

WordPress 実習課題 01

作成するサイトの説明

今回は、典型的なブログサイトを作成します。作成するサイトは、宇宙についてのトピックを扱うブログサイト「Cosmic Blog」というサイトです。

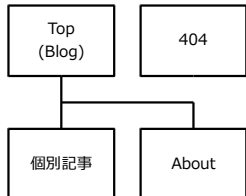


サイトに対する要件

- 基本的なブログ機能を備える。
 - 管理画面での記事の管理
 - 記事に対するコメントの投稿
 - カテゴリーや年月による記事の分類
 - 新着の投稿やコメントの表示
- サイト紹介のページを入れる。
- Twitter、Facebook 等のソーシャルボタンを設置する。
- スマートフォン向けの画面も用意する（オリジナルデザインでなくてよい）。

サイト構成図

このサイトの構成図を以下に示します。



1. 「Top」ページ

このページはサイトのトップページです。このページでは、ブログの記事のリストを表示します。また、サイドバーには、ブログ内検索ボックス、最近の投稿の一覧、最近のコメントの一覧、月別アーカイブへのリンクを表示します。



2. 個別記事ページ

このページではブログの単体記事を表示します（下図）。記事の下には前の記事、次の記事へのリンクを表示し、さらにこの記事に投稿されたコメントとコメント投稿フォームを表示します。



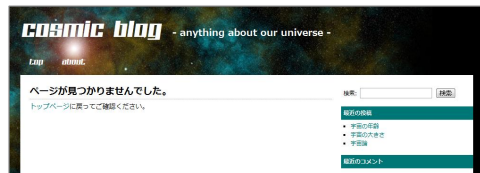
3. 「About」 ページ

このページは、サイトの紹介文を掲載するページです。About ページは固定ページとして作成し、ページの内容は管理ページから更新できるようにします。



4. 「404 Not Found」 ページ

このページは、指定されたページが存在しない場合に表示するページです。



WordPress の観点から見た機能

今回のサイトでは、以下の WordPress の機能を使用します。

- ブログ機能
- コメント機能
- トラックバック機能
- 投稿カテゴリー
- 固定ページ
 - About ページ
- カスタムメニュー
- ウィジェット

その他

- プラグインを利用し、ソーシャルボタンを配置します。
- スマートフォン向けサイトは、プラグインを使って構築します。

作業の流れ

このサイトを作成するにあたり、以下のような流れで作業を行います。

1. WordPress のインストール
2. ブログ記事の登録と固定ページの作成
3. オリジナルテーマ作成の準備
4. ブログページ用テンプレートの作成
5. カスタムメニューの設定
6. サイドバーの登録とウィジェットの表示
7. ヘッダーとフッターをパーツへ分解
8. 固定ページ用テンプレートの作成
9. 404 Not Found ページの作成
10. プラグインの導入
11. 不要なファイルやフォルダの削除

「WordPress.Project.01.Data」フォルダ内の WordPress 化する前の HTML ファイルをよく観察しておきましょう。どういう構造で HTML が構成されているかを把握しておくことで、これから行う作業に対する理解が深まります。

ToBeUploaded フォルダ内には、この後の作業で使う画像が入っています。

課題 01-1: WordPress のインストール

データベースの準備とユーザー権限の設定

まず、今回のサイト用に phpMyAdmin で新規のデータベース「wp01db」を作成します。

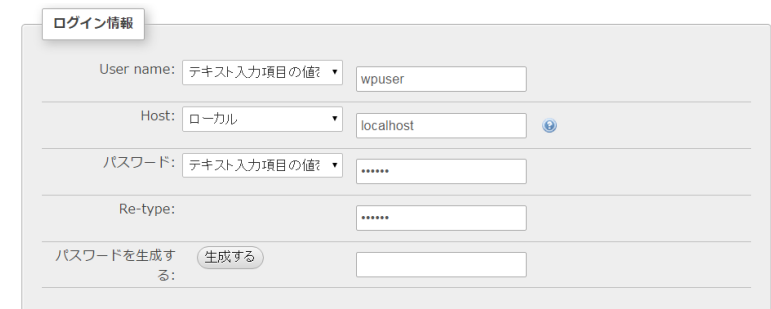


次に、データベースアクセス用ユーザーを作成し、「wp01db」に対する操作権限を設定します。今回はユーザー名を「wpuser」、パスワードは「secret」としておきます。以下に phpMyAdmin でのユーザー追加手順を紹介します。（SQL 文を利用したユーザーの作成でも構いません）

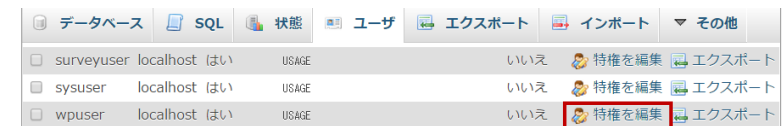
① ユーザタブからユーザを追加するをクリック



② ログイン情報にユーザー名とパスワードを入力します。パスワードは Re-type の欄にも入力してください。Host はローカルを設定します。ページ下部にある実行ボタンを押して、ユーザーの作成を完了します。



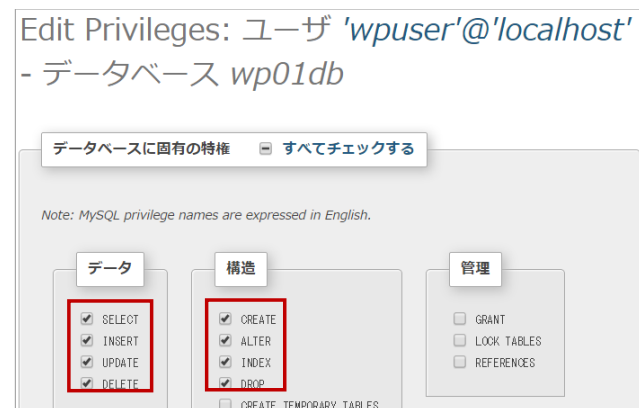
③ 次はデータベースに対する操作権限を設定します。先ほど作成したユーザーの特権を編集します。



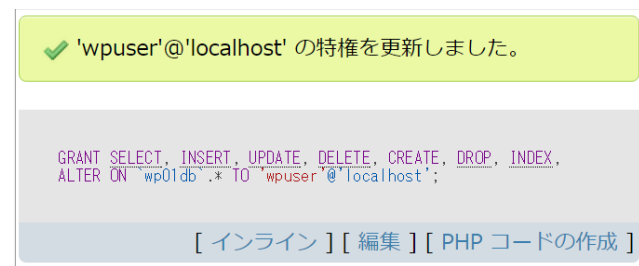
③ データベースをクリックし、Add privileges on the following database（次のデータベースに権限を付与する）の欄で「wp01db」を選択します。



④画面が自動的に遷移するので、ユーザー名・データベース名を確認して、付与する権限にチェックを入れます。



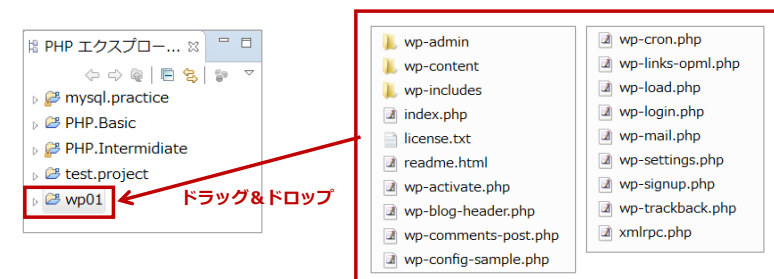
⑤**実行**ボタンを押して、権限の付与を完了します。以下のような画面が表示されれば成功です。MySQL Monitor を利用してユーザーの設定を行う場合は、ここに表示されている SQL 文を参考にしてください。



データベースの準備は以上で完了です。

WordPress のインストール

今回のサイト用に新たに WordPress をインストールします。まず、Eclipse 内で新たに「wp01」というプロジェクトを作成します。次に WordPress の ZIP ファイルを展開して作成されるフォルダとファイルをすべてプロジェクト内に移動します。



ブラウザで「http://localhost/wp01/」を開き、WordPress のインストール作業を行います。

データベースの接続設定では、上記で準備したデータベースの接続情報を入力します。テーブル接頭辞はデフォルトのままでも構いませんが、ここでは本番サーバへの移行を考え「cb_」としています。

| | | |
|-------------|-----------|---|
| データベース名 | wp01db | WordPress を動作させるデータベースの名 |
| ユーザー名 | wpuser | MySQL のユーザー名 |
| パスワード | secret | ...そして、あなたの MySQL パスワード。 |
| データベースのホスト名 | localhost | もし localhost という値では動かない場合、ホスティングサービスから情報が入手できるはずです。 |
| テーブル接頭辞 | cb_ | ひとつのデータベースに複数の WordPress をインストールしたい場合、これを変えてください。 |

サイト名は「Cosmic Blog」とします。その他の情報は適宜設定してください。ここでのユーザー名・パスワードはWordPressの管理画面に入るためのものです。先に作成したデータベースのユーザー名・パスワードとは異なるもので構いません。覚えやすいものを設定してください。

必要情報

次の情報を入力してください。ご心配なく、これらの情報は後からいつでも変更できます。

サイトのタイトル

サイトのキャッチフレーズの設定

インストール作業が完了したら、管理画面にログインし、「設定」→「一般」にてキャッチフレーズを「anything about our universe」に変更して保存しておきます。

設定

一般

投稿設定

表示設定

ディスカッション

メディア

パーマリンク設定

一般設定

サイトのタイトル

キャッチフレーズ
このサイトの簡単な説明。

課題 01-2: ブログ記事の登録と固定ページの作成

動作確認に使用する仮のブログ記事を何件か投稿し、About ページ用の固定ページを作成します。

テスト用ブログ記事の作成

管理画面の「投稿」→「新規追加」画面で、仮のブログ記事を数件投稿しておきます。長い文章になる場合は、「続きを読む」タグを挿入しましょう。

新規投稿を追加

宇宙とは何か？

パーマリンク: <http://localhost/wp01/?p=5> [パーマリンクの変更](#) [投稿](#)

[メディアを追加](#) [ビジュアル](#) [テキスト](#)

「続きを読む」タグを挿入

宇宙とは何か？ この広大で果てしない問いは、長きに渡って人類を魅了し続けている。

About ページ用固定ページの作成

管理画面の「固定ページ」→「新規追加」画面で、About ページ用の固定ページを作成します。必要な画像はToBeUploaded フォルダに入っています。WordPressのエディタ上で、画像をクリックすると、配置に関するアイコンが表示されます。

新規固定ページを追加

About

パーマリンク: http://localhost/wp01/?page_id=14 [パーマリンクの変更](#) [固定ページを表示](#)

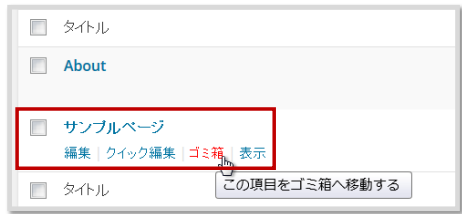
[メディアを追加](#) [ビジュアル](#) [テキスト](#)

配置に関するアイコン

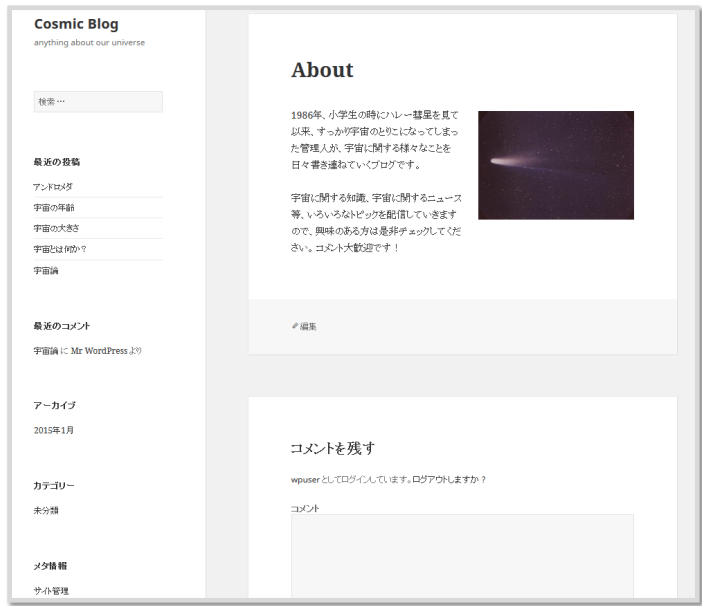
1986年、小学生の時にハレー彗星を見て以来、すっかり宇宙のとりこになってしまった管理人が、宇宙に関する様々なことを日々書き連ねていくブログです。

宇宙に関する知識、宇宙に関するニュース等、いろいろなトピックを配信していきますので、興味のある方は是非チェックしてください。コメント大歓迎です！

固定ページのリストにあるサンプルページは不要なので削除しておきます。



この状態でサイトを表示すると、デフォルトのテーマ（下図は「Twenty Fifteen」）が適用された状態でページが表示されます。



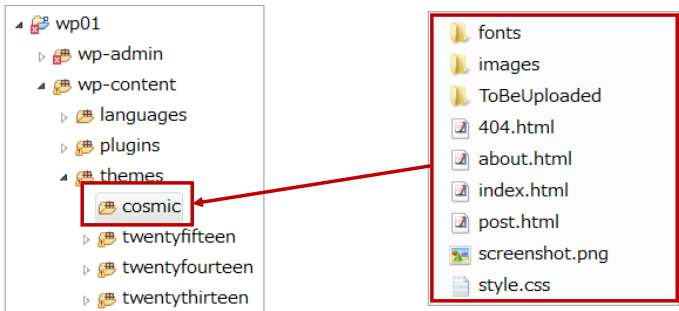
課題 01-3: オリジナルテーマ作成の準備

オリジナルテーマの元となる HTML データから、オリジナルテーマを作成する準備を行います。
「WordPress.Project.01.Data」フォルダ内に、以下のようなファイルやフォルダが含まれていることを確認します。

| ファイル名 | 説明 |
|----------------|---|
| index.html | トップページの HTML データ |
| post.html | ブログ個別記事ページの HTML データ。このファイルは最後に削除します |
| about.html | About ページの HTML データ |
| 404.html | 404 Not Found ページの HTML データ |
| style.css | CSS ファイル |
| screenshot.png | テーマ用スクリーンショット画像 |
| fonts | フォントファイルが格納されているフォルダ |
| images | 更新性のない画像ファイルが格納されているフォルダ |
| ToBeUploaded | About ページに挿入するための画像ファイルが格納されているフォルダ。このフォルダは最後に削除します |

テーマ用フォルダの作成

wp-content/themes フォルダ内に「cosmic」フォルダを作成し、「WordPress.Project.01.Data」フォルダ内のファイルとフォルダを全て cosmic フォルダ内にコピーします。



index.php の準備

cosmic フォルダ内の index.html のファイル名を「**index.php**」に変更します。

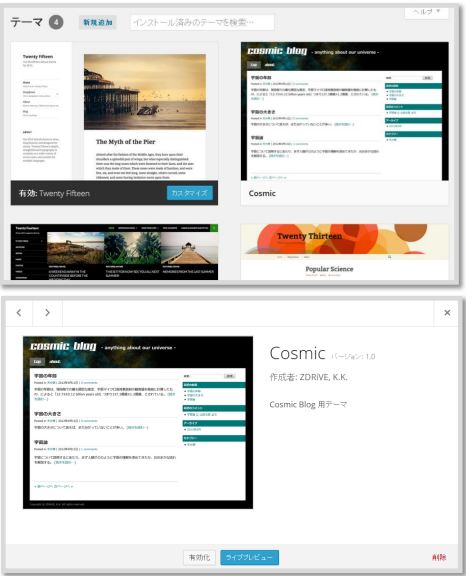
style.css の準備

cosmic フォルダ内の style.css ファイルの先頭に以下の記述を追加して保存します。

style.css

```
/*
Theme Name: Cosmic
Description: Cosmic Blog 用テーマ
Author: ZDRIVE, K.K.
Version: 1.0
*/
```

これで WordPress は、themes フォルダ内に配置した cosmic フォルダをテーマフォルダであると認識します。管理画面の「外観」→「テーマ」を見ると以下のように表示されます（下図）ので、有効化します。



この時点では、元々の HTML に記述されていたものが表示されますが、CSS が効いていません。



この状態を頭の隅に留めておいて、次のページの作業に進みましょう。作業ごとにこまめに「サイトを表示」をすると理解が深まります。

課題 01-4: ブログページ用テンプレートの作成

このサイトのメインのページであるブログページのテンプレートを作成します。今回作成するテーマでは、記事の一覧ページと個別記事のページ共に index.php 内で処理します。

head 要素の変更

まず、head 要素内のいくつかの要素を、WordPress で生成される情報を利用する形に変更します。

1. 文字コードを示す meta 要素

index.php

```
<meta charset="<?php bloginfo('charset'); ?>">
```

ここで使われるテンプレートタグの説明は以下の通りです。

| テンプレートタグ | 説明 |
|---------------------|-----------------|
| bloginfo('charset') | ページの文字コードを出力する。 |

2. スタイルシートの URL

index.php

```
<link rel="stylesheet" href="<?php echo get_stylesheet_uri(); ?>">
```

ここで使われるテンプレートタグの説明は以下の通りです。

| テンプレートタグ | 説明 |
|----------------------|--|
| get_stylesheet_uri() | テーマ用 CSS (style.css) の URL を絶対パスとして取得する。 |

この時点で CSS が有効になるはずですので、確認してみましょう。

3. ページのタイトル

index.php

```
<title><?php bloginfo('name'); ?><?php wp_title('|'); ?></title>
```

ここで使われるテンプレートタグの説明は以下の通りです。

| テンプレートタグ | 説明 |
|------------------|--------------------------------------|
| bloginfo('name') | 管理画面の「設定」→「一般」で設定された「サイトのタイトル」を出力する。 |
| wp_title() | ページのタイトルを出力する。最初の引数で区切り文字を指定できる。 |

今回は「サイト名 | ページタイトル」という形式でタイトルを設定しています。

サイト名とサイトの説明テキスト

index.php

```
...
<header>
  <h1>
    <a href="<?php echo home_url(); ?>"><?php bloginfo('name'); ?></a>
    <span>- <?php bloginfo('description'); ?> -</span>
  </h1>
</header>
```

ここで使われるテンプレートタグの説明は以下の通りです。

| テンプレートタグ | 説明 |
|-------------------------|--------------------------------------|
| home_url() | サイトのホーム URL を取得する。 |
| bloginfo('description') | 管理画面の「設定」→「一般」で設定された「キャッチフレーズ」を出力する。 |

wp_head()と wp_footer()の挿入

プラグインを導入した際に正しく動作させるために、ページ内に wp_head() と wp_footer() を挿入します。

wp_head() は、head 要素の終了タグ (</head>) の直前の行に挿入します。

index.php

```
...
<?php wp_head(); ?>
</head>
```

wp_footer() は、body 要素の終了タグ (</body>) の直前の行に挿入します。

index.php

```
...
<?php wp_footer(); ?>
</body>
</html>
```


記事リストの表示

index.php 内で、まずは記事リストを表示する部分を記述していきます。

1. 記事リスト表示の基本構造の記述

このテーマでは、記事の表示部分は以下のような HTML 構造になっています：

```
<div id="contents">
  <article="post">
    <h2>
      <a href="記事の URL">記事のタイトル</a>
    </h2>
    <p class="postmeta">
      Posted in カテゴリー名 | 日付 |
      <a href="コメントの URL">コメント数 comments</a>
    </p>
    <div class="entry">
      記事の内容
    </div>
  </article><!-- post -->
  <article class="post">
    ... 2つ目の記事 ...
  </article>
  <article class="post">
    ... 3つ目記事 ...
  </article>
  <footer>
    <div id="link">
      <a href="#">&laquo; 前ページへ</a>
      <a href="#">次ページへ &raquo;</a>
    </div>
  </footer>
</div><!-- contents -->
```

1 件の記事の情報

id="contents" の div 要素内に、1 件の記事の情報を表す class="post" の article 要素が記事の数だけ繰り返されているという構造です。

現時点では index.php 内にはダミーのデータが記述されています。まずは、id="contents" の div 要素の中に class="post" の article のブロックを一つだけ残して、あとは削除します。青字の部分は、後から WordPress のテンプレートタグに置き換わります。

index.php

```
<div id="contents">
  <article class="post">
    <h2>
      <a href="#">宇宙の年齢</a>
    </h2>
    <p class="postmeta">
```

```
      Posted in <a href="#">未分類</a> | 2012 年 4 月 11 日 |
      <a href="#">0 comments</a>
    </p>
    <div class="entry">
      <p>宇宙の年齢は、現段階での最も秘密な（以下略）</p>
    </div>
  </article><!-- post -->
  <footer>
    <div id="link">
      <a href="#">&laquo; 前ページへ</a>
      <a href="#">次ページへ &raquo;</a>
    </div>
  </footer>
</div><!-- contents -->
```

次は WordPress のテンプレートタグを記述していきます。赤字の部分が class="post" の article 要素を繰り返すための記述、青字が元々のテキストをテンプレートタグで置き換えている部分です。まず、赤字部分だけ記述して、同じ内容の記事が繰り返し表示されることを確認しましょう。次に青字部分を記述して、WordPress の管理画面から投稿した内容が反映されることを確認します。

index.php

```
<div id="contents">
  <?php if (have_posts()); ?>
  <?php while (have_posts()); the_post(); ?>
  <article class="post">
    <h2>
      <a href="<?php echo get_permalink(); ?>"><?php the_title(); ?></a>
    </h2>
    <p class="postmeta">
      Posted in <?php the_category(' '); ?> |
      <?php echo get_the_date(); ?> |
      <a href="<?php comments_link(); ?>">
        <?php echo get_comments_number(); ?> comments</a>
    </p>
    <div class="entry">
      <?php the_content("（続きを読む...）"); ?>
    </div>
  </article><!-- post -->
  <?php endwhile; ?>
  <?php endif; ?>
  <footer>
    <div id="link">
      <a href="#">&laquo; 前ページへ</a>
      <a href="#">次ページへ &raquo;</a>
    </div>
  </footer>
</div><!-- contents -->
```

この中に登場する以下のコードブロックは、投稿記事が存在する数だけ繰り返し処理を行うための定石的な書き方ですので、そのまま覚えましょう。

```
<?php if (have_posts()): ?>
<?php while (have_posts()): the_post(); ?>
...
<?php endwhile; ?>
<?php endif; ?>
```

その他のテンプレートタグの説明は以下の通りです。

| テンプレートタグ | 説明 |
|-----------------------|--|
| get_permalink() | 記事の個別ページの URL を取得する。 |
| the_title () | 記事のタイトルを出力する。 |
| the_category() | 記事の属するカテゴリーを出力する。引数として複数カテゴリーを列挙する際の区切り文字が指定できる。 |
| get_the_date() | 記事が投稿された日付を取得する。 |
| comments_link() | 記事に対するコメントの URL を出力する。 |
| get_comments_number() | 記事に対するコメントの数を取得する。 |
| the_content() | 記事の本文を出力する。今回は引数として「（続きを読む...）」を渡しています。特に指定がなければ「続きを読む」タグを挿入した箇所には「さらに...」と表示されます。 |

2. ページ移動リンクの表示

記事の件数が「設定」→「表示設定」で設定できる 1 ページ当たりの表示件数を超えた場合に次のページや前のページに移動するためのナビゲーションリンクを表示します。

index.php

```
...
<footer>
<div id="link">
  <?php previous_posts_link(); ?>
  <?php next_posts_link(); ?>
</div>
</footer>
</div><!-- contents -->
```

ここで使われるテンプレートタグの説明は以下の通りです。

| テンプレートタグ | 説明 |
|-----------------------|--|
| previous_posts_link() | 前のページへのテキストリンクを出力する。前のページは、より最近に投稿された記事のリストとなる。デフォルトでは「<< 前ページへ」というテキストリンクが表示されるが、引数でテキストリンクの表示を指定できる。 |
| next_posts_link() | 次のページへのテキストリンクを出力する。次のページは、より過去に投稿された記事のリストとなる。デフォルトでは「次ページへ >>」というテキストリンクが表示されるが、引数でテキストリンクの表示を指定できる。 |

個別記事の表示

次に、個別記事の内容を表示する記述を追加します。記事の内容自体は先に作成したループの記述で処理されるので、ここではそれ以外の部分（コメント、トラックバック URL、前後の記事へのリンク）を作成します。

1. 個別記事用のテキストリンク

まず、個別記事の表示かどうかを調べる関数「**is_single()**」を使って、個別記事の場合と記事リストの場合での表示を分けます。

index.php

```
...
<footer>
<?php if (is_single()): ?>

<?php else: ?>
  <div id="link">
    <?php previous_posts_link(); ?>
    <?php next_posts_link(); ?>
  </div>
<?php endif; ?>
</footer>
</div><!-- content -->
```

個別記事の場合は、前の記事と次の記事へのテキストリンクを表示します。

index.php

```
<?php if (is_single()): ?>
<div id="link">
  <?php previous_post_link('前の記事: %link'); ?>
  <?php next_post_link('次の記事: %link'); ?>
</div>
<?php else: ?>
<div id="link">
  <?php previous_posts_link(); ?>
  <?php next_posts_link(); ?>
</div>
<?php endif; ?>
```

ここで使われるテンプレートタグの説明は以下の通りです。

| テンプレートタグ | 説明 |
|----------------------|---|
| previous_post_link() | 前の記事へのテキストリンクを出力する。前の記事とは、より古い記事のことを指す。デフォルトでは「<< (前の記事のタイトル)」というテキストリンクが表示されるが、引数でテキストリンクの表示を指定できる。その際、「%link」という表現を組み込むと、その場所に記事のタイトルを表示させることができる。 |
| next_post_link() | 次の記事へのテキストリンクを出力する。次の記事とは、より新しい記事のことを指す。デフォルトでは「(次の記事のタイトル) >>」というテキストリンクが表示されるが、引数でテキストリンクの表示を指定できる。その際、「%link」という表現を組み込むと、その場所に記事のタイトルを表示させることができる。 |

2. コメントの表示

個別記事に対するコメントとコメント投稿フォームを設置します。

```
<?php if (is_single()): ?>
<div id="link">
  <?php previous_post_link('前の記事: %link'); ?>
  <?php next_post_link('次の記事: %link'); ?>
</div>
<?php comments_template(); ?>
<?php else: ?>
  ...
```

ここで使われるテンプレートタグの説明は以下の通りです。

| テンプレートタグ | 説明 |
|---------------------|-----------------------|
| comments_template() | コメントテンプレートを読み込んで表示する。 |

※ログイン状態では、名前・メールアドレス・ウェブサイトの欄は表示されません。

トラックバック URL の表示

個別記事に対するトラックバック URL を表示します。トラックバック URL は、その記事がトラックバックを許可している時だけ表示するようにするために、「**pings_open()**」という関数を使ってチェックを行います。

```
<?php if (is_single()): ?>
<div id="link">
  <?php previous_post_link('前の記事: %link'); ?>
  <?php next_post_link('次の記事: %link'); ?>
</div>
<?php comments_template(); ?>
<?php if (pings_open()): ?>
<div id="trackback">
  <p>トラックバック URL</p>
  <p><input type="text" value="<?php trackback_url(); ?>"></p>
</div>
<?php endif; ?>
```

ここで使われるテンプレートタグの説明は以下の通りです。

| テンプレートタグ | 説明 |
|------------------|---------------------|
| trackback_url () | トラックバック用 URL を出力する。 |

ここまでで、記事リストの表示と個別記事の表示部が完成しました。次はナビゲーションを作っていきます。

課題 01-5: カスタムメニューの設定

ここでは、Cosmic Blog のメインナビゲーションを作成します。まず、テーマに機能を加えるために必要な「**functions.php**」という名前のファイルを cosmic フォルダ内に作成します。

1. カスタムメニューの登録

functions.php に以下のように記述します。PHP の終了タグは記述しません。

functions.php

```
<?php
/**
 * カスタムメニューの登録
 */
register_nav_menus();
```

ここで使われるテンプレートタグの説明は以下の通りです。

| テンプレートタグ | 説明 |
|-----------------------|----------------|
| register_nav_menus () | カスタムメニューを登録する。 |

2. カスタムメニューの表示

index.php 内にある id="navi" の div の内容を削除し、代わりに以下のように記述します。

index.php

```
...
<nav id="global">
  <?php wp_nav_menu (array ("container"=>false)); ?>
</nav>
<div id="main">
```

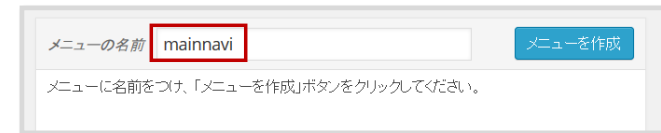
ここで使われるテンプレートタグの説明は以下の通りです。

| テンプレートタグ | 説明 |
|---------------|---|
| wp_nav_menu() | カスタムメニューを表示する。引数は連想配列で渡すことができるが、今回は「 " container " =>false」とした。デフォルトでは ul, li を div が囲む形でメニュー出力されるが、この設定によって div の出力は行われない。 |

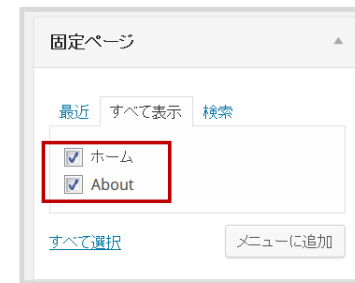
これでカスタムメニューが表示されるようになりました。

3. メニューに表示する項目の設定

次に、カスタムメニューに表示させる項目の設定を行います。管理画面で「外観」→「メニュー」を選択し、画面の右上にあるパネルの「メニューの名前」という欄に、今回作成するメニューの名前を「mainnavi」と入力し、「メニューを作成」ボタンをクリックします。



今回はトップページへのリンクと About ページへのリンクをメニュー項目として表示させます。画面左下にある「固定ページ」パネルで「すべて表示」を選び、「ホーム」と「About」にチェックを入れ「メニューに追加」ボタンをクリックします（下図）。



「ホーム」と「About」の項目が追加されます。



ここで、「ホーム」の項目の表示を「Top」に変更します。「ホーム」項目の右端にある下向きの三角のボタンをクリックし、「ナビゲーションラベル」の欄に「Top」と入力してメニューを保存します。

Top

カスタムリンク ▲

URL

http://localhost/wp01/

ナビゲーションラベル

Top

移動

ひとつ下△

削除

キャンセル

これで表示するメニュー項目の設定ができました。サイト画面で確認するとナビゲーション部に「Top」と「About」の項目が表示されていることが確認できます（下図）。



課題 01-6: サイドバーの登録とウィジェットの表示

Cosmic Blog では、画面右側にサイドバーとして検索ボックス、最近投稿された記事へのリンク、カテゴリーごとの記事へのリンクなどを表示しますが、これらは WordPress の「ウィジェット」機能を利用します。

オリジナルテーマ内でウィジェットを扱えるようにするためには、サイドバーの登録と表示設定が必要になります。

1. サイドバーの登録

functions.php に、以下の記述を追加します。

functions.php

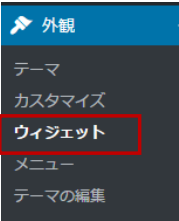
```
...
/**
 * サイドバーの登録
 */
register_sidebar();
```

これで、今回作成するテーマでウィジェットを扱えるようになりました。

ここで使われるテンプレートタグの説明は以下の通りです。

| テンプレートタグ | 説明 |
|--------------------|-------------|
| register_sidebar() | サイドバーを登録する。 |

これでウィジェットが登録できるようになりました。まずは「最近の投稿」「最近のコメント」「アーカイブ」「カテゴリー」のウィジェットを表示させたいので、管理画面の「外観」→「ウィジェット」の画面で「サイドバー1」と表示されているボックスにそれらのウィジェットを登録します。「検索」は別の方法で表示するので、今回は登録しません。



2. index.php の修正

index.php 内でウィジェットを表示する記述を追加します。検索のブロックの下に、下記（青字部分）のように追記します。

index.php

```
...
<aside>
  <div class="widget widget_search">
    <form method="get" id="searchform" action="">
      <div>
        <label class="screen-reader-text" for="s">検索:</label>
        <input type="text" value="" name="s" id="s">
        <input type="submit" id="searchsubmit" value="検索">
      </div>
    </form>
  </div>
  <?php dynamic_sidebar(); ?>
</aside>
</main>
```

ここで使われるテンプレートタグの説明は以下の通りです。

| テンプレートタグ | 説明 |
|-------------------|-------------|
| dynamic_sidebar() | サイドバーを表示する。 |

ここまでの作業ができれば、実際の画面で確認し、さらに要素を検証してみます。すると、下記（左）のような HTML の構成でウィジェットが出力されていることがわかります。これは元々の構成（右）とは異なるので、次の作業で修正していきます。

現在の HTML 構成

```
<li>
<h2>最近の投稿</h2>
<ul>
  <li>宇宙の年齢</li>
  <li>宇宙の大きさ</li>
  <li>宇宙論</li>
</ul>
</li>
```

元々の HTML 構成

```
<section class="widget">
<h2>最近の投稿</h2>
<nav>
  <ul>
    <li>宇宙の年齢</li>
    <li>宇宙の大きさ</li>
    <li>宇宙論</li>
  </ul>
</nav>
</section>
```

3. functions.php の修正

先ほど functions.php 内に記述した register_sidebar() にいくつかのパラメーターを連想配列で渡します。

functions.php

```
...
/**
 * サイドバーの登録
 */
register_sidebar(
  array(
    'before_widget' => '<section class="widget">',
    'after_widget' => '</nav></section>',
    'after_title' => '</h2><nav>'
  )
);
```

これでタグの構成が意図したものになりました。以下に配列のキーと関連する箇所を示します。今回の場合、before_title は特に指定していません。

| キー | 説明 |
|---------------|---|
| before_widget | ウィジェットの前に出力されるテキスト。デフォルトでは、 が出力される。 |
| after_widget | ウィジェットの後に出力されるテキスト。デフォルトは |
| before_title | タイトルの前に出力されるテキスト。デフォルトは <h2> |
| after_title | タイトルの後に出力されるテキスト。デフォルトは </h2> |

4. 検索ボックスの表示

次に検索ボックスを表示します。今回は検索ボックスを任意の div タグ内に表示させるため、dynamic_sidebar とは切り分けています。

index.php

```
<aside>
  <div class="widget widget_search">
    <?php get_search_form(); ?>
  </div>
  <?php dynamic_sidebar(); ?>
</aside>
```

ここで使われるテンプレートタグの説明は以下の通りです。

| テンプレートタグ | 説明 |
|-------------------|--------------|
| get_search_form() | 検索ボックスを表示する。 |

以上でサイドバーは完成です。

課題 01-7: ヘッダーとフッターをパーツへ分解

オリジナルテーマを作成する際、ページによっては、レイアウトや表示内容を他のページと変えた場合があります。その場合は、index.php 以外にもテンプレートファイルを作成しますが、その場合でもヘッダー部分とフッター部分は同じデザインを使いたいということがよくあります。

そのような場合、ヘッダー部分とフッター部分をそれぞれ別ファイルとして用意し、各テンプレートファイルでインクルードするという形にします。

WordPress では、ヘッダー用の PHP ファイルとフッター用の PHP ファイルには決まった名前を使うことになっています。ヘッダー用の PHP ファイルは「**header.php**」、フッター用の PHP ファイルは「**footer.php**」という名前です。ファイル名に「header」「footer」とついていますが、共通で利用していくものをパーツ化することが目的ですので、header タグや footer タグの外の要素が含まれていても問題ありません。

それでは、これら 2 つのファイルを作成していきます。

1. header.php の準備

まず、cosmic フォルダ内に header.php を新規で作成します。次に index.php の先頭行から **<div id="contents">** の行までを切り取って、header.php に貼り付けます。

header.php

```
<!DOCTYPE html>
<html lang="ja">
<head>
<meta charset="<?php bloginfo('charset'); ?>">
<link rel="stylesheet" href="<?php echo get_stylesheet_uri(); ?>">
<title><?php bloginfo('name'); ?><?php wp_title('|'); ?></title>
<?php wp_head(); ?>
</head>
<body>
<div id="container">
  <header>
    <h1>
      <a href="<?php echo home_url(); ?>"><?php bloginfo('name'); ?></a>
```

```
<span>- <?php bloginfo('description'); ?> -</span>
</h1>
</header>
<nav id="global">
  <?php wp_nav_menu(array("container"=>false)); ?>
</nav>
<main>
  <div id="contents">
```

2. footer.php の準備

同様に footer.php を新規作成し、index.php の **</div><!--contents -->** の行から最後の行までを切り取り、footer.php に貼り付けます。

footer.php

```
</div><!-- contents -->
<aside>
  <div class="widget widget_search">
    <?php get_search_form(); ?>
  </div>
  <?php dynamic_sidebar(); ?>
</aside>
</main>
<footer>
  <p>Copyright &copy; ZDRIVE, K.K. All rights reserved.</p>
</footer>
</div>
<?php wp_footer(); ?>
</body>
</html>
```

3. index.php の変更

header.php と footer.php にコードを移動させた index.php で以下のように記述し、ヘッダー部分とフッター部を読み込みます。

index.php

```
<?php get_header(); ?>
<?php if(have_posts()): ?>
  ...
</footer>
<?php get_footer(); ?>
```

ここで使われるテンプレートタグの説明は以下の通りです。

| テンプレートタグ | 説明 |
|--------------|------------------------------|
| get_header() | テーマ内の header.php を読み込んで表示する。 |
| get_footer() | テーマ内の footer.php を読み込んで表示する。 |

課題 01-8: 固定ページ用テンプレートの作成

現状では、固定ページである「About」ページを表示させると、ブログ記事と同じように投稿日などが表示されます。



固定ページの場合はこのような投稿日などの情報を表示しないようにするために、固定ページ用のテンプレートを作成します。固定ページ専用のテンプレートは「**page.php**」という名前のファイルに記述することになっています。今回は about.html を page.php に変更し固定ページ「About」を作成します。

page.php を作成したら、<div id="contents">までを header.php に、</div><!-- contents -->以降を footer.php に置き換えます。

page.php

```
<?php get_header(); ?>
<article class="post">
  <h2>
    <a href="about.html">About</a>
  </h2>
  <div class="entry">
    <p>
      (…中略…)
    ……コメント大歓迎です！</p>
  </div>
</article><!-- post -->
<?php get_footer(); ?>
```

次に、about ページのメインコンテンツを WordPress のテンプレートタグを使って出力します。

page.php

```
<?php get_header(); ?>
<?php if (have_posts()): ?>
<?php while (have_posts()): the_post(); ?>
  <article class="post">
    <h2>
      <a href="<?php echo get_permalink(); ?>"><?php the_title(); ?></a>
    </h2>
    <div class="entry">
      <?php the_content(); ?>
    </div>
  </article><!-- post -->
<?php endwhile; ?>
<?php endif; ?>
<?php get_footer(); ?>
```

これで About ページは完成です。

課題 01-9: 404 Not Found ページの作成

テーマフォルダ内にある「404.html」ファイルの名前を「**404.php**」に変更し、ヘッダーとフッターの部分 get_header() および get_footer() に置き換えます。

404.php

```
<?php get_header(); ?>
<article class="post">
  <h2>ページが見つかりませんでした。</h2>
  <div class="entry">
    <p><a href="<?php echo home_url(); ?>">トップページ</a>に戻ってご確認ください。</p>
  </div>
</article>
<?php get_footer(); ?>
```

課題 01-10: プラグインの導入

このサイトでは以下のプラグインを使用します。

- **WP Multibyte Patch**
 - デフォルトでインストールされているマルチバイト環境用不具合修正を行うプラグイン。
 - 日本語のサイトを作る場合はとりあえずこのプラグインを有効化しておきます。
- **spam-byebye**
 - コメントスパムを排除するプラグイン

- WordPress をインストールすると最初からインストールされている「Akismet」という同種のプラグインがありますが、使用に際してユーザー登録が必要であったり、商用では有料となることから、代わりにこのプラグインを使用します。

● WP Social Bookmarking Light

- Facebook、Twitter、はてなブックマークなどのソーシャルボタンが手軽に設置できる。

● WPTouch

- スマートフォン向けに専用画面を表示するプラグイン
- ページデザインのカスタマイズはあまりできませんが、ブログサイトを手っ取り早くスマートフォン対応させられます。

WP Multibyte Patch プラグインの導入

このプラグインは WordPress にデフォルトでインストールされているので、プラグイン画面で有効化します（下図）。



spam-byebye プラグインの導入

下記から spam-byebye をダウンロードします。

<https://wordpress.org/plugins/spam-byebye/>

「プラグイン」→「新規追加」の画面で「プラグインのアップロード」からダウンロードした zip ファイルを選択し「いますぐインストール」をクリックします。



インストールが完了したら「プラグインを有効化」をします。

WP Social Bookmarking Light プラグインの導入

spam-byebye プラグインと同様の手順でインストールと有効化を行います。WP Social Bookmarking Light は下記からダウンロードできます。

<https://ja.wordpress.org/plugins/wp-social-bookmarking-light/>

有効化するだけで、個別記事ページの記事上部に 3 つのソーシャルボタンが追加されます。



ボタンの配置場所や表示するボタンの種類は、設定画面から変更することができます。



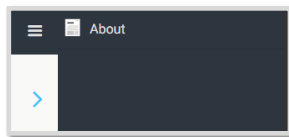
WPtouch プラグインの導入

同様の手順でインストールと有効化を行います。WPtouch は下記からダウンロードできます。

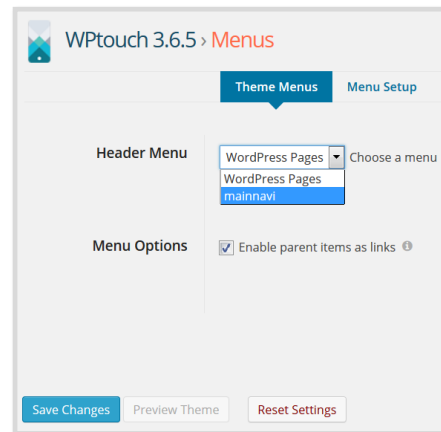
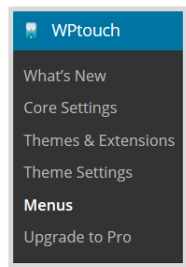
<https://wordpress.org/plugins/wptouch/>

WPtouch プラグインを有効化した後、Chrome の Developer Tool などを利用してスマートフォン用の表示をすると、右図のようになります。

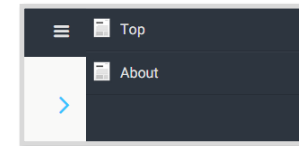
このままの状態では「mainnavi」で設定したメニューが表示されない（下図）ので、管理画面から WPtouch の設定を行っていきます。



管理画面の「設定」→「WPtouch」→「Menus」の画面に移動します。「Theme Menus」の「Header Menu」から「mainnavi」を選択し、「Save Changes」で変更を保存します。



「mainnavi」で設定したメニューが反映されます。



以上で「Cosmic Blog」は完成です。

課題 01-11: 不要なファイルやフォルダの削除

cosmic フォルダに含まれる不要なファイルやフォルダを削除し、テーマとして完成させます。今回不要なファイルとフォルダは以下の通りです

- post.html
- ToBeUploaded フォルダ

これで Cosmic Blog サイトが完成しました。