# WordPress Custom Post Type for Glossary Project

I've been playing with the idea of creating a technical glossary for refering technical terms on my posts. After a lot of research I came back to a simple tool: WordPress custom post types (CPT).

The idea is as follows:

1. Create a custom post type plugin for glossary entries
    1. Create the post type
    2. Change the labels as appropriate
    3. Ensure that it will appear in the REST API and use Gutenberg if desired
2. Create a corresponding taxonomy
    1. Change the labels as appropriate
    2. Ensure that it will appear in the REST API
3. Explore how to incorporate the CPT and taxonomy to an existing theme
    1. Ensure that it will appear on the home page if needed or desired
    2. Create additional templates to work with the custom post type
    3. Flush the cache on plugin activation (and only on activation)

## Create Glossary Entry Custom Post Type

Creating the Custom Post Type is not hard, it's just somewhat cumbersome.

I first created an array of names and internationalized values for the different labels that I want to change from the default. I put the array in a separate variable to make the code easier to read.

```
<?php
function rivendellweb_custom_glossary_type() {
  $labels = array(
    'name' => __( 'Glossary Entries', RWPDOMAIN ),
    'singular_name' => __( 'Glossary Entry', RWPDOMAIN ),
```

```
    'featured_image' => __( 'Entry Image', RWPDOMAIN ),
    'set_featured_image' => __( 'Set Entry Image', RWPDOMAIN ),
    'remove_featured_image' => __( 'Remove Entry Image', RWPDOMAIN ),
    'use_featured_image' => __( 'Use Glosary Entry Image', RWPDOMAIN ),
    'archives' => __( 'Glossary', RWPDOMAIN ),
    'add_new' => __( 'Add Glosary Entry', RWPDOMAIN ),
    'add_new_item' => __( 'Add Glosary Entry', RWPDOMAIN ),
  );
```

The second variable holds an array of the arguments for the CPT that we will use when registering the post

> The two important attributes in the array are:
>
> - `supports` tells WordPress what areas of the editor are available in this Custom Post Type
> - `show_in_rest` makes the CPT available through the REST API and activates the Gutenberg editor for this CPT when set to true

```
<?php
  $args = array(
    'labe'labels'  => $labels,
    'public'       => true,
    'has_archive'public'lossary',
    'rewrite'      => array(
      'has_front' => true ),
    'menu_icon'    => 'dashicons-book',
   ',
    'ts'     => array(
      'title',
      'editor',
      'thumbnail',
     ),
    '     => array(
      '// Line below makes CPT available in rest
    // Line below makes CPT available to
    //Gutenberg/Block editor
    'show_in_rest' => true,
```

```
  );
```

We need to run the `register_post_type` function. It takes two parameter: the name of the CPT and an array of arguments we defined in the `$args` variable.

```
<?php
  register_post_type( 'glossary', $args );
}
```

The final step is to add our Custom Post Type to WordPress. We do this by adding an the function we created to the init hook using `add_action`.

```
<?php
add_action( 'init',
 'rivendellweb_custom_glossary_type' );
```

We have a custom post type. We will next look at creating a taxonomy for the CPT and how to integrate them.

For more information, check the reference page for register_post_type

# Create Taxonomy for Glossary Entries

In WordPress a taxonomy is a way of grouping posts together based on a select number of relationships. WordPress ships with two default taxonomies, categories and tags but these are not enough for customized or highly specialized post types.

WordPress allows you to create custom taxonomies to go along with your CPTs. In this case we'll create a taxonomy for the books CPT. For this example, we will create a genre taxonomy for our books.

Just like with the CPT, we first register the labels that we want to update. This is a basic subset, there are more labels we can customize.

```php
<?php
function rivendellweb_custom_book_genre_tax() {
  $tax_labels = array(
    'name'          => 'Genres',
    'singular_name' => 'Genre',
    'search_items'  => 'Search Genres',
    'edit_item'     => 'Edit Genre',
    'add_new_item'  => 'Add New Genre'
  );
```

Next we define an array of arguments. The biggest difference between this array and the one we used to define the book CPT is that we are making the taxonomy hierarchical. This will allow to create parent-child relationships like **non-fiction -> technology** or **fiction -> science fiction**

```php
<?php
  $args = array(
    'labels' => $tax_labels,
    'show_in_rest' => true,
    'hierarchical' => true,
    'query_var'    => true,
    'rewrite' => array(
      'has_front' => true ),
    'supports' => array(
      'title',
      'editor',
      'thumbnail'
    ),
  );
```

The `register_taxonomy` function takes three parameters: the name of the taxonomy, the name of the post type it's associated with and an array of arguments for the taxonomy (that we defined in the `$args` variable)

```php
<?php
  register_taxonomy('genre', 'book', $args);
```

```
}
```

like we did with the CPT, we need to tell WordPress to register the taxonomy so it can use it. We do this with the `add_action` function. The first parameter is the action we want to attach the code to and the second one is the function that we want to attach to the init event, in this case `rivendellweb_custom_book_genre_tax`

```php
<?php
add_action( 'init',
  'rivendellweb_custom_book_genre_tax'
);
```

This process did two things, it created the `genre` taxonomy and it associated it with the book post type. We can also associate our custom taxonomies to post and pages.

> See register_taxonomy for more information

# Flush the cache on plugin activation

```php
<?php
function my_rewrite_flush() {
    // initialize the custom post type
    rivendellweb_custom_glossary_type();

    flush_rewrite_rules();
}

register_activation_hook( __FILE__,
 'my_rewrite_flush' );
'my_rewrite_flush'
```

# Use the Taxonomy in the CPT

When you register the taxonomy you've already associated it with the CPT you want to use it with. To make sure all functionality works properly, you must also register the association from the CPT arguments by adding the following element to the CPT `$args` array:

```php
<?php
  'taxonomies' => array(
    'genre',
  ),
```

The array contains all the custom taxonomies that we want to use with the CPT. For books we could also add authors and publishers taxonomies.

# Adding the CPT and Taxonomies to an existing theme

In the last post we covered the technical part of creating Custom Post Types (CPTs) and Taxonomies. This post will cover integrating the CPT into a theme. We will also discuss issues regarding using CPTs as a plugin and modifying an existing theme or child theme versus incorporating the code for the CPT into a theme.

## Choosing deployment strategies

The most common way to deploy CPTs is to to bundle the code in a plugin and deploy them independent of the theme you're using.

This means that you will have to customize whatever theme you choose by creating a child theme for each theme you want to add the CPT to.

Creating a child theme is beyond the scope of this post but it's well documented in the theme handbook.

We'll leverage the template hierarchy to creaate templates specific to the CPT and taxonomies we created.

The process is simple:

1. Create a new template and name it as appropriate or copy the existing template (`single.php` or `archive.php`) and rename it to match your CPT or taxonomy
2. Customize the template to suit your needs

For the CPT, we'll create two type-specific templates:

- `single-book.php` for single entries of type book
- `archive-book.php` for listings of books

We can start these file from the corresponding generic versions (`single.php` and `archive.php`) and then modify them to suit our needs. Using these customized templates we can leverage the whole set of tools available to WordPress

templates.

This assumes that we want the book post type to have its own template. Another option is to use conditional tags to branch an existing template based on post type or any other criteria we want to use.

This process reduces the number of files at the cost of complexity and more difficulty in reading and understanding the files in question, particularly if you want to share your work.

# Adding the CPT to an existing theme's homepage

There may be times when you just want to add the custom post type to the home page without any type of customization; for example, you may want to add new business listings to the blog homepage or you may want to format your glossary the same way you format your regular content.

The sample function checks if we're home and if the query is the primary one. If the query matches the condition then we modify the query to add the CPT so it will appear on the home page. We could genralize this function to include all the CPTs that we want to add to the homepage.

Put the same function below on your theme's `functions.php` file.

```
<?php
function rivendellweb_add_glossary_to_query( $query ) {
    if ( $query->is_home() &&
         $query->is_main_query() ) {
        $query->set( 'post_type', array(
          'post',
          'glossary'
        ) );
    }
}

add_action( 'pre_get_posts',
            'rivendellweb_add_glossary_to_query' );
```