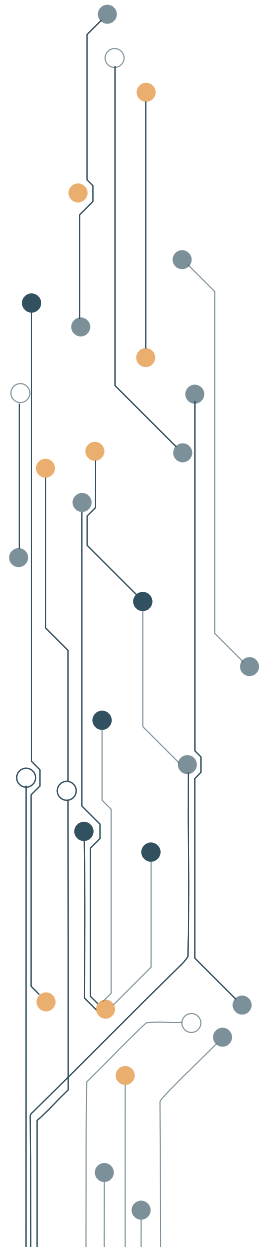
Desarrollar en WordPress

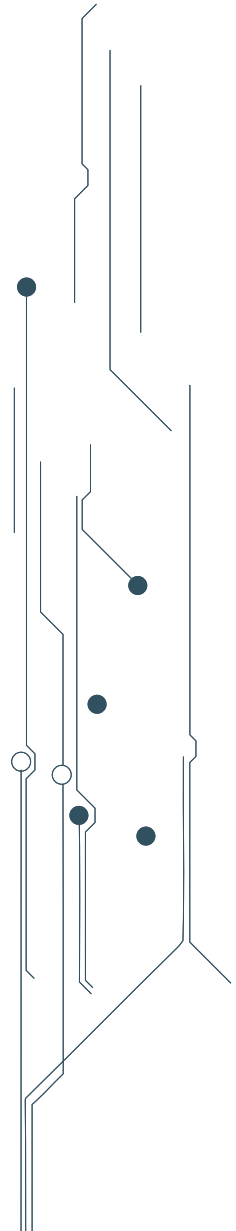
Telefónica
EDUCACIÓN DIGITAL

Índice

Desarrollar en WordPress



Índice (cont.)



Desarrollar en WordPress

1 Recursos

1.1 El núcleo

Casi todo lo que obtenemos al instalar Wordpress es parte del núcleo (core en inglés). La instalación incluye un par de *plugins* y varios *themes* básicos, además de la información de configuración que aportamos durante el proceso; pero todo lo demás es el "motor" de nuestro CMS.

El núcleo de WordPress es la base del CMS, que nos aporta la mayor parte de la lógica y las funcionalidades. No debemos modificarlo, ya que podríamos causar un funcionamiento inesperado. Además, cualquier modificación podría perderse al instalar una nueva actualización.

Sin embargo, conviene acceder a él para comprender su funcionamiento y con él, la forma en la que podemos modificarlo mediante nuestro propio código en los temas y plugins. Casi todo el código está comentado. Muchos archivos incluyen al inicio un bloque de comentario aportando una descripción de lo que hacen y otra información útil. También la mayoría de funciones cuenta con una descripción de lo que hacen así como de los parámetros que esperan y que devuelven o la versión de Wordpress en la que fueron incluidas.

Veamos un par de ejemplos de comentarios en diferentes archivos:

/wp-login.php (información del archivo, al inicio)

```
<?php
/**
 * WordPress User Page
 *
 * Handles authentication, registering, resetting passwords, forgot password,
 * and other user handling.
 *
 * @package WordPress
 */
```

/wp-includes/user.php (función: email_exists)

```
...
/**
 * Checks whether the given email exists.
 *
 * @since 2.1.0
 *
 * @param string $email Email.
 * @return int|false The user's ID on success, and
false on failure.
 */
function email_exists( $email ) {
     if ( $user = get_user_by( 'email', $email) ) {
          return $user->ID;
     }
     return false;
}
...
```

Estos comentarios tienen un formato específico llamado DocBlock[1] de PHPDoc.

PHPDoc es una forma de añadir documentación al código mediante bloques de comentario. Estos pueden ser usados por herramientas como phpDocumentator[2] para generar un archivo con la documentación fuera del propio código.

Dado que la mayor parte de Wordpress está incluida en grandes archivos con multitud de funciones que se llaman unas a otras, conviene aprender a "saltar" fácilmente de a sus definiciones. La mayoría de IDEs y editores de código permiten buscar entre los archivos de un proyecto la definición de una función. Muchos permiten acceder directamente a esa definición usando el menú contextual[3] al poner el cursor sobre una llamada a la función.

Muchos temas y plugins también cuentan con documentación o con un código claro que permite entender su funcionamiento. Sin embargo, debemos estar seguros de su calidad antes de tomarlos por referencia.

1 https://en.wikipedia.org/wiki/PHPDoc#DocBlock
2 https://phpdoc.org/
3 Generalmente accesible al hacer clic con el botón secundario del ratón.

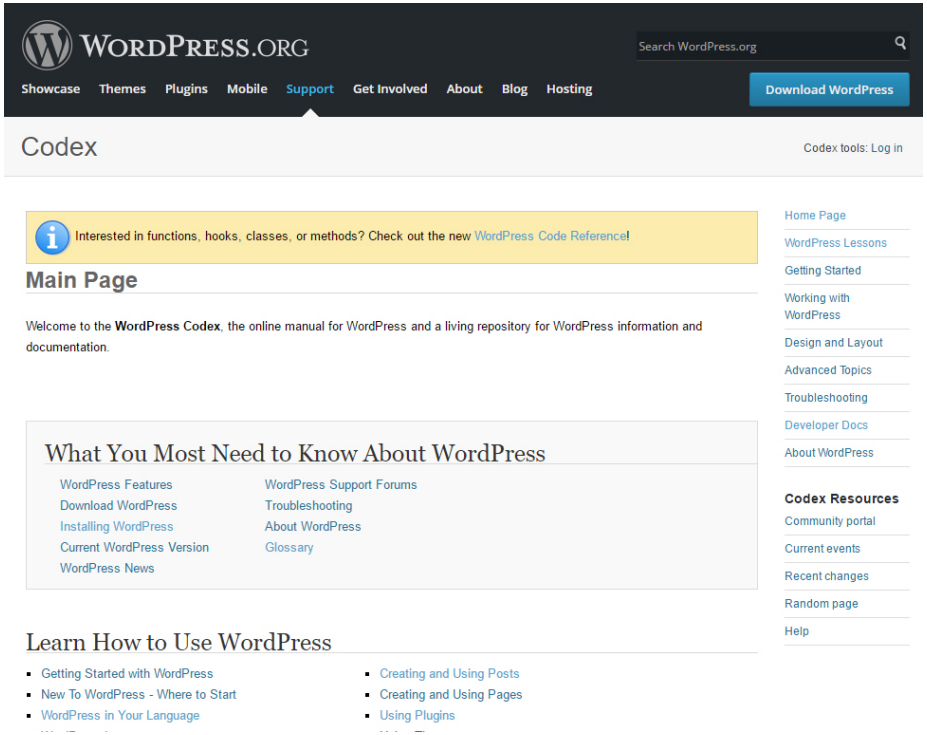
1.2 El Codex

El Codex[4] de WordPress es la documentación online de la plataforma. En ella podemos encontrar artículos a modo de guía que van desde la instalación y el uso básico, a la creación de temas y plugins.

También cuenta con un glosario de términos[5] que puede ser muy útil sobretodo al inicio, para entender muchos de los términos que se usan en la plataforma y en propia documentación.

La web es una wiki mantenida por la comunidad y traducida, al menos parcialmente, a diversos idiomas. Sin embargo, hay que tener en cuenta que a veces la documentación no está todo lo actualizada que debiera. Pese a eso, sigue siendo una referencia ineludible a la hora de trabajar con WordPress.

El Codex cuenta con un buscador para encontrar fácilmente aquello que buscamos, pero hay que tener en cuenta que buscará en todo el sitio de wordpress.org y no solo en el Codex.



Página principal del Codex

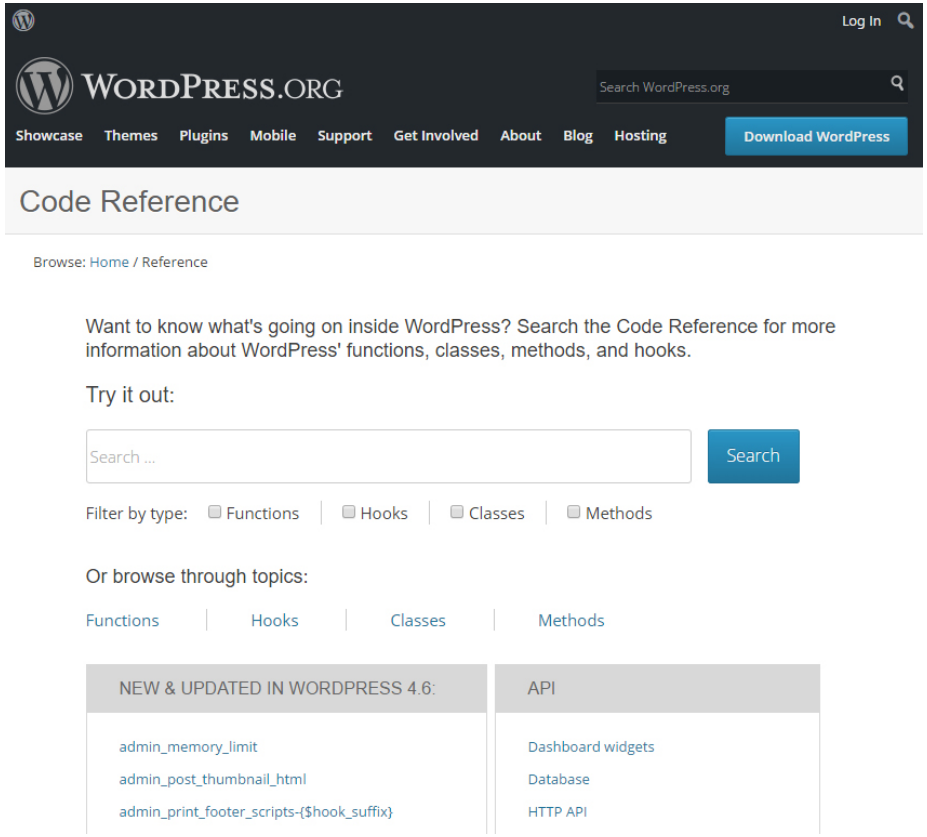
4 https://codex.wordpress.org/
5 https://codex.wordpress.org/Glossary

1.3 Referencia de funciones

La referencia de funciones[6] es el lugar perfecto para buscar información sobre todas las funciones disponibles en Wordpress. Esta web viene a sustituir a la antigua versión[7] que incluía el Codex y que sigue activa y es preferible evitar.

Podremos buscar fácilmente una función y consultar su descripción, los parámetros, el valor devuelto, el changelog[8] y la propia definición de la función. Esta es la misma información que vemos en los archivos PHP, y no es casualidad. Esta referencia está construida a partir de la información del propio código, haciendo uso de PHPDoc, lo que nos asegura que esté siempre actualizada. En ese sentido, será cuestión de comodidad acudir según el caso a esta referencia o al propio código de los archivos.

Sin embargo, la referencia nos aporta unos interesantes "extras": ejemplos de uso aportados por la comunidad y un listado de las partes del código desde la que es llamada la función. Ambos serán recursos muy valiosos.



Portada de la Referencia de Funciones de WordPress

6 https://developer.wordpress.org/reference/
7 https://codex.wordpress.org/Function_Reference/
8 La versión en la que se introdujo la versión y aquellas en las que sufriera cambios (indicando cuales) en caso de que las hubiera.

1.4 Coding Standards

Al tratarse de una plataforma en la que participan un gran número de desarrolladores, es importante buscar la legibilidad y la sencillez del código, a lo que ayuda un estilo común. WordPress cuenta con sus propios estándares de código no solo para PHP sino también para HTML, CSS y JavaScript[9]. Conviene tener presentes todas estas reglas cuando desarrollamos.

Algunos ejemplos de las reglas de PHP son:

Convenciones de nombres

- Los nombres de las variables se escriben en letras minúsculas.
- Las clases comienzan las palabras en mayúsculas.
- Las constantes están escritas íntegramente en mayúsculas.
- En los tres casos anteriores, las palabra se separan mediante guiones bajos ("snake_case").
- En los nombres de archivo las palabras se separan con guiones medios.

Uso de espacios

- Se deben usar espacios a ambos lados de los paréntesis de los condicionales y bucles.
- En los lados interiores de los paréntesis al definir y llamar a funciones.
- A ambos lados de los comparadores.

Condiciones "Yoda"[10]

En las comparaciones lógicas se deben poner las variables en la parte izquierda y las constantes, literales o funciones en el lado derecho. Así, por ejemplo, en caso de que olvidemos un signo de igual (=) al ir a hacer una comparación de igualdad (==), se lanzará un error en lugar de asignarse un valor de forma errónea.

```
if ( true == $var ) {
    $other_var = "value";
}
```

9 Los estándares completos de WordPress pueden consultarse en https://make.wordpress.org/core/handbook/best-practices/coding-standards/

10 Recibe su nombre del personaje de "Star Wars" por la similitud con su peculiar forma de hablar.

2 Cómo funciona WordPress

Pese a la gran variedad de personalización que podemos hacer en WordPress, lo cierto es que casi todas las páginas trabajan de una forma muy similar.

2.1 La base de datos

Además del propio código del núcleo, es interesante ver como está estructurada la base de datos para comprender como se almacenan los datos. Al hacerlo una de las cosas que puede llamar la atención es encontrar una tabla llamada wp_posts pero no encontrar ninguna cuyo nombre nos haga pensar que almacene las páginas.

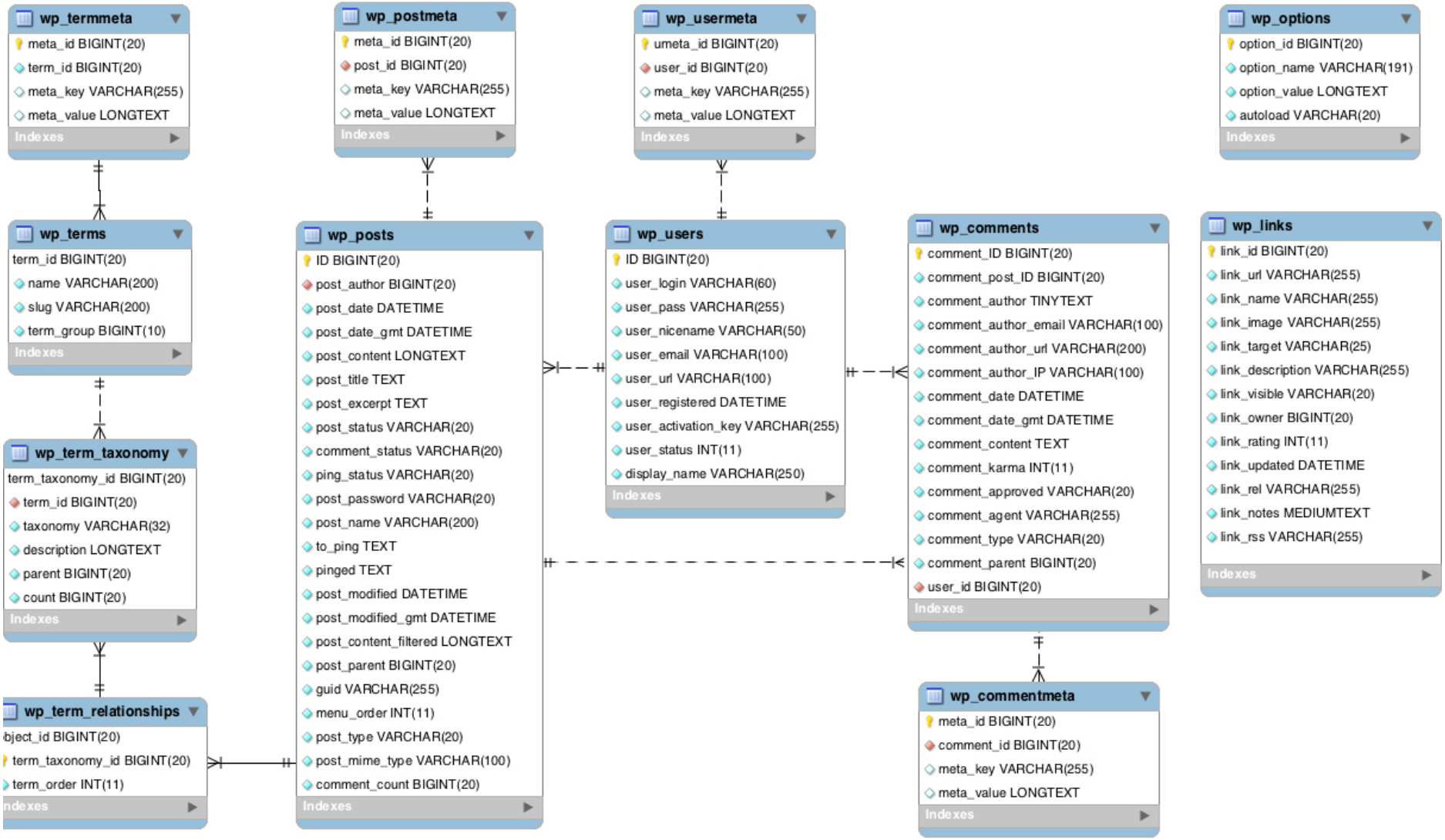
Aunque a nivel de usuario un post y una página sean cosas diferentes, internamente Wordpress las trata del mismo modo. Ambos tipos de contenido se guardan en la misma tabla (wp_posts) con los mismos campos de información. Lo único que los diferencia y permite que puedan ser tratados (y sobre todo, mostrados) de forma diferentes es el campo post_type.

Pero las páginas y las entradas no son dos únicos post_types posibles. Por defecto, Wordpress cuenta con cinco tipos diferentes[11]:

- Entrada/Post ('post')
- Página ('page')
- Adjunto ('attachment')
- Revisión ('revision')
- Elemento de menú ('nav_menu_item')

Comprendida esta diferenciación entre lo que supone un post para el usuario y lo que se entiende internamente como post (en un sentido más genérico, veamos la estructura completa de tablas.

11 A los que , como veremos más adelante, podremos añadir nuevos tipos.



Esquema de la base de datos de WordPress a partir de la versión 4.4.2 (https://codex.wordpress.org/File:WP4.4.2-ERD.png)

Los nombres de las tablas don bastante descriptivos, pero repasemos cada una de las tablas para ver qué contenido almacenan. Ten en cuenta que el prefijo "wp_" no tiene por qué coincidir en tu instalación (de hecho, no debería), sino que tomará el valor que le indicaras en la instalación.

wp_posts

Como hemos comentado, a nivel interno casi todo se almacena como un post. Por tanto, esta es la tabla que contiene la mayor parte de la información de nuestra web.

wp_comments

Almacena los comentarios, relacionándolos con su autor (si estaba logueado), el post comentado y el comentario "padre" en caso de ser una respuesta.

wp_terms

Almacenas las categorías y etiquetas y cualquier otro término relativo a una taxonomía que usemos para clasificar nuestra información.

wp_term_taxonomy

Si la tabla wp_terms almacena todas los términos que usamos para clasificar nuestra información, esta tabla almacena esas diferentes clasificaciones (taxonomías) a la que pueden pertenecer. Por defecto solo tiene dos registros: "category" (categoría) y "post_tag" (etiqueta).

wp_term_relationships

Esta tabla relaciona los términos con las taxonomías a las que pertenecen

wp_users

Contiene la lista de usuarios de nuestra web.

wp_commentmeta, wp_ postmeta, wp_termmeta, wp_usermeta

Contienen metadatos relativos a comentarios, posts, términos de taxonomías y usuarios. Estos metadatos muchas veces son generados por plugins que necesitan almacenar su propia información relativa a estos registros.

wp_options

Almacena las opciones configuradas desde el panel de administración (WP-Admin). Esto incluye las opciones propias del

núcleo como las de los plugins y temas (incluyendo cuales están activados).

wp_links

Antes de la versión 3.5, Wordpress contaba con una sección de links para construir lo que se conoce como un *blogroll*[12]. Sin embargo las versiones actuales han delegado este servicio a un plugin opcional[13], por lo que, salvo que lo instalemos, esta es una tabla que no vamos a usar.

12 https://es.wikipedia.org/wiki/Blogroll
13 https://codex.wordpress.org/Links_Manager

2.2 Query & Loop

El llamado "Loop" es una de las partes principales de las páginas de Wordpress. Es la parte que se usa para mostrar cada uno de los posts[14], independientemente del tipo de página en la que nos encontremos.

Como su nombre indica se trata básicamente de un bucle, que recorre los posts a manejar (ya sean varios o uno solo) y actúa sobre cada uno de ellos de la forma que se precise. Así es como se presenta habitualmente el Loop:

```
<?php if ( have_posts() ) : while ( have_posts() )
: the_post(); ?>
    <!-- Contenido de cada post -->
<?php endwhile; else: ?>
    <!-- Alternativa si no hay post que mostrar -->
<?php endif; ?>
```

La lectura no es lo más clara posible, ya que abre y cierra la etiqueta de php muchas veces y en una sola linea (la primera) llama a una función dentro de un bucle while que a su vez está contenido en un condicional. Veamos su código más claramente.

14 Entendidos en sentido general, no solo "artículos" sino también páginas, etc.

```
<?php
if ( have_posts() ) {
    while ( have_posts() ) {
        the_post();
        // Contenido de cada post
    }
} else {
    // Alternativa si no hay post que mostrar
}
```

El condicional y el bucle en sí no tienen complejidad. Solo es preciso saber lo que hacen las funciones have_posts() y the_post().

/wp-includes/query.php

```
...
function have_posts() {
    global $wp_query;
    return $wp_query->have_posts();
}
```

```
...
function the_post() {
    global $wp_query;
    $wp_query->the_post();
}
```

El nombre have_posts resulta bastante descriptivo y esperamos que nos diga si hay posts que mostrar. Pero al consultar la información de ésta y de la función the_post, vemos que ambas se encuentran definidas en el mismo archivo (wp-includes/query.php) y que están llamando a métodos una variable global llamada $wp_query.

La referencia de estas funciones nos indica donde podemos encontrar eso métodos.[15] [16]

> Related # Top ↑
>
> Uses # Top ↑
>
> wp-includes/query.php: WP_Query::have_posts()

15 https://developer.wordpress.org/reference/functions/have_posts/#uses
16 https://developer.wordpress.org/reference/functions/the_post/#uses

WP_Query

La variable global $wp_query es una instancia de la clase WP_Query que está definida en el mismo archivo wp-includes/query.php. WordPress.

Esta clase de más de 4000 líneas de código y comentarios es la que se encarga de realizar las consultas a base de datos y almacenar el resultado en la variable global $wp_query.

Cuando accedemos a una dirección de WordPress, ya sea para consultar una página estática, leer una entrada específica, o consultar el archivo de entradas para una determinada categoría, WordPress realiza automáticamente la consulta necesaria a la base de datos.

Si, por ejemplo, queremos ver todas las entradas categorizadas como "tecnología", el sistema lo detectará gracias a la url y rescatará los últimos artículos[17] de esa categoría. La propia consulta, sus resultados e información extra se guardan en la variable $wp_query, que queda preparada para ser usada en el "Loop".

En cada iteración del bucle, otra variable global llamada $post va tomando los datos de cada uno de los posts a mostrar[18] permitiendo acceder a ellos fácilmente desde las plantillas.

17 El número de artículos a solicitar a base de datos vendrá determinado por lo indicado desde el WP-Admin.
18 A través del método the_post()

Si estás trabajando con una instalación local de Wordpress puedes probar a imprimir el contenido de $wp_query. También puedes probar a mostrar el valor de $post dentro del bucle (Loop) y como este cambia con cada iteración[19].

Pero la clase WP_Query no solo nos facilita acceder al contenido principal solicitado por el usuario (página/artículo o listado), sino que nos permite realizar cualquier otra consulta en diferentes momentos. Así podremos, por ejemplo, mostrar en la página de un artículo un listado con otros artículos relacionados, o listar las últimas entradas en una barra lateral (sidebar) o en el pie de página[20]. Veamos un ejemplo en el que buscamos

Para conocer todo lo necesario sobre esta clase es conveniente revisar su documentación[21]. Veamos unos ejemplos de uso:

```
// Posts con la etiqueta "codigo" de 5 en 5
$query = new WP_Query( array( 'tag' => 'codigo' ), 'posts_per_page' => 5 );


// páginas hijas de la página con id 42[22]
$query = new WP_Query( array( 'post_type' => 'page', 'post_parent' => 42 ) );


// páginas que no son hijas ordenadas por título
$query = new WP_Query( array( 'post_type' => 'page', 'post_parent' => 0, 'order' => 'ASC', 'orderby' => 'title' ) );
```

Hay que tener en cuenta que al hacer esto, estaremos sobreescribiendo el valor que tuviera $query. Si no queremos cambiar su valor crearemos una nueva variable (con otro nombre).

19 Puedes usar la variable var_dump() en la página index.php de la plantilla que estés usando (ejemplo: wp-content\themes\twentysixteen\index.php). Recuerda devolver después la página a su estado original.
Si estás trabajando en un servidor accesible por los usuarios usa un sistema de logs para consultar esta información sin mostrarla directamente en el navegador.

20 Por ahora solo necesitas tener una idea de lo que hace, pero si mas adelante quieres conocer más a fondo WP_Query puedes leer una serie de artículos dedicados a esta clase en: https://code.tutsplus.com/series/mastering-wp_query--cms-818 (Los primeros de la serie disponen de una traducción al español, accesible desde la cabecera de los artículos)

21 https://codex.wordpress.org/Class_Reference/WP_Query

22 El id de cualquier elemento se puede saber consultando la url en su página de edición de WP-Admin (o consultando la base de datos)

3 Hooks

La forma en la que una pieza de código interactúa con otra en WordPress son los hooks. Los usan los plugins para interactuar con el núcleo de la plataforma o con otros plugins, pero también se usan dentro del propio núcleo para relacionar unas partes con otras. Son eventos disparados por acciones y filtros, que nos permiten asociar nuestras propias funciones.

3.1 Acciones (Actions)

Las acciones nos permiten agregar o quitar código en diferentes puntos de la ejecución. Para añadir nuestro propio código en una determinada acción usaremos la función add_action[23] indicando el nombre del filtro al que se "enganchará" y el nombre de la función que se ejecutará.

Como parámetros adicionales podemos indicar la prioridad (para indicar el orden de ejecución en caso de que haya varios funciones asociadas al filtro), y el número de argumentos que aceptará la función.

```
add_action( string $tag, callable $function_to_add,
int $priority = 10, int $accepted_args = 1 )
```

El listado completo de acciones disponibles en el núcleo está incluido en la documentación[24].

Veamos como podríamos, mediante la acción save_post[25], enviar un email al administrador cada vez que alguien añada o edite un post.

```
add_action( 'save_post', 'myplugin_save_post', 10,
3 );
function wpdocs_my_save_post( $post_ID, $post,
$update ) {
    // Enviar email
}
```

23 https://developer.wordpress.org/reference/functions/add_action/

24 https://codex.wordpress.org/Plugin_API/Action_Reference

25 https://developer.wordpress.org/reference/hooks/save_post/

3.2 Filtros (Filters)

Los filtros nos permiten modificar los datos que procesa Wordpress. Al contrario que las acciones, no nos permiten añadir ni eliminar código. Solo podemos reemplazar datos (generalmente variables).

Podemos añadir nuestro propio código a un filtro mediante la función add_filter[26], que es llamada de la misma forma que add_action, pero que debe asociarse a un filtro en lugar de a una acción

```
add_filter( string $tag, callable $function_to_add,
int $priority = 10, int $accepted_args = 1 )
```

El listado íntegro de filtros de WordPress puede consultarse en la documentación[27].

Por ejemplo, podemos tomar las clases que tiene la etiqueta <body> de una página de WordPress y filtrarlas, añadiendo o eliminando alguna clase.

```
add_filter( 'body_class', 'myplugin_add_body_class'
);
function myplugin_add_body_class($classes) {
        $classes[] = "weekday" . date("w");
        return $classes;
}
```

La función del ejemplo se asocia al *filter hook* llamado body_class[28]. Este hook pasa a nuestra función el listado de clases de la etiqueta <body> por medio del argumento $classes. Y nuestra función agrega una clase extra que indica el día de la semana en el que nos encontramos[29].

26 https://developer.wordpress.org/reference/functions/add_filter/
27 https://codex.wordpress.org/Plugin_API/Filter_Reference

28 https://developer.wordpress.org/reference/hooks/body_class/
29 La función date de PHP devuelve la fecha/hora actual formateada según el argumento que pasemos. En este caso, dado que la "w" referencia al día de la semana en formato decimal, obtendremos un número entre el 0 (domingo) y el 6 (sábado). Más información: http://php.net/manual/es/function.date.php

4 Dónde añadir nuestro código

Cada vez que vayamos a añadir código a nuestro WordPress debemos plantearnos dónde debe ir ese código. En ocasiones, hay varios sitios donde podemos colocar nuestro código para que funcione. Es importante conocer las diferencias entre extender nuestro Wordpress de una forma u otra.

Incluso aunque estemos siguiendo un manual, debemos plantearnos si la forma en la que se actúa en él es la más práctica para nosotros en nuestra situación. Veamos las ubicaciones habituales para nuestro código en WordPress:

functions.php

Este archivo forma parte de cada tema de wordpress. Nos permite extender funcionalidad, pero debemos tener en cuenta que, al tratarse de un archivo dependiente de un tema específico, las modificaciones que hagamos se perderán si se cambiamos de tema. Probablemente esta sea la mejor opción cuando añadamos algo con un fuerte componente visual, muy asociado al tema en concreto. Hay que tener en cuenta que si cometemos algún error en este archivo es probable que no podamos acceder a nuestro wordpress (nos aparecerá la página en blanco)

Plugins

Los plugins son independientes de los temas, por lo que si queremos añadir un comportamiento y que este se pueda mantener independientemente del tema que usemos probablemente esta sea una opción a considerar. Hay que tener en cuenta que los plugins pueden activarse y desactivarse a voluntad en el panel de administración, lo cual aporta un control extra. Si cometemos algún error en el código de un plugin, wordpress lo deshabilitará pero la web funcionará sin problema.

Existe un tipo de plugins especial que no aparecen en el listado junto con el resto y por lo tanto no pueden desactivarse. Estos plugins de uso obligatorio (*Must Use Plugins*)[30] se instalan y activan al copiarlos en el directorio "wp-content/mu-plugins".

30 https://codex.wordpress.org/Must_Use_Plugins

Telefónica

EDUCACIÓN DIGITAL