college CDI

Development of a dynamic web site (DWD)

Class 04

The WordPress loop

The WordPress loop is a short PHP snippet allowing to display the entire content of a page. It gets data ready (title, text, images, categories, link, etc.) and also calls it using easy to remember functions.

The loop is the same in every templates, all you need to do is to copy and paste it.

Integrating the loop

Let's see how to integrate the loop in the home page's template. Copy and paste the following codes in both ***front-page.php*** and ***page.php***. Doing so will result in the page's content showing in the browser upon testing.

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

<?php if( have_posts() ) : while( have_posts() ) : the_post(); ?>

        <h1><?php the_title(); ?></h1>
        <?php the_content(); ?>

<?php endwhile; endif; ?>

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

Explanation :

Let's see what these codes are all about.

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

have_posts()
This function retrieves the page's content to display it.

while
This loop ensures that all the content will be display. For instance, for a blog page, it will loop withing the content until there are no more posts to show.

the_post()
Prepares the data to be displayed within the loop through the various template tags.

The template tags

Templates Tags are WordPress functions that must be used within the loop allow to display the content accordingly with the templates. These functions (there are tens of them) can be used by developers to make their pages dynamic. In our example, two are used : *the_title()* and *the_content()*.

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

<?php if( have_posts() ) : while( have_posts() ) : the_post(); ?>

        <h1><?php the_title(); ?></h1>
        <?php the_content(); ?>

<?php endwhile; endif; ?>

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

the_title()
Allows to display the page's title.

the_content()
Allows to display the page's content.

For more details about the various template tags :
https://codex.wordpress.org/Template_tags

The loop within the blog's template

The blog's template shows the 10 most recent posts and this requires a few more codes. Template Hierarchy use a different template, the blog's home page, using *home.php* just in case you would like the first page to be different (showing post sorted by author or by categories, for instance).

So, for now, copy the following codes in *archive.php, home.php* and *archive.php*. We will correct the repetition later.

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

<h1>My project</h1>

<?php if( have_posts() ) : while( have_posts() ) : the_post(); ?>

    <article class="post">
            <h2><?php the_title(); ?></h2>
            <?php the_post_thumbnail(); ?>
            <p class="post__meta">
              Published the:  ?php the_time( get_option( 'date_format' ) ); ?>
              By: <?php the_author(); ?> • <?php comments_number(); ?>
            </p>

            <?php the_excerpt(); ?>

            <p>
              <a href="<?php the_permalink(); ?>" class="post__link">More</a>
            </p>
    </article>

<?php endwhile; endif; ?>

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

In the dashboard settings, select ***Your latest posts*** in the section *Your homepage displays*, and view the results. Since there are still no menu in the web site, we will need to access the blog from the dashboard menu **Pages > All pages** (hover the Blog page and select **View**). You may also simply **/blog/** at the end of the URL in the browser's address bar. If all worked well you should now see the 10 last posts from those we created using *FakerPress*.

the_post_thumbnail()
Displays the feature image.

the_date() and **the_time()**
As you can probably guess, these functions display the publishing date and time but in usual PHP format. get_options() formats them accordingly with the settings in **Settings > General > Date format**.

the_author()
Displays the author's name as defined in **Users > Your profile > Display name publicly as**.

comments_number()
This function displays the number of comments for a given post. By default, WordPress will display : *No comments*, *1 comment*, or *x comments*. These messages can be customized as you can see below.

```
<?php comments_number( 'no responses', 'one response', '% responses' ); ?>

OR

<?php comments_number( '0', '1', '%' ); ?>
```

the_excerpt()
While *the_content()* shows the entire post, *the_excerpt()* shows a teaser only set in the Document settings of Posts. Visitor needs to click to access the entire post.

the_permalink()
Placed within a <a> tag, this function generates automatically and displays a hyperlink to the post.

In our example, it was used on the word « More », but it could have been used on any of the page's element such as a heading or on the feature image.

Displaying a post (single template)

When the user clicks on an post, the entire post is displayed using the *single.php* template. See below what it could be like.

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

<?php if( have_posts() ) : while( have_posts() ) : the_post(); ?>

        <article class="post">
                <?php the_post_thumbnail(); ?>

                <h1><?php the_title(); ?></h1>

                <div class="post__meta">
                <?php echo get_avatar( get_the_author_meta( 'ID' ), 40 ); ?>
                        <p>
                          Publié le <?php the_date(); ?>
                          par <?php the_author(); ?>
                          Dans la catégorie <?php the_category(); ?>
                          Avec les étiquettes <?php the_tags(); ?>
                        </p>
                </div>

                <div class="post__content">
                        <?php the_content(); ?>
                </div>
        </article>

<?php endwhile; endif; ?>

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

get_avatar()
Retrieves the author's avatar. The first parameter that's needed is the user ID which can be retrieved using the function ***get_the_author_meta('ID')***. The second parameter define the size of the avatar in pixels.

the_category() and **the_tags**()
Displays the categories and labels attached to the post in an unordered list using <ul> and <li> tags. Clicking a category brings the user to a list of posts falling within the given category.

Equivalents of PHP «get»

You will sometimes need to retrieve data (values) to process it in PHP without wanting to display it in the template. For instance, in conditional structure.

Let's say you want to retrieve the title. Using the function *the_title()* won't work. The function ***get_the_title()*** will need to be used.

```
<?php
        $title = get_the_title();

        // If title contains the word "Mac", add a symbol
        if( strpos( $title, 'Mac' ) ) {
                $title = '' . $title;
        }
?>
        <h1><?php echo $title; ?></h1>
```

Remember

Functions starting with «**the_**» display data in the template.

Functions starting with «**get_**» retrieve the data without displaying it.

Template parts

In order to avoid repeating the same codes over an over again from one page to another, it is possible to call sub-templates using the function **get_template_part**(). which can basically content anything you want.

Calling a template part

Let's suppose you created a small newsletter subscription form you want to place in different pages of your theme. What is needed to be done is to code this form into a template part that will be called whenever needed.

Step 1 : create the template part file

For our example, we will create the file *newsletter.php* that could look like the following :

```
<form class="newsletter-form">
<p>Subscribe to our newsletter :</p>
   <input type="email" name="email" placeholder="E-mail">
   <input type="submit" value="I subscribe!">
</form>
```

Step 2 : call the template part from any page

To call a template part in order to include it into any template page wanted, simply use the function **get_template_part**().

```
<?php get_template_part( 'parts/newsletter' ); ?>
```

Note that the extension isn't used while calling the template part. Also, in order to avoid confusion, it is strongly suggested that you group your template parts into a folder (parts, in the example above).

You can even use this function to make your template pages less confusing, like we use include() or require() in PHP to reduce the amount of information.

Useful tip

It is quite useful to use **get_template_part**() to include CSS custom styles in *header.php* so it applies to the entire pages.