Trabajar con campos personalizados en WordPress sin utilizar Plugins

Teléfono: 900 103 253
www.hostalia.com

En alguna ocasión, hemos comentado que WordPress se ha convertido en los últimos años en el CMS preferido para bloggers y desarrolladores de sitios web. Uno de los principales motivos son los miles de plugins desarrollados por terceras personas o empresas que aportan nuevas funcionalidades a la herramienta, pero además de esto, también hay que destacar una serie de características pensadas para desarrolladores y que permiten la personalización del sitio que se está desarrollando con WordPress. Dentro de estas características, podemos destacar el uso de custom fields o campos personalizados, que permiten añadir nuevos campos de recogida de información a la hora de publicar una nueva entrada.

Hay plugins desarrollados que nos facilitan el trabajo con los campos personalizados, pero como no es bueno abusar de estos añadidos, hoy os vamos a explicar cómo crear vuestros propios custom fields desde cero, mostrando cómo almacenar la información y recuperarla para mostrarla en el sitio que necesitemos.

¿Qué son los campos personalizados?

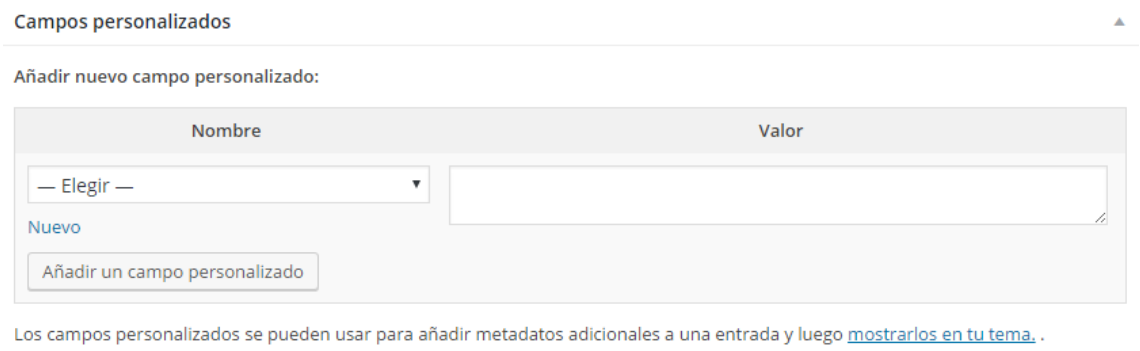
Quien haya trabajado con WordPress se habrá percatado de que cuenta con una serie de campos por defecto como son el nombre de la entrada, el slug, la fecha de publicación, las categorías, los tags... Estos datos suelen ser suficientes pero hay ocasiones en las que se pueden quedar cojos, necesitando utilizar campos extras.

Este CMS permite hacer uso de campos personalizados para conseguir esa otra información que por defecto WordPress no permite. Esto le aporta una gran potencia ya que cualquier persona puede crear

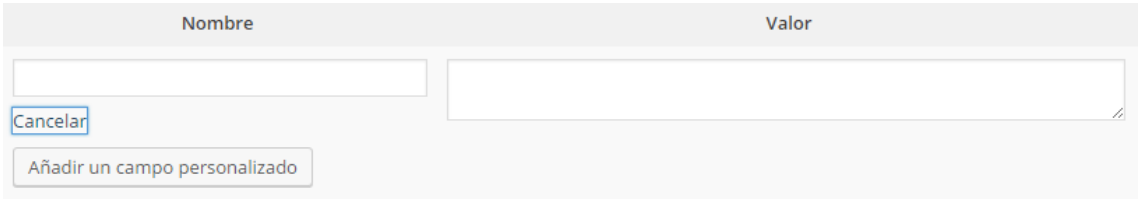
todos los campos que necesite sin mucho esfuerzo. Estos datos pueden ser utilizados posteriormente para mostrarlos por pantalla, para crear filtros o para cualquier cosa que se nos pase por la cabeza.

Pongamos un ejemplo. Si queremos desarrollar una página web sobre libros en nuestro alojamiento web, el usuario podría estar interesado en guardar el ISBN de la publicación y el nombre del autor. Por defecto estos campos no aparecerían en la instalación básica de WordPress, pero gracias al uso de los campos personalizados podemos crearlos y adaptar el desarrollo a nuestras necesidades.

Toda instalación de WordPress ofrece por defecto este tipo de campos, aunque su uso no es todo lo cómodo que pudiéramos esperar. Los podemos encontrar dentro de cada entrada, a continuación del editor de texto, en una sección con el nombre de "**Campos personalizados**" (si no aparece ve a “Opciones de pantalla” y activa la casilla).



Desde esa sección, podemos crear nuevos campos pulsando en la opción "Nuevo".



Como vemos en la imagen, nos pedirán el nombre del campo y el valor que le queremos asignar. Tras guardarlo, en la base de datos se almacenará en forma de pares del tipo key - value, datos que luego podrán ser recuperados.

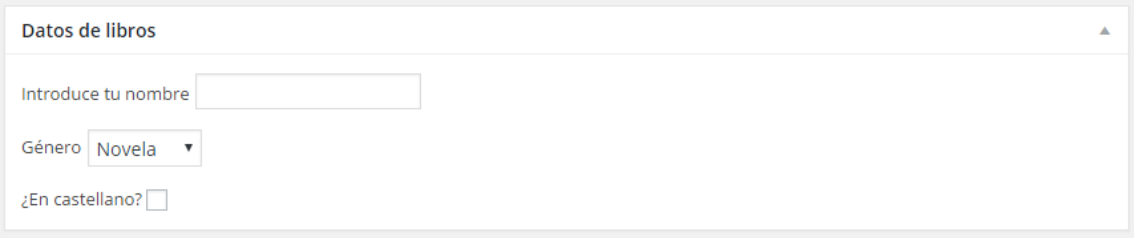
El problema de este sistema básico es que todo se mostrará de la misma forma, pero si queremos crear un campo personalizado de tipo checkbox o lista desplegable, no lo podemos hacer así, sino que deberemos utilizar los custom fields y los meta boxes, que será en lo que nos centraremos en el punto siguiente.

Cómo crear campos personalizados sin utilizar plugins

Ya hemos comentado que crear nuevos campos personalizados es muy sencillo, lo único que necesitamos es tener algunos conocimientos de PHP y HTML, además de no tener miedo a tocar dentro de nuestra instalación de WordPress, más concretamente dentro del archivo **functions.php** que hay dentro de la plantilla. De todas formas, os recomendamos que os hagáis un backup del contenido para poder volver hacia atrás en caso de surgir algún problema.

Antes de empezar el proceso de creación, es importante saber conocer que los diferentes campos de texto utilizados para la introducción de contenido están agrupados en secciones que se conocen como meta boxes y que utilizaremos para crear nuestro ejemplo. Dicho esto, es el momento de empezar con los pasos para su creación.

1.- Registrar los campos personalizados que vamos a crear



Lo primero que haremos, será registrar los campos personalizados que vamos a crear. En nuestro caso nos basaremos en el ejemplo del registro de libros, donde queremos guardar el autor del libro, el género al que pertenece y si está traducido al castellano. Son los campos que podéis ver en la imagen que acompaña a estas líneas.

Para ello debemos utilizar el método "**register_meta**" que ofrece WordPress y cuya estructura es la siguiente.

register_meta($meta_type, $meta_key, $sanitize_callback, $auth_callback);

El significado de cada uno de los parámetros es el siguiente:

- **$meta_type**: Sirve para indicar el tipo de objeto para el que va destinado el nuevo dato. Entre sus valores nos podemos encontrar post, user, category...
- **$meta_key**: Aquí se indicará el nombre del custom field que estamos creando
- **$sanitize_callback**: Se trata del nombre de la función que se debe ejecutar antes de que el valor sea guardado en base de datos para eliminar todos aquellos caracteres que puedan dar problemas, por ejemplo el uso de código HTML
- **$auth_callback**: Es el único parámetro opcional y sirve para indicar el nombre de una función que se encargará de comprobar si el usuario tiene permisos suficientes para añadir, modificar o borrar la información de esos campos

En nuestro ejemplo, el código necesario para registrar esos tres campos sería el siguiente.

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

function libros_register_meta_fields() {

    register_meta( 'post', 'libros_autor', 'sanitize_text_field',
    'libros_custom_fields_auth_callback' );

    register_meta( 'post', 'libros_genero', 'sanitize_text_field',
    'libros_custom_fields_auth_callback' );
```

```
        register_meta( 'post', 'libros_traduccion', 'cyb_sanitize_text_field',
        'libros_custom_fields_auth_callback' );

}
```

Si nos fijamos, al principio del código nos encontramos la llamada al método:

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

Con esto estamos diciendo que cuando se inicie WordPress, se haga la llamada al método encargado de registrar los campos personalizados.

2.- Registro del meta box

Como ya hemos comentado anteriormente, cada grupo de campos se organiza en cajas conocidas como meta boxes. Lo que habrá que hacer es registrar la nuestra donde queremos que aparezcan los campos que estamos creando. Para ello, debemos hacer uso de la función de WordPress **add_meta_box** y cuya estructura es la siguiente.

```
add_meta_box( $id, $title, $callback, $post_type, $context, $priority, $callback_args );
```

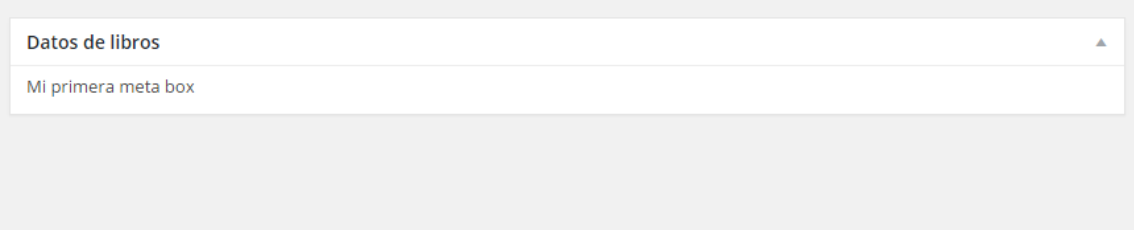
El significado de cada uno de los parámetros es el siguiente:

- **$id**: Se pueden utilizar tanto letras como números y sirve para indicar el valor que tendrá el ID de la sección meta box que vayamos a crear
- **$title**: Mediante este atributo indicaremos el título que tendrá nuestro meta box
- **$callback**: Aquí indicaremos el nombre de la función encargada de mostrar el código HTML que irá dentro del meta box
- **$post_type**: Puede tomar el valor de "post", "page" o "custom-post", y sirve para indicar en qué tipo de entradas aparecerá el meta box creado. Por ejemplo, si ponemos "post" sólo aparecerá en la zona de entradas, pero no en las páginas
- **$context**: Sirve para indicar la posición donde queremos que aparezca el meta box. Puede tomar el valor "normal" o "advanced" para que la caja aparezca en la misma columna que el editor donde escribimos, o "side" si queremos que aparezca en el lateral
- **$priority**: Sirve para indicar la altura donde queremos que aparezca el meta box, aunque esta posición dependerá también de otros plugins. Sus posibles valores son "core", "low", "high" o "default"
- **$callback_args**: Es el único parámetro opcional y en él podemos pasar parámetros que serán utilizados dentro del método indicado en el parámetro $callback

Un ejemplo de uso sería el siguiente:

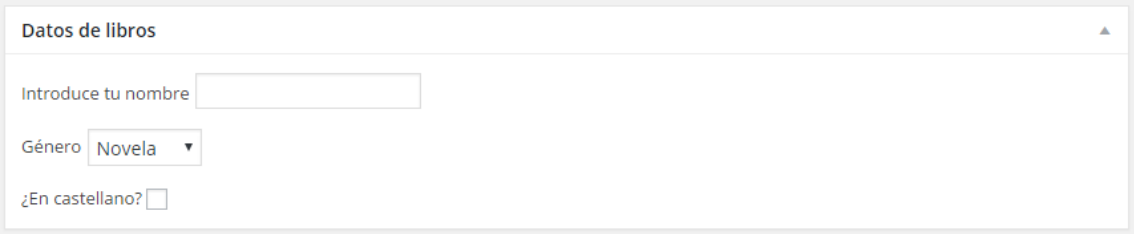
```
add_action( 'add_meta_boxes', 'libros_meta_boxes' );

function libros_meta_boxes() {

   add_meta_box( 'libros-meta-box', __( 'Datos de libros', 'libros_textdomain' ),
'libros_meta_box_callback', 'post', 'side', 'high', array( 'arg' => 'value') );

}
```

```
function libros_meta_box_callback( $post ) {
    echo 'Mi primera meta box';
}
```



En la imagen superior, podéis ver el resultado de la ejecución de ese código.

3.- Generar el HTML del meta box



Una vez que sabemos cómo generar las cajas meta box, el siguiente paso será poner el código HTML encargado de pintarlas. Todo esto se realiza dentro del método que hemos llamado dentro del parámetro $callback de la función "**add_meta_box**" y que en nuestro ejemplo hemos llamado "libros_meta_box_callback".

El proceso es tan sencillo como el de crear un formulario para cualquier página web, ya que se tratan de campos normales, como pueden ser input, select, radio button, checkbox, textarea... Para conseguir la apariencia de la imagen que hemos dejado más arriba, el código que deberíamos poner en el interior del método "libros_meta_box_callback" sería el siguiente.

```
function libros_meta_box_callback( $post ) {
    ?>
    <p>
      <label class="label" for="libros_autor"><?php _e( 'Introduce tu nombre', 'cyb_textdomain' ); ?></label>
      <input  name="libros_autor" id="libros_autor" type="text" value="">
    </p>
    <?php
```

```
    ?>
    <p>
      <label class="label" for="libros_genero"><?php  _e( 'Género', 'libros_textdomain' );
?></label>
      <select name="libros_genero" id="libros_genero">
        <option value="Novela"><?php _e( 'Novela', 'libros_textdomain' ); ?></option>
        <option value="Histórica" ><?php _e( 'Histórica', 'libros_textdomain' ); ?></option>
        <option value="Infantil"><?php _e( 'Infantil', 'libros_textdomain' ); ?></option>
      </select>
    </p>
    <?php
    ?>
    <p>
                  <label class="label" for="libros_castellano"><?php  _e( '¿En castellano?',
'libro_textdomain' ); ?></label>
      <input type="checkbox" name=" libros_traduccion " id=" libros_traduccion " value="1">
    </p>
    <?php
}
```

A cada uno de los campos le indicaremos un nombre identificativo que utilizaremos para recuperar el valor y guardarlo en base de datos.

Cómo almacenar y recuperar los valores de los campos personalizados

En el punto anterior hemos visto los pasos que debemos seguir para crear nuestra estructura de campos personalizados, una estructura que puede ser todo lo compleja que queramos, añadiendo los campos que necesitemos. Lo siguiente que debemos hacer es conocer los pasos que debemos seguir para almacenar la información en la base de datos y luego recuperarla para mostrarla a los visitantes.

1.- Guardar la información

Cada custom field se recibe en el servidor como **$_POST['nombre_del_campo']**, por ejemplo, para el campo que tenía como nombre "libros_autor", su valor se recuperaría con $_POST['libros_autor'].

Para almacenar los valores, lo que utilizaremos será la función "**update_post_meta**", cuya estructura es la siguiente:

```
update_post_meta($id, $key, $value);
```

El significado de cada uno de los parámetros es el siguiente:

- **$id**: Corresponde con el ID del post al que pertenece. El valor suele ser $post_id, una variable de WordPress donde está almacenada esa información
- **$key**: Nombre del campo que suele coincidir con el nombre del campo HTML que queremos almacenar
- **$value**: Valor que vamos a guardar en la base de datos

Un ejemplo de esto sería el siguiente código:

```
update_post_meta( $post_id, 'libros_autor', $_POST['libros_autor'] );
```

2.- Recuperar los valores

Recuperar los valores almacenados en la base de datos es un proceso sencillo ya que WordPress ofrece varias funciones para ello, que podemos utilizar en cualquier parte de nuestra plantilla. Los métodos más importantes son:

- **get_post_custom()**: Devuelve un array con todos los valores de los campos personalizados de una determinada entrada
- **get_post_meta()**: En este caso, nos permite recuperar algún campo en particular

Un ejemplo de uso de "**get_post_custom()**" sería el siguiente:

```
$campos_personalizados = get_post_custom();

echo $campos_personalizados['libros_autor'];
```

Lo que hemos hecho ha sido recuperar todos los campos personalizados del post y asignarlos al array $campos_personalizados. Luego solo debemos recuperar el valor del campo que nos interesa por medio de su nombre.

Si por el contrario queremos recuperar un valor de un campo mediante el uso de la función "**get_post_meta**", el proceso sería el siguiente:

```
$nombreAutor = get_post_meta(get_the_ID(), 'libros_autor', true);

echo $nombreAutor;
```

En este caso, a ese método hay que indicarle el ID de la entrada cuyo dato queremos recuperar y que lo hacemos mediante el uso del método **get_the_ID()** que ofrece WordPress, el nombre del campo y el valor "true", con el que indicamos que se trata de un dato simple.

A lo largo de este WhitePaper hemos visto todos los pasos que debemos seguir para ser capaces de crear nuestros campos personalizados de una forma más profesional, campos con los que poder recoger toda aquella información que necesitemos en nuestros desarrollos. De todas formas, si preferimos hacer uso de plugins, siempre podemos utilizar algunos ya desarrollados como pueden ser Advanced Custom Fields o Custom Field Suite.