



## **Development of a dynamic web site (DWD)**

**Class 10**

# Types of posts and taxonomies

With a WordPress website or blog, at first at least, you can create two types of content : articles and pages.

## **Pages**

Pages are all these different pages containing different type of contents including the various templates.

## **Articles**

Articles are also pages, but regrouped in archives, sorted by dates and categories. They are like blog posts.

## **Attachments**

Attachments are all the medias within your library (images, sounds, videos, etc.) as well as menus which also are types of posts.

Although, these types of post often aren't sufficient when creating custom themes. Therefore, WordPress makes it possible to create your own *custom post type*.

## **Custom post type**

Very often, pages and articles won't be enough for your needs. Maybe will you need a daily menu for a restaurant, a portfolio page, a products page for a company, etc. These different « sections » would probably have there specificities and, therefore, would need their own layout. These are the type of pages called *custom post type*.

Most of the time, the contents of every of those different types of posts would be presented like blog's articles and each type of page may have it's very own template if needed. In other cases, the content may simply be put one after another in a one-pager type of layout.

## **Taxonomies**

Taxonomies make it possible to sort all the web site's contents within categories and labels (key words). Although categories are usually predefined, labels can be freely created.

For instance, a photography web site may define a photograph in the category «Portrait», but the labels could be : woman, black and white, short hair, comedian...

## Declaring a custom post type

As you may already have guessed, in order to be available and used, a custom post type must first be declared in *functions.php*. So, let's say you will create the portfolio part of your personal web site.

### Step 1: Creating the «init» hook

First, it is needed to create the hook in the file *functions.php*. The *init* hook is launched after WordPress has finished loading but before any headers are displayed.

```
<?php
function myproject_register_post_types() {
    // Declaration of custom post types and taxonomies go here
}

add_action( 'init', 'myproject_register_post_types' );
```

### Step 2: Declaring the post type

The declaration itself is made within the function *register\_post\_type()*. The function can contain several parameters, but we keep it here to a minimum. Your page should then appear in the dashboard menu.

```
<?php
function myproject_register_post_types() {

    $labels = array(
        'name' => 'Portfolio',
        'all_items' => 'My projects',           // Displayed in the sub-menu
        'singular_name' => 'Project',
        'add_new_item' => 'Add a project',
        'edit_item' => 'Modify project',
        'menu_name' => 'Portfolio'
    );
    $args = array(
        'labels' => $labels,
        'public' => true,
        'show_in_rest' => true,
        'has_archive' => true,
        'supports' => array( 'title', 'editor', 'thumbnail' ),
        'menu_position' => 5,
        'menu_icon' => 'dashicons-admin-customizer',
    );

    register_post_type( 'portfolio', $args );
}

add_action( 'init', 'myproject_register_post_types' ); // Launches the function
```

**\$labels and \$args**

This array contains what will show in the admin interface instead of the default labels.

**name**

The name of the page.

**all\_items**

How is called the group of all items.

**singular name**

How is called one item (here *project* instead of *projects*).

**public**

Since a custom theme is being developed, of course we want the page to be public (true). Only in a situation such as the development of a plugin *false* could be used.

**has\_archive**

This parameter indicates whether this page will behave like a post (*true*, using archive and singles), or has a page (*false*, like a one-pager).

**show\_in\_rest**

Allows the database to display the type of post within the REST API.

**menu\_position**

This sets where the custom post type will appear in the dashboard's menu :

- 5 : Just after Posts
- 10 : After Medias ;
- 20 : After Pages ;
- 65 : After Plugins ;
- 70 : After Users ;
- 80 : After Settings ;
- 100 : At the very bottom

**menu\_icon**

This sets the icon to be used before the custom post type's name in the dashboard. You may choose from WordPress dashicons library (<https://developer.wordpress.org/resource/dashicons/>), use a png file or a svg (base64).

**Examples :**

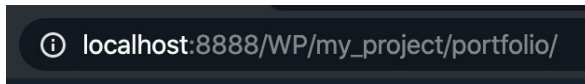
```
'menu_icon' => 'dashicons-portfolio',  
'menu_icon' => get_template_directory_uri() . '/img/cpt-icon.png',  
'menu_icon' => 'data:image/svg+xml;base64,' . base64_encode( "<svg>...</svg>" ),
```

**For more information :**

[https://developer.wordpress.org/reference/functions/register\\_post\\_type/](https://developer.wordpress.org/reference/functions/register_post_type/)

**Step 3: Test the page**

Type the URL in the address bar so you can visualize the result.



If you get a **404 error message**, go to **Dashboard>Settings>Permalinks**, and click the save button at the bottom of the page without changing anything. This should fix everything.

**Step 4: Add the page to your menu**

Go to **Dashboard>Appearance>Customize**, and modify the existing menu by adding the new page.