Post Types

Languages: English • 日本語 • Português do Brasil • Slovenčina • (Add your language)

WordPress can hold and display many different types of content. Internally, these are all stored in the same place, in the `wp_posts` table. These are differentiated by a column called `post_type`.

WordPress 3.0 gives you the capability to add your own custom post types and to use them in different ways.

Default Types

There are five major types that WordPress uses by default.

Post

A "post" in WordPress is the main type used by the blog. Posts are normally displayed in the blog in reverse sequential order by time (newest posts first). Posts are also used for creating the feeds.

Page

A "page" in WordPress is like a post, but it lives outside the normal time based structure of posts. They have their own URLs directly off the main site URL. They can also use special Page Templates to display them. Pages can also be organized in a hierarchical structure, with Pages being parents to other Pages.

Attachment

An "attachment" is a special post that holds information about files uploaded through the Media upload system. They hold all the description and name and other information about uploaded files. For images, this is also linked to metadata information about the size of the images and thumbnails generated from the images, the location of the files, and even information obtained from EXIF data embedded in the images.

Revisions

A "revision" is used to hold draft posts as well as any past revisions of existing posts or pages. These are basically identical to the main post/page that they are for, but have that post/page as their parent.

Nav Menus

The "nav_menu_item" type holds information about a single item in the Navigation Menu system. These are the first examples of entries in the posts table to be used for something other than an otherwise displayable content on the blog.

Custom Types

Adding a custom type to WordPress is done via the register_post_type function. This function allows you to define the post type and how it operates within WordPress.

Here's a basic example of adding a custom post type:

```
add_action( 'init', 'create_post_type' );
function create_post_type() {
        register_post_type( 'acme_product',
                array(
                        'labels' => array(
                                'name' => __( 'Products' ),
                                'singular_name' => __( 'Product' )
```

Home Page
WordPress Lessons
Getting Started
Working with WordPress
Design and Layout
Advanced Topics
Troubleshooting
Developer Docs
About WordPress

Codex Resources
Community portal
Current events
Recent changes
Random page
Help

```
                        ),
                'public' => true,
                'has_archive' => true,
                )
        );
}
```

This creates a post type named "Product". It has two major arguments with it. The first one is the "labels", which define the name of the type in both plural and singular forms. The second one is "public", which is a predefined flag to show the post type in the admin section and to make it show up on the main site itself, if it's queried for.

There are many more parameters you can add to the register_post_type function, to do things like set up hierarchy, show the new post type in searches, change the URLs of the new posts, and to hide or show meta boxes in the post editing screen. These parameters are optional, and you can use them to configure your post type on a detailed level.

Naming Best Practices

While it's convenient to name your custom post type a simple name like "product" which is consistent with the core post types "post", "page", "revision", "attachment" and "nav_menu_item", it is better if you prefix your name with a short "*namespace*" that identifies your plugin, theme or website that implements the custom post type.

For example:

- "acme_product" or "aw_product" - Products post type used by hypothetical ACMEWidgets.com.
- "eightfold_product" or "eft_product" - Products post type provided by the hypothetical "EightFold" Theme.
- "ai1m_product" - Products post type provided by the hypothetical "All-in-One Merchant" Plugin.

Without namespacing of your custom post types your website's post types will more likely conflict with custom post types defined in a theme you fall in love with later or a plugin you realize that you absolutely need to use. Or if you are developing custom post types or themes there's a much greater chance your plugin or theme will conflict with custom post types defined in other plugins or themes and/or custom post types defined for your prospective user's website. Namespacing your custom post type won't guarantee against conflicts but will certainly minimize their likelihood.

Do pay close attention to not having your namespace exceed 20 characters though, as the post_type column in the DB is currently a varchar field of the latter length.

Reserved Post Type Names

Although the core development team has yet to make a final decision on this, it has been proposed on the wp-hackers list that future core post type names will be namespaced with "**wp_**", i.e. if the core team decides to add an "Event" post type then according to this suggestion they would use the internal name "wp_event." Even though this hasn't been finalized, it'll be a good idea to avoid any custom post types whose name begins with "wp_."

Admin UI

When a post type is created like this, it gets a new top level entry in the Admin section to create posts of that new type. From there, you'll have a full post editor and everything that comes along with it by default. You can customize the UI for a post type with several hooks and filters, see this Custom Post Type snippets post by Yoast for an explanation and code examples.

URLs

The new post type will also get its own special section of the site layout. In the above example, posts of this new "product" type can be displayed at http://example.com/product/%product_name% (where %product_name% is the URL-ized name of your product, i.e. http://example.com/product/foobrozinator.) You can see this link appear in the edit pages for your new type, just like other posts.

URLs with Namespaced Custom Post Types

When you namespace a URL and still want to use a "clean" URL structure, you need to add the "rewrite" element to the array. For example, assuming the "ACME Widgets" example from above:

```
add_action( 'init', 'create_post_type' );
function create_post_type() {
        register_post_type( 'acme_product',
                array(
                        'labels' => array(
                                'name' => __( 'Products' ),
                                'singular_name' => __( 'Product' )
                        ),
                        'public' => true,
                        'has_archive' => true,
                          'rewrite' => array('slug' => 'products')
                  )
        );
}
```

The above will result in a URL like http://example.com/products/%product_name% (see description of %product_name% above.) Note that we used a plural form here which is a format that some people prefer because it implies a more logical URL for a page that embeds a list of products, i.e. http://example.com/products/.

Also note using a generic name like "products" here can potentially conflict with other plugins or themes that use the same name but most people would dislike longer and more obscure URLs like http://example.com/acme_products/foobrozinator and resolving the URL conflict between two plugins is easier simply because the URL structure is not stored persistently in each post record in the database in the same way custom post type names are stored.

Template Files

The theme system supports post types too. WordPress added support for single-type-template in Version 3.0 and for archive-type-template in Version 3.1.

Note: In some cases the permalink structure must be updated in order for the new template files to be accessed when viewing a custom post type. To do this go to the permalink menu, change the option to a different option, save the changes, and change it back to the desired option.

Single template

In the form of the single-type-template. In the same way that posts are shown on their own page with **single.php**, custom post types will use **single-{posttype}.php** if it's available.

So for the above example, you could create a **single-acme_product.php** file and the product posts would be shown using that template.

Archive template

In the form of the archive-type-template. In the same way that posts are shown on their own archive with **archive.php**, custom post types will use **archive-{posttype}.php** if it's available.

So for the above example, you could create a **archive-acme_product.php** file and the product posts would be shown using that template.

Note: Use the is_post_type_archive() function to check if the query shows post type archive page, and the post_type_archive_title() to echo the post type title.

Querying by post type

In the rest of the theme system, you can also create new queries to display posts from a specific post type. This is done via the "post_type" parameter to a WP_Query.

Example:

```
$args = array( 'post_type' => 'product', 'posts_per_page' => 10 );
$loop = new WP_Query( $args );
while ( $loop->have_posts() ) : $loop->the_post();
        the_title();
        echo '<div class="entry-content">';
        the_content();
        echo '</div>';
endwhile;
```

This simply loops through the latest 10 product posts and displays the title and content of them.

Function Reference

Post Types: register_post_type(), add_post_type_support(), remove_post_type_support(), post_type_supports(), post_type_exists(), set_post_type(), get_post_type(), get_post_types(), get_post_type_object(), get_post_type_capabilities(), get_post_type_labels(), is_post_type_hierarchical(), is_post_type_archive(), post_type_archive_title()

More Information

- Showing custom post types on your home/blog page
- Custom post types in WordPress
- Custom Post Types in WordPress 3.0
- Extending Custom Post Types in WordPress 3.0
- Change Order for Custom Post Types in WordPress 3.0 and up
- Custom Post Type snippets

Category: Advanced Topics