A medida que un theme para WordPress se hace más complejo y aumenta el número de templates que utiliza, es habitual que aparezcan **partes que comiencen a repetirse**. Para evitar la duplicidad de código, y todos los inconvenientes que conlleva, podemos **extraer esas partes**, poner cada una en su propio archivo PHP y luego **incluirlos en cuantos templates lo necesitemos**.

Los archivos PHP con el código de esas "partes" se conocen como **template parts** y la función `get_template_part()` se encarga de su inclusión en otros templates. Se podría realizar directamente con `require` o `include`, pero estaríamos restringiendo la flexibilidad del sistema de theming de WordPress, especialmente en lo que respecta a child themes.

`get_template_part()` utiliza internamente `locale_template()` , función que busca primero en el tema hijo, si existe, y luego en el tema padre, por lo que se **permite que los template parts sean sobreescritos en temas hijos**.

Nomenclatura de los template parts

Para utilizar la función `get_template_part()` hay que seguir algunas reglas de nomenclatura para nombrar sus archivos PHP. El nombre de estos archivos puede ser de dos formas:

- **{slug}.php**: conocido como template part genérico
- **{slug}-{name}.php**: conocido como template part especializado

Por ejemplo, podríamos crear un template part con los botones para compartir en redes sociales. El nombre de este template part podría ser `share.php`. En este caso el slug es igual a `share` y name a `null`.

Ahora imagina que quieres mostrar los botones sociales arriba y abajo del contenido y que ambas localizaciones utilizan el mismo código con una ligera variación. Se podría **subdividir el template part general** en otros dos **template parts más especializados**, `share-above.php` y `share-bellow.php`. En este caso el slug seguiría siendo `share` y name sería igual a `above` y `bellow` respectivamente.

¿Cómo utilizar `get_template_part()`?

La sintaxis general es la siguiente:

```
get_template_part ( string $slug, string $name = null )
```

Parámetros

$slug

(string) (requerido) Es el slug, o nombre genérico, del template part. Por ejemplo, si el template

part es content.php, el slug es `content`.

$name

(string) (opcional). Predeterminado `null`. Es el nombe especializado del template part. Por ejemplo, si el template part es content-image.php, name es `image`.

Uso

El uso más básico de `get_template_part()` sería para cargar un template part genérico, por ejemplo `content.php`:

```
get_template_part( 'content' );
```

Para cargar template parts especializados se pasaría el slug y name correspondiente, por ejemplo si el template part es content-image.php

```
get_template_part( 'content', 'image' );
```

Es importante tener en cuenta que **si se utiliza un template part especializado inexistente**, `get_template_part()` intentará **cargar el template part genérico**. Por ejemplo, si content-image.php no existe, se intentará cargar content.php.

En el ejemplo anterior, se seguiría esta secuencia:

1. Se busca content-image.php en el **child theme**
2. Si no existe, se busca content-image.php en el **parent theme**
3. Si no existe, se busca content.php en el **child theme**
4. Si no existe, se busca content.php en el **parent theme**
5. Si ninguno de ellos existe, **NO PASA NADA**. Absolutamente nada; no se genera ningún error ni aviso; **tampoco se carga template alternativo**.

Ejemplo: template parts según el post format y el post type

El ejemplo que venimos utilizando con el template part content.php es muy habitual para **cargar template parts en función del post format**. Por ejemplo, podríamos tener el siguiente loop en un template y cargar template parts tipo content-image.php, content-video.php, content-gallery.php, etc:

```
while ( have_posts() ) {
  the_post();
  // Cargar el template part content-{post_format}.php
  get_template_part( 'content', get_post_format() );
```

```
}
```

De forma similar se podría utilizar para **cargar template parts en función del post type**:

```
while ( have_posts() ) {
  the_post();
  // Cargar el template part content-{post_type}.php
  get_template_part( 'content', get_post_type() );
}
```

También es habitual para diferentes listados de posts o "archives". Por ejemplo, en el template search.php podríamos tener:

```
while ( have_posts() ) {
  the_post();
  get_template_part( 'content', 'search' );
}
```

Y en el template category.php:

```
while ( have_posts() ) {
  the_post();
  get_template_part( 'content', 'category' );
}
```

Cargar template parts en subdirectorios

Los template parts se suelen poner en el directorio raíz del theme o del child theme pero también se pueden poner en subdirectorios. En este caso, como parámetro `$slug` hay que utilizar la **ruta relativa al directorio raíz del theme**.

Por ejemplo, si los template parts content.php y content-image.php están en `/mitheme/partials/` (o `childtheme/partials/`), se incluirían del siguiente modo:

```
// Carga /mitheme/partials/content.php
get_template_part( 'partials/content' );
// Carga /mitheme/partials/content-image.php
get_template_part( 'partials/content', 'image' );
```

Hooks disponibles

Para modificar el comportamiento de `get_template_part()` disponemos del action `get_template_part_{$slug}`. Por ejemplo, si el slug es `content`:

```
add_action( 'get_template_part_content', 'hacer_algo' );
function hacer_algo() {
  // ....
}
```

Ten en cuenta que el action anterior se ejecutaría tanto para content.php como para content-image.php o culquier otro content-{name}.php, ya que todos comparten el slug `content`.

Algunas funciones comunes en el desarrollo de themes, como `get_header()` y `get_footer()`, se pueden entender como **versiones especializadas de** `get_template_part()` y tienen también actions especializados:

```
// Carga header.php y ejecuta el action get_header
get_header();
// Carga header-big.php y ejecuta el action get_header
get_header( 'big' );
// Carga header-big.php SIN ejecutar el action get_header
// Se ejecutaría el action get_template_part_header
get_template_part( 'header', 'big' );
// Carga footer.php y ejecuta el action get_footer
get_footer();
// Carga footer-grid.php y ejecuta el action get_footer
get_footer( 'grid' );
// Carga footer-grid.php SIN ejecutar el action get_header
// Se ejecutaría el action get_template_part_header
get_template_part( 'footer', 'grid' );
```

Con esto llegamos al final. A partir de ahora comienza a utilizar `get_template_part()` en tus themes, ¡qué no se te olvide!