?>
// workのコンテンツ本体
// ここにHTMLを書き込んでいく。
<div class="container">
    <?php the_content(); ?>
</div>
// フッター
<?php get_footer(); ?>
```

- WPに認識させる。
 - WP／ダッシュボード／固定ページ／から新規作成。
 - タイトルには任意（日本語可）につける。WP内で管理する名称になる。
 - 固定ページ編集サイドバーからテンプレートを選択。ダイアログのプルダウン・メニューにファイル作成時につけた<?php /* Template Name: page */ ?>のpageがリストされるはず。
 - プルダウン・メニューからpageを選択。
 - 公開をクリックしてWPに設定したパーマリンクがファイルと紐付けられる。

::::::::::::::

スカスカのページのフッターを底ベタさせる方法

```
.page, .single
position: relative
// min-heightが肝
min-height: 100vh // footerの高さ
padding-bottom: 100px // footerの高さ
footer
    position: absolute
    bottom: 0
    left: 0
    width: 100%
    height: 100px // ここに注意
    color: #fff
    background-color: #333
```

```
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

投稿の一覧ページ・カテゴリーに紐づく投稿を表示させる。

投稿一覧、カテゴリーに紐づいた投稿は、**default**でファイル名が決まっている。

- archive.php
- category.php

一覧ページを表示

まずは、投稿一覧ページを作成する。

留意点として、**function.php**での設定は、投稿の一覧ページやカテゴリー一覧ページの生成とは **何の関係もない**ということ。

function.php

```
// post_has_archive()関数の定義
function post_has_archive($args, $post_type) {
    if ('post' == $post_type) {
        $args['rewrite'] = true;
        // 混乱するので、アーカイブに対するスラッグ名(url)は
        // ファイル名と同じにする。
        $args['has_archive'] = 'archive';
    }
    return $args;
}
add_filter('register_post_type_args', 'post_has_archive', 10, 2);
```

archive.php

ページネーションをつけて投稿一覧を出力するやり方。フォーマットで持っておいてコピペすべし。

```
<?php get_header(); ?>
<div class="container">
    <div class="contents">
        <h1>投稿一覧</h1>
        <dl class="posts-list">
            // get_query_var()関数に'paged'という引数を渡すと
            // ページ数が数値で返ってくる。
            // ただ、最初は『0』が入るので、それを三項演算子を使い『1』に変更する。
            // get_query_var('paged')がtrueだったらget_query_var('paged')を代入。
            // get_query_var('paged')がfalseだったら『1』を代入する。
            $recent_page = get_query_var('paged') ? get_query_var('paged') : 1;

            // phpはここから埋め込まれ -----
            <?php
                // このページ内で表示する投稿の数を指定する。この場合は『3件』
                $args = array('posts_per_page' => 3);
                // WPのDBからのインスタンスに『設定した制限数』を引数として入れる。
                $my_query = new WP_Query($args);
```

```

    // 3件の投稿を持っていたら、3件を最後まで回す。
    if ($my_query->have_posts()) : while ($my_query->have_posts()) :
$my_query->the_post(); ?>
    // phpはここまで埋め込まれている。-----

        // the_permalink()関数を呼んだら、現在の投稿のリンクを出力して、
<a href="php the_permalink(); ?&gt;"&gt;
        // the_title()関数を呼んだらタイトルを出力して、
&lt;dt&gt;&lt;?php the_title(); ?&gt;&lt;/dt&gt;
        // the_excerpt()関数を呼んだら本文を出力する。
&lt;dd&gt;&lt;?php the_excerpt(); ?&gt;&lt;/dd&gt;
&lt;/a&gt;
&lt;?php endwhile; endif; ?&gt;
&lt;/dl&gt;

// ページネーションのリストを生成してくれる。
// 必要な要素はWPが書き出すので、それに合わせてスタイルをつけていくこと。
&lt;?php
$args = array(
    'type' =&gt; 'list',
    'current' =&gt; $recent_page,
    'total' =&gt; $my_query-&gt;max_num_pages,
    'prev_text' =&gt; '&lt;',
    'next_text' =&gt; '&gt;'
);
echo paginate_links($args);
?&gt;
&lt;/div&gt;
&lt;/div&gt;
&lt;?php get_footer(); ?&gt;
</pre

```

中の肝心な箇所で使われるコードの意味を解説しておく。三項演算子を理解するためのサンプルコード この原理を使った三項演算子での解決。

```

<?php
if (get_query_var('paged')) {
    var_dump(get_query_var('paged'));
    var_dump('数字が入っているだからtrue');
} else {
    var_dump(get_query_var('paged'));
    var_dump('数字が0だからfalse');
}
?>

```

カテゴリーに紐づく投稿

archive.phpの一部だけ改造してやる。

category.php

```

<div class="container">
  <div class="contents">
    <!-- 投稿の一覧ページの`カテゴリー`をクリックするとこのページに飛ぶようにデフォルトで紐づけられている。
      このページ自体が『選択した`カテゴリー`』の`インスタンス`である、それに対する関数を`WP`は定義している。
      それらを使って欲しい情報を適宜出力してページにレイアウトすると考える。 -->
      <!-- 『single_cat_title()』関数でカテゴリー名を出力する。 -->
      <h1><?php single_cat_title(); ?>カテゴリーの一覧</h1>
      <dl>
        <?php // ページ番号を取得する。1ページ目は『0』になるので条件節で変換する。
        $recent_page = get_query_var('paged') ? get_query_var('paged') : 1;
        // 現在カテゴリーの情報を『get_the_category()』関数で引き出す。
        $category = get_the_category();
        // $categoryはオブジェクトで値は一つだけなのでインデックス[0]と特定して呼び出す。
        // キー『slug(url)』に紐づいた値=現在いるカテゴリーの名称を引き出すという流れ。
        // var_dump($category[0]->slug);
        // var_dump($category[0]);
        $args = array(
          // これはカスタム投稿じゃないからデフォルトのまま。なお省略可能。
          // 'post_type' => 'post',
          // 現在のカテゴリー(オブジェクト)をslug(url)(キー)を参照して値を変数に格納する。
          'category_name' => $category[0]->slug,
          // 『-1』は、あるだけ全部出してという意味
          'posts_per_page' => -1,
          // このインスタンスで定義された変数。現在のカテゴリーに対して現在のページ番号が格納されている。
          'paged' => $recent_page
          // オプションが多数あり、下記参照。
          // 'orderby' =>
        );
        $my_query = new WP_Query($args); ?>

        <?php if ($my_query->have_posts()) : while ($my_query->have_posts())
        : $my_query->the_post(); ?>
          <a href="php the_permalink(); ?&gt;"&gt;
            &lt;dt&gt;&lt;?php the_title(); ?&gt;&lt;/dt&gt;
            &lt;dd&gt;&lt;?php the_excerpt(); ?&gt;&lt;/dd&gt;
          &lt;/a&gt;
        &lt;?php endwhile; ?&gt;
        &lt;?php endif; ?&gt;
      &lt;/dl&gt;

      &lt;!-- パンくずリスト --&gt;
      &lt;!-- このリストもWPが構造を書き出す。それに合わせてスタイルをつける。 --&gt;
      &lt;?php
        $args = array(
          'type' =&gt; 'list',
          'current' =&gt; $recent_page,
          // 現在のインスタンス、つまり全ページの総数を変数に格納している。
          'total' =&gt; $my_query-&gt;max_num_pages,
          // 記号は適宜
          'prev_text' =&gt; '&lt;',
</pre

```

```

    'next_text' => '>
');
// 『paginate_links()』関数の1行でパンくずリストを出力してる。
echo paginate_links($args);
?>
</div>
</div>

```

the_category()関数で留意すべき点

WPのthe_permalink()関数は、HTML要素を新たに生成させて構造を作る。the_category()関数を入れ子にすると、the_permalink()関数で生成した構造を破壊してしまう。the_category()関数からの出力を配列に変換してして出力して回避する。

```

<ul class="post-archive">
<?php
$args = array('posts_per_page' => 3);
$my_query = new WP_Query($args);
if ($my_query->have_posts()) : while ($my_query->have_posts()) :
$my_query->the_post();
?>
<li>
<a href="<?php the_permalink(); ?>">
<div class="frame">
<?php the_post_thumbnail(); ?>
</div>
<div class="header-sub">
<ul class="post-categorie">
// この部分 /////////////////////////////////
<?php
$category = get_the_category();
// name属性をキーにして値を取り出す。
// 文字列になったカテゴリー名を渡す。
foreach ($category as $attr) {
    echo '<li>' . $attr->name . '</li>';
}
?>
// ↑ ここまで /////////////////////////////////
</ul>
<time datetime="<?php echo get_the_date("Y-m-d") ?>"><?php
echo get_the_date("Y年m月d日") ?></time>
</div>
<h4 class="shrinkLine"><?php the_title(); ?></h4>
<p><?php the_excerpt(); ?></p>
<p><?php the_category(); ?></p>
</a>
</li>
<?php endwhile; ?>
<?php endif; ?>
</ul>

```

'offset'=>1と投稿インデックスの文字数制限

最初のページをスキップするオプション

```
<ul class="post-archive">
<?php
$recent_page = get_query_var('paged') ? get_query_var('paged') : 1;
$args = array(
    // defaultでは'post'。例えば'mesg'という名称のカスタム投稿を追加した場合は、
    // 'post_type' => 'mesg'として宣言する。
    'post_type' => 'post',
    'posts_per_page' => 3,
    'paged' => $recent_page,
    'offset' => 1
);
$my_query = new WP_Query($args);
?>
<?php if ($my_query->have_posts()) : ?>
<?php while ($my_query->have_posts()) : ?>
<?php $my_query->the_post(); ?>
<li>
<a href="<?php the_permalink(); ?>">
<div class="frame">
<?php the_post_thumbnail(); ?>
</div>
<div class="header-sub">
<ul class="post-categorie">
<?php
$category = get_the_category();
foreach ($category as $attr) {
    echo '<li>' . $attr->name . '</li>';
}
?>
</ul>
<time datetime="<?php echo get_the_date("Y-m-d") ?>"><?php
echo get_the_date("Y.m.d") ?></time>
</div>
<h4 class="shrinkLine"><?php the_title(); ?></h4>
<?php
add_filter('excerpt_length', function ($length) {
    return 50; //表示する文字数
}, 999);
?>
<p><?php the_excerpt(); ?></p>
</a>
</li>
<?php endwhile; ?>
<?php endif; ?>
</ul>
```

⋮

カスタム投稿

- ・ プラグインCustom Post Type UIを使用する。
- ・ ダッシュボード／CPT UI／新規投稿タイプを追加する。
 - 投稿タイプスラッグ => slugと紐づくので英語で記入する。
 - 複数形ラベル ダッシュボード内での管理名 日本語で記入する。 単数形ラベル ダッシュボード内での管理名 日本語で記入する。
- ・ 投稿タイプを追加ボタンをクリックする。
- ・ ダッシュボードに複数形ラベルで付けた名称の項目が追加される。
- ・ 該当項目をクリックして編集ページへ
- ・ ページを追加するため、新規追加ボタンを押してページ編集画面へ。
- ・ タイトル・本文を入れて公開する。
- ・ ダッシュボードの項目CTP UIからタクソノミーの追加と編集をクリックしタクソノミーの編集タブに入る。
- ・ 選択のプルダウンメニューから複数形ラベルで付けた名称の項目を選択する。
- ・ 管理画面が便利になるので、追加ラベルの欄で以下の項目をTrueにする。
 - 管理画面でカラムを表示 => true
 - クイック編集/一括編集パネルに表示 => true
- ・ 複数形ラベルで付けた名称の項目のカスタムページを作成する。

```

<?php
$args = array(
    // 投稿タイプスラッグ => slugと紐づくので英語で記入するして名称を登録する。
    'post_type' => 'meeting',
    'posts_per_page' => 3
);
// 変数名をユニークにしておく。
$meeting_query = new WP_Query($args)
?>

<ul>
    // 投稿をカスタム投稿に限定する設定。
    <?php if ($meeting_query->have_posts()) : while ($meeting_query->have_posts()) : $meeting_query->the_post(); ?>
        <li>
            <a href="php the_permalink(); ?"&gt;"&gt;
                &lt;h2&gt;&lt;?php the_title(); ?&gt;&lt;/h2&gt;
                &lt;p&gt;&lt;?php the_content(); ?&gt;&lt;/p&gt;
            &lt;/a&gt;
        &lt;/li&gt;
    &lt;?php endwhile;
    endif; ?&gt;
&lt;/ul&gt;
</pre

```

::

ページを作成してURLを付けWPDBへ認識させる

- ・ 『page-ページ名称（英語）.php』でページのファイルを作成。
- ・ ダッシュボード／固定ページ／新規追加でページのインスタンスをWEBページ上で運用する名称（日本語可能）で生成する。


```

// 特定の「カテゴリー」に関連付けられた投稿を表示する場合
'cat' => 5, // カテゴリーIDを指定
'category_name' => 'daily, news', // カテゴリースラッグを指定(複数の場合
は「,」で区切る)
'category_and' => array(2, 6), // カテゴリーIDを配列で指定(カテゴリーID
を含む記事を絞り込む)
'category_in' => array(2, 6), // カテゴリーIDを配列で指定(カテゴリーID
を含む記事を絞り込む)
'category_not_in' => array(2, 6), // カテゴリーIDを配列で指定(カテゴリーID
を含まない記事を絞り込む)

// 特定の「タグ」に関連付けられた投稿を表示する場合
'tag' => 'cooking', // タグスラッグを指定
'tag_id' => 5, // タグIDを指定
'tag_and' => array(2, 6), // タグIDを配列で指定(タグIDを含む記事
を絞り込む)
'tag_in' => array(2, 6), // タグIDを配列で指定(タグIDを含む記事
を絞り込む)
'tag_not_in' => array(2, 6), // タグIDを配列で指定(タグIDを含まない記
事を絞り込む)
'tag_slug_and' => array('red', 'blue'), // タグスラッグを配列で指定(タグスラッグ
を含む記事を絞り込む)
'tag_slug_in' => array('red', 'blue'), // タグスラッグを配列で指定(タグスラッグ
を含む記事を絞り込む)

// 特定の「タクソノミー」に関連付けられた投稿を表示する場合(以下は複数のタクソノミーにてAND検
索)
'tax_query' => array( // タクソノミーパラメーターを指定
  'relation' => 'AND', // タクソノミーの検索条件に 'AND' か
'OR' が使用可能
  array(
    'taxonomy' => 'color', // タクソノミーを指定
    'field' => 'slug', // term_id(デフォルト), name, slug のい
すれかのタームの種類を選択
    'terms' => array('red', 'blue'), // ターム(文字列かIDを指定)
    'include_children' => true, // 階層を持つタクソノミーの場合に、子孫タク
ソノミーを含めるかどうか
    'operator' => 'IN' // 演算子'IN', 'NOT
IN', 'AND', 'EXISTS'(4.1.0以降), 'NOT EXISTS'(4.1.0以降)が利用可能
  ),
  array(
    'taxonomy' => 'actor',
    'field' => 'id',
    'terms' => array(103, 115, 206),
    'include_children' => false,
    'operator' => 'NOT IN'
  )
),
),

// 特定の「投稿&固定ページ」に関連付けられた投稿を表示する場合
'p' => 1, // 投稿IDを指定
'name' => 'hello-world', // 投稿スラッグを指定
'page_id' => 1, // 固定ページのIDを指定
'pagename' => 'sample-page', // ページスラッグを指定

```

```

'pagename' => 'contact_us/canada',           // 子ページを表示する場合、スラッシュ
区切りで親と子のスラッグを指定
'post_parent' => 1,                          // ページIDを指定した子ページを表示
'post_parent_in' => array(1, 2, 3),          // 配列の親ページIDを含む投稿を表示
'post_parent_not_in' => array(1, 2, 3),       // 配列の親ページIDを含まない投稿を表
示
'post_in' => array(1, 2, 3),                  // 配列の投稿IDを含む投稿を表示
'post_not_in' => array(1, 2, 3),              // 配列の投稿IDを含まない投稿を表示

// 特定の「パスワード」に関連付けられた投稿を表示する場合
'has_password' => true,                      // パスワード付きの投稿を表示( true
or false )
'post_password' => 'zxcvbn',                 // 特定のパスワードが付いた投稿を表
示

// 特定の「タイプ」に関連付けられた投稿を表示する場合
'post_type' => array(
    'post',                                // 投稿
    'page',                                // 固定ページ
    'revision',                             // リビジョン
    'attachment',                           // 添付ファイル
    'custom-post-type'                     // カスタム投稿タイプ
),
'post_type' => 'any', // すべてのタイプを含めて表示(リビジョン
と'exclude_from_search'がtrueにセットされたものを除く)

// 特定の「投稿ステータス」に関連付けられた投稿を表示する場合
'post_status' => array( // 投稿ステータスを指定 (デフォルト'publish')
    'publish',                            // 公開された投稿、または固定ページを表示
    'pending',                            // レビュー待ちの投稿を表示
    'draft',                              // 下書きの投稿を表示
    'auto-draft',                         // コンテンツのない、新しく作成された投稿を表示
    'future',                             // 予約公開設定された投稿を表示
    'private',                            // ログインしていないユーザーには見えない投稿を表示
    'inherit',                            // リビジョンを表示
    'trash',                              // ゴミ箱に入った投稿を表示
),
'post_status' => 'any', // すべてのステータスを表示(投稿タイプ
で'exclude_from_search'がtrueにセットされたものを除く)

// ページ送りパラメーターを設定する場合
'posts_per_page' => 10,                   // 1ページあたりに表示する投稿数を指定(-1を指
定するとすべての投稿を表示)
'posts_per_archive_page' => 10,           // 1ページあたりに表示する投稿数(アーカイブペー
ジのみ)
'nopaging' => false,                     // ページ送りを使用するか、すべての投稿を表示す
るか、(デフォルトはfalseでページ送りを使用)
'paged' => 6,                           // ページ番号6の記事を表示
'paged' => get_query_var('paged'), // 現在のページから投稿を表示
'offset' => 3,                           // 設定した数だけ、ずらして表示(例では4番目の投
稿から表示)
'ignore_sticky_posts' => false,         // 先頭固定表示投稿を無視するかどうか(デフォルト
値は0で先頭固定表示投稿を無視しない)

// 「投稿の並び順」を指定する場合

```

```

'order' => 'DESC',      // 'ASC' 昇順 (1, 2, 3; a, b, c)
// 'DESC' 降順 (3, 2, 1; c, b, a)

'orderby' => 'date',   // デフォルト値'date' 複数のオプションを渡すことが可能
// 例:'orderby' => 'menu_order title'
// その他のオプション ↓
///'none'      並び替えなし
///'ID'        投稿IDで並び替え
///'author'    著者で並び替え
///'title'     タイトルで並び替え
///'name'      Order by post name(post slug)
///'modified'  更新日で並び替え
///'parent'    親ページIDで並び替え
///'rand'      ランダム順
///'comment_count' コメント数で並び替え
///'menu_order' ページの表示順で並び替え
///'meta_value' アルファベット順で並び替え(数値ではうまくいかない)
///'meta_value_num' 数値で並び替え
///'post_in'    post_inで配列で指定された投稿IDの並び順を維持して表示

// 特定の「時間や日付の期間」に関連付けられた投稿を表示する場合
'year' => 2015,          // 4桁の年を数字で指定(2015など)
'monthnum' => 4,         // 月を数字で指定( 1~12 )
'w' => 25,                // 年内の週を数字で指定( 0~53 )
'day' => 17,               // 月内の日を数字で指定( 1~31 )
'hour' => 13,              // 時間を数字で指定( 0~23 )
'minute' => 19,             // 分を数字で指定( 0~60 )
'second' => 30,             // 秒を数字で指定( 0~60 )
'm' => 201404,             // 年と月を数字で指定 ( 201508など )

// 「○年○月○日から○年○月○日の範囲の投稿情報」を表示する場合(投稿日の検索が自由自在!)
'date_query' => array(
    array(
        'year' => 2015,           // 4桁の年を数字で指定(2015など)
        'month' => 8,            // 月を数字で指定( 1~12 )
        'week' => 31,             // 年内の週を数字で指定( 0~53 )
        'day' => 5,               // 月内の日を数字で指定( 1~31 )
        'hour' => 2,              // 時間を数字で指定( 0~23 )
        'minute' => 3,             // 分を数字で指定( 0~60 )
        'second' => 36,             // 秒を数字で指定( 0~60 )
        'after' => 'January 1st, 2013', // 指定した日付以降の投稿を取得。
        strtotime()と互換性のある文字列で'after'=>'2015/08/31'などでもOK
        'before' => array(           // 指定した日付以前の投稿を取得。
        strtotime()と互換性のある文字列で'before'=>'2015/08/31'などでもOK
            'year' => 2013,           // 4桁の年を数字で指定(2015など) デフォルトは空
            'month' => 2,            // 年内の月を数字で指定( 1~12 ) デフォルトは12
            'day' => 28,              // 月内の日を数字で指定( 1~31 ) デフォルトは月内
            の末日
        ),
        'inclusive' => true,        //「after」または「before」パラメーターで指定さ
        れた値を含むかどうか
        'compare' => '=',           // 使用可能な値は '=' , '!=', '>', '>=' ,
        '<', '<=' , 'LIKE' , 'NOT LIKE' , 'IN' , 'NOT IN' , 'BETWEEN' , 'NOT BETWEEN' ,
        'EXISTS' , and 'NOT EXISTS'
        'column' => 'post_date',       // 照会するカラムを指定。デフォルトは
    )
)

```

```

「post_date」
    'relation' => 'AND', // OR または AND デフォルトは「AND」
),
),

// 特定の「カスタムフィールド」に関連付けられた投稿を表示する場合
'meta_key' => 'key', // カスタムフィールドのキーを指定
'meta_value' => 'value', // カスタムフィールドの値を指定
'meta_value_num' => 10, // カスタムフィールドの値を指定
'meta_compare' => '=', // 「meta_value」をテストする演算子。使える値
は'!=', '>', '>=' , '<', '<=' デフォルト値は'='
'meta_query' => array( // カスタムフィールドパラメーター
    'relation' => 'AND', // 「AND」または「OR」を指定。meta_query内の配列が「2つ以上」の場合に限る。meta_query配列が1つの場合は使用しない。
    array(
        'key' => 'color', // カスタムフィールドのキー。
        'value' => 'blue', // カスタムフィールドの値（注意 compareの値が'IN'、'NOT IN'、'BETWEEN'、'NOT BETWEEN'のみ配列をサポート）
        'type' => 'CHAR', // カスタムフィールドタイプ。タイプについては以下の
「meta_queryで使えるデータ型」参照
        'compare' => '=', // 演算子を指定 デフォルト値は'=' 演算子の種類については以下
下「meta_queryで指定できる演算子の種類」参照
    ),
    array(
        'key' => 'price',
        'value' => array(1, 200),
        'compare' => 'NOT LIKE',
    )
),
),

// 適切な権限を持っているユーザーのプライベートの記事を表示する場合
'perm' => 'readable', // 使える値は'readable'と'editable'

// キャッシュ系のパラメーター
'cache_results' => true, // 投稿情報をキャッシュするかどうか デフォルトは
true
'update_post_term_cache' => true, // 投稿タームキャッシュを更新するかどうか デフォ
ルトはtrue
'update_post_meta_cache' => true, // 投稿メタキャッシュを更新するかどうか デフォルト
はtrue
'no_found_rows' => false, // カウントをスキップする? trueでパフォーマンス
が向上する可能性があるかも デフォルトはfalse

// 検索系のパラメーター
's' => $s, // 検索からクエリーストリング値を渡します。
'exact' => true, // タイトル／投稿の全体から正確なキーワードで検索するか デフォルト値は
false
'sentence' => true, // 語句(フレーズ検索)で検索するか デフォルト値はfalse

// 投稿フィールドパラメーター
'fields' => 'ids' // 1つのフィールドで返すか全てのフィールドで返すか デフォルトでは全てのフ
ィールドが返される
// 使用できる値
// 'ids' 投稿のIDの配列を返します
// 'id=>parent' 連想配列を返します

```

```
);

$query = new WP_Query($args);

if ($the_query->have_posts()) :
    while ($the_query->have_posts()) : $the_query->the_post();
        // 何かしらの処理
    endwhile;
endif;

// 投稿データのリセット
wp_reset_query();
?>
```

構文

```
<!-- 宣言 -->
<!-- ここから始まる、ここで終わる宣言が必要。 -->
<?php ?>

<!-- 文字を出力する。要素も出力できる。 -->
<p><?php echo 'hello world!' ?></p>

<!-- 変数を使って文字列を出力する。 -->
<?php $str = 'WordPress'; ?>
<p><?php echo $str; ?></p>

<!-- 文字列を結合する。 -->
<?php
    $text = 'hello, ';
    $str = 'world!';
?
<p><?php echo $text . $str; ?></p>

<!-- 配列 -->
<?php
$arr1 = array('晴れ', '曇り', '雨');
$arr2 = ['晴れ', '曇り', '雨'];
var_dump($arr2);
echo $arr1[0] . '<br>';
foreach ($arr1 as $key => $val) {
    echo $val . '<br>';
}
?>

<!-- 配列はechoできない。var_dump()で確認。 -->
<?php var_dump($arr2); ?>

<!-- if条件文 -->
<?php
$a = 2;
```

```
$b = 1;
if ($a > $b) {
    echo 'aはbより大きい';
} elseif ($a === $b) {
    echo 'aはbと同じ';
} else {
    echo 'aはbより小さい';
}
?>

<!-- 関数 -->
<!-- 関数を定義してみる -->
<?php
function calc($a, $b) {
    echo '<p>' . ($a + $b) . '</p>';
}
calc(5, 20);
?>
```