



EBook Gratis

APRENDIZAJE WordPress

Free unaffiliated eBook created from
Stack Overflow contributors.

#wordpress

Tabla de contenido

Acerca de.....	1
Capítulo 1: Empezando con WordPress.....	2
Observaciones.....	2
Versiones.....	3
Examples.....	5
Introducción a WordPress.....	5
Temas de WordPress.....	5
Mapeo de URLs a plantillas específicas.....	5
Tema básico estructura de directorio.....	6
Ejemplo de un "Single" (plantilla para una publicación individual).....	6
Ejemplo de un "Archivo" (plantilla para una lista de publicaciones múltiples).....	7
Publicaciones, páginas, tipos de publicaciones personalizadas y campos personalizados.....	7
Capítulo 2: ¿Cómo puedo integrar el editor de Markdown con el complemento del repetidor de... 9	9
Examples.....	9
Añadir MarkDown Editor.....	9
Capítulo 3: Acciones y Filtros.....	10
Sintaxis.....	10
Parámetros.....	10
Observaciones.....	10
Examples.....	13
add_action - init.....	13
add_action - init - función anónima.....	13
add_action - init - dentro del objeto de clase.....	14
add_action - init - dentro de la clase estática.....	14
Capítulo 4: Actualizar WordPress manualmente.....	15
Examples.....	15
VIA FTP.....	15
Capítulo 5: add_action ().....	16
Sintaxis.....	16

Parámetros.....	16
Examples.....	16
Función directa de devolución de llamada.....	16
Llamada de referencia de nombre de función.....	16
Devolución de llamada de método estático de clase.....	17
Método de devolución de llamada de método.....	17
Capítulo 6: add_editor_style ().....	18
Introducción.....	18
Sintaxis.....	18
Parámetros.....	18
Examples.....	18
Cargando un solo archivo css.....	18
Capítulo 7: add_menu_page ().....	19
Introducción.....	19
Sintaxis.....	19
Parámetros.....	19
Observaciones.....	19
Examples.....	20
Agregar el elemento "Título de la página del tema" a la barra de navegación.....	20
OOP y cómo cargar scripts / estilos en la página del menú.....	21
Capítulo 8: add_submenu_page ().....	23
Introducción.....	23
Sintaxis.....	23
Parámetros.....	23
Observaciones.....	23
Examples.....	24
Agregar la "Página del submenú" como una subpágina de "Herramientas" a la barra de navegac.....	24
Capítulo 9: add_theme_support ().....	26
Introducción.....	26
Sintaxis.....	26
Parámetros.....	26

Observaciones.....	26
Examples.....	26
Agregando soporte de tema para formatos de post.....	26
Agregar soporte de tema para las miniaturas de las publicaciones.....	26
Capítulo 10: Admin Dashboard Widgets.....	28
Introducción.....	28
Sintaxis.....	28
Parámetros.....	28
Examples.....	28
Widget simple (muestra texto).....	28
Capítulo 11: Agregar / eliminar información de contacto para usuarios con gancho de filtro.....	30
Examples.....	30
Habilitar las redes sociales más populares.....	30
Eliminar el método de contacto.....	31
Capítulo 12: AJAX.....	32
Examples.....	32
Solicitud AJAX con respuesta JSON.....	32
AJAX con .ajax () y WordPress Nonce.....	33
wp_ajax - funcionalidad principal + _wpnonce check.....	34
OOP ajax presentación utilizando una clase simple con nonce.....	35
Capítulo 13: Añadir Shortcode.....	39
Sintaxis.....	39
Parámetros.....	39
Observaciones.....	39
Examples.....	39
Código corto simple para la publicación reciente.....	39
Código abreviado avanzado para publicaciones recientes.....	40
Capítulo 14: API de opciones.....	41
Introducción.....	41
Sintaxis.....	41
Observaciones.....	41

Examples.....	41
get_option.....	42
add_option.....	42
delete_option.....	42
update_option.....	42
Capítulo 15: API REST.....	43
Introducción.....	43
Observaciones.....	43
Examples.....	44
Ejemplo de trabajo completo.....	44
Capítulo 16: Asegure su instalación.....	46
Observaciones.....	46
Examples.....	46
Deshabilitar el editor de archivos.....	46
Mueve wp-config.php.....	46
Establecer un prefijo personalizado para las tablas de WordPress.....	47
Capítulo 17: Barras laterales.....	52
Sintaxis.....	52
Parámetros.....	52
Observaciones.....	52
Examples.....	52
Registro de barras laterales.....	53
Obtener barra lateral.....	53
Capítulo 18: Bucle principal alternante (filtro pre_get_posts).....	54
Sintaxis.....	54
Parámetros.....	54
Observaciones.....	54
Examples.....	54
Aún más específica la orientación de bucle.....	54
Mostrar publicaciones de una sola categoría.....	55
Filtrar el uso básico antes de las entradas.....	55
Excluir categoría de la lista de publicaciones editar compartir.....	55

Cambiar posts_per_page por bucle principal.....	56
Apuntando solo al bucle principal de WordPress.....	56
Capítulo 19: Código corto.....	57
Examples.....	57
Registro de shortcode.....	57
Usando Shortcodes en el Backend de WordPress.....	57
Añadiendo nuevos códigos cortos.....	57
Uso de códigos cortos dentro del código PHP (temas y complementos).....	58
Usando Shortcodes en Widgets.....	58
Capítulo 20: Código corto con atributo.....	59
Sintaxis.....	59
Parámetros.....	59
Observaciones.....	59
Examples.....	59
Ejemplos de códigos cortos.....	59
Creando un shortcode de cierre automático.....	60
Creación de un shortcode de cierre automático con parámetros.....	60
Creando un shortcode adjunto.....	60
Shortcodes en Widgets.....	61
Capítulo 21: Códigos cortos.....	63
Examples.....	63
Introducción al código corto.....	63
Button shortcode.....	63
Capítulo 22: Conceptos básicos del personalizador (Agregar panel, Sección, Configuración, ...	65
Examples.....	65
Añadir un panel de personalización.....	65
Agregar una sección de personalizador con la configuración básica y sus controles.....	65
Capítulo 23: Consulta de mensajes.....	69
Sintaxis.....	69
Parámetros.....	69
Observaciones.....	69
Nunca use query_posts ().....	69

Examples.....	70
Usando el objeto WP_Query ().....	70
Usando get_posts ().....	70
Capítulo 24: Creación de plugins de WordPress.....	72
Introducción.....	72
Examples.....	72
Configuración mínima de una carpeta de plugin y archivos.....	72
Capítulo 25: Creando una plantilla personalizada.....	74
Examples.....	74
Creando plantilla básica en blanco.....	74
Incluyendo encabezado y pie de página en nuestra plantilla.....	75
Plantilla personalizada con contenido.....	76
Capítulo 26: Crear plantilla para tipo de publicación personalizada.....	78
Examples.....	78
Creación de una plantilla personalizada para libro de tipo Publicación personalizada.....	78
Plantillas personalizadas de tipo de publicación.....	78
Archivo de tipo de publicación personalizado:.....	78
Tipo de entrada personalizado Plantilla única:.....	80
Capítulo 27: Crear una publicación programáticamente.....	82
Sintaxis.....	82
Parámetros.....	82
Observaciones.....	82
Argumentos.....	82
Evitar publicaciones duplicadas.....	84
Explicación.....	84
Examples.....	84
Introducción.....	84
Crear una publicación básica.....	85
Crear una página básica.....	85
Capítulo 28: Depuración.....	86
Introducción.....	86

Observaciones.....	86
Examples.....	86
WP_DEBUG.....	86
WP_DEBUG_LOG.....	86
WP_DEBUG_DISPLAY.....	86
SCRIPT_DEBUG.....	87
Guardias.....	87
Ejemplo wp-config.php y buenas prácticas para la depuración.....	87
Ver registros en un archivo separado.....	88
Capítulo 29: Desarrollo de plugins.....	89
Sintaxis.....	89
Parámetros.....	89
Observaciones.....	89
Examples.....	90
Filtrar.....	90
Acción.....	90
Ejemplos de desarrollo de plugins: Widget de canción favorita.....	90
Capítulo 30: Ejecutar WordPress local con XAMPP.....	93
Introducción.....	93
Examples.....	93
1. Instalando XAMPP.....	93
2. Configurar una base de datos después de instalar XAMPP.....	93
3. Instalar WordPress después de configurar la base de datos.....	93
Capítulo 31: El Loop (bucle principal de WordPress).....	95
Examples.....	95
Estructura básica del bucle de WordPress.....	95
Sintaxis de bucle alternativa.....	95
Manejando ningunos artículos en el bucle.....	95
Capítulo 32: El objeto \$wpdb.....	97
Observaciones.....	97
Examples.....	97
Seleccionando una variable.....	97

Seleccionando multiples filas.....	97
Capítulo 33: el título()	99
Introducción.....	99
Sintaxis.....	99
Parámetros.....	99
Observaciones.....	99
Examples.....	99
Uso simple de the_title.....	99
Imprimiendo el título con código antes y después.....	99
Capítulo 34: Eliminar saltos de línea automáticos del contenido y extracto	101
Introducción.....	101
Observaciones.....	101
Examples.....	101
Quitar los filtros.....	101
Función para eliminar los filtros.....	101
Capítulo 35: Eliminar versión de Wordpress y hojas de estilo	103
Introducción.....	103
Sintaxis.....	103
Parámetros.....	103
Observaciones.....	103
Examples.....	103
Eliminar números de versión de css / js.....	103
Eliminar números de versión de WordPress.....	104
Capítulo 36: en eso	105
Sintaxis.....	105
Observaciones.....	105
Examples.....	105
Procesando \$_POST datos de solicitud.....	105
Procesando \$_GET datos de solicitud.....	105
Registro de un tipo de mensaje personalizado.....	105
Capítulo 37: Encolando guiones	106

Sintaxis.....	106
Parámetros.....	106
Examples.....	106
Poner en cola los scripts en functions.php.....	106
Encolar scripts solo para IE.....	107
Encolando scripts condicionalmente para páginas específicas.....	107
Capítulo 38: Estilos de puesta en cola.....	109
Sintaxis.....	109
Parámetros.....	109
Examples.....	109
Incluyendo archivo css interno con otro archivo css como una dependencia.....	109
Incluyendo archivo css interno.....	109
Incluyendo archivo css externo.....	110
Encolar hojas de estilo solo para IE.....	110
Incluyendo archivo css interno para su clase de Plugin.....	110
Añadir hojas de estilo alternativas.....	110
Capítulo 39: Extractos personalizados con excerpt_length y excerpt_more.....	112
Examples.....	112
Limitar la longitud del extracto a 50 palabras.....	112
Adición de un enlace Leer más al final del extracto.....	112
Agregando algunos puntos al final del extracto.....	113
Capítulo 40: Función: add_action ().....	115
Sintaxis.....	115
Parámetros.....	115
Observaciones.....	115
Examples.....	115
Gancho de acción básico.....	115
Prioridad de gancho de acción.....	116
Enganchar métodos de clase y objeto a acciones.....	116
Método del objeto Acción ganchos.....	117
Método de clase de ganchos de acción.....	118
Capítulo 41: Función: wp_trim_words ().....	120

Sintaxis.....	120
Parámetros.....	120
Examples.....	120
Recortar contenido de la publicación.....	120
Capítulo 42: Fundamentos del tema infantil.....	121
Sintaxis.....	121
Observaciones.....	121
Examples.....	121
2) El propósito.....	121
Definición y requisitos.....	122
3) sobrescritura de plantillas.....	122
Reemplazo de activos.....	123
Capítulo 43: get_bloginfo ().....	125
Introducción.....	125
Sintaxis.....	125
Parámetros.....	125
Observaciones.....	125
Examples.....	127
Obtener el título del sitio.....	127
Obtener el lema del sitio.....	128
Obteniendo la URL del tema activo.....	129
Obtener url del sitio.....	130
Obtenga la dirección de correo electrónico del administrador del sitio.....	130
Capítulo 44: get_home_path ().....	131
Introducción.....	131
Parámetros.....	131
Observaciones.....	131
Diferencia importante entre get_home_path() y ABSPATH.....	131
Usandolo en tu codigo.....	132
Examples.....	132
Uso.....	132

Capítulo 45: <code>get_option ()</code>	133
Introducción	133
Sintaxis	133
Parámetros	133
Observaciones	133
Examples	133
Mostrar título del blog	133
Nombre del blog en estilo H1	134
Mostrar conjunto de caracteres	134
Manejo de opciones no existentes	134
Capítulo 46: <code>get_permalink ()</code>	135
Introducción	135
Sintaxis	135
Parámetros	135
Observaciones	135
Examples	135
Uso simple de <code>get_permalink</code>	135
Especificando la publicación para obtener el enlace	135
Obtén el enlace sin el nombre del post	136
Capítulo 47: <code>get_template_part ()</code>	137
Sintaxis	137
Parámetros	137
Examples	137
Cargar una parte de la plantilla desde una subcarpeta	137
Obtener un archivo específico	137
Capítulo 48: <code>get_template_part ()</code>	138
Introducción	138
Sintaxis	138
Parámetros	138
Examples	138
Incluyendo una plantilla personalizada	138

Incluyendo una plantilla personalizada con un nombre de archivo separado por guión	138
Incluyendo una plantilla personalizada desde dentro de un directorio.....	139
Incluyendo una plantilla personalizada con un nombre de archivo separado por guión ubicado.....	139
Pasando la variable al ámbito de la plantilla personalizada.....	139
Capítulo 49: get_template_part ()	140
Sintaxis.....	140
Parámetros.....	140
Examples.....	140
Cargando parte de la plantilla.....	140
Capítulo 50: get_the_category ()	141
Introducción.....	141
Sintaxis.....	141
Parámetros.....	141
Observaciones.....	141
Examples.....	142
Consigue todos los nombres de las categorías de la publicación.....	142
Consigue todas las identificaciones de las categorías de la publicación.....	142
Capítulo 51: get_the_title ()	143
Introducción.....	143
Sintaxis.....	143
Parámetros.....	143
Observaciones.....	143
Examples.....	143
Uso simple de get_the_title.....	143
Obtener el título de una ID de publicación especificada.....	143
Capítulo 52: Hacer solicitudes de red con API HTTP	145
Sintaxis.....	145
Parámetros.....	145
Observaciones.....	145
Devoluciones.....	145
Examples.....	145

OBTENER un recurso JSON remoto.....	145
Capítulo 53: home_url ().....	146
Sintaxis.....	146
Parámetros.....	146
Examples.....	146
Obteniendo la URL de inicio.....	146
Sitio URL.....	146
Capítulo 54: Instalacion y configuracion.....	147
Examples.....	147
WordPress en LAMP.....	147
Instalación WP en MAMP.....	148
Capítulo 55: Jerarquía de plantillas.....	150
Observaciones.....	150
Examples.....	150
Introducción.....	150
Depuración.....	152
Capítulo 56: La barra de administración (también conocida como "La barra de herramientas").....	153
Observaciones.....	153
Examples.....	153
Eliminar la barra de herramientas de administración de todos, excepto los administradores.....	153
Eliminar la barra de herramientas de administración usando filtros.....	153
Cómo quitar el logotipo de WordPress de la barra de administración.....	153
Agregue su logotipo personalizado y enlace personalizado en la página de inicio de sesión.....	154
Capítulo 57: Meta Box.....	155
Introducción.....	155
Sintaxis.....	155
Observaciones.....	155
Examples.....	155
Un ejemplo simple con una entrada regular y una entrada de selección.....	155
Capítulo 58: Migración del sitio.....	158
Sintaxis.....	158

Examples.....	158
Actualización de las tablas con una nueva URL.....	158
Capítulo 59: Personalizador Hello World.....	159
Parámetros.....	159
Examples.....	159
Hola mundo ejemplo.....	159
Capítulo 60: Post Formatos.....	161
Observaciones.....	161
Examples.....	161
Agregar tipo de publicación al tema.....	161
Añadir post-formatos a post_type 'página'.....	161
Registrar tipo de publicación personalizada 'my_custom_post_type'.....	162
Agregue post-formatos a post_type 'my_custom_post_type'.....	162
Registre el tipo de publicación personalizada 'my_custom_post_type' con el parámetro 'supp'.....	162
Añadir soporte de tema para la publicación.....	162
Llamada de función.....	162
Capítulo 61: Seguridad en WordPress - Escape.....	163
Sintaxis.....	163
Observaciones.....	163
Examples.....	163
escape de datos en código HTML.....	163
escapar de una url.....	163
datos de escape en código js.....	164
atributos de escape.....	164
datos de escape en textarea.....	164
Capítulo 62: Seguridad en WordPress - Sanitización.....	165
Sintaxis.....	165
Observaciones.....	165
Examples.....	165
Desinfectar el campo de texto.....	165
Desinfectar título.....	165
Desinfectar el correo electrónico.....	166

Desinfectar clase html.....	166
Desinfectar nombre de archivo.....	166
Desinfectar nombre de usuario.....	166
Capítulo 63: Taxonomías.....	167
Sintaxis.....	167
Parámetros.....	167
Examples.....	167
Ejemplo de registro de una taxonomía para géneros.....	167
Añadir categoría en la página.....	168
Agregue categorías y etiquetas a las páginas e insértelas como clase en.....	168
Agregue categorías y etiquetas a las páginas e inserte como clase en.....	169
Capítulo 64: Tema de WordPress y desarrollo de temas infantiles.....	171
Introducción.....	171
Examples.....	171
Desarrollando tu propio tema.....	171
Capítulo 65: Temas.....	174
Introducción.....	174
Examples.....	174
Temas de WordPress.....	174
Cómo elegir un tema.....	174
Actualización disponible.....	174
Instalar temas.....	175
Creación de un tema personalizado.....	176
Capítulo 66: <code>template_include</code>.....	177
Parámetros.....	177
Observaciones.....	177
Examples.....	177
Ejemplo simple.....	177
Más ejemplo de Adv.....	178
Capítulo 67: Tipos de correos personalizados.....	179
Sintaxis.....	179

Parámetros.....	179
Examples.....	179
Registro de un tipo de mensaje personalizado.....	179
Agregar tipos de publicaciones personalizados a la consulta principal.....	180
Agregar tipos de publicaciones personalizadas a la fuente RSS principal.....	181
Registrar tipo de mensaje personalizado.....	181
Tipo de publicación personalizada utilizando Twenty Fifteen WordPress Theme.....	182
Tipo de publicación personalizada en la búsqueda por defecto.....	183
Capítulo 68: wp_get_current_user ().....	184
Sintaxis.....	184
Examples.....	184
Obteniendo el usuario actual.....	184
Utilice el bucle foreach para obtener información de usuario de wp_get_current_user ().....	184
Capítulo 69: wp_get_current_user ().....	185
Examples.....	185
Obtener información actual de usuario registrado.....	185
Capítulo 70: WP_Query () Loop.....	186
Introducción.....	186
Examples.....	186
Recuperar las últimas 10 publicaciones.....	186
Capítulo 71: WP-CLI.....	187
Introducción.....	187
Examples.....	187
Gestionar temas.....	187
Administrar complementos.....	187
Administrar el propio WP-CLI.....	188
Descarga, instala, actualiza y gestiona una instalación de WordPress.....	189
Gestionar usuarios.....	189
Realice operaciones básicas de la base de datos utilizando las credenciales almacenadas en.....	189
Capítulo 72: WP-Cron.....	191
Examples.....	191
Ejemplo de wp_schedule_event ().....	191

intervalo de repetición personalizado en wp_schedule_event ().....	191
Creditos	193

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [wordpress](#)

It is an unofficial and free WordPress ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official WordPress.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con WordPress

Observaciones



WordPress es un sistema de gestión de contenido de código abierto (CMS) que se utiliza para crear y administrar sitios web. WordPress es el CMS más popular en Internet por una milla de un país, alimentando [aproximadamente la mitad](#) de todos los sitios web de CMS al momento de escribir y aproximadamente [una cuarta parte de todos los sitios web](#) en Internet.

WordPress comenzó su vida como una plataforma para los blogs, pero ha evolucionado a lo largo de los años para adaptarse a la mayoría de los tipos de sitios web. La interfaz se puede utilizar sin conocimientos de codificación, lo que lo hace popular para los principiantes y desarrolladores que desean capacitar a sus clientes para administrar su propio sitio web.

Otro factor importante en la popularidad de WordPress es su flexibilidad, principalmente debido a los complementos del núcleo y los sistemas temáticos. El sistema de complementos facilita la extensión de la funcionalidad del núcleo sin modificar el código del núcleo. De manera similar, el sistema de temática facilita el cambio del diseño y la estética del sitio web. Ahora hay miles de complementos y temas de WordPress gratuitos y premium disponibles. Muchos de estos se encuentran en el [repositorio de plugins de wordpress.org](#) y en el [repositorio de temas](#), respectivamente.

WordPress es desarrollado por su propia comunidad, pero está fuertemente asociado con la compañía [Automattic](#), que emplea a muchos de los desarrolladores principales de WordPress.

Código

WordPress se basa en el lenguaje de scripting del servidor [PHP](#) y el lenguaje de consulta [MySQL](#). WordPress usa MySQL como almacén de datos para el contenido y la configuración del usuario. El PHP organiza los datos de contenido en una página web [HTML](#) con todos los recursos necesarios.

wordpress.com vs wordpress.org

Puede usar WordPress [registrándose en el servicio wordpress.com de Automattic](#) y alojando su sitio web en sus servidores, o puede descargar el software de [WordPress](#) desde [wordpress.org](#) y hospedar su sitio web en un servidor bajo su control. La primera opción es fácil, pero no puede editar ningún código de sitio. Solo puedes hacer cambios a través de la interfaz de WordPress. La segunda opción requiere más trabajo, pero le brinda flexibilidad para hacer lo que quiera con el código de su sitio web. Si usted es un usuario de StackOverflow, probablemente opte por la segunda opción.

Fuente abierta

WordPress es un software de código abierto, lo que significa que es de uso gratuito y cualquier persona puede ver el código fuente y contribuir a él. Los contribuyentes potenciales pueden comenzar leyendo la [página de Contribución del código de WordPress](#).

Los errores se pueden informar enviando un error en el [rastreador de tickets de WordPress](#).

Documentación

WordPress está documentado oficialmente en el [Código de WordPress](#) en [WordPress.org](#). Los desarrolladores que trabajan con WordPress estarán particularmente interesados en la sección de [Codex](#) para [desarrolladores](#) y en la sección de [referencia para desarrolladores](#) de [wordpress.org](#).

Versiones

Versión	Fecha de lanzamiento
1.0	2004-01-03
1.2	2004-05-22
1.5	2005-02-17
2.0	2005-12-26
2.1	2007-01-22
2.2	2007-05-16
2.3	2007-09-24

Versión	Fecha de lanzamiento
2.5	2008-03-29
2.6	2008-07-15
2.7	2008-12-10
2.8	2009-06-10
2.9	2009-12-18
3.0	2010-06-17
3.1	2011-02-23
3.2	2011-07-04
3.3	2011-12-12
3.4	2012-06-13
3.5	2012-12-11
3.6	2013-08-01
3.7	2013-10-24
3.8	2013-12-12
3.9	2014-04-16
4.0	2014-09-04
4.1	2014-12-17
4.2	2015-04-23
4.3	2015-08-18
4.4	2015-12-08
4.5	2016-04-12
4.6	2016-08-16
4.7	2016-12-06
4.8	2017-06-08

Examples

Introducción a WordPress

[WordPress](#) [WP] es un sistema de gestión de contenido de código abierto para crear aplicaciones, sitios web y blogs. WP está escrito en PHP y utiliza MySQL como el almacén de datos para el contenido y la configuración del usuario. Cuenta con un rico ecosistema de [complementos](#) y [temas](#), y disfruta de una vibrante comunidad de código abierto, buena documentación y pocas barreras de entrada. La documentación de usabilidad y del desarrollador se puede encontrar en el [Código WP](#) .

Una parte de WordPress que lo hace diferente de la mayoría de los otros productos de CMS es su [Programación dirigida por eventos](#) . Esta es una forma diferente de programación y representación lógica que la arquitectura MVC (Model View Controller) que es utilizada por la mayoría de los sistemas CMS. WordPress utiliza los conceptos de Acciones y Filtros. Forman una cola de eventos que permiten a los complementos y temas insertar, modificar o incluso eliminar partes de la página de la aplicación web final y / o partes. Un concepto similar es JIT o compilación Just-In-Time.

Si bien históricamente WordPress ha sido conocido como una plataforma de blogs, y nunca puede perder este estigma, el enfoque del equipo central de WordPress ha cambiado claramente. Con el [Estado de la Palabra 2016](#) , por el fundador [Matthew Mullenweg](#) , podemos ver un claro cambio en los objetivos, la visión y el esfuerzo. En 2016, vimos un progreso asombroso cuando el núcleo de WordPress adoptó la mayoría del popular [complemento API REST](#) . Esta fue claramente la intención del equipo central desde el principio, cuando comenzaron un esfuerzo audaz de construir un panel de administración de JavaScript CMS de primera línea, que rompe con el estándar de oro que hemos visto durante tantos años; Lo llamaron [Calpyso](#) .

Temas de WordPress

Mapeo de URLs a plantillas específicas

Para comprender completamente los temas de WordPress, debe comprender dos conceptos principales:

1. Enlaces permanentes
2. La jerarquía de plantillas

Un enlace permanente es una URL permanente, no cambiante (o enlace, a un recurso específico). Por ejemplo:

- [example.com/about-us/](#) (una página en WP)
- [example.com/services/](#) (una lista de varios elementos, también llamada "archivo" en la jerga de WP)
- [example.com/services/we-can-do-that-for-you/](#) (un elemento individual)

Cuando un usuario solicita una URL, WordPress aplica ingeniería inversa al enlace permanente para averiguar qué plantilla debe controlar su diseño. WordPress busca los distintos archivos de plantilla que *podrían* controlar este contenido en particular y, en última instancia, da preferencia a la más específica que encuentra. Esto se conoce como la jerarquía de plantillas.

Una vez que WP encuentra la plantilla de vista coincidente en la jerarquía, usa ese archivo para procesar y renderizar la página.

Por ejemplo: `index.php` (la plantilla predeterminada, "catch-all") será reemplazada por `archive.php` (la plantilla predeterminada para el contenido basado en listas), que a su vez será reemplazada por `archive-services.php` (una plantilla archivo específicamente para el archivo llamado "servicios").

[Aquí hay una gran referencia visual para la jerarquía de plantillas.](#)

Tema básico estructura de directorio

Un tema simple se ve algo como esto:

```
// Theme CSS
style.css

// Custom functionality for your theme
functions.php

// Partials to include in subsequent theme files
header.php
footer.php
sidebar.php
comments.php

// "Archives", (listing views that contain multiple posts)
archive.php
author.php
date.php
taxonomy.php
tag.php
category.php

// Individual content pages
// Note that home and frontpage templates are not recommended
// and they should be replaced by page templates
singular.php
single.php
page.php
front-page.php
home.php

// Misc. Utility Pages
index.php (a catch-all if nothing else matches)
search.php
attachment.php
image.php
404.php
```


Ejemplo de un "Single" (plantilla para una publicación individual)

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

<?php if ( have_posts() ) while ( have_posts() ) : the_post(); ?>
    <h1><?php the_title(); ?></h1>
    <?php the_content(); ?>
    <?php comments_template( '', true ); ?>
<?php endwhile; ?>

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

¿Que esta pasando aqui? Primero, carga `header.php` (similar a un PHP incluye o requiere), configura The Loop, muestra `the_title` y `the_content` , luego incluye `comments.php` , `sidebar.php` , y `footer.php` . The Loop hace el trabajo pesado, configurando un objeto `Post` , que contiene toda la información del contenido que se ve actualmente.

Ejemplo de un "Archivo" (plantilla para una lista de publicaciones múltiples)

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

<?php if ( have_posts() ) while ( have_posts() ) : the_post(); ?>
    <a href="<?php the_permalink(); ?>"><?php the_title(); ?></a>
    <?php the_excerpt(); ?>
<?php endwhile; ?>

<?php
    next_posts_link( 'Older Entries', $the_query->max_num_pages );
    previous_posts_link( 'Newer Entries' );
?>

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

Primero, incluye `header.php` , configura The Loop e incluye `sidebar.php` y `footer.php` . Pero en este caso, hay varias publicaciones en el bucle, así que en su lugar se muestra un extracto con un enlace a la publicación individual. `next_posts_link` y `previous_posts_link` también se incluyen para que el archivo pueda paginar los resultados.

Publicaciones, páginas, tipos de publicaciones personalizadas y campos

personalizados

Fuera de la caja, WordPress admite dos tipos de contenido: `Posts` y `Pages` . Las publicaciones se utilizan normalmente para contenido no jerárquico como las publicaciones de blog. Las páginas se utilizan para contenido estático e independiente, como una página Acerca de nosotros o la página de Servicios de una empresa con subpáginas anidadas debajo.

A partir de la versión 3.0, los desarrolladores pueden definir sus propios tipos de publicaciones personalizadas para ampliar la funcionalidad de WordPress más allá de lo básico. Además de los tipos de publicaciones personalizadas, también puede crear sus propios campos personalizados para adjuntarlos a sus publicaciones / páginas / tipos de publicaciones personalizadas, lo que le permite proporcionar una forma estructurada de agregar y acceder a los metadatos dentro de sus plantillas. Ver: [Campos personalizados avanzados](#) .

Lea [Empezando con WordPress en línea](#):

<https://riptutorial.com/es/wordpress/topic/304/empezando-con-wordpress>

Capítulo 2: ¿Cómo puedo integrar el editor de Markdown con el complemento del repetidor del campo personalizado avanzado?

Examples

Añadir Markdown Editor

Encontré la solución. Por favor considere a continuación los pasos de mención.

Instalar el plugin [wp Markdown Editor](#) .

Luego instale " [acf-wp-wysiwyg](#) " para el campo del repetidor. Ahora tienes que actualizar en algunos archivos. Abra este archivo y vaya al número de línea "180" o vaya a la función "create_field" agregar

```
echo '<script> var simplemde = new SimpleMDE({element:
document.getElementById("'.$id.'")});jQuery(".quicktags-
toolbar").css("display","none");</script>';
```

Ahora bajo el archivo "input.js" del complemento "acf-repeater" abierto, número de línea "142"

reemplazar

```
new_field_html = this.$el.find('> table > tbody > tr.row-clone').html().replace(/(=["]*[\w-
\[\]]*?)(acfcloneindex)/g, '$1' + new_id),
```

Con

```
new_field_html = this.$el.find('> table > tbody > tr.row-clone').html().replace(/(["]*[\w-
\[\]]*?)(acfcloneindex)/g, '$1' + new_id),
```

Lea [¿Cómo puedo integrar el editor de Markdown con el complemento del repetidor del campo personalizado avanzado?](#) en línea: <https://riptutorial.com/es/wordpress/topic/6602/-como-puedo-integrar-el-editor-de-markdown-con-el-complemento-del-repetidor-del-campo-personalizado-avanzado->

Capítulo 3: Acciones y Filtros

Sintaxis

- `add_action` (etiqueta, `function_to_call`, prioridad, `num_of_args`);
- `add_filter` (etiqueta, `function_to_call`, prioridad, `num_of_args`);

Parámetros

Parámetro	Explicación
\$ etiqueta	(cadena) (Requerido) El nombre de la acción a la que se engancha la función \$.
\$ función	(callable) (Requerido) Requiere una cadena que contenga el nombre de la función o la función anónima. Ver ejemplos para agregar funciones dentro de las clases.
\$ prioridad	(int) por defecto = 10. Las funciones asociadas a los ganchos / filtros se ejecutarán en la prioridad asignada. Puede tener una situación en la que desee trabajar con código antes de cualquier otra acción, establecer prioridad = 1 o después de todas las demás funciones vinculadas prioridad = 100, etc. Al igual que con todas las funciones php, puede usar la función sin pasar un valor para una variable donde se ha establecido un valor predeterminado, pero si desea cambiar el número de parámetros devueltos, debe especificar!
\$ parámetros	(int) por defecto = 1. El número de parámetros devueltos a su función adjunta. Los parámetros devueltos dependerán del número adjunto donde se creó el gancho. Vea <code>apply_filters()</code> y <code>do_action()</code> para más detalles.

Observaciones

Wordpress Hooks

Algo que a menudo confunde a los desarrolladores cuando comienzan a trabajar con WordPress es el uso de `apply_filters()` y `add_action()`. A menudo verá complementos / temas que los utilizan en el código y, si no entiende el concepto, le resultará difícil trabajar con ellos.

En resumen (muy breve, busque el diagrama de flujo de carga de WordPress para conocer el proceso en detalle), WordPress se carga de la siguiente manera:

1. `wp-load.php` - funciones, etc.
2. `mu-plugins`: cualquier archivo que se encuentre en la carpeta `mu-plugins`, a menudo se usa para servir objetos almacenados en caché

3. Complementos: sin ningún orden en particular, se cargarán los complementos instalados y activados
4. Tema infantil activo / tema principal
5. init - resto de datos
6. modelo

Si usted es un desarrollador y está trabajando con un archivo de funciones, puede ver que ambos se cargan antes en el proceso que los archivos con los que está trabajando. Lo que significa que no puede modificar procesos (tenga en cuenta que no puede sobrescribir funciones) o variables que se ejecutan más tarde o que aún no se hayan definido. También los desarrolladores de temas pueden colocar enlaces en su código para permitir que los complementos se conecten o los complementos podrían permitir que otros complementos sobrescriban sus variables. Ahora esto puede ser confuso hasta ahora, pero aguanta.

Para entender `add_filter()` y `add_action()` necesitamos ver cómo se crean los ganchos en primer lugar.

```
$arga= 'hello';  
do_action('im_a_hook', $arga );
```

Cuando encuentre lo anterior en WordPress, llamará a cualquier función adjunta al gancho `im_a_hook` (busque `$wp_filter` para obtener información sobre el proceso). En su función adjunta, `$arga` estará disponible para que funcione la función adjunta.

```
add_action('im_a_hook', 'attached_function');  
  
function attached_function($arga){  
    echo $arga;  
}
```

Esto abre nuevas y poderosas oportunidades para modificar variables en ciertos puntos del proceso de carga. ¿Recuerda que dijimos anteriormente que las plantillas se cargan después de los complementos / temas? Un complemento común es WooCommerce, que crea pantallas más adelante en el proceso. No voy a documentar cómo, pero se puede encontrar un ejemplo de `do_action` en el complemento.

```
do_action( 'woocommerce_after_add_to_cart_button' );
```

Aquí tenemos un gancho creado que no devuelve ninguna variable, pero aún podemos divertirnos con él:

```
add_action( 'woocommerce_after_add_to_cart_button', 'special_offer');  
  
function special_offer(){  
    echo '<h1>Special Offer!</h1>';  
}
```

El `add_action` anterior se hará `echo` un encabezado de oferta especial donde se encuentra `do_action('woocommerce_after_add_to_cart_button')` que es cuando se crea una pantalla de

WooCommerce. Así que podemos usar este gancho para insertar html. Otros usos podrían incluir redirigir a una pantalla diferente por completo, etc.

También se pueden pasar múltiples variables a la función. Prueba esto en tus funciones de temas. Tenga en cuenta el último parámetro que estamos configurando en 3, porque queremos trabajar con los 3 parámetros disponibles. Si cambiamos esto a 2, solo se devolverían 2 y obtendríamos un error indefinido.

```
add_action('custom_hook', 'attached_function', 10, 3);

function attached_function($a,$b,$c){

    var_dump($a);
    var_dump($b);
    var_dump($c);

}

$args = 1;
$args = 2;
$args = 3;

do_action('custom_hook', $args, $args, $args);
exit;
```

Hay otro tipo de gancho WP llamado filtro. Un filtro es diferente de una acción en su uso, una acción solo puede recibir variables, obviamente estas variables están dentro del alcance de las funciones (usted debe saber qué alcance de php es, si no Google). Los filtros devuelven los datos devueltos, por lo que puede utilizar para modificar las variables.

```
$filter_me= apply_filters('im_a_filter', $variable_to_filter);
```

Cuando vea lo anterior, puede modificar el valor de `$filter_me` ya que cualquier dato que devuelva será el valor almacenado en la variable. Entonces, por ejemplo (note que estamos cambiando `$variable_to_filter` a `$filter_me` en el ejemplo):

```
add_filter('im_a_filter', 'attached_function', 100);

function attached_function($filter_me){

    $filter_me= 'ray';

    return $filter_me;

}

$filter_me = 'bob';
$filter_me= apply_filters('im_a_filter', $filter_me);
```

La variable `$filter_me` ahora contendrá *'ray'* en lugar de *'bob'*, hemos establecido una prioridad de 100, por lo que estamos razonablemente seguros de que nadie cambiará el valor después de su

uso (puede haber varios filtros ejecutándose en el mismo gancho). ahora puede cambiar las variables utilizadas más adelante en el proceso si `apply_filters()` está presente.

También puede pasar varios parámetros, pero solo puede cambiar el valor de uno. También debe devolver un valor, o de lo contrario su variable no contendrá nada. Si entiendes cómo usar php para asignar valores / arrays / objetos a las variables, esto será obvio para ti, por ejemplo:

```
add_filter('im_a_filter', 'attached_function', 100, 3);

function attached_function($filter_me, $arga, $argb){

    $filter_me= 'ray'.$arga.$argb;

    $arga= 'you fool';

    return $filter_me;

}

$filter_me = 'bob';

$arga = ' middlename';
$argb = ' surname';

$filter_me= apply_filters('im_a_filter', $filter_me, $arga, $argb);
```

La variable `$filter_me` ahora contiene *'ray middlename surname'*. Pero ¿qué pasa con `$arga`? Esto todavía contiene *'middlename'*, cambiar un `$arga` a *'you fool'* dentro de nuestra función no tiene efecto en el valor definido fuera de su alcance (hay formas, google globals, etc.)

add_action (\$ hook_name, \$ function, \$ prioridad, \$ parámetros)

add_filter (\$ hook_name, \$ function, \$ priority, \$ parameters);

Examples

add_action - init

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

function process_post(){
    if($_POST)
        var_dump($_POST);
}
```

add_action - init - función anónima

```
add_action('init' , function(){
    echo 'i did something';
});
```

add_action - init - dentro del objeto de clase

```
class sample{

    public function __construct(){
        add_action('init', array($this, 'samp') );
    }

    public function samp(){ // must be public!!
        echo 'i did something';
    }
}

new sample();
```

add_action - init - dentro de la clase estática

```
class sample{

    public static function add_action_func(){
        //note __CLASS__ will also include any namespacing
        add_action('init', array(__CLASS__, 'samp') );
    }

    public static function samp(){
        echo 'i did something';
    }

}

sample::add_action_func();
```

Lea Acciones y Filtros en línea: <https://riptutorial.com/es/wordpress/topic/2692/acciones-y-filtros>

Capítulo 4: Actualizar WordPress manualmente

Examples

VIA FTP

1. Descargue la versión deseada de WordPress desde www.wordpress.org a su computadora local y descomprima el archivo.
 - También mantenga una copia de seguridad de su versión actual ... por si acaso.
2. Conéctese a su sitio web con su cliente FTP favorito (FileZilla es popular y fácil, pero cualquier cliente FTP estará bien).
 - *Las instrucciones para esto están fuera del alcance de WordPress, pero se pueden encontrar en una fecha futura en el tema FTP propuesto .*
3. Suba las carpetas (y su contenido) tituladas "wp-admin" y "wp-includes" a sus directorios correspondientes en su servidor. Asegúrese de sobrescribir las carpetas actuales.
 - Puede omitir la carga de la carpeta "wp-content" a menos que elija utilizar uno de los temas incluidos. Si desea actualizar / cargar los temas predeterminados incluidos con la versión elegida, también debe cargar esta carpeta.
4. Cargue los archivos individuales en la carpeta de inicio (index.php, wp - *. Php, etc.).
 - Puede omitir los archivos titulados "license.txt" y "readme.html", ya que no se requiere que funcionen y se pueden usar como métodos para determinar su versión de WP para ataques de seguridad.
5. Visite e inicie sesión en su sitio web para realizar las actualizaciones necesarias de la base de datos.
 - No todas las actualizaciones de WP tienen cambios en la base de datos, pero algunas sí.

Nota : este método creará archivos huérfanos que pueden / se acumularán con el tiempo y pueden presentar riesgos de seguridad. Asegúrese de realizar una comparación de archivos después de la finalización y elimine los archivos antiguos de su servidor de versiones anteriores de WP que ya no están en uso.

Lea [Actualizar WordPress manualmente en línea:](https://riptutorial.com/es/wordpress/topic/8663/actualizar-wordpress-manualmente)

<https://riptutorial.com/es/wordpress/topic/8663/actualizar-wordpress-manualmente>

Capítulo 5: add_action ()

Sintaxis

- add_action (\$ tag, \$ function_to_add)
- add_action (\$ tag, \$ function_to_add, \$ prioridad)
- add_action (\$ tag, \$ function_to_add, \$ prioridad, \$ aceptado en args)

Parámetros

Parámetro	Detalles
\$ etiqueta	(cadena) El nombre de la acción a la que se enganchará el procedimiento \$function_to_add .
\$ function_to_add	(llamada) La función / procedimiento de llamada que desea que se llame.
\$ prioridad	(int) El nivel de prioridad al que se ejecutará \$function_to_add . Opcional. Predeterminado 10.
\$ accept_args	(int) El número de argumentos que acepta la \$function_to_add invocable \$function_to_add . Opcional. Predeterminado 1.

Examples

Función directa de devolución de llamada

```
add_action( 'init', function() {  
    // do something here  
} );
```

Usando un bloque de funciones para enganchar un conjunto de instrucciones. Con el gancho de `init` , el conjunto de instrucciones se ejecutará justo después de que wordpress haya terminado de cargar los componentes necesarios.

Llamada de referencia de nombre de función

```
function my_init_function() {  
    // do something here  
}  
  
add_action( 'init', 'my_init_function' );
```

Usando el nombre de la función para enlazar un conjunto de instrucciones. Con el gancho de `init`

, el conjunto de instrucciones se ejecutará justo después de que wordpress haya terminado de cargar los componentes necesarios.

Devolución de llamada de método estático de clase

```
class MyClass {
    static function my_init_method() {
        // do something here
    }
}

add_action( 'init', array( 'MyClass', 'my_init_method' ) );
```

Utilizando un método estático de una clase para enganchar un conjunto de instrucciones. Con el gancho de `init`, el conjunto de instrucciones se ejecutará justo después de que wordpress haya terminado de cargar los componentes necesarios.

Método de devolución de llamada de método

```
class MyClass{
    function my_init_method() {
        // do something here
    }
}

$obj = new MyClass();

add_action( 'init', array( $obj, 'my_init_method' ) );
```

Utilizando un método de un objeto para enganchar un conjunto de instrucciones. Con el gancho de `init`, el conjunto de instrucciones se ejecutará justo después de que wordpress haya terminado de cargar los componentes necesarios.

Lea `add_action ()` en línea: <https://riptutorial.com/es/wordpress/topic/6264/add-action--->

Capítulo 6: add_editor_style ()

Introducción

La función permite al usuario cargar hojas de estilo para el editor TinyMCE

Sintaxis

- add_editor_style (\$ hoja de estilo)

Parámetros

Parámetro	Detalles
\$ hoja de estilo	(matriz o cadena) (Opcional) Nombre de la hoja de estilo o matriz del mismo, en relación con la raíz del tema. Por defecto es 'editor-style.css'

Examples

Cargando un solo archivo css

Código

```
function add_new_style() {  
    add_editor_style( 'file-name-here.css' );  
}  
add_action( 'admin_init', 'add_new_style' );
```

Explicación

En el código anterior, usamos add_editor_style para cargar el archivo css. También usamos add_action para asegurarnos de que WordPress ejecuta nuestra función.

Lea add_editor_style () en línea: <https://riptutorial.com/es/wordpress/topic/9215/add-editor-style--->

Capítulo 7: add_menu_page ()

Introducción

Esta función es para agregar un elemento en la barra de navegación del panel de administración.

Sintaxis

- add_menu_page (\$ page_title, \$ menu_title, \$ capacidad, \$ menu_slug, \$ function, \$ icon_url, \$ position)

Parámetros

Parámetro	Detalles
\$ page_title	(cadena) El texto que se mostrará en las etiquetas de título de la página cuando se seleccione el menú.
\$ menu_title	(cadena) El texto que se utilizará para el menú.
\$ capacidad	(cadena) La capacidad requerida para que este menú se muestre al usuario.
\$ menu_slug	(cadena) El nombre del slug para referirse a este menú por (debe ser único para este menú).
\$ función	(callable) (opcional) La función que se llamará para generar el contenido de esta página.
\$ icon_url	(cadena) (opcional) La URL del icono que se utilizará para este menú.
\$ posición	(int) (opcional) La posición en el orden del menú debería aparecer.

Observaciones

Aquí hay una lista de las posiciones predeterminadas (para \$ posición)

- 2 - Dashboard
- 4 - Separador
- 5 - Mensajes
- 10 - Medios
- 15 - Enlaces
- 20 - Páginas
- 25 - Comentarios
- 59 - Separador

- 60 - Apariencia
- 65 - Complementos
- 70 - Usuarios
- 75 - Herramientas
- 80 - Configuraciones
- 99 - Separador

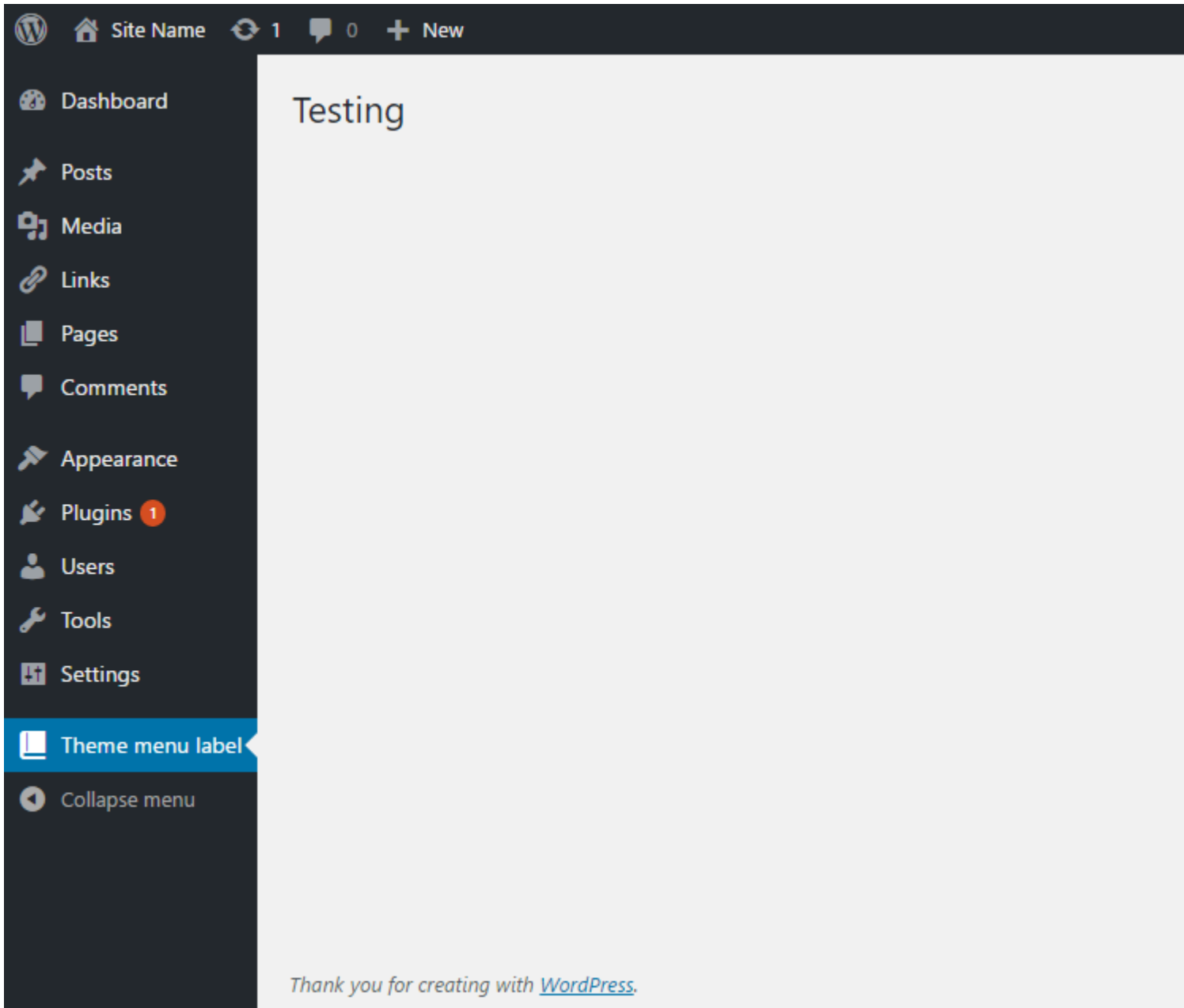
Examples

Agregar el elemento "Título de la página del tema" a la barra de navegación

Código

```
function add_the_theme_page(){
    add_menu_page('Theme page title', 'Theme menu label', 'manage_options', 'theme-options',
'page_content', 'dashicons-book-alt');
}
add_action('admin_menu', 'add_the_theme_page');
function page_content(){
    echo '<div class="wrap"><h2>Testing</h2></div>';
}
```

Salida



Explicación

En el código, creamos una función llamada `add_the_theme_page` y usamos `add_menu_page` para agregar el elemento a la barra de navegación. Verifique la parte de parámetros en esta página para conocer los argumentos que pasamos. Luego usamos `add_action` para ejecutar nuestra función `add_the_theme_page`. Finalmente, creamos la función `page_content` para mostrar los contenidos en la página.

OOP y cómo cargar scripts / estilos en la página del menú

```
<?php
/*
 * Plugin Name: Custom Admin Menu
 */

class SO_WP_Menu {
```

```

private $plugin_url;

public function __construct() {
    $this->plugin_url = plugins_url( '/', __FILE__ );
    add_action( 'plugins_loaded', array( $this, 'init' ) );
}

public function init() {
    add_action( 'admin_menu', array( $this, 'add_menu' ) );
}

public function add_menu() {
    $hook = add_menu_page(
        'My Menu', // Title, html meta tag
        '<span style="color:#e57300;">My Menu</span>', // Menu title, hardcoded style
        'edit_pages', // capability
        'dummy-page-slug', // URL
        array( $this, 'content' ), // output
        null, // icon, uses default
        1 // position, showing on top of all others
    );
    add_action( "admin_print_scripts-$hook", array( $this, 'scripts' ) );
    add_action( "admin_print_styles-$hook", array( $this, 'styles' ) );
}

public function content() {
    ?>
    <div id="icon-post" class="icon32"></div>
    <h2>Dummy Page</h2>
    <p> Lorem ipsum</p>
    <?php
}

# Printing directly, could be wp_enqueue_script
public function scripts() {
    ?><script>alert('My page');</script><?php
}

# Enqueuing from a CSS file on plugin directory
public function styles() {
    wp_enqueue_style( 'my-menu', $this->plugin_url . 'my-menu.css' );
}
}

new SO_WP_Menu();

```

Lo que es importante tener en cuenta en este ejemplo es que, al usar `add_menu_page()`, devuelve un `add_menu_page()` que puede usarse para apuntar a nuestra página exacta y cargar estilos y secuencias de comandos allí.

Un error común es poner en cola sin segmentar y eso derrama los scripts y los estilos en todo `/wp-admin`.

Usando OOP podemos almacenar variables comunes para ser usadas entre métodos internos.

Lea `add_menu_page()` en línea: <https://riptutorial.com/es/wordpress/topic/9189/add-menu-page--->

Capítulo 8: add_submenu_page ()

Introducción

Esta función es agregar un subelemento a un elemento existente en la barra de navegación de los paneles de administración.

Sintaxis

- add_submenu_page (\$ parent_slug, \$ page_title, \$ menu_title, \$ capacidad, \$ menu_slug, \$ function)

Parámetros

Parámetro	Detalles
\$ parent_slug	(cadena) El nombre del slug para el menú principal (o el nombre de archivo de una página de administrador de WordPress estándar).
\$ page_title	(cadena) El texto que se mostrará en las etiquetas de título de la página cuando se seleccione el menú.
\$ menu_title	(cadena) El texto que se utilizará para el menú.
\$ capacidad	(cadena) La capacidad requerida para que este menú se muestre al usuario.
\$ menu_slug	(cadena) El nombre del slug para referirse a este menú por (debe ser único para este menú).
\$ función	(callable) (Opcional) La función que se llamará para generar el contenido de esta página.

Observaciones

Aquí hay una lista de las babosas para \$ parent_slug

- Panel de control: 'index.php'
- Publicaciones: 'edit.php'
- Medios: 'upload.php'
- Páginas: 'edit.php? Post_type = page'
- Comentarios: 'edit-comments.php'
- Tipos de publicaciones personalizadas: 'edit.php? Post_type = your_post_type'
- Apariencia: 'themes.php'
- Plugins: 'plugins.php'

- Usuarios: 'users.php'
- Herramientas: 'tools.php'
- Configuraciones: 'options-general.php'
- Configuración de red: 'settings.php'

Examples

Agregar la "Página del submenú" como una subpágina de "Herramientas" a la barra de navegación

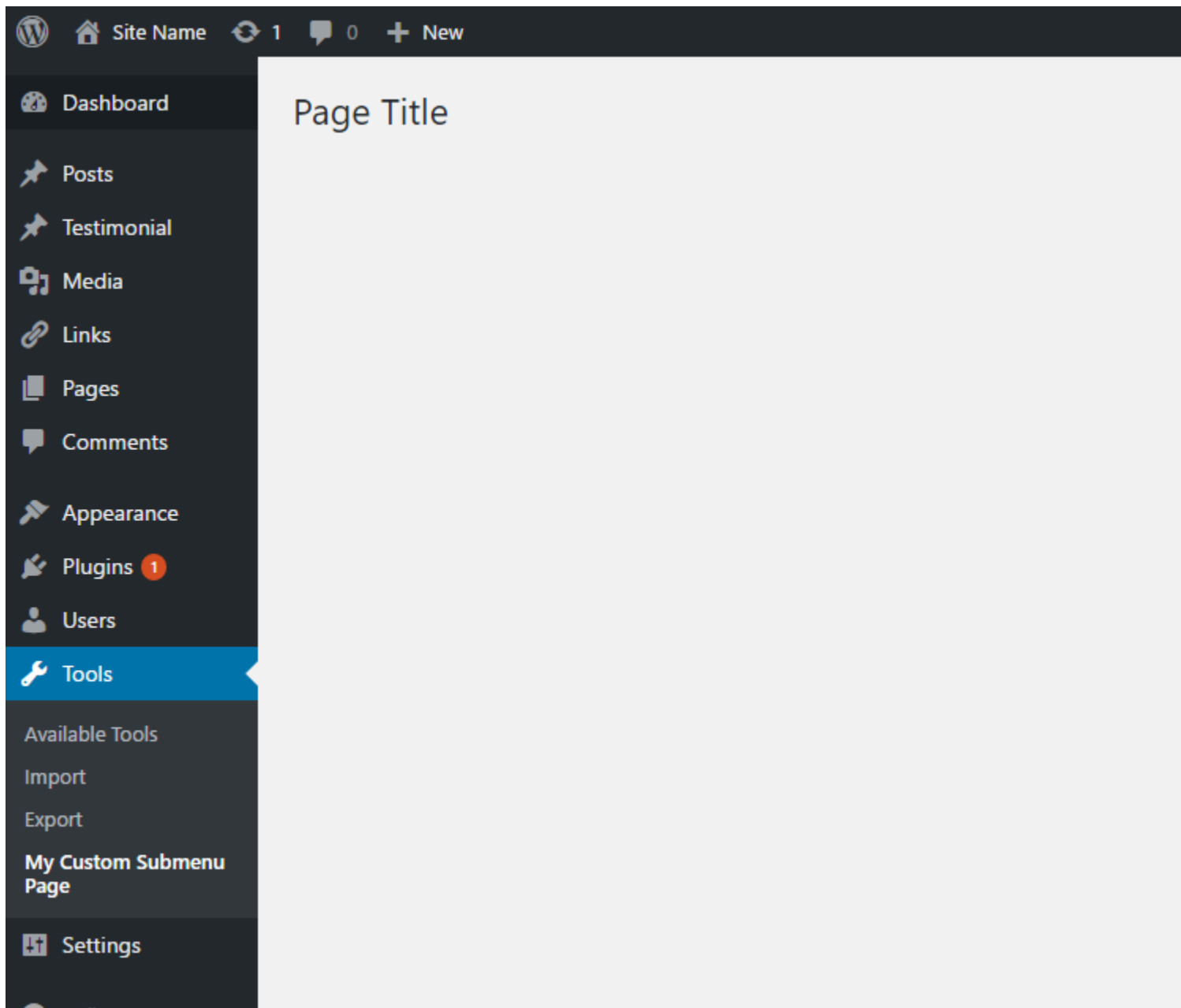
Código

```
add_action('admin_menu', 'register_my_custom_submenu_page');

function register_my_custom_submenu_page() {
    add_submenu_page(
        'tools.php',
        'Submenu Page',
        'My Custom Submenu Page',
        'manage_options',
        'my-custom-submenu-page',
        'my_custom_submenu_page_content' );
}

function my_custom_submenu_page_content() {
    echo '<div class="wrap">';
    echo '<h2>Page Title</h2>';
    echo '</div>';
}
```

Salida



Explicación

En el código, creamos una función llamada `register_my_custom_submenu_page` y usamos `add_submenu_page` para agregar el elemento a la barra de navegación como elemento secundario de `tools.php`, que es la página de Herramientas.

Verifique la parte de parámetros en esta página para conocer los argumentos que pasamos. Luego usamos `add_action` para ejecutar nuestra función `register_my_custom_submenu_page`. Finalmente, creamos la función `my_custom_submenu_page_content` para mostrar los contenidos en la página.

Lea `add_submenu_page ()` en línea: <https://riptutorial.com/es/wordpress/topic/9193/add-submenu-page--->

Capítulo 9: add_theme_support ()

Introducción

Esta función registra las características que admite el tema.

Sintaxis

- add_theme_support (\$ feature)

Parámetros

Parámetro	Detalles
\$ característica	(cadena) La característica que se está agregando.

Observaciones

Lista de características que se utilizarán en \$ característica:

- 'post-formatos'
- 'post-miniaturas'
- 'html5'
- 'logotipo personalizado'
- 'subidas de encabezado personalizadas'
- 'cabecera personalizada'
- 'fondo personalizado'
- 'etiqueta de título'
- 'contenido de inicio'

Examples

Agregando soporte de tema para formatos de post

```
add_theme_support( 'post-formats', array( 'formatone', 'formattwo' ) );
```

Agregar soporte de tema para las miniaturas de las publicaciones

```
add_theme_support( 'post-thumbnails', array( 'post' ) );
```

El código anterior solo permite publicar mensajes en todas las publicaciones. Para habilitar la característica en todos los tipos de publicaciones, haga:

```
add_theme_support ( 'post-thumbnails' );
```

Lea `add_theme_support ()` en línea: <https://riptutorial.com/es/wordpress/topic/9216/add-theme-support--->

Capítulo 10: Admin Dashboard Widgets

Introducción

Con un widget del panel de administración, puede mostrar cualquier tipo de información en el panel de administración. Puedes hacer múltiples widgets si quieres. Puede agregar el código a las funciones.php de su tema o a su complemento.

Sintaxis

- `add_action` (`$ tag`, `$ function_to_add`, `$ priority`, `$ accept_args`);
- `wp_add_dashboard_widget` (`$ widget_id`, `$ widget_name`, `$ callback`, `$ control_callback`, `$ callback_args`);

Parámetros

Parámetro	Detalles
<code>\$ etiqueta</code>	(<i>cadena requerida</i>) Nombre de la acción donde se engancha <code>\$ function_to_add</code>
<code>\$ function_to_add</code>	(se <i>requiere</i> llamada) Nombre de la función que desea llamar.
<code>\$ prioridad</code>	(<i>int opcional</i>) Lugar de la llamada de función en todas las funciones de acción (predeterminado = 10)
<code>\$ accept_args</code>	(<i>int opcional</i>) Número de parámetros que acepta la función (predeterminado = 1)
<code>\$ widget_id</code>	(<i>cadena requerida</i>) Slug único para tu widget
<code>\$ widget_name</code>	(<i>cadena requerida</i>) Nombre de su widget (mostrado en la cabeza)
<code>\$ devolución de llamada</code>	(se puede <i>llamar</i>) Nombre de la función que muestra el contenido de su widget
<code>\$ control_callback</code>	(<i>llamable opcional</i>) Nombre de la función que maneja los formularios de opciones de widget
<code>\$ callback_args</code>	(<i>array opcional</i>) Parámetros de la función <code>\$ control_callback</code>

Examples

Widget simple (muestra texto)

Esto agregará un widget simple que muestra solo un pequeño mensaje.

```
add_action('wp_dashboard_setup', 'register_my_dashboard_widgets');

function register_my_dashboard_widgets() {
    wp_add_dashboard_widget('myInfo_widget', 'Important Information', 'display_infoWidget');
}

function display_infoWidget() {
    echo '<p>At the first of february this site gets a new design.
    Therefore is wont be available this day. To see the current progress you can visit
    <a href="http://www.justanexample.com" >this site</a></p>';
}
```

Lea Admin Dashboard Widgets en línea: <https://riptutorial.com/es/wordpress/topic/9571/admin-dashboard-widgets>

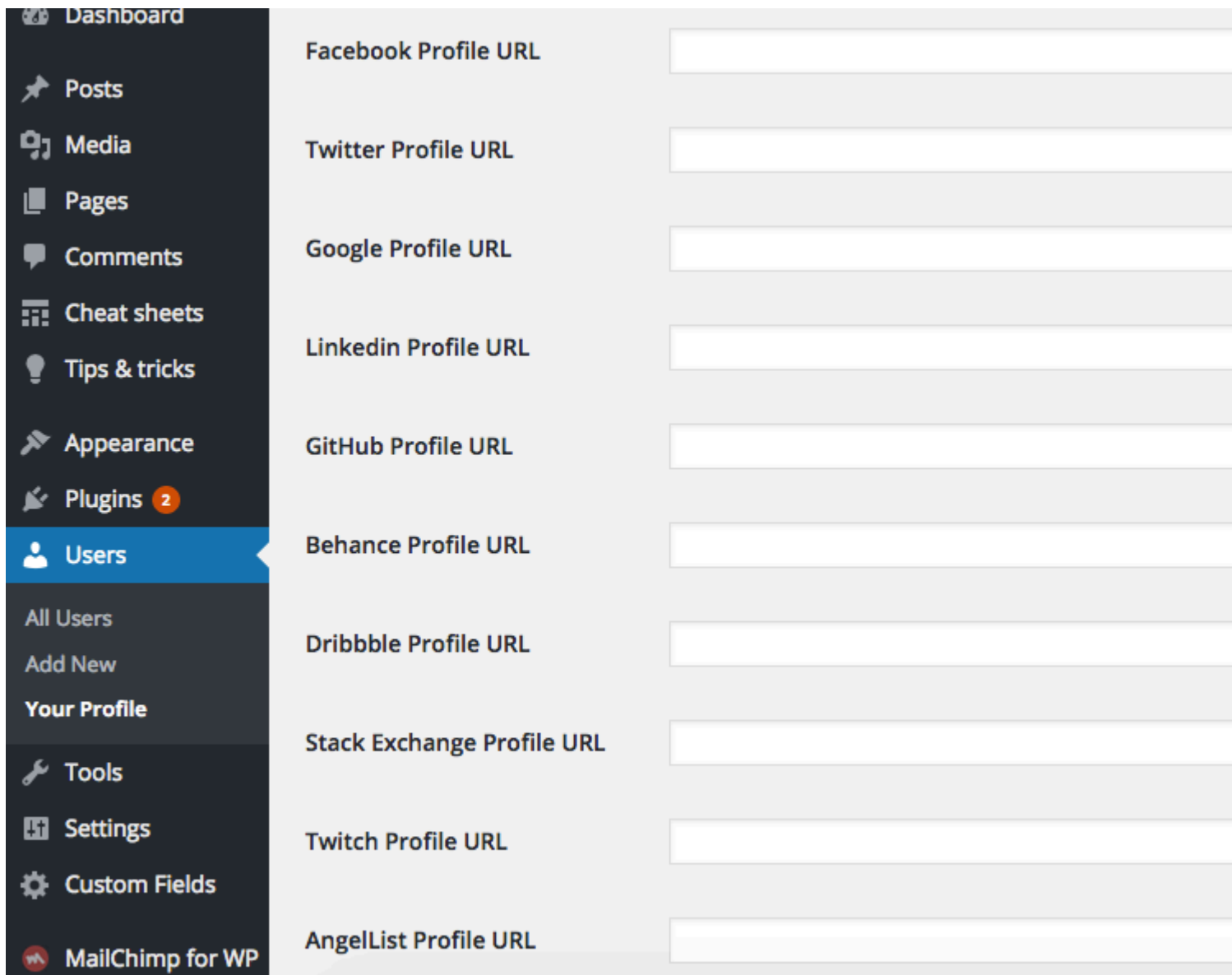
Capítulo 11: Agregar / eliminar información de contacto para usuarios con gancho de filtro `user_contactmethods`

Examples

Habilitar las redes sociales más populares.

```
function social_profiles( $contactmethods ) {  
  
    $contactmethods['facebook_profile'] = 'Facebook Profile URL';  
    $contactmethods['twitter_profile']  = 'Twitter Profile URL';  
    $contactmethods['google_profile']   = 'Google Profile URL';  
    $contactmethods['linkedin_profile']  = 'Linkedin Profile URL';  
    $contactmethods['github_profile']    = 'GitHub Profile URL';  
    $contactmethods['behance_profile']   = 'Behance Profile URL';  
    $contactmethods['dribbble_profile']  = 'Dribbble Profile URL';  
    $contactmethods['stack_profile']     = 'Stack Exchange Profile URL';  
    $contactmethods['twitch_profile']    = 'Twitch Profile URL';  
    $contactmethods['angellist_profile'] = 'AngelList Profile URL';  
  
    return $contactmethods;  
}  
  
add_filter( 'user_contactmethods', 'social_profiles', 10, 1);
```

Obtendrá estos archivos en su panel de control:



Y así es como lo recuperas en código.

```
<?php $user_stack_exchange = get_the_author_meta( 'stack_profile' ); ?>
```

Eliminar el método de contacto

```
function remove_contact_methods( $contactmethods ) {  
  
    unset( $contactmethods['facebook_profile'] );  
    unset( $contactmethods['twitter_profile'] );  
  
    return $contactmethods;  
}  
  
add_filter( 'user_contactmethods', 'remove_contact_methods', 10, 1 );
```

Lea Agregar / eliminar información de contacto para usuarios con gancho de filtro `user_contactmethods` en línea: <https://riptutorial.com/es/wordpress/topic/2694/agregar---eliminar-informacion-de-contacto-para-usuarios-con-gancho-de-filtro-user-contactmethods>

Capítulo 12: AJAX

Examples

Solicitud AJAX con respuesta JSON

funciones.php:

```
// We add the action twice, once for logged in users and once for non logged in users.
add_action( 'wp_ajax_my_action', 'my_action_callback' );
add_action( 'wp_ajax_nopriv_my_action', 'my_action_callback' );

// Enqueue the script on the front end.
add_action( 'wp_enqueue_scripts', 'enqueue_my_action_script' );
// Enqueue the script on the back end (wp-admin)
add_action( 'admin_enqueue_scripts', 'enqueue_my_action_script' );

function my_action_callback() {
    $json = array();

    if ( isset( $_REQUEST['field2'] ) ) {
        $json['message'] = 'Success!';
        wp_send_json_success( $json );
    } else {
        $json['message'] = 'Field 2 was not set!';
        wp_send_json_error( $json );
    }
}

function enqueue_my_action_script() {
    wp_enqueue_script( 'my-action-script', 'path/to/my-action-script.js', array( 'jquery' ),
    null, true );
    wp_localize_script( 'my-action-script', 'my_action_data', array(
        'ajaxurl' => admin_url( 'admin-ajax.php' ),
    ) );
}
```

my-action-script.js:

```
(function( $ ) {
    'use strict';

    $( document ).on( 'ready', function() {
        var data = {
            action: 'my_action',
            field2: 'Hello World',
            field3: 3
        };

        $.getJSON( my_action_data.ajaxurl, data, function( json ) {
            if ( json.success ) {
                alert( 'yes!' );
            } else {
                alert( json.data.message );
            }
        }
    )
}
```

```
    } );  
  } );  
  
})( jQuery );
```

AJAX con .ajax () y WordPress Nonce

funciones.php

```
//Localize the AJAX URL and Nonce  
add_action('wp_enqueue_scripts', 'example_localize_ajax');  
function example_localize_ajax(){  
    wp_localize_script('jquery', 'ajax', array(  
        'url' => admin_url('admin-ajax.php'),  
        'nonce' => wp_create_nonce('example_ajax_nonce'),  
    ));  
}  
  
//Example AJAX Function  
add_action('wp_ajax_example_function', 'example_function');  
add_action('wp_ajax_nopriv_example_function', 'example_function');  
function example_function(){  
    if ( !wp_verify_nonce($_POST['nonce'], 'example_ajax_nonce') ){  
        die('Permission Denied.');    }  
  
    $firstname = sanitize_text_field($_POST['data']['firstname']);  
    $lastname = sanitize_text_field($_POST['data']['lastname']);  
  
    //Do something with data here  
    echo $firstname . ' ' . $lastname; //Echo for response  
    wp_die(); // this is required to terminate immediately and return a proper response:-  
    https://codex.wordpress.org/AJAX_in_Plugins  
}
```

example.js

```
jQuery(document).on('click touch tap', '.example-selector', function(){  
    jQuery.ajax({  
        type: "POST",  
        url: ajax.url,  
        data: {  
            nonce: ajax.nonce,  
            action: 'example_function',  
            data: {  
                firstname: 'John',  
                lastname: 'Doe'  
            },  
        },  
    },  
    success: function(response){  
        //Success  
    },  
    error: function(XMLHttpRequest, textStatus, errorThrown){  
        //Error  
    },  
    timeout: 60000  
    });
```

```
return false;
});
```

wp_ajax - funcionalidad principal + _wpnonce check

funciones.php :

```
function rm_init_js() {
    wp_enqueue_script( 'custom-ajax-script', get_template_directory_uri() . '/js/ajax.js',
array( 'jquery', 'wp-util' ), '1.0', true );
    // pass custom variables to JS
    wp_localize_script( 'custom-ajax-script', 'BEJS', array(
        'action' => 'custom_action',
        'nonce' => wp_create_nonce( 'test-nonce' )
    ) );
}

add_action( 'wp_enqueue_scripts', 'rm_init_js' );

function rm_ajax_handler() {
    check_ajax_referer( 'test-nonce' );

    extract( $_POST );
    $data = compact( 'first_name', 'last_name', 'email' );

    foreach ( $data as $name => $value ) {
        switch ( $name ) {
            case 'first_name':
            case 'last_name':
                $data[ $name ] = ucfirst( sanitize_user( $value ) );
                break;
            case 'email':
                $data[ $name ] = sanitize_email( $value );
                break;
        }
    }

    $userID = email_exists( $data['email'] );

    if ( ! $userID ) {
        wp_send_json_error( sprintf( __( 'Something went wrong! %s try again!', 'textdomain'
), $data['first_name'] . ' ' . $data['last_name'] ) );
    }

    wp_update_user( array(
        'ID'            => $userID,
        'display_name' => $data['first_name'] . ' ' . $data['last_name'],
        'first_name'   => $data['first_name'],
        'last_name'    => $data['last_name'],
    ) );

    wp_send_json_success( sprintf( __( 'Welcome Back %s', 'textdomain' ), $data['first_name']
. ' ' . $data['last_name'] ) );
}

add_action( 'wp_ajax_custom_action', 'rm_ajax_handler' );
add_action( 'wp_ajax_nopriv_custom_action', 'rm_ajax_handler' );
```

ajax.js

```
;(function() {
    wp.ajax.post(BEJS.action, {
        first_name: 'john',
        last_name: '%65doe',
        email: 'john.doe@example.com',
        _ajax_nonce: BEJS.nonce
    }).done( function( response ) {
        alert(`Success: ${response}`);
    }).fail( function( response ) {
        alert(`Error: ${response}`);
    });
})();
```

OOP ajax presentación utilizando una clase simple con nonce

Puedes copiar y pegar todo este plugin para probarlo. El esqueleto de clase se usa desde [aquí](#) .

class-oop-ajax.cpp

```
<?php

/**
 * The plugin bootstrap file
 *
 * This file is read by WordPress to generate the plugin information in the plugin
 * Dashboard. This file defines a function that starts the plugin.
 *
 * @wordpress-plugin
 * Plugin Name:      Oop Ajax
 * Plugin URI:       http://
 * Description:      A simple example of using OOP principles to submit a form from the
 * front end.
 * Version:          1.0.0
 * Author:           Digvijay Naruka
 * Author URI:       http://
 * License:           GPL-2.0+
 * License URI:      http://www.gnu.org/licenses/gpl-2.0.txt
 * Text Domain:      oop-ajax
 * Domain Path:      /languages
 */

// If this file is called directly, abort.
if ( ! defined( 'WPINC' ) ) {
    die;
}

class Oop_Ajax {

    // Put all your add_action, add_shortcode, add_filter functions in __construct()
    // For the callback name, use this: array($this, '<function name>')
    // <function name> is the name of the function within this class, so need not be globally
```

```

unique
// Some sample commonly used functions are included below
public function __construct() {

    // Add Javascript and CSS for front-end display
    add_action('wp_enqueue_scripts', array($this,'enqueue'));

    // Add the shortcode for front-end form display
    add_action( 'init', array( $this, 'add_form_shortcode' ) );
    // Add ajax function that will receive the call back for logged in users
    add_action( 'wp_ajax_my_action', array( $this, 'my_action_callback' ) );
    // Add ajax function that will receive the call back for guest or not logged in users
    add_action( 'wp_ajax_nopriv_my_action', array( $this, 'my_action_callback' ) );

}

// This is an example of enqueueing a JavaScript file and a CSS file for use on the front
end display
public function enqueue() {
    // Actual enqueues, note the files are in the js and css folders
    // For scripts, make sure you are including the relevant dependencies (jquery in this
case)
    wp_enqueue_script('my-ajax-script', plugins_url('js/oop-ajax.js', __FILE__),
array('jquery'), '1.0', true);

    // Sometimes you want to have access to data on the front end in your Javascript file
    // Getting that requires this call. Always go ahead and include ajaxurl. Any other
variables,
    // add to the array.
    // Then in the Javascript file, you can refer to it like this:
my_php_variables.ajaxurl
    wp_localize_script( 'my-ajax-script', 'my_php_variables', array(
        'ajaxurl' => admin_url('admin-ajax.php'),
        'nonce' => wp_create_nonce('_wpnonce')
    ));

}

/**
 * Registers the shortcode for the form.
 */
public function add_form_shortcode() {

    add_shortcode( "oop-ajax-add-form", array( $this, "add_form_front_end" ) );

}

/**
 * Processes shortcode oop-ajax-add-form
 *
 * @param array $atts The attributes from the shortcode
 *
 * @return mixed $output Output of the buffer
 */
function add_form_front_end($atts, $content) {

    echo "<form id='my_form'>";

    echo "<label for='name'>Name: </label>";
    echo "<input id='name' type='text' name='name' ";

```

```

        echo "<br>" ;

        echo "<label id='email' for='email'>Email: </label>" ;
        echo "<input type='text' name='email'>";

        echo "<br>" ;

        echo "<input type='hidden' name='action' value='my_action' >" ;
        echo "<input id='submit_btn' type='submit' name='submit' value='submit'> ";

        echo "</form><br><br>";
        echo "<div id='response'>ajax response will be here</div> ";
    }

    /**
     * Callback function for the my_action used in the form.
     *
     * Processes the data recieved from the form, and you can do whatever you want with it.
     *
     * @return echo response string about the completion of the ajax call.
     */
    function my_action_callback() {
        // echo wp_die('<pre>' . print_r($_REQUEST) . "<pre>");

        check_ajax_referer( '_wpnonce', 'security' );

        if( ! empty( $_POST ) ){

            if ( isset( $_POST['name'] ) ) {

                $name = sanitize_text_field( $_POST['name'] ) ;
            }

            if( isset( $_POST['email'] ) ) {

                $email = sanitize_text_field( $_POST['email'] ) ;
            }

            ////////////////////////////////////////////////////////////////////
            // do stuff with values
            // example : validate and save in database
            //           process and output
            ////////////////////////////////////////////////////////////////////

            $response = "Wow <strong style= 'color:red'>". $name . "!!</style></strong> you
            rock, you just made ajax work with oop.";
            //this will send data back to the js function:
            echo $response;

        } else {

            echo "Uh oh! It seems I didn't eat today";
        }

        wp_die(); // required to terminate the call so, otherwise wordpress initiates the
        termination and outputs weird '0' at the end.

    }

}

//initialize our plugin

```

```
global $plugin;

// Create an instance of our class to kick off the whole thing
$plugin = new Oop_Ajax();
```

oop-ajax.js

Coloque el archivo js dentro del directorio js, es decir, oop-ajax / js / oop-ajax.js

```
(function($) {
    'use strict';

    $("#submit_btn").on('click', function() {
        // set the data
        var data = {
            action: 'my_action',
            security: my_php_variables.nonce,
            name: $("#name").val(),
            email: $("#email").val()
        }

        $.ajax({
            type: 'post',
            url: my_php_variables.ajaxurl,
            data: data,
            success: function(response) {
                //output the response on success
                $("#response").html(response);
            },
            error: function(err) {
                console.log(err);
            }
        });

        return false;
    });
})(jQuery);
```

Lea AJAX en línea: <https://riptutorial.com/es/wordpress/topic/2335/ajax>

Capítulo 13: Añadir Shortcode

Sintaxis

- `add_shortcode($tag , $func);`

Parámetros

Parámetro	Detalles
\$ etiqueta	<i>(cadena) (requerido)</i> Etiqueta de código corto para buscar en el contenido de la publicación
\$ func	<i>(se puede llamar) (requerido)</i> Gancho para ejecutarse cuando se encuentra el código corto

Observaciones

- A la devolución de llamada de shortcode se le pasarán tres argumentos: los atributos de shortcode, el contenido de shortcode (si corresponde) y el nombre del shortcode.
- Solo puede haber un gancho para cada shortcode. Lo que significa que si otro complemento tiene un código abreviado similar, anulará el tuyo o el tuyo anulará el suyo, según el orden en que se incluyan o se ejecuten.
- Los nombres de atributo de Shortcode siempre se convierten en minúsculas antes de pasarlos a la función de controlador. Los valores están intactos.
- Tenga en cuenta que la función llamada por el código abreviado nunca debe producir ningún tipo de salida. Las funciones de código abreviado deben devolver el texto que se utilizará para reemplazar el código abreviado. Producir la salida directamente conducirá a resultados inesperados. Esto es similar a la forma en que deben comportarse las funciones de filtro, ya que no deben producir los efectos secundarios esperados de la llamada, ya que no puede controlar cuándo y desde dónde se llaman.

Examples

Código corto simple para la publicación reciente

`add_shortcode` es una palabra clave wp.

```
// recent-posts is going to be our shortcode.
add_shortcode('recent-posts', 'recent_posts_function');

// This function is taking action when recent post shortcode is called.
function recent_posts_function() {
    query_posts(array('orderby' => 'date', 'order' => 'DESC' , 'showposts' => 1));
```

```

if (have_posts()) :
    while (have_posts()) : the_post();
        $return_string = '<a href="'.get_permalink().'">'.get_the_title().'</a>';
    endwhile;
endif;
wp_reset_query();
return $return_string;
}

```

Este fragmento se puede colocar en las `functions.php` su tema.php.

[recent-posts] Este es un código corto para publicaciones recientes. Podemos aplicar este código abreviado en el backend (como páginas, publicaciones, widgets).

También podemos usar el mismo shortcode dentro de nuestro código. con la ayuda de `do_shortcode`.

P.ej. `echo do_shortcode('[recent-posts]');`

Código abreviado avanzado para publicaciones recientes

Esta función toma el parámetro de cuántas publicaciones recientes desea mostrar.

Ej: desea mostrar solo cinco publicaciones recientes. Acaba de pasar los argumentos con `posts = "5"` (puede pasar cualquier número de publicaciones recientes que desee mostrar).

La función busca solo cinco publicaciones recientes de la base de datos.

```

// recent-posts is going to be our shortcode.
add_shortcode('recent-posts', 'recent_posts_function');

// Functions takes parameter such as posts="5".
function recent_posts_function($atts){
    extract(shortcode_atts(array(
        'posts' => 1,
    ), $atts));

    $return_string = '<ul>';
    query_posts(array('orderby' => 'date', 'order' => 'DESC' , 'showposts' => $posts));
    if (have_posts()) :
        while (have_posts()) : the_post();
            $return_string .= '<li><a href="'.get_permalink().'">'.get_the_title().'</a></li>';
        endwhile;
    endif;
    $return_string .= '</ul>';

    wp_reset_query();
    return $return_string;
}

```

P.ej. `echo do_shortcode('[recent-posts posts="5"]');`

Lea Añadir Shortcode en línea: <https://riptutorial.com/es/wordpress/topic/6580/anadir-shortcode>

Capítulo 14: API de opciones

Introducción

Las opciones son partes de datos que WordPress utiliza para almacenar varias preferencias y configuraciones. La API de opciones es una forma simple y estandarizada de almacenar datos en la base de datos. La API facilita la creación, el acceso, la actualización y la eliminación de opciones.

Sintaxis

- // Crear nueva opción dentro de WordPress
`add_option ($ option, $ value =, $ deprecated =, $ autoload = 'yes');`
- // Elimina una opción de la base de datos.
`delete_option ($ option);`
- // Recuperar una opción guardada
`get_option ($ option, $ default = false);`
- // Actualizar el valor de una opción que ya fue agregada.
`update_option ($ option, $ newvalue);`
- // También hay versiones * `_site_option ()` de estas funciones,
// para manipular las opciones de toda la red en WordPress Multisite
- // Crear nueva opción de red
`add_site_option ($ option, $ value =, $ deprecated =, $ autoload = 'yes');`
- // Elimina una opción de red
`delete_site_option ($ option);`
- // Recuperar una opción de red guardada
`get_site_option ($ opción, $ por defecto = falso);`
- // Actualizar el valor de una opción que ya fue agregada.
`update_site_option ($ option, $ newvalue);`

Observaciones

La API de opciones es una forma simple y estandarizada de trabajar con datos almacenados en la tabla de opciones de la base de datos MySQL. La API facilita la creación, lectura, actualización y eliminación de opciones.

Examples

get_option

La función `get_option` se utiliza para recuperar un valor de la tabla de opciones en función del nombre de la opción.

Puede usar el siguiente código para obtener la dirección de correo electrónico de un administrador del sitio de WordPress.

```
<?php echo get_option('admin_email'); ?>
```

`get_option()` tiene un segundo argumento opcional, que le permite establecer un valor predeterminado para devolver en el caso de que la opción solicitada no esté establecida. Por defecto, este argumento es `false`.

Para recuperar una cadena de texto, y usar una cadena de repetición, si el texto no está configurado en la tabla de opciones, puede hacer esto:

```
<?php get_option( 'my_text', "I don't have anything written. Yet." ); ?>
```

add_option

La función `add_option` se usa para insertar una nueva fila en la tabla de opciones.

Esto insertará una nueva fila en la tabla de opciones con el nombre de opción `some_option_name` y valor como `some_option_value`

```
<?php add_option( 'some_option_name', 'some_option_value' ); ?>
```

delete_option

La función `delete_option` se usa para eliminar una opción de la tabla de opciones.

Esto eliminará `my_custom_option` de la tabla de opciones.

```
<?php delete_option( 'my_custom_option' ); ?>
```

update_option

La función `update_option` se usa para actualizar un valor que ya existe en la tabla de opciones. Si la opción no existe, entonces la opción se agregará con el valor de la opción.

Esto establecerá el estado de comentario predeterminado en 'cerrado':

```
update_option( 'default_comment_status', 'closed' );
```

Lea API de opciones en línea: <https://riptutorial.com/es/wordpress/topic/7854/api-de-opciones>

Capítulo 15: API REST

Introducción

La API REST de WordPress proporciona puntos finales de API para los tipos de datos de WordPress que permiten a los desarrolladores interactuar con sitios de forma remota mediante el envío y la recepción de objetos JSON (notación de objetos de JavaScript).

Cuando envíe contenido o realice una solicitud a la API, la respuesta se devolverá en JSON. Esto permite a los desarrolladores crear, leer y actualizar contenido de WordPress desde JavaScript del lado del cliente o desde aplicaciones externas, incluso aquellas escritas en idiomas más allá de PHP.

Observaciones

Para que este ejemplo simple de la API REST de WordPress funcione, necesita aprender cómo funciona con más detalle. La documentación oficial recomienda aprender sobre:

1. Rutas / puntos finales, que son asignaciones de métodos HTTP individuales a rutas conocidas como "puntos finales", lo hace utilizando la función [register_rest_route \(\)](#) , y aquí puede encontrar más información sobre [rutas y puntos finales](#).
2. Solicitudes: la API REST de WordPress define la clase `WP_REST_Request` que se utiliza para almacenar y recuperar información para la solicitud actual. `WP_REST_Request` objetos `WP_REST_Request` se generan automáticamente cuando realiza una solicitud HTTP a una ruta registrada. Los datos especificados en la solicitud determinarán qué respuesta obtendrá de la API. Aquí puede aprender más sobre la [clase WP_REST_Request](#) .
3. Respuestas: son los datos que obtiene de la API. `WP_REST_Response` proporciona una forma de interactuar con los datos de respuesta devueltos por los puntos finales. En su definición de punto final, asigna un nombre a la función de devolución de llamada (respuesta) para servir a su interacción. Aquí puede obtener más información sobre la [clase WP_REST_Response](#) .
4. Esquema: cada punto final requiere y proporciona estructuras de datos ligeramente diferentes, y esas estructuras se definen en el esquema de API. Si desea puntos finales mantenibles, detectables y fácilmente extensibles, se recomienda utilizar el esquema. Aquí puedes aprender más sobre el [esquema](#) .
5. Clases de controladores: reúnen todos los elementos en un solo lugar. Con una clase de controlador, puede administrar el registro de rutas y puntos finales, manejar solicitudes, utilizar esquemas y generar respuestas API. Ya has aprendido sobre dos clases de controlador: `WP_REST_Request` y `WP_REST_Response` . Aquí puedes aprender más sobre las [clases de controladores](#)

Nota: parte de esta información está tomada del [manual](#) oficial de [REST API de Wordpress](#).

Examples

Ejemplo de trabajo completo

```
add_action('rest_api_init', 'my_rest_validate_endpoint' );
function my_rest_validate_endpoint() {

    // Declare our namespace
    $namespace = 'myrest/v1';

    // Register the route
    // Example URL matching this route:
    // http://yourdomain/wp-json/myrest/v1/guides/tag=europe/price=29
    register_rest_route($namespace,
        // Using regular expressions we can initially validate the input
        '/guides/tag=(?P<tag>[a-zA-Z0-9-]+)/price=(?P<price>[0-9]+)',
        // We can have multiple endpoints for one route
        array(
            array(
                'methods'   => 'GET',
                'callback'  => 'my_get_guides_handler'
            )
        ),
        // We can register our schema callback
        // 'schema' => 'my_get_guide_schema',

    );

    // You can register another route here the same way
}

// The callback handler for the endpoint
function my_get_guides_handler(WP_REST_Request $request) {

    // Get the parameters:
    $tag = $request->get_param('tag');
    $price = $request->get_param('price');

    // Do something with the parameters
    // for instance: get matching guides from the DB into an array - $results
    // ...

    // Prepare the response
    $message = "We've found " . count($results) . " guides ";
    $message .= "(searching for a tag: " . $tag . ", with a price tag: " . $price . ")";

    // The response gets automatically converted into a JSON format
    return new WP_REST_Response(
        array(
            'results' => $results,
            'message' => $message,
            'status' => 'OK'
        ),
        200 );
}
```

Lea API REST en línea: <https://riptutorial.com/es/wordpress/topic/10645/api-rest>

Capítulo 16: Asegure su instalación

Observaciones

Los sitios web de WordPress son frecuentemente hackeados. Este tema es para técnicas y prácticas que aumentan la seguridad de su instalación de WordPress más allá de lo que se logra en una instalación básica.

Además de este tema, otro buen lugar para leer acerca de cómo asegurar una instalación de WordPress es la [página del Codex de WordPress de Hardening](#) .

Examples

Deshabilitar el editor de archivos

El editor de archivos que viene con WordPress es un riesgo de seguridad. Si un atacante obtiene acceso de administrador a su sitio web de WordPress, podrá insertar fácilmente código malicioso en los archivos de temas y complementos. También es un riesgo para los clientes que no saben lo que están haciendo. Una vez colocados dos puntos en el editor de archivos, se puede romper un sitio y hacerlo inaccesible desde el navegador.

En su archivo `wp-config.php` WordPress, deshabilite el editor de archivos agregando la siguiente línea de código.

```
define( 'DISALLOW_FILE_EDIT', true );
```

Esa línea tendrá el efecto deseado cuando se agregue también al archivo `functions.php` su tema, pero es mejor agregarlo a `wp-config.php` .

Si está utilizando [WordPress CLI](#) para instalar WordPress, puede usar el siguiente comando para crear un archivo `wp-config.php` con la edición de archivos deshabilitada.

```
/* declare variables beforehand or substitute strings in */
wp core config --dbname="$MYSQL_DBNAME" --dbuser="$MYSQL_USERNAME" --dbpass="$MYSQL_PASS" --
dbprefix="$WP_DBPREFIX"_ --locale=en_AU --extra-php <<PHP
define( 'DISALLOW_FILE_EDIT', true );
PHP
```

Este método es útil si instala WordPress con una secuencia de comandos.

Mueve wp-config.php

La información más confidencial de una instalación de WordPress se almacena en el archivo `wp-config.php` . Si un pirata informático obtiene acceso a este archivo, tendrá el control total de su sitio web.

Por defecto, `wp-config.php` se almacena en la carpeta de instalación de WordPress. Para hacer que este archivo sea más difícil de robar, puede sacarlo de la carpeta accesible desde la web. Si lo mueve solo una carpeta arriba, WordPress lo encontrará automáticamente. Si mueve `wp-config.php` a una ubicación diferente, cree un archivo vacío llamado `wp-config.php` en la carpeta de instalación de WordPress. Luego agregue lo siguiente:

```
define('ABSPATH', dirname(__FILE__) . '/');
// '../..wp-config.php' defines location two folders above installation folder.
// Substitute with actual location of wp-config.php file as necessary.
require_once(ABSPATH . '../..wp-config.php');
```

Es posible que deba hacer `php` ejecutable en la carpeta donde coloca `wp-config.php`. Debe hacer `php` ejecutable en la menor cantidad de carpetas posible. Un buen sistema coloca la instalación de WordPress en `/path/to/wordpress/install/` y la configuración en `/path/to/wordpress/config`. Asegúrate de que la carpeta de configuración no sea accesible desde la web y no coloques ninguna otra información confidencial que se coloque en `/path/to/` o superior en la jerarquía de carpetas. En ese caso, escribiría una línea similar a la siguiente en su `php.ini`:

```
open_basedir = "/path/to/wordpress/install;/path/to/wordpress/config"
```

Esta técnica es controvertida y algunas personas no creen que mejore la seguridad. Se puede leer una amplia discusión sobre el tema en [esta pregunta de WordPress StackExchange](#).

Establecer un prefijo personalizado para las tablas de WordPress

Cuando instala WordPress en su servidor, la secuencia de comandos de instalación colocará un prefijo delante de todos los nombres de las tablas de WordPress MySQL. Este prefijo se establece en `'wp_'` por defecto. La tabla de publicaciones de WordPress se llamará `wp_posts` por ejemplo. Al cambiar el prefijo de la tabla, puede crear cierta seguridad por oscuridad. De esta manera, cuando un pirata informático intente ataques de inyección de SQL, tendrá que adivinar el prefijo de su tabla en lugar de simplemente usar `'wp_'`. Puedes configurar este prefijo para que sea lo que quieras.

Establecer Prefijo en Nueva Instalación de WordPress

Si utiliza la famosa instalación de 5 minutos, cambie el prefijo en el campo durante la instalación.



Below you should enter your database connection details. If you're not sure about these, contact your host.

Database Name	<input type="text" value="wordpress"/>	The name of the database you want to use with WordPress.
Username	<input type="text" value="username"/>	Your database username.
Password	<input type="text" value="password"/>	Your database password.
Database Host	<input type="text" value="localhost"/>	You should be able to get this info from your web host, if localhost doesn't work.
Table Prefix	<input type="text" value="wp_"/>	If you want to run multiple WordPress installations in a single database, change this.

Si realiza la instalación a través de la CLI de WordPress use el siguiente comando:

```
// set other variables above, or substitute your strings in.
WP_DBPREFIX=foo
wp core config --dbname="$MYSQL_DBNAME" --dbuser="$MYSQL_USERNAME" --dbpass="$MYSQL_PASS" --
dbprefix="$WP_DBPREFIX_" --locale=en_AU
```

Cambiar prefijo en una instalación existente

Cambiar el prefijo es un poco más difícil. Primero use un programa FTP como FileZilla para editar el archivo `wp-config.php`. Cambie la entrada `$table_prefix = 'wp_';` a `$table_prefix = 'foo_';` sustituyendo 'foo' por su prefijo deseado.

A continuación tendremos que editar la base de datos. Si tiene acceso a phpMyAdmin, inicie sesión y haga lo siguiente:

- Seleccione la base de datos de WordPress

<input type="checkbox"/> wp_links	Browse Structure
<input type="checkbox"/> wp_options	Browse Structure
<input type="checkbox"/> wp_postmeta	Browse Structure
<input type="checkbox"/> wp_posts	Browse Structure

- Seleccione todas las tablas y en el menú desplegable seleccione reemplazar el prefijo de la

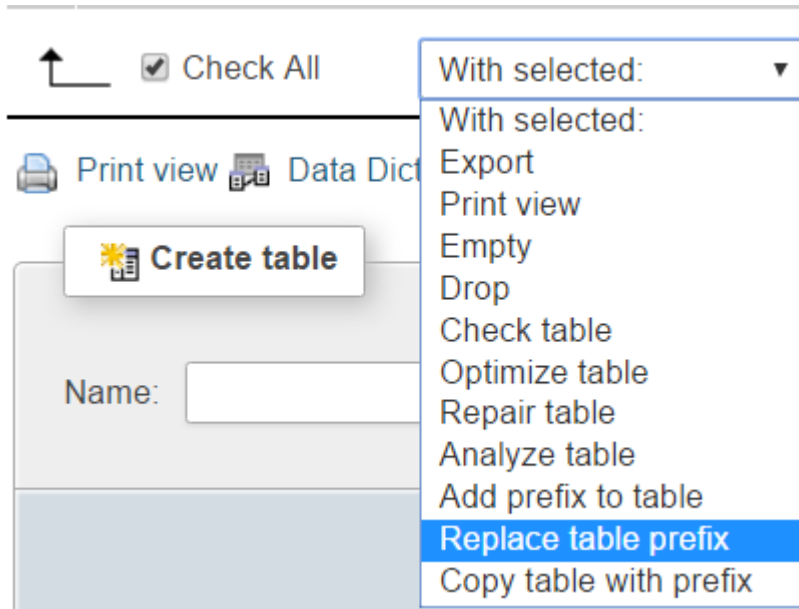


tabla.

- En "De" escriba 'wp_'. En "Para" escriba su prefijo, 'foo_' en este ejemplo y presione

Replace table prefix:

From

To

"Enviar".

- Las tablas ahora deberían verse así:

<input type="checkbox"/> foo_links	Browse Structure
<input type="checkbox"/> foo_options	Browse Structure
<input type="checkbox"/> foo_postmeta	Browse Structure
<input type="checkbox"/> foo_posts	Browse Structure

Si no puede usar phpMyAdmin, use el siguiente comando MySQL:

```
RENAME table `wp_comments` TO `foo_comments`
```





































Tendrá que ejecutar ese comando para cada tabla, sustituyendo 'comentarios' por los otros nombres de tabla.

A continuación necesitamos cambiar algunas entradas en algunas tablas. Ejecute esta consulta en la tabla 'foo_options'





































```
SELECT * FROM foo_options WHERE option_name LIKE '%user_roles%'
```

Debe aparecer una entrada con option_name de 'wp_user_roles'. En esa entrada, cambie la entrada 'wp_user_roles' de wp_user_roles a foo_user_roles .

Luego abre la tabla 'foo_usermeta' y encuentra cada entrada con 'wp_' en la parte delantera.

 Edit  Copy  Delete	10	1	wp_capabilities	a:1:{s
 Edit  Copy  Delete	26	2	wp_capabilities	a:1:{s
 Edit  Copy  Delete	75	3	wp_capabilities	a:1:{s
 Edit  Copy  Delete	14	1	wp_dashboard_quick_press_last_post_id	2595
 Edit  Copy  Delete	29	2	wp_dashboard_quick_press_last_post_id	1283
 Edit  Copy  Delete	11	1	wp_user_level	10
 Edit  Copy  Delete	27	2	wp_user_level	10
 Edit  Copy  Delete	76	3	wp_user_level	0
 Edit  Copy  Delete	15	1	wp_user-settings	advlm
 Edit  Copy  Delete	41	2	wp_user-settings	editor
 Edit  Copy  Delete	16	1	wp_user-settings-time	14753
 Edit  Copy  Delete	42	2	wp_user-settings-time	14134

y cambiarlo a 'foo_'. La cantidad de entradas que debe cambiar dependerá de cuántos usuarios tenga.

<input type="checkbox"/>	 Edit	 Copy	 Delete	10	1	foo_capabilities	a:1
<input type="checkbox"/>	 Edit	 Copy	 Delete	26	2	foo_capabilities	a:1
<input type="checkbox"/>	 Edit	 Copy	 Delete	75	3	foo_capabilities	a:1
<input type="checkbox"/>	 Edit	 Copy	 Delete	14	1	foo_dashboard_quick_press_last_post_id	25
<input type="checkbox"/>	 Edit	 Copy	 Delete	29	2	foo_dashboard_quick_press_last_post_id	12
<input type="checkbox"/>	 Edit	 Copy	 Delete	11	1	foo_user_level	10
<input type="checkbox"/>	 Edit	 Copy	 Delete	27	2	foo_user_level	10
<input type="checkbox"/>	 Edit	 Copy	 Delete	76	3	foo_user_level	0
<input type="checkbox"/>	 Edit	 Copy	 Delete	15	1	foo_user-settings	ad
<input type="checkbox"/>	 Edit	 Copy	 Delete	41	2	foo_user-settings	ed
<input type="checkbox"/>	 Edit	 Copy	 Delete	16	1	foo_user-settings-time	14
<input type="checkbox"/>	 Edit	 Copy	 Delete	42	2	foo_user-settings-time	14

Eso debería ser todo lo que necesita para cambiar el prefijo en una instalación existente

Lea Asegure su instalación en línea: <https://riptutorial.com/es/wordpress/topic/7594/asegure-su-instalacion>

Capítulo 17: Barras laterales

Sintaxis

- register_sidebar (\$ args)
- get_sidebar (string \$ name = null)

Parámetros

Parámetro	Detalles
\$ args	(string array) (Opcional) Construye una barra lateral basada en los valores de name e id
\$ nombre	* (cadena) (Opcional) El nombre de la barra lateral especializada. Valor por defecto: nulo

Observaciones

Las opciones de argumento son:

- **nombre** : nombre de la barra lateral (*predeterminado*: 'barra lateral' localizada e ID numérica) .
- **id** - ID de la barra lateral: debe estar todo en minúsculas, sin espacios (*por defecto*: una identificación numérica con incremento automático) . Si no establece el valor del argumento id, obtendrá mensajes E_USER_NOTICE en modo de depuración, comenzando con la versión 4.2.
- **description** : descripción de texto de qué / dónde está la barra lateral. Se muestra en la pantalla de gestión de widgets. (Desde 2.9) (*por defecto*: vacío)
- **class** : clase CSS para asignar a la barra lateral en la página Apariencia -> Administración de widgets. Esta clase solo aparecerá en la fuente de la página de administración del widget de WordPress. No se incluirá en la parte delantera de su sitio web. **Nota** : la sidebar valor se añadirá al valor de la clase. Por ejemplo, una clase de tal resultará en un valor de clase de sidebar-tal . (*por defecto*: vacío) .
- **before_widget** : HTML para colocar antes de cada widget (*predeterminado*: <li id="%1\$s" class="widget %2\$s">) **Nota** : usa sprintf para la sustitución de variables
- **after_widget** : HTML para colocar después de cada widget (*predeterminado*: \n) .
- **before_title** : HTML para colocar antes de cada título (*predeterminado*: <h2 class="widgettitle">) .
- **after_title** : HTML para colocar después de cada título (*predeterminado*: </h2>\n) .

Examples

Registro de barras laterales

En tus `functions.php` puedes registrar nuevas barras laterales con este código

```
/**
 * Registers sidebars
 *
 * @param array Array with default or specified array values
 * @since      1.0.0
 */
if ( function_exists( 'register_sidebar' ) ) {
    register_sidebar( array (
        'name'          => esc_html__( 'Primary Sidebar', 'mytheme' ),
        'id'            => 'primary-widget-area',
        'description'   => esc_html__( 'The Primary Widget Area', 'mytheme' ),
        'before_widget' => '<div id="%1$s" class="widget %2$s">',
        'after_widget'  => '</div>',
        'before_title'  => '<div class="sidebar-widget-heading"><h3>',
        'after_title'   => '</h3></div>',
    ) );

    register_sidebar( array (
        'name'          => esc_html__( 'Secondary Sidebar', 'mytheme' ),
        'id'            => 'secondary-widget-area',
        'description'   => esc_html__( 'The Secondary Widget Area', 'mytheme' ),
        'before_widget' => '<div id="%1$s" class="widget %2$s">',
        'after_widget'  => '</div>',
        'before_title'  => '<div class="sidebar-widget-heading"><h3>',
        'after_title'   => '</h3></div>',
    ) );
}
```

Puede agregar tantas barras laterales como desee.

Obtener barra lateral

También puede crear su propio archivo de barra lateral en el tema para llamarlo en diferentes plantillas. Copie y pegue `sidebar.php` del tema actual y cambie el nombre (es decir, `sidebar-book.php`)

En la plantilla, puedes llamar a esta barra lateral usando `get_sidebar('book')`. Usando esto puedes llamar a diferentes barras laterales en diferentes páginas.

Lea Barras laterales en línea: <https://riptutorial.com/es/wordpress/topic/6293/barras-laterales>

Capítulo 18: Bucle principal alternante (filtro `pre_get_posts`)

Sintaxis

- `add_action('pre_get_posts', 'callback_function_name');`
- función `callback_function_name($ consulta) {}`
- // para PHP 5.3.0 o superior
- `add_action('pre_get_posts', función($ consulta) {});`

Parámetros

Parámetro	Detalles
<code>\$ consulta</code>	(<code>WP_Query</code>) objeto de bucle

Observaciones

Si está utilizando PHP 5.3.0 o superior, puede usar cierres ([funciones anónimas](#))

```
add_action( 'pre_get_posts', function( $query ) {
    if( !$query->is_main_query() || is_admin() ) return;

    // this code will run only if
    // - this query is main query
    // - and this is not admin screen
});
```

Examples

Aún más específica la orientación de bucle

Digamos que queremos cambiar el *bucle principal* , solo para taxonomía específica o tipo de publicación.

Apuntando solo al bucle principal en la página de archivo del tipo de publicación de `book` .

```
add_action( 'pre_get_posts', 'my_callback_function' );

function my_callback_function( $query ) {
    if( !$query->is_main_query() || is_admin() ) return;
    if( !is_post_type_archive( 'book' ) ) return;

    // this code will run only if
    // - this query is main query
```



```
// - and this is not admin screen
// - and we are on 'book' post type archive page
}
```

También puede consultar la categoría, etiqueta o página de archivo de taxonomía personalizada utilizando `is_category()` , `is_tag()` e `is_tax()` .

Puedes usar cualquier *etiqueta condicional* disponible en WordPress.

Mostrar publicaciones de una sola categoría

```
add_action( 'pre_get_posts', 'single_category' );

function single_category( $query ) {
    if( !$query->is_main_query() || is_admin() ) return;

    $query->set( 'cat', '1' );
    return;
}
```

Filtrar el uso básico antes de las entradas.

A veces te gustaría cambiar la consulta principal de WordPress.

Filtrar `pre_get_posts` es el camino a seguir.

Por ejemplo, al usar `pre_get_posts` , puede indicar al *bucle principal* que muestre solo 5 publicaciones. O para mostrar publicaciones solo de una categoría, o excluyendo cualquier categoría, etc.

```
add_action( 'pre_get_posts', 'my_callback_function' );

function my_callback_function( $query ) {
    // here goes logic of your filter
}
```

Como puede ver, estamos pasando el objeto de consulta del *bucle principal* a nuestro argumento de función de devolución de llamada.

Nota importante aquí: **estamos pasando el argumento como referencia** . Esto significa que no es necesario que devolvamos la consulta ni configuremos ninguna variable global para que funcione. Como `$query` es una referencia al objeto de consulta principal, todos los cambios que hacemos en nuestro objeto se reflejan inmediatamente en el objeto de bucle principal.

Excluir categoría de la lista de publicaciones editar compartir

```
add_action( 'pre_get_posts', 'single_category_exclude' );

function single_category_exclude( $query ) {
    if( !$query->is_main_query() || is_admin() ) return;
}
```

```
$query->set( 'cat', '-1' );  
return;  
}
```

Cambiar posts_per_page por bucle principal

Todo lo que necesitamos hacer es usar el método `set()` del objeto `$query`.

Toma dos argumentos, primero lo que queremos establecer y segundo qué valor configurar.

```
add_action( 'pre_get_posts', 'change_posts_per_page' );  
  
function change_posts_per_page( $query ) {  
    if( !$query->is_main_query() || is_admin() ) return;  
  
    $query->set( 'posts_per_page', 5 );  
    return;  
}
```

Apuntando solo al bucle principal de WordPress

WordPress está aplicando el filtro `pre_get_posts` a literalmente cualquier bucle que genere. Significa que todos los cambios que estamos haciendo en nuestra función de devolución de llamada se aplican a todos los bucles que salen.

Obviamente no es lo que queremos en la mayoría de los escenarios.

En la mayoría de los casos, nos gustaría dirigirnos solo al *bucle principal*, y solo para pantallas que no sean de administrador.

Podemos utilizar `is_main_query()` método y `is_admin()` la función global para comprobar si estamos en el lugar correcto.

```
add_action( 'pre_get_posts', 'my_callback_function' );  
  
function my_callback_function( $query ) {  
    if( !$query->is_main_query() || is_admin() ) return;  
  
    // this code will run only if  
    // - this query is main query  
    // - and this is not admin screen  
}
```

Lea Bucle principal alternante (filtro `pre_get_posts`) en línea:

<https://riptutorial.com/es/wordpress/topic/4418/bucle-principal-alternante--filtro-pre-get-posts->

Capítulo 19: Código corto

Examples

Registro de shortcode

Shortcode es un pequeño fragmento de código que se puede agregar al editor de WordPress y generará algo diferente una vez que la página se publique o se obtenga una vista previa.

Con frecuencia, los códigos cortos se agregan al archivo `functions.php` del tema, pero **no es una buena práctica** ya que se espera que los códigos cortos sigan funcionando después de cambiar los temas. En su lugar, **escriba un complemento** para agregar esta funcionalidad.

La estructura para el registro de shortcode es:

```
function new_shortcode($atts, $content = null){
    // if parameters are needed in the shortcode
    // parameters can be set to default to something
    extract( shortcode_atts( array(
        'param_one' => 'h1'
    ), $atts ) );
    $shortcode = '<'. $param_one '>'. $content . '</'. $param_one . '>';
    return $shortcode;
}
// this is what registers the shortcode with wordpress
add_shortcode('demo-shortcode', 'new_shortcode');
```

Dentro del editor de WordPress, puede escribir:

```
[demo-shortcode param_one="h2"]Demo[/demo-shortcode]
// you don't need to insert param_one into the editor if it has a default value.
// having it in the editor will override the default
```

Una vez publicada la página, esta se convertirá en

```
<h2>Demo</h2>
```

Usando Shortcodes en el Backend de WordPress

```
[footag foo="value of 1" attribute-2="value of 2"]
```

En el administrador de wordpress, usamos códigos abreviados predefinidos al escribir el nombre del código abreviado entre corchetes y, opcionalmente, agregarle atributos separados por espacio.

Añadiendo nuevos códigos cortos

```
function footag_func( $atts ) {
    return "foo = {$atts['foo']}";
}
add_shortcode( 'footag', 'footag_func' );
```

En los complementos podemos agregar códigos cortos usando la función `add_shortcode`.

El código abreviado se puede utilizar en cualquier página o publicación de Wordpress simplemente encerrándolo entre corchetes.

```
[footag]
```

Uso de códigos cortos dentro del código PHP (temas y complementos)

```
<?php echo do_shortcode("[footag foo='Hi! I am a foo output']"); ?>
```

Para imprimir un shortcode usando php, use la función `do_shortcode` y `do_shortcode` eco del valor devuelto.

Usando Shortcodes en Widgets

```
add_filter( 'widget_text', 'shortcode_unautop' );
add_filter( 'widget_text', 'do_shortcode' );enter code here
```

Agregue esto a un complemento o al archivo `functions.php` para habilitar códigos cortos en widgets. El código primero detiene WordPress convirtiendo los saltos de línea en etiquetas de párrafos y luego permite códigos cortos para analizar widgets. El orden de las dos líneas es importante.

Lea Código corto en línea: <https://riptutorial.com/es/wordpress/topic/4952/codigo-corto>

Capítulo 20: Código corto con atributo

Sintaxis

- `add_shortcode ('your_short_code', 'your_function_name');`

Parámetros

Parámetros	Descripción y uso
\$ etiqueta	(cadena) (requerido) Etiqueta de código corto para buscar en el contenido de la publicación Predeterminado: Ninguno
\$ func	(se puede llamar) (requerido) Se engancha para ejecutarse cuando se encuentra el shortcode Predeterminado: Ninguno

Observaciones

IMPORTANTE: no utilice camelCase o UPPER-CASE para sus atributos

Puedes generar un shortcode con atributo [aquí](#)

Examples

Ejemplos de códigos cortos

Los códigos cortos de WordPress fueron introducidos en 2.5

Aquí viene un ejemplo de shortcode.

```
[button]
```

para usar shortcode directamente en el tema tienes que usar `do_shortcode()`

```
<?php echo do_shortcode('[button]'); ?>
```

Para personalizar el botón, simplemente podríamos agregar algo como:

```
[button type="twitter"]
```

O para hacerlo aún mejor, podríamos usar un shortcode adjunto:

```
[button type="twitter"]Follow me on Twitter![/button]
```

Creando un shortcode de cierre automático

El código corto más simple es el de cierre automático. Vamos a crear un enlace simple a nuestra cuenta de Twitter y luego lo agregaremos en una publicación de blog. Todo el código va en `functions.php` , que se encuentra en `/wp-content/themes/your-theme/` . Si no tienes uno, simplemente créalo y pon el código en él.

```
<?php
function button_shortcode() {
return '<a href="http://twitter.com/rupomkhondaker" class="twitter-button">Follow me on
Twitter!</a>';
}
add_shortcode('button', 'button_shortcode');
?>
```

Uso: `[button]`

Creación de un shortcode de cierre automático con parámetros.

Creación de un shortcode de cierre automático con parámetros.

```
<?php
function button_shortcode( $type ) {

    extract( shortcode_atts(
        array(
            'type' => 'value'
        ), $type ) );

    // check what type user entered
    switch ($type) {

        case 'twitter':
            return '<a href="http://twitter.com/rupomkhondaker" class="twitter-button">Follow
me on Twitter!</a>';
            break;

        case 'rss':
            return '<a href="http://example.com/rss" class="rss-button">Subscribe to the
feed!</a>';
            break;

    }
}
add_shortcode( 'button', 'button_shortcode' );
?>
```

Ahora puede elegir qué botón mostrar al definir el tipo en su shortcode.

```
[button type="twitter"]
[button type="rss"]
```

Creando un shortcode adjunto

adjuntando shortcode

El código abreviado adjunto le permite incrustar contenido dentro de su código abreviado, como BBCode si alguna vez lo ha usado.

```
<?php
function button_shortcode( $attr, $content = null ) {
return '<a href="http://twitter.com/filipstefansson" class="twitter-button">' . $content .
'</a>';
}
add_shortcode('button', 'button_shortcode');
?>
```

Para usar este código abreviado, incrusta el texto que desea usar así:

```
[button]Follow me on Twitter![/button]
```

Para mejorar aún más este botón, podríamos agregar parámetros como lo hicimos en el ejemplo anterior.

```
<?php
function button_shortcode( $atts, $content = null ) {
extract( shortcode_atts( array(
'account' => 'account',
'style' => 'style'
), $atts ) );
return '<a href="http://twitter.com/' . esc_attr($account) . '" class="twitter-button ' .
esc_attr($style) . '">' . $content . '</a>';
}
add_shortcode('button', 'button_shortcode');
?>
```

Uso:

```
[button account="rupomkhondaker" style="simple"]Follow me on Twitter![/button]
```

Shortcodes en Widgets

De forma predeterminada, WordPress no admite códigos cortos dentro de los Widgets de la barra lateral. Solo expande los códigos cortos dentro del contenido de una publicación, página o publicación personalizada. Para agregar soporte de shortcode a los widgets de la barra lateral, puede instalar un complemento o usar el siguiente código:

```
add_filter( 'widget_text', 'shortcode_unautop' );
add_filter( 'widget_text', 'do_shortcode' );
```

Es importante que estas líneas se agreguen en este orden. La primera línea evita que WordPress convierta los saltos de línea en etiquetas de párrafo, ya que esto evita que los códigos cortos funcionen. La segunda línea es la que hace que los shortcodes funcionen.

Lea Código corto con atributo en línea: <https://riptutorial.com/es/wordpress/topic/6291/codigo->

Capítulo 21: Códigos cortos

Examples

Introducción al código corto

Los códigos cortos son útiles cuando desea poder agregar elementos más complejos en línea en el editor de contenido normal.

Un shortcode en su forma más simple para se vería así:

```
function my_shortcode( ){
    return "This is a shortcode";
}
add_shortcode( 'my_shortcode', 'my_shortcode' );
```

Mostraría el texto "This is a shortcode" y lo utilizarías escribiendo [my_shortcode] en el editor de contenido.

Button shortcode

Aquí hay un ejemplo de un simple código abreviado de botón:

```
<?php
function my_button_shortcode( $atts ) {

    // Parse the input attributes and assign default values for the
    // attributes that are not specified on the shortcode
    $a = shortcode_atts( array(
        'id' => '',
        'url' => '#',
        'class' => '',
        'text' => ''
    ), $atts );

    // Open the anchor tag and add role=button for better accessibility
    $btn_html = '<a role="button"';

    // Add the href(link) attribute
    $btn_html .= ' href="' . $a['url'] . '"';

    // Add id attribute to output if specified
    if ( !empty( $a['id'] ) ) {
        $btn_html .= ' id="' . $a['id'] . '"';
    }

    $btn_classes = 'button';

    // Add class attribute to output
    $btn_html .= ' class="button ' . ( !empty( $a['class'] ) ? $a['class'] : '' ) . '"';

    // Close opening anchor tag
```

```
$btn_html .= '>'.
    // Add button text
    $a['text'].
    // Add closing anchor tag
    '</a>'. "\n";

return $btn_html;
}
add_shortcode( 'button', 'my_button_shortcode' );
```

Este código abreviado se puede usar al escribir `[button url="/my-other-page" id="my-other-page-button" class="dark" text="Click me!"]` En el editor y se mostrará el siguiente HTML:

```
<a role="button" href="/my-other-page" id="my-other-page-button"
class="button dark">Click me!</a>
```

Lea Códigos cortos en línea: <https://riptutorial.com/es/wordpress/topic/6070/codigos-cortos>

Capítulo 22: Conceptos básicos del personalizador (Agregar panel, Sección, Configuración, Control)

Examples

Añadir un panel de personalización

```
<?php
/**
 * Panel: WPCustomize
 *
 * Basic Customizer panel with basic controls.
 *
 * @since 1.0.0
 * @package WPC
 */

// Exit if accessed directly.
if ( ! defined( 'ABSPATH' ) ) {
    exit;
}

// Customize function.
if ( ! function_exists( 'wpc_panel_wpcustomize' ) ) {
    // Customize Register action.
    add_action( 'customize_register', 'wpc_panel_wpcustomize' );
    /**
     * Customize Panel.
     *
     * Adds a Panel, Section with basic controls.
     *
     * @param object WP_Customize $wp_customize Instance of the WP_Customize_Manager class.
     * @since 1.0.0
     */
    function wpc_panel_wpcustomize( $wp_customize ) {
        // Panel: Basic.
        $wp_customize->add_panel( 'wpc_panel_wpcustomize', array(
            'priority'      => 10,
            'title'         => __( 'WPCustomize Panel Title', 'WPC' ),
            'description'   => __( 'Panel Description', 'WPC' ),
            'capability'    => 'edit_theme_options'
        ) );
    }
}
```

Agregar una sección de personalizador con la configuración básica y sus controles

Los paneles pueden tener secciones, las secciones pueden tener configuraciones y las configuraciones pueden tener controles. Las configuraciones se guardan en la base de datos,

mientras que los controles para configuraciones particulares solo se usan para mostrar su configuración correspondiente al usuario.

Este código crea una `section` básica en el `panel` desde arriba. Dentro hay unas cuantas `settings` básicas con `controls` adjuntos.

```
<?php
/**
 * Section: Basic
 *
 * Basic Customizer section with basic controls.
 *
 * @since 1.0.0
 * @package WPC
 */

// Exit if accessed directly.
if ( ! defined( 'ABSPATH' ) ) {
    exit;
}

// Customize function.
if ( ! function_exists( 'wpc_customize_panel_basic' ) ) {
    // Customize Register action.
    add_action( 'customize_register', 'wpc_customize_panel_basic' );

    /**
     * Customize Panel.
     *
     * Adds a Panel, Section with basic controls.
     *
     * @param object WP_Customize $wp_customize Instance of the WP_Customize_Manager class.
     * @since 1.0.0
     */
    function wpc_customize_panel_basic( $wp_customize ) {
        // Section: Basic.
        $wp_customize->add_section( 'wpc_section_basic', array(
            'priority'      => 10,
            'panel'         => 'wpc_panel_wpcustomize',
            'title'         => __( 'Basic Section Title', 'WPC' ),
            'description'   => __( 'Section Description.', 'WPC' ),
            'capability'    => 'edit_theme_options'
        ) );

        // Setting: Text.
        $wp_customize->add_setting( 'wpc_text', array(
            'type'          => 'theme_mod',
            'default'       => 'Placeholder.',
            'transport'     => 'refresh', // Options: refresh or postMessage.
            'capability'    => 'edit_theme_options',
            'sanitize_callback' => 'esc_attr'
        ) );

        // Control: Text.
        $wp_customize->add_control( 'wpc_text', array(
            'label'         => __( 'Text', 'WPC' ),
            'description'  => __( 'Description', 'WPC' ),
            'section'      => 'wpc_section_basic',
            'type'         => 'text'
        ) );
    }
}
```

```

// Setting: Textarea.
$wp_customize->add_setting( 'wpc_textarea', array(
    'type'           => 'theme_mod',
    'default'        => 'Placeholder textarea.',
    'transport'      => 'refresh', // Options: refresh or postMessage.
    'capability'     => 'edit_theme_options',
    'sanitize_callback' => 'exc_textarea'
) );

// Control: Textarea.
$wp_customize->add_control( 'wpc_textarea', array(
    'label'          => __( 'Textarea', 'WPC' ),
    'description'    => __( 'Description', 'WPC' ),
    'section'        => 'wpc_section_basic',
    'type'           => 'textarea'
) );

// Setting: Checkbox.
$wp_customize->add_setting( 'wpc_checkbox', array(
    'type'           => 'theme_mod',
    'default'        => 'enable',
    'transport'      => 'refresh', // Options: refresh or postMessage.
    'capability'     => 'edit_theme_options',
    'sanitize_callback' => 'wpc_sanitize_checkbox' // Custom function in
customizer-sanitization.php file.
) );

// Control: Checkbox.
$wp_customize->add_control( 'wpc_checkbox', array(
    'label'          => __( 'Checkbox', 'WPC' ),
    'description'    => __( 'Description', 'WPC' ),
    'section'        => 'wpc_section_basic',
    'type'           => 'checkbox'
) );

// Setting: Radio.
$wp_customize->add_setting( 'wpc_radio', array(
    'type'           => 'theme_mod',
    'default'        => 'on',
    'transport'      => 'refresh', // Options: refresh or postMessage.
    'capability'     => 'edit_theme_options',
    'sanitize_callback' => 'wpc_sanitize_select', // Custom function in customizer-
sanitization.php file.
) );

// Control: Radio.
$wp_customize->add_control( 'wpc_radio', array(
    'label'          => __( 'Radio', 'WPC' ),
    'description'    => __( 'Description', 'WPC' ),
    'section'        => 'wpc_section_basic',
    'type'           => 'radio',
    'choices'        => array(
        'enable' => 'Enable',
        'disable' => 'Disable'
    )
) );

// Setting: Select.
$wp_customize->add_setting( 'wpc_select', array(
    'type'           => 'theme_mod',

```

```

        'default'           => 'enable',
        'transport'        => 'refresh', // Options: refresh or postMessage.
        'capability'       => 'edit_theme_options',
        'sanitize_callback' => 'wpc_sanitize_select' // Custom function in customizer-
sanitization.php file.
    ) );

    // Control: Select.
    $wp_customize->add_control( 'wpc_select', array(
        'label'           => __( 'Select', 'WPC' ),
        'description'    => __( 'Description', 'WPC' ),
        'section'        => 'wpc_section_basic',
        'type'           => 'select',
        'choices'        => array(
            'enable'     => 'Enable',
            'disable'    => 'Disable'
        )
    ) );
}
}

```

Lea Conceptos básicos del personalizador (Agregar panel, Sección, Configuración, Control) en línea: <https://riptutorial.com/es/wordpress/topic/2930/conceptos-basicos-del-personalizador--agregar-panel--seccion--configuracion--control->

Capítulo 23: Consulta de mensajes

Sintaxis

- `$ the_query = new WP_Query ($ args);`
- `$ posts_array = get_posts ($ args);`

Parámetros

Parámetro	Descripción
<code>\$ args</code>	<i>(matriz)</i> Una matriz de argumentos necesarios para una consulta: puede personalizarse según sus necesidades, por ejemplo, consultar publicaciones de una sola categoría, desde el tipo de publicación personalizada o incluso consultar cierta taxonomía.

Observaciones

Los argumentos de consulta son numerosos. [La página de código WP_Query \(\)](#) tiene una lista de parámetros. Algunos de ellos son

- [Parámetros del autor](#)
- [Parámetros de categoría](#)
- [Parámetros de etiqueta](#)
- [Parámetros de taxonomía](#)
- [Parámetro de búsqueda](#)
- [Post & Parámetros de página](#)
- [Parámetros de contraseña](#)
- [Tipo de parámetros](#)
- [Parámetros de estado](#)
- [Parámetros de paginación](#)
- [Parámetros de Order & Orderby](#)
- [Parámetros de fecha](#)
- [Parámetros de campo personalizados](#)
- [Parámetros de permiso](#)
- [Parámetros de tipo Mime](#)
- [Parámetros de almacenamiento en caché](#)
- [Parámetros de los campos de retorno](#)

Una de las cosas más importantes a tener en cuenta es:

Nunca use `query_posts ()`

`query_posts ()`

anula la consulta principal y puede causar problemas en el resto de su tema. Cada vez que necesite modificar la consulta principal (o cualquier consulta para esa materia) es usar el filtro [pre_get_posts](#) . Esto le permitirá modificar la consulta antes de que se ejecute.

Además, cuando consulta las publicaciones, siempre debe restablecerlo con [wp_reset_postdata \(\)](#) . Esto restaurará la variable `$post` global del bucle de consulta principal, y no tendrá ningún problema más adelante (como las categorías que se excluyen, porque en su bucle secundario las ha excluido y se olvidó de restablecer la consulta).

Examples

Usando el objeto WP_Query ()

Crear una instancia separada del objeto `WP_Query` es fácil:

```
$query_args = array(
    'post_type' => 'post',
    'post_per_page' => 10
);

$my_query = new WP_Query($query_args);

if( $my_query->have_posts() ):
    while( $my_query->have_posts() ): $my_query->the_post();
        //My custom query loop
    endwhile;
endif;

wp_reset_postdata();
```

Tenga en cuenta que debe crear la matriz de argumentos de consulta según su especificación. Para más detalles, mira la [página de código WP_Query](#) .

Usando get_posts ()

`get_posts ()` es un contenedor para una instancia separada de un objeto `WP_Query` . El valor devuelto es una matriz de objeto de publicación.

```
global $post;

$args = array(
    'numberposts' => 5,
    'offset'=> 1,
    'category' => 1
);

$myposts = get_posts( $args );

foreach( $myposts as $post ) :
    setup_postdata($post); ?>
    <h2><a href="<?php the_permalink(); ?>"><?php the_title(); ?></a></h2>
<?php endforeach;
wp_reset_postdata(); ?>
```


Para obtener más información, consulte la [página de código get_posts \(\)](#) .

Lea Consulta de mensajes en línea: <https://riptutorial.com/es/wordpress/topic/4002/consulta-de-mensajes>

Capítulo 24: Creación de plugins de WordPress

Introducción

Los complementos de WordPress deben tener un enfoque en la lógica del servidor y / o las partes administrativas de su aplicación de sitio web. Los buenos complementos son como las buenas aplicaciones, hacen una cosa realmente bien. Su objetivo es mejorar y automatizar partes del CMS de forma modular, ya que puede activarlas y desactivarlas. Los buenos complementos hacen uso de las acciones principales de WordPress, los filtros y los marcos de trabajo javascript y css existentes.

Examples

Configuración mínima de una carpeta de plugin y archivos

El primer paso para crear un complemento es crear la carpeta y el archivo desde donde se cargará el complemento.

Los complementos se encuentran en `/wp-content/plugins/` .

El estándar de WordPress es crear una carpeta y un nombre de archivo que se reflejen mutuamente de la siguiente manera:

```
/wp-content/plugins/myplugin/  
/wp-content/plugins/myplugin/myplugin.php
```

Después de crear su archivo de complemento, debe iniciar su complemento con un `Plugin Header` . Esto permite a WordPress escanear su archivo de complemento y almacenar los metadatos sobre el complemento, y permite a los usuarios usar esto y determinar si quieren que su complemento esté activo o inactivo. Copie esta plantilla en la parte superior de su archivo de complemento principal que creó y modifíquela según sea necesario:

```
<?php  
/**  
 * Plugin Name: PLUGIN-NAME  
 * Plugin URI: HTTP-LINK-TO-WEBSITE-PLUGIN-PAGE-OR-REPO  
 * Description: BREIF DESCRIPTION - KEEP IT SHORT  
 * Author: WORDPRESS-DOT-ORG-USERNAME  
 * Version: 0.0.1  
 * Author URI: HTTP-LINK-TO-MAINTAINER  
 * License: GNU General Public License v2 or later  
 * License URI: http://www.gnu.org/licenses/gpl-2.0.html  
 * Text Domain: short_prefix  
 */  
  
// Begin custom PHP WordPress plugin work
```

Tenga en cuenta que los complementos de WordPress normalmente deben tener una licencia como GPL. Sin embargo, la licencia no debe ser discutida como parte de este tema.

En este punto, ya debería poder ver su nuevo complemento en el área de administración de WordPress. En una configuración estándar, ubicaría esta área en `/wp-admin/plugins.php`. ¡Adelante, active su complemento y estará listo para avanzar a los próximos pasos de la construcción de su complemento!

Solo para finalizar nuestro ejemplo en algo que se puede hacer, ahora puede agregar lo siguiente a la parte inferior de su archivo de complemento:

```
die('My custom plugin is loaded : '. __FILE__);
```

La actualización de su sitio después de este cambio debería dar como resultado que todas las páginas impriman este texto. *Nunca haga esto en sitios de producción (en vivo), y siempre recuerde retirar esto antes de continuar.*

Lea [Creación de plugins de WordPress en línea](https://riptutorial.com/es/wordpress/topic/9420/creacion-de-plugins-de-wordpress):

<https://riptutorial.com/es/wordpress/topic/9420/creacion-de-plugins-de-wordpress>

Capítulo 25: Creando una plantilla personalizada

Examples

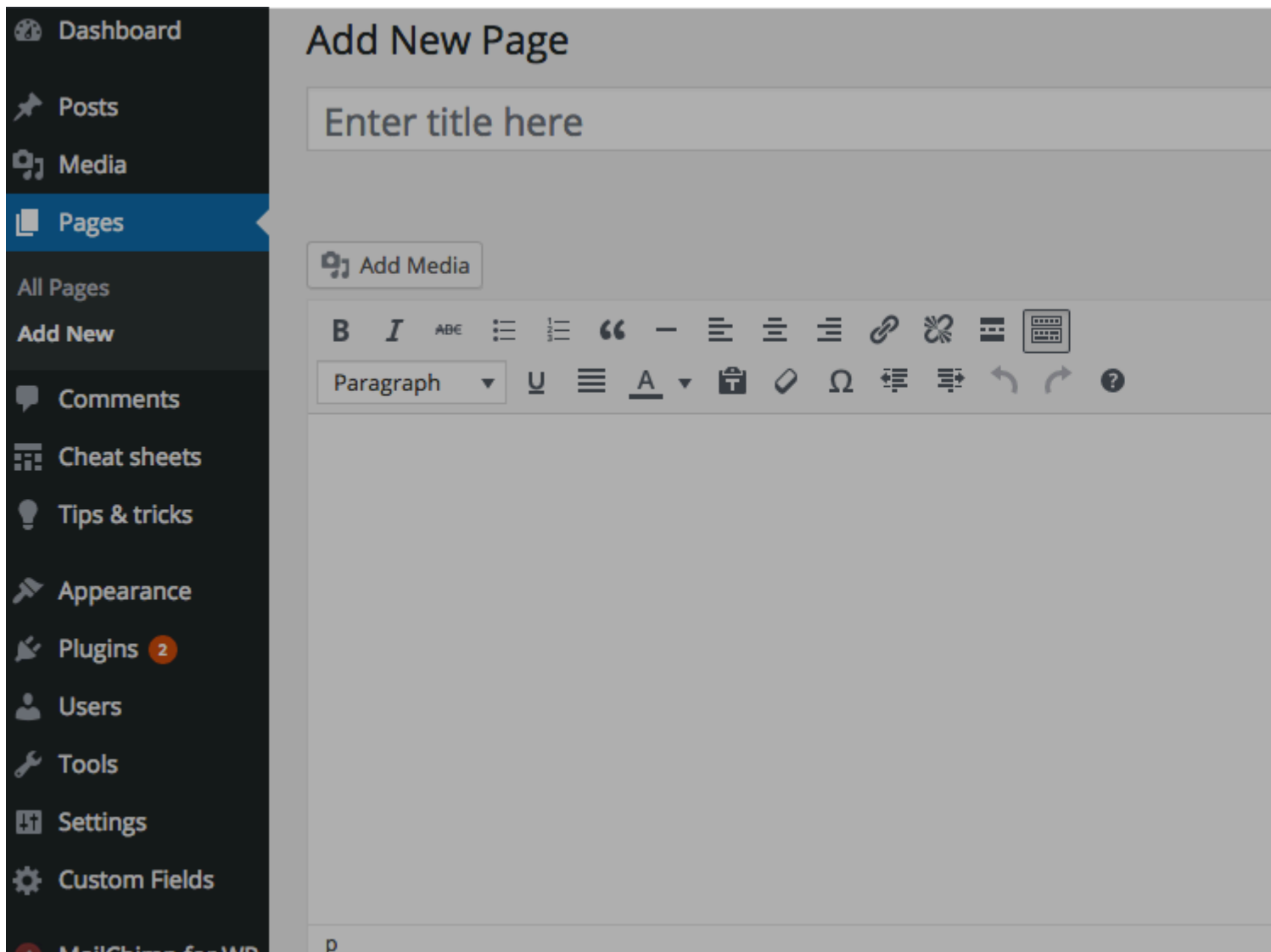
Creando plantilla básica en blanco

Para crear una plantilla personalizada, primero necesitamos crear un archivo php en un directorio de temas. Puedes nombrarlo de la forma que quieras. Para este ejemplo crearemos **example.php**.

Una y única cosa que debemos definir dentro de example.php, para que WordPress lo reconozca como plantilla, es el nombre de la plantilla. Hacemos esa compra poniendo un comentario especial en la parte superior de un archivo, como esto:

```
<?php
/*
Template Name: Example
*/
?>
```

Y ahora, cuando deberíamos ver nuestra plantilla en el **menú desplegable Plantilla** en el **cuadro Atributos de página**



Incluyendo encabezado y pie de página en nuestra plantilla.

Extendamos nuestra plantilla desde arriba e **incluimos** contenido de **header.php** y **footer.php**

Incluyendo cabecera:

Incluiremos el encabezado justo después del **comentario del nombre de la plantilla**.

Hay dos formas comunes de hacer esto. Ambos son correctos y funcionan igual, solo se trata de tu estilo y de cómo se ve el código

Primera forma:

```
<?php
/*
Template Name: Example
*/
get_header();
?>
```

Segunda forma:

```
<?php
/*
Template Name: Example
*/
?>
<?php get_header(); ?>
```

Incluyendo pie de página:

Incluir pie de página funciona de la misma manera, solo hay una cosa de la que debemos preocuparnos, y es que incluimos pie de página después de incluir encabezado. Así que la plantilla final debería verse algo como esto.

```
<?php
/*
Template Name: Example
*/
get_header();
?>

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

Plantilla personalizada con contenido

Ampliaremos nuestra plantilla e incluiremos el título de la página y un contenido.

```
<?php
/*
Template Name: Example
*/
get_header();

the_title();
the_content();

get_footer();
```

Y si lo deseas puedes envolverlos con elementos HTML como este.

```
<?php
/*
Template Name: Example
*/
get_header();

echo '<h1>' . the_title() . '</h1>';
echo '<section>' . the_content() . '</section>';

get_footer();
```

O si prefieres trabajar como un archivo HTML normal, sin usar eco

```
<?php
/*
```

```
Template Name: Example
*/
get_header();
?>

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

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

Lea **Creando una plantilla personalizada en línea:**

<https://riptutorial.com/es/wordpress/topic/2791/creando-una-plantilla-personalizada>

Capítulo 26: Crear plantilla para tipo de publicación personalizada

Examples

Creación de una plantilla personalizada para libro de tipo Publicación personalizada

Para crear una plantilla para las publicaciones individuales de nuestro tipo de publicación personalizada, necesitamos crear un archivo llamado `single-post_type_name.php` donde **post_type_name** es el nombre de nuestro tipo de publicación personalizada.

Por ejemplo, si nuestro tipo de publicación personalizada se llama "Libros", necesitamos crear un archivo PHP llamado `.php` de **libro** único. Tenga en cuenta que utilizamos el nombre singular de nuestro tipo de publicación personalizada.

Copie el contenido del archivo `single.php` de la carpeta de temas, péguelo en la nueva plantilla y guárdelo, luego la plantilla se aplicará a la página individual de tipo de publicación personalizada.

Plantillas personalizadas de tipo de publicación

Archivo de tipo de publicación personalizado:

Para crear una plantilla de archivo para un tipo de publicación personalizada, debe establecer el argumento `has_archive` igual a `true` en su función `register_post_type()`. En el siguiente ejemplo, se crea un tipo de publicación personalizada para un tipo de publicación de evento.

```
add_action( 'init', 'create_events_post_type' );
function create_events_post_type() {
    register_post_type( 'event',
        array(
            'labels' => array(
                'name' => __( 'Events' ),
                'singular_name' => __( 'Event' )
            ),
            'public' => true,
            'has_archive' => true,
        )
    );
}
```

Para [crear una plantilla](#) para nuevos tipos de publicaciones personalizadas, tendrá que crear un nuevo archivo de plantilla. Para crear una plantilla para las páginas de una sola publicación, debe

single-{post_type}.php y archive-{post_type}.php para el archivo.

El nombre de archivo de nuestra plantilla de archivo será `archive-event.php` y para la página del evento será `single-event.php`. Ambos archivos deben estar en su directorio raíz de temas.

Una plantilla de archivo de ejemplo se vería así. Tirado de los [veintisiete tema](#).

```
<?php
/**
 * The template for displaying archive pages
 *
 * @link https://codex.wordpress.org/Template_Hierarchy
 *
 * @package WordPress
 * @subpackage Twenty_Seventeen
 * @since 1.0
 * @version 1.0
 */

get_header(); ?>

<div class="wrap">

    <?php if ( have_posts() ) : ?>
        <header class="page-header">
            <?php
                the_archive_title( '<h1 class="page-title">', '</h1>' );
                the_archive_description( '<div class="taxonomy-description">', '</div>' );
            ?>
        </header><!-- .page-header -->
    <?php endif; ?>

    <div id="primary" class="content-area">
        <main id="main" class="site-main" role="main">

            <?php
                if ( have_posts() ) : ?>
                    <?php
                        /* Start the Loop */
                        while ( have_posts() ) : the_post();

                            /*
                             * Include the Post-Format-specific template for the content.
                             * If you want to override this in a child theme, then include a file
                             * called content-____.php (where ____ is the Post Format name) and that will be
                             used instead.
                             */
                            get_template_part( 'template-parts/post/content', get_post_format() );

                        endwhile;

                        the_posts_pagination( array(
                            'prev_text' => twentyseventeen_get_svg( array( 'icon' => 'arrow-left' ) ) .
                            '<span class="screen-reader-text">' . __( 'Previous page', 'twentyseventeen' ) . '</span>',
                            'next_text' => '<span class="screen-reader-text">' . __( 'Next page',
                            'twentyseventeen' ) . '</span>' . twentyseventeen_get_svg( array( 'icon' => 'arrow-right' ) ),
                            'before_page_number' => '<span class="meta-nav screen-reader-text">' . __(
                            'Page', 'twentyseventeen' ) . ' </span>',
                        ) );
```

```

else :

    get_template_part( 'template-parts/post/content', 'none' );

endif; ?>

</main><!-- #main -->
</div><!-- #primary -->
<?php get_sidebar(); ?>
</div><!-- .wrap -->

<?php get_footer();

```

Tipo de entrada personalizado Plantilla única:

Aquí hay un ejemplo de una sola plantilla. Tirado de los [veintisiete tema](#) .

```

<?php
/**
 * The template for displaying all single posts
 *
 * @link https://developer.wordpress.org/themes/basics/template-hierarchy/#single-post
 *
 * @package WordPress
 * @subpackage Twenty_Seventeen
 * @since 1.0
 * @version 1.0
 */

get_header(); ?>

<div class="wrap">
    <div id="primary" class="content-area">
        <main id="main" class="site-main" role="main">

            <?php
                /* Start the Loop */
                while ( have_posts() ) : the_post();

                    get_template_part( 'template-parts/post/content', get_post_format() );

                    // If comments are open or we have at least one comment, load up the
comment template.
                    if ( comments_open() || get_comments_number() ) :
                        comments_template();
                    endif;

                    the_post_navigation( array(
                        'prev_text' => '<span class="screen-reader-text">' . __( 'Previous
Post', 'twentyseventeen' ) . '</span><span aria-hidden="true" class="nav-subtitle">' . __(
'Previous', 'twentyseventeen' ) . '</span> <span class="nav-title"><span class="nav-title-
icon-wrapper">' . twentyseventeen_get_svg( array( 'icon' => 'arrow-left' ) ) .
'</span>%title</span>',
                        'next_text' => '<span class="screen-reader-text">' . __( 'Next Post',
'twentyseventeen' ) . '</span><span aria-hidden="true" class="nav-subtitle">' . __( 'Next',
'twentyseventeen' ) . '</span> <span class="nav-title">%title<span class="nav-title-ic
on-wrapper">' . twentyseventeen_get_svg( array( 'icon' => 'arrow-right' ) ) . '</span></span>',

```

```
        ) );

        endwhile; // End of the loop.
    ?>

    </main><!-- #main -->
</div><!-- #primary -->
<?php get_sidebar(); ?>
</div><!-- .wrap -->

<?php get_footer();
```

Ambos ejemplos de plantillas se están realizando en [parciales](#) para mostrar el contenido interno.

Si su tema hijo / padre tiene una plantilla única / de archivo, debe usar ese código como plantilla para sus nuevas plantillas.

Lea [Crear plantilla para tipo de publicación personalizada en línea](#):

<https://riptutorial.com/es/wordpress/topic/6390/crear-plantilla-para-tipo-de-publicacion-personalizada>

Capítulo 27: Crear una publicación programáticamente

Sintaxis

- `wp_insert_post (array $ args, bool $ wp_error);`

Parámetros

Parámetro	Descripción
<code>\$ args</code> (Array Required)	Una matriz de valores clave de los siguientes elementos.
<code>\$ wp_error</code> (booleano opcional)	Devuelve un error <code>WP_Error</code> en caso de fallo.

Observaciones

Argumentos

La siguiente tabla muestra una lista de elementos que puede usar dentro del primer parámetro (Array).

Parámetro	Descripción
CARNÉ DE IDENTIDAD	(Int) El ID de la publicación. Si es igual a algo distinto de 0, la publicación con esa ID se actualizará. Por defecto 0.
<code>post_autor</code>	(Int) El ID del usuario que agregó la publicación. El valor predeterminado es el ID de usuario actual.
<code>posfechar</code>	(Cadena) La fecha de la publicación. El valor predeterminado es la hora actual.
<code>post_date_gmt</code>	(Cadena) La fecha de la publicación en la zona horaria GMT. El valor predeterminado es el valor de <code>\$ post_date</code> .
Publicar Contenido	(Mixto) El contenido del post. Predeterminado vacío.
<code>post_content_filtered</code>	(Cadena) El contenido del post filtrado. Predeterminado vacío.
título de la entrada	(Cadena) El título del post. Predeterminado vacío.

Parámetro	Descripción
post_category	(Array) Array de valores de categoría de entrada.
post_excerpt (String)	El extracto del post. Predeterminado vacío.
post_status	(Cadena) El estado de la publicación. Proyecto por defecto.
tipo de mensaje	(Cadena) El tipo de publicación. Publicación por defecto.
comment_status	(Cadena) Si la publicación puede aceptar comentarios. Acepta abierto o cerrado. El valor predeterminado es el valor de la opción default_comment_status.
estado de ping	(Cadena) Si la publicación puede aceptar pings. Acepta abierto o cerrado. El valor predeterminado es el valor de la opción default_ping_status.
post_password	(Cadena) La contraseña para acceder a la publicación. Predeterminado vacío.
Nombre del puesto	(Cadena) El nombre de la publicación o slug. El valor predeterminado es el título de la publicación saneada al crear una nueva publicación.
to_ping	(Cadena) Espacio o lista de direcciones URL separadas por retorno de carro para hacer ping. Predeterminado vacío.
hizo ping	(Cadena) Lista de direcciones URL a las que se ha hecho ping. Predeterminado vacío.
post_modificado	(Cadena) La fecha en que se modificó por última vez la publicación. El valor predeterminado es la hora actual.
post_modified_gmt	(Cadena) La fecha en que la publicación se modificó por última vez en la zona horaria GMT. El valor predeterminado es la hora actual.
post_parente	(Int) Establezca esto para la publicación a la que pertenece, si corresponde. Por defecto 0.
menu_order	(Int) El orden en que se debe mostrar la publicación. Predeterminado 0.
post_mime_type	(Cadena) El tipo mime de la publicación. Predeterminado vacío.

Parámetro	Descripción
guid	(Cadena) ID única global para hacer referencia a la publicación. Predeterminado vacío.
tax_input	(Array) Array de términos de taxonomía marcados por su nombre de taxonomía. Predeterminado vacío.
meta_input	(Array) Array de valores de metadatos de entrada codificados por su clave de metadatos de publicación. Predeterminado vacío.

Evitar publicaciones duplicadas

Cuando ejecute esta función, probablemente podría obtener una publicación duplicada, al menos eso me pasó a mí. (Puedes comprobarlo en la sección [Publicar WordPress](#))

Encontré una [solución](#) :

```
if( !get_page_by_title( $title, 'OBJECT', 'post' ) ){
    $my_post = array('post_title' => $title,
        'post_content' => 'Content',
        'tags_input' => $tags,
        'post_category' => array(2),
        'post_status' => 'publish'
    );

    $result = wp_insert_post( $my_post );
}
```

Explicación

Antes de guardar una nueva publicación, valide si la nueva publicación ya existe usando el título de la publicación como un parámetro, si no hay un título de la publicación, puede guardar su nueva publicación.

Consulte la documentación de `get_page_by_title` [aquí](#) .

Examples

Introducción

A veces tenemos otro editor en otro lugar en lugar de TinyMCE (Wordpress Default Editor). Eso sucede cuando estamos creando nuestro propio tema, complemento o algo específico; y necesitamos escribir y manipular un tipo de publicación y guardarla en nuestra base de datos WP.

Entonces, si estás en esa situación, puedes usar una función de WordPress llamada:

```
wp_insert_post( array $args, bool $wp_error );
```

Crear una publicación básica

```
$basic_post_args = array(  
    'post_title' => 'My Basic Post',  
    'post_content' => 'This is a basic content',  
    'post_status' => 'publish',  
    'post_author' => 1,  
    'post_category' => array(8, 59)  
);  
  
wp_insert_post( $basic_post_args );
```

Crear una página básica

```
$basic_page_args = array(  
    'post_title' => 'My Basic Page',  
    'post_content' => 'This is a basic content',  
    'post_type' => 'page',  
    'post_status' => 'publish',  
    'post_author' => 1  
);  
  
wp_insert_post( $basic_page_args );
```

Lea [Crear una publicación programáticamente en línea:](https://riptutorial.com/es/wordpress/topic/5860/crear-una-publicacion-programaticamente)

<https://riptutorial.com/es/wordpress/topic/5860/crear-una-publicacion-programaticamente>

Capítulo 28: Depuración

Introducción

https://codex.wordpress.org/Debugging_in_WordPress

La depuración del código PHP es parte de cualquier proyecto, pero WordPress viene con sistemas de depuración específicos diseñados para simplificar el proceso, así como para estandarizar el código en el núcleo, los complementos y los temas.

Observaciones

Plugins para depuración en WordPress:

- <https://wordpress.org/plugins/query-monitor/>
- <https://wordpress.org/plugins/debug-bar/>
- <https://wordpress.org/plugins/debug-bar-console/>
- <https://wordpress.org/plugins/kint-debugger/>
- <https://wordpress.org/plugins/rest-api-console/>

Examples

WP_DEBUG

`WP_DEBUG` es una constante de PHP (una variable global permanente) que se puede usar para activar el modo "depurar" en WordPress. Se asume que es falso por defecto y generalmente se establece en verdadero en el archivo `wp-config.php` en las copias de desarrollo de WordPress.

```
define( 'WP_DEBUG', true );  
define( 'WP_DEBUG', false );
```

WP_DEBUG_LOG

`WP_DEBUG_LOG` es un complemento de `WP_DEBUG` que hace que todos los errores también se guarden en un archivo de registro `debug.log` dentro del directorio `/wp-content/`. Esto es útil si desea revisar todos los avisos más adelante o si necesita ver los avisos generados fuera de la pantalla (por ejemplo, durante una solicitud AJAX o la ejecución de `wp-cron`).

```
//enable  
define( 'WP_DEBUG_LOG', true );  
  
//disable  
define( 'WP_DEBUG_LOG', false );
```

WP_DEBUG_DISPLAY

`WP_DEBUG_DISPLAY` es otro compañero de `WP_DEBUG` que controla si los mensajes de depuración se muestran dentro del HTML de las páginas o no. El valor predeterminado es "verdadero", que muestra los errores y advertencias a medida que se generan. Establecer esto en falso ocultará todos los errores. Esto debe usarse junto con `WP_DEBUG_LOG` para que los errores puedan revisarse más adelante. Nota: para que `WP_DEBUG_DISPLAY` haga algo, `WP_DEBUG` debe estar habilitado (verdadero).

```
//enable
define( 'WP_DEBUG_DISPLAY', true );

//disable
define( 'WP_DEBUG_DISPLAY', false );
```

SCRIPT_DEBUG

`SCRIPT_DEBUG` es una constante relacionada que obligará a WordPress a usar las versiones "dev" de los archivos principales de CSS y JavaScript en lugar de las versiones mínimas que normalmente se cargan. Esto es útil cuando está probando modificaciones a cualquier archivo .js o .css integrado. El valor predeterminado es falso.

```
//enable
define( 'SCRIPT_DEBUG', true );

//disable
define( 'SCRIPT_DEBUG', false );
```

Guardias

La definición de `GUARDADO` guarda las consultas de la base de datos en una matriz y esa matriz se puede mostrar para ayudar a analizar esas consultas. La constante definida como verdadera hace que cada consulta se guarde, cuánto tiempo tardó en ejecutarse y qué función la llamó. NOTA: Esto tendrá un impacto en el rendimiento de su sitio, así que asegúrese de desactivarlo cuando no esté realizando la depuración.

```
define( 'SAVEQUERIES', true );
```

La matriz se almacena en el

```
global $wpdb->queries;
```

Ejemplo wp-config.php y buenas prácticas para la depuración

El siguiente código, insertado en su archivo `wp-config.php`, registrará todos los errores, avisos y advertencias en un archivo llamado `debug.log` en el directorio `wp-content`. También ocultará los errores para que no interrumpan la generación de páginas.

```
// Enable WP_DEBUG mode
define( 'WP_DEBUG', true );
```

```
// Enable Debug logging to the /wp-content/debug.log file
define( 'WP_DEBUG_LOG', true );

// Disable display of errors and warnings
define( 'WP_DEBUG_DISPLAY', false );
@ini_set( 'display_errors', 0 );

// Use dev versions of core JS and CSS files (only needed if you are modifying these core files)
define( 'SCRIPT_DEBUG', true );
```

Buena práctica Si desea agregar mensajes personalizados al registro de depuración, agregue el siguiente código en su complemento o tema.

```
//Checking is function_exists
if ( !function_exists( 'print_to_log' ) ) {
    //function writes a message to debug.log if debugging is turned on.
    function print_to_log( $message )
    {
        if ( true === WP_DEBUG ) {
            if ( is_array( $message ) || is_object( $message ) ) {
                error_log( print_r( $message, true ) );
            } else {
                error_log( $message );
            }
        }
    }
}
}
```

Ver registros en un archivo separado

Cuando tiene una llamada ajax, es extremadamente difícil obtener un registro desde dentro de la función de devolución de llamada. Pero si habilitas la depuración

```
define('WP_DEBUG', true);
```

y luego, después de eso, agrega

```
ini_set('log_errors',TRUE);
ini_set('error_reporting', E_ALL);
ini_set('error_log', dirname(__FILE__) . '/error_log.txt');
```

tendrá un archivo `error.log.txt` en su carpeta raíz donde se encuentran todos sus registros. Incluso puedes registrarlos con

```
error_log( print_r( 'what I want to check goes here', true ) );
```

dentro de tu código. Esto hará que tu vida sea mucho más fácil.

Lea **Depuración en línea**: <https://riptutorial.com/es/wordpress/topic/9170/depuracion>

Capítulo 29: Desarrollo de plugins

Sintaxis

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

Parámetros

Parámetro	Detalle
\$ etiqueta	(cadena) (Requerido) El nombre del filtro para enlazar la devolución de llamada \$ function_to_add a.
\$ function_to_add	(callable) (Requerido) La devolución de llamada se ejecuta cuando se aplica el filtro.
\$ prioridad	(int) (Opcional) Se utiliza para especificar el orden en que se ejecutan las funciones asociadas con una acción en particular. Los números más bajos se corresponden con la ejecución anterior y las funciones con la misma prioridad se ejecutan en el orden en que se agregaron a la acción. Valor predeterminado: 10
\$ accept_args	(int) (Opcional) El número de argumentos que acepta la función. Valor predeterminado: 1

Observaciones

La forma en que funcionan los enganches de los complementos es que, en varias ocasiones, mientras WordPress se está ejecutando, WordPress verifica si los complementos tienen funciones registradas para ejecutar en ese momento y, de ser así, se ejecutan las funciones. Estas funciones modifican el comportamiento predeterminado de WordPress.

Hay dos tipos de ganchos:

Los filtros le permiten cambiar el valor de un dato durante la ejecución de WordPress. Las funciones de devolución de llamada para los filtros se pasarán a través de una variable, se modificarán y luego se devolverán. Están diseñados para funcionar de manera aislada y nunca deben afectar las variables globales o cualquier otra cosa fuera de la función.

Las acciones, en contraste, le permiten agregar o cambiar el funcionamiento de WordPress. Su función de devolución de llamada se ejecutará en un punto específico en la ejecución de WordPress y puede realizar algún tipo de tarea, como hacer eco en el resultado para el usuario o

insertar algo en la base de datos.

[Referencia de filtro](#)

[Referencia de acción](#)

[Manual](#)

[API de plugin](#)

[Filtros vs Acciones](#)

Examples

Filtrar

```
add_filter('comment_text','before_comment');
add_filter('comment_text','after_comment');
function before_comment($comment_text){
    return 'input before comment'.$comment_text;
}
function after_comment($comment_text){
    return $comment_text.'input after comment';
}
```

Acción

```
add_action('wp_head','hook_javascript');

function hook_javascript() {
    $output="<script> alert('Page is loading...'); </script>";
    echo $output;
}
```

Ejemplos de desarrollo de plugins: Widget de canción favorita

```
<?php
function wpsnout_register_widgets() {
    register_widget( 'Favorite_Song_Widget' );
}

add_action( 'widgets_init', 'wpsnout_register_widgets' );

class Favorite_Song_Widget extends WP_Widget {

function Favorite_Song_Widget() {
    // Instantiate the parent object
    parent::__construct(
        'favorite_song_widget', // Base ID
        __('Favorite Song', 'text_domain'), // Name
        array( 'description' => __( 'Widget for playable favorite song', 'text_domain' ),
    ) // Args
    );
}
```

```

}

function widget( $args, $instance ) {
    echo $args['before_widget'];
    echo '<h3>Favorite Song Lists:</h3>';
    echo $instance['songinfo'];
    echo '<a href="" . $instance['link'] . "">Download it</a><br>';
    echo $instance['description'];
    echo $args['after_widget'];
}

function update($new_abc,$old_abc) {
    $instance = $old_abc;
    // Fields
    $instance['link'] = strip_tags($new_abc['link']);
    $instance['songinfo'] = strip_tags($new_abc['songinfo']);
    $instance['description'] = strip_tags($new_abc['description']);
    return $instance;
}

// Widget form creation
function form($instance) {
    $link = '';
    $songinfo = '';
    $description = '';
    // Check values
    if( $instance ) {
        $link = esc_attr($instance['link']);
        $songinfo = esc_textarea($instance['songinfo']);
        $description = esc_textarea($instance['description']);
    } ?>

    <p>
        <label for="<?php echo $this->get_field_id('link'); ?>"><?php _e('Link',
'wp_widget_plugin'); ?></label>
        <input class="widefat" id="<?php echo $this->get_field_id('link'); ?>" name="<?php
echo $this->get_field_name('link'); ?>" type="text" value="<?php echo $link; ?>" />
    </p>

    <p>
        <label for="<?php echo $this->get_field_id('songinfo'); ?>"><?php _e('Song Info:',
'wp_widget_plugin'); ?></label>
        <input class="widefat" id="<?php echo $this->get_field_id('songinfo'); ?>" name="<?php
echo $this->get_field_name('songinfo'); ?>" type="text" value="<?php echo $songinfo; ?>" />
    </p>

    <p>
        <label for="<?php echo $this->get_field_id('description'); ?>"><?php
_e('Description:', 'wp_widget_plugin'); ?></label>
        <textarea class="widefat" id="<?php echo $this->get_field_id('description'); ?>"
name="<?php echo $this->get_field_name('description'); ?>" type="text" value="<?php echo
$description; ?>"></textarea>
    </p>

    <p><a href="#" id="add-more-tabs"><?php _e('Add More Tabs', 'wp_widget_plugin');
?></a></p>

<?php }
}

```

Lea Desarrollo de plugins en línea: <https://riptutorial.com/es/wordpress/topic/6108/desarrollo-de-plugins>

Capítulo 30: Ejecutar WordPress local con XAMPP

Introducción

Con XAMPP puede instalar un servidor web en su PC local. Tiene un servidor web Apache, la base de datos MariaDB (MySQL) y funciona con Perl y PHP. Después de la instalación, puede ejecutar y depurar, por ejemplo, sistemas de administración de contenido como WordPress en su PC local. Puede usarlo con Windows, Linux, Mac OS X y Solaris.

Examples

1. Instalando XAMPP

1. Descarga el instalador para la [versión actual](#) de XAMPP
2. Vaya a través del asistente de instalación y no cambie nada si no está seguro de lo que está cambiando.
3. Después de la instalación, verá el panel de control de XAMPP. Simplemente haga clic en el botón de inicio de Apache y MySQL para ejecutar ambos con la configuración básica.
4. Si su computadora está conectada a una red local, y también desea acceder al servidor web desde otras computadoras, asegúrese de que su firewall lo permita.

2. Configurar una base de datos después de instalar XAMPP

Para ejecutar WordPress en su computadora, primero debe configurar una base de datos. Asegúrese de que Apache y MySQL se estén ejecutando (consulte [Instalación de XAMPP](#) paso 3)

1. Inicie un navegador web de su elección e ingrese "localhost" en la dirección.
2. Elija su idioma preferido y seleccione en la categoría "Herramientas" -> `phpMyAdmin` .
3. Seleccione la pestaña `Databases` .
4. Ingrese el nombre de su base de datos (necesitará este nombre más adelante) debajo de "Crear base de datos" y elija `utf8_general_ci` en el menú desplegable "Intercalación".
5. Haga clic en la base de datos creada en el lado izquierdo y seleccione la pestaña `Users` .
6. Agregue un nuevo usuario haciendo clic en `Add user` .
7. Ingrese su nombre de usuario, elija `local` en el menú desplegable "Host" (si desea acceder a la base de datos desde otras computadoras en la red, elija `Any host`) e ingrese su contraseña (también la necesitará más adelante).
8. Marque si debajo de "Base de datos para usuario" `Grant all privileges on wildcard name` está marcado
9. Presione `Go` .

3. Instalar WordPress después de configurar la base de datos

Después de instalar XAMPP y configurar la base de datos MySQL, puede instalar WordPress.

1. [Descarga](#) la última versión de WordPress.
2. Descomprima el archivo e inserte la carpeta en el directorio raíz de su servidor web (si utilizó la ruta sugerida durante la configuración de XAMPP, es "c: \xampp \htdocs").
3. Todos los archivos que existen dos veces pueden ser reemplazados.
4. Ingrese el campo de dirección de su navegador web "localhost / wordpress / wp-admin / install.php".
5. Después de elegir su idioma, haga clic en `Continue` y luego `Let's go`.
6. Reemplace todos los campos de texto predeterminados con la configuración elegida (consulte " [Configurar una base de datos](#) "). No cambie "Host de base de datos" y "Prefijo de tabla", excepto que desea migrar su instalación local de WordPress en línea. Entonces tenga en cuenta que la seguridad es importante. Encontrará más información en [Establecer prefijo en Nueva instalación de WordPress](#).
7. Presiona `Submit` y luego `Run the install`.
8. Ahora debe ingresar el nombre de su sitio, un nombre de usuario, contraseña y un correo electrónico válido.
9. Haga clic en `Install WordPress` para finalizar la configuración e iniciar sesión en su nuevo sitio local de WordPress.

Lea [Ejecutar WordPress local con XAMPP en línea](#):

<https://riptutorial.com/es/wordpress/topic/9551/ejecutar-wordpress-local-con-xampp>

Capítulo 31: El Loop (bucle principal de WordPress)

Examples

Estructura básica del bucle de WordPress

Cada vez que WordPress carga la página, se ejecutará el *bucle principal* .

El bucle es la forma de iterar sobre todos los elementos relacionados con la página en la que se encuentra actualmente.

El bucle principal funcionará en un objeto `WP_Query` global. La consulta tiene un método globalizado `have_posts()` , que nos permite recorrer todos los resultados. Finalmente, dentro del bucle puede llamar `the_post()` método `the_post()` (también como una función global), que establece el objeto de publicación global en la publicación actual dentro del bucle, y establece la postdata en la publicación actual. Gracias a esto, puede llamar a funciones como `the_title` , `the_content` , `the_author` (*etiquetas de plantilla*) directamente dentro del bucle.

Por ejemplo, si está en listas de publicaciones, el bucle principal contendrá un objeto de consulta con todas las publicaciones.

Si está en una sola publicación (o página), contendrá una consulta con una única publicación (página) en la que se encuentra actualmente.

```
if ( have_posts() ) :
    while ( have_posts() ) :
        the_post();
        var_dump( $post );
    endwhile;
endif;
```

Sintaxis de bucle alternativa

También puede utilizar bucle con llaves como esto:

```
if ( have_posts() ) {
    while ( have_posts() ) {

        the_post();
        var_dump( $post );

    }
}
```

Manejando ningunos artículos en el bucle

Si desea manejar este escenario, simplemente agregue una sentencia `if/else` .

```
if ( have_posts() ) : while ( have_posts() ) :  
  
    the_post();  
    var_dump( $post );  
  
endwhile; else :  
  
    __( 'This Query does not have any results' );  
  
endif;
```

Lea El Loop (bucle principal de WordPress) en línea:

<https://riptutorial.com/es/wordpress/topic/1803/el-loop--bucle-principal-de-wordpress->

Capítulo 32: El objeto \$wpdb

Observaciones

Hay dos formas de hacer referencia al objeto `$wpdb`. El primero es utilizar la palabra clave `global` PHP para actuar en la instancia global del objeto.

```
global $wpdb;
echo $wpdb->prefix;
// Outputs the prefix for the database
```

La segunda forma de usar el objeto `$wpdb` es hacer referencia a la `$GLOBALS` súper global `$GLOBALS` PHP.

```
echo $GLOBALS['wpdb']->prefix;
// This will also output the prefix for the database
```

Se desalienta la segunda forma, ya que puede no considerarse la mejor práctica.

Examples

Seleccionando una variable

En la forma más básica, es posible seleccionar una sola variable de una tabla llamando al método `get_var` del objeto pasando una consulta SQL.

```
global $wpdb;
$user = $wpdb->get_var( "SELECT ID FROM $wpdb->users WHERE user_email='foo@bar.com' " );
```

Es muy importante tener en cuenta que cualquier valor que no sea de confianza utilizado en las consultas *debe* escaparse para protegerse contra ataques. Esto se puede hacer usando el método de `prepare` del objeto.

```
global $wpdb;
$email = $_POST['email'];
$user = $wpdb->get_var(
    $wpdb->prepare( "SELECT ID FROM $wpdb->users WHERE user_email=%s", $email )
);
if( !is_null( $user ) ){
    echo $user;
} else {
    echo 'User not found';
}
```

Seleccionando multiples filas

Puede usar `get_results` para obtener varias filas de la base de datos.

```
global $wpdb;  
$userTable = $wpdb->prefix."users";  
  
$selectUser = $wpdb->get_results("SELECT * FROM $userTable");
```

Esto mostrará la lista de todos los usuarios en la matriz.

Lea El objeto \$wpdb en línea: <https://riptutorial.com/es/wordpress/topic/2691/el-objeto---wpdb>

Capítulo 33: el título()

Introducción

Esta función devuelve el título de la publicación actual.

Sintaxis

- `the_title` (\$ before, \$ after, \$ echo);

Parámetros

Parámetro	Detalles
\$ antes	(cadena) (opcional) Texto a colocar antes del título.
\$ después	(cadena) (opcional) Texto para colocar después del título.
\$ eco	(Booleano) (opcional) Muestra el título o lo devuelve para su uso en PHP

Observaciones

- Para el parámetro \$ echo, use true para mostrar el título y use false para devolverlo para su uso en PHP
- Tenga en cuenta que `the_title` solo se puede usar en bucles, si desea usarlo fuera de los bucles, use `get_the_title`

Examples

Uso simple de `the_title`

Código

```
the_title( );
```

Salida

El título de la publicación o página actual.

Imprimiendo el título con código antes y después.

Código

```
the_title( '<h1>', '</h1>' );
```

Salida

El título de la publicación o página actual en las etiquetas h1.

Lea el título() en línea: <https://riptutorial.com/es/wordpress/topic/9213/el-titulo-->

Capítulo 34: Eliminar saltos de línea automáticos del contenido y extracto

Introducción

Para los sitios que dependen del HTML a mano en el editor o en los extractos, los que desea codificar usted mismo, los saltos de línea automáticos pueden ser una molestia. Puede deshabilitarlos eliminando estos filtros.

Observaciones

Estos deben ser ejecutados directamente en un archivo de inclusión. Ya sea en functions.php o en otro archivo de inclusión, estos no se pueden envolver en un gancho. No funcionarán en init ni en ningún otro que haya encontrado hasta ahora.

También se pueden incluir directamente en una plantilla como page.php para ejecutar solo para esa plantilla.

NOTA: NO INCLUYA ESTO EN UN TEMA O PLUGIN DISTRIBUIDO (a menos que esté deshabilitado de forma predeterminada, como no incluir el archivo de inclusión en el que se encuentra a menos que el usuario lo especifique).

Esta es una mala práctica incluir en un sitio que no controla porque puede y romperá la salida de cualquier otro tema o complemento.

Examples

Quitar los filtros

```
// Remove the auto-paragraph and auto-line-break from the content
remove_filter( 'the_content', 'wpautop' );

// Remove the auto-paragraph and auto-line-break from the excerpt
remove_filter( 'the_excerpt', 'wpautop' );
```

Función para eliminar los filtros.

```
/**
 * Remove the automatic line breaks from content and excerpts.
 *
 * @since 1.0.0
 */
function remove_content_auto_line_breaks() {
    // Remove the auto-paragraph and auto-line-break from the content
    remove_filter( 'the_content', 'wpautop' );
```

```
// Remove the auto-paragraph and auto-line-break from the excerpt
remove_filter( 'the_excerpt', 'wpautop' );
}

// Execute the function
remove_content_auto_line_breaks();
```

Lea [Eliminar saltos de línea automáticos del contenido y extracto en línea](https://riptutorial.com/es/wordpress/topic/9614/eliminar-saltos-de-linea-automaticos-del-contenido-y-extracto):

<https://riptutorial.com/es/wordpress/topic/9614/eliminar-saltos-de-linea-automaticos-del-contenido-y-extracto>

Capítulo 35: Eliminar versión de Wordpress y hojas de estilo

Introducción

Para hacer que sea más difícil para otros piratear su sitio web, puede eliminar el número de versión de WordPress de su sitio, su css y js. Sin ese número, no es posible ver si ejecuta la versión actual para explotar los errores de las versiones anteriores.

Además, puede mejorar la velocidad de carga de su sitio, ya que sin cadenas de consulta en la URL, los archivos css y js se pueden almacenar en caché.

Sintaxis

- `add_filter` (etiqueta \$, \$ `function_to_add`, \$ prioridad, \$ aceptado)

Parámetros

Parámetro	Detalles
\$ etiqueta	(cadena requerida) Nombre del filtro donde se enlaza \$ <code>function_to_add</code> a
\$ <code>function_to_add</code>	(se puede llamar) Nombre de la función que se ejecuta cuando se aplica el filtro
\$ prioridad	(int opcional) lugar de \$ <code>function_to_add</code> entre otras funciones en una acción (predeterminado = 10)
\$ <code>accept_args</code>	(int opcional) Número de parámetros que acepta \$ <code>function_to_add</code> (predeterminado = 1)

Observaciones

Destinado a mejorar la velocidad y seguridad del sitio.

Examples

Eliminar números de versión de css / js

Solo agrega esta función a tus funciones.php. Se eliminará la versión de todos los archivos js y css en cola.

```
function remove_cssjs_ver( $src ) {
    if( strpos( $src, '?ver=' ) )
        $src = remove_query_arg( 'ver', $src );
    return $src;
}

add_filter( 'style_loader_src', 'remove_cssjs_ver', 999 );
add_filter( 'script_loader_src', 'remove_cssjs_ver', 999 );
```

Eliminar números de versión de WordPress

Si agrega esto a su functions.php, elimina el número de versión de WordPress de la fuente RSS y el encabezado.

```
function remove_wordpress_ver() {
    return '';
}

add_filter('the_generator', 'remove_wordpress_ver', 999);
```

Lea [Eliminar versión de Wordpress y hojas de estilo en línea](https://riptutorial.com/es/wordpress/topic/6218/eliminar-version-de-wordpress-y-hojas-de-estilo):

<https://riptutorial.com/es/wordpress/topic/6218/eliminar-version-de-wordpress-y-hojas-de-estilo>

Capítulo 36: en eso

Sintaxis

1. `add_action('init', callable $ function)`

Observaciones

`init` es un gancho de acción que se activa después de que WordPress haya terminado de cargarse pero antes de que se envíen los encabezados HTTP.

Examples

Procesando `$_POST` datos de solicitud

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

```
function process_post_data() {  
    if( isset( $_POST ) ) {  
        // process $_POST data here  
    }  
}
```

Procesando `$_GET` datos de solicitud

```
add_action('init', 'process_get_data');  
  
function process_get_data() {  
    if( isset( $_GET ) ) {  
        // process $_GET data here  
    }  
}
```

Registro de un tipo de mensaje personalizado

```
add_action( 'init', function() {  
    register_post_type( 'event', array(  
        'public' => true,  
        'label'  => 'Events'  
    ) );  
});
```

Registra un nuevo tipo de publicación personalizada con una etiqueta `Events` y un `event` slug

Lea en eso en línea: <https://riptutorial.com/es/wordpress/topic/6375/en-eso>

Capítulo 37: Encolando guiones

Sintaxis

- `wp_enqueue_script` (`$ handle`, `$ src`, `$ deps`, `$ ver`, `$ in_footer`)

Parámetros

Parámetro	Detalles
<code>\$ manejar</code>	<i>(cadena)</i> (Requerido) Nombre de la secuencia de comandos. Debería ser único.
<code>\$ src</code>	<i>(cadena)</i> (Opcional) URL completa de la secuencia de comandos o ruta de la secuencia de comandos relativa al directorio raíz de WordPress. <i>Valor por defecto: falso</i>
<code>\$ deps</code>	<i>(matriz)</i> (Opcional) Una matriz de secuencia de comandos registrada que maneja esta secuencia de comandos depende de. <i>Valor por defecto: array ()</i>
<code>\$ ver</code>	<i>(string bool null)</i> (Opcional) Cadena que especifica el número de versión de la secuencia de comandos, si tiene uno, que se agrega a la URL como una cadena de consulta para propósitos de almacenamiento en memoria caché. Si la versión se establece en falso, un número de versión se agrega automáticamente igual a la versión de WordPress instalada actual. Si se establece en nulo, no se agrega ninguna versión. <i>Valor por defecto: falso</i>
<code>\$ in_footer</code>	<i>(bool)</i> (Opcional) Si se pone en cola el script antes <code></body></code> lugar de en <code><head></code> . <i>Valor por defecto: falso</i>

Examples

Poner en cola los scripts en functions.php

Si desea agregar el script `custom.js` que se encuentra en la carpeta `js/` de su tema, deberá ponerlo en cola. En `functions.php` añadir

```
<?php

add_action( 'after_setup_theme', 'yourtheme_theme_setup' );

if ( ! function_exists( 'yourtheme_theme_setup' ) ) {
    function yourtheme_theme_setup() {

        add_action( 'wp_enqueue_scripts', 'yourtheme_scripts' );
        add_action( 'admin_enqueue_scripts', 'yourtheme_admin_scripts' );
    }
}
```

```

    }
}

if ( ! function_exists( 'yourtheme_scripts' ) ) {
    function yourtheme_scripts() {

        wp_enqueue_script( 'yourtheme_custom', get_template_directory_uri().'/js/custom.js',
array( 'jquery' ), '1.0.0', true );

    }
}

if ( ! function_exists( 'yourtheme_admin_scripts' ) ) {
    function yourtheme_admin_scripts() {

        wp_enqueue_script( 'yourtheme_custom', get_template_directory_uri().'/js/custom.js',
array( 'jquery-ui-autocomplete', 'jquery' ), '1.0.0', true );

    }
}
}

```

Encolar scripts solo para IE

```

add_action( 'wp_enqueue_scripts', 'enqueue_my_styles_and_scripts' );

/**
 * Enqueue scripts (or styles) conditionally.
 *
 * Load scripts (or stylesheets) specifically for IE. IE10 and above does
 * not support conditional comments in standards mode.
 *
 * @link https://gist.github.com/wpscholar/4947518
 * @link https://msdn.microsoft.com/en-us/library/ms537512(v=vs.85).aspx
 */
function enqueue_my_styles_and_scripts() {

    // Internet Explorer HTML5 support
    wp_enqueue_script( 'html5shiv', get_template_directory_uri().'/js/html5shiv.js', array(),
'3.7.3', false);
    wp_script_add_data( 'html5shiv', 'conditional', 'lt IE 9' );

    // Internet Explorer 8 media query support
    wp_enqueue_script( 'respond', get_template_directory_uri().'/js/respond.js', array(),
'1.4.2', false);
    wp_script_add_data( 'respond', 'conditional', 'lt IE 9' );

}

```

Encolando scripts condicionalmente para páginas específicas

Puede usar operadores condicionales en WordPress para poner en cola scripts en páginas específicas de su sitio web.

```

function load_script_for_single_post() {
    if(is_single()){
        wp_enqueue_script(

```

```
        'some',
        get_template_directory_uri().'/js/some.js',
        array('jquery'),
        '1.0.0',
        false
    );
}
}
add_action('wp_enqueue_scripts','load_script_for_single_post');
```

En el ejemplo anterior, si la página web actual es una publicación única, la secuencia de comandos se pondrá en cola. De lo contrario, la función *wp_enqueue_script* no se ejecutará.

Lea Encolando guiones en línea: <https://riptutorial.com/es/wordpress/topic/1103/encolando-guiones>

Capítulo 38: Estilos de puesta en cola

Sintaxis

1. `wp_enqueue_style` (\$ handle, \$ src, \$ dependency, \$ version, \$ media);

Parámetros

Parámetro	Detalles
\$handle	(Cadena) (Requerido) Nombre único para la hoja de estilo.
\$src	(Cadena) (Opcional) URL de la hoja de estilo que se usará dentro del atributo src de la etiqueta del enlace .
\$deps	(Cadena) (Opcional) Una matriz de hojas de estilo de las que depende esta hoja de estilos.
\$ver	(Cadena) (Opcional) Cadena que especifica la versión de la hoja de estilo de la hoja de estilo.
\$media	(Cadena) (Opcional) El medio para el que se crea esta hoja de estilo. por ejemplo, 'todo', 'imprimir', 'pantalla', etc.

Examples

Incluyendo archivo css interno con otro archivo css como una dependencia

```
function themeSlug_enqueue_scripts() {
    wp_enqueue_style( 'themeSlug-reset', get_template_directory_uri() .'/css/reset.css',
    '1.0.0' );
    wp_enqueue_style( 'themeSlug-style', get_template_directory_uri() .'/style.css',
    'themeSlug-reset', '1.0.0');
}
add_action('wp_enqueue_scripts', 'themeSlug_enqueue_scripts');
```

Incluyendo archivo css interno

En este caso, `style.css` se encuentra en la raíz de la carpeta del tema

```
function themeSlug_enqueue_scripts() {
    wp_enqueue_style( 'themeSlug-style', get_template_directory_uri() .'/style.css', '1.0.0');
}
add_action('wp_enqueue_scripts', 'themeSlug_enqueue_scripts');
```

Incluyendo archivo css externo

En este ejemplo queremos incluir la fuente de icono impresionante

```
function themeSlug_enqueue_scripts() {
    wp_enqueue_style( 'font-awesome', '//cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.6.3/css/font-awesome.css');
}
add_action('wp_enqueue_scripts', 'themeSlug_enqueue_scripts');
```

Encolar hojas de estilo solo para IE

```
add_action( 'wp_enqueue_scripts', 'enqueue_my_styles_and_scripts' );

/**
 * Enqueue styles (or scripts) conditionally.
 *
 * Load stylesheets (or scripts) specifically for IE. IE10 and above does
 * not support conditional comments in standards mode.
 *
 * @link https://gist.github.com/wpscholar/4947518
 * @link https://msdn.microsoft.com/en-us/library/ms537512(v=vs.85).aspx
 */
function enqueue_my_styles_and_scripts() {

    // Internet Explorer specific stylesheet.
    wp_enqueue_style( 'themename-ie', get_stylesheet_directory_uri() . '/css/ie.css', array(
'twentyfifteen-style' ), '20141010' );
    wp_style_add_data( 'themename-ie', 'conditional', 'lte IE 9' );

    // Internet Explorer 7 specific stylesheet.
    wp_enqueue_style( 'themename-ie7', get_stylesheet_directory_uri() . '/css/ie7.css', array(
'twentyfifteen-style' ), '20141010' );
    wp_style_add_data( 'themename-ie7', 'conditional', 'lt IE 8' );

}
```

Incluyendo archivo css interno para su clase de Plugin

```
class My_Plugin() {
    function __construct() {
        add_action( 'wp_enqueue_scripts', array( $this, 'init_fe_assets' ) );
    }

    public function init_fe_assests() {
        wp_enqueue_style( 'my-plugin', plugin_dir_url( __FILE__ ) .
'assets/css/frontend/plugin.css', array(), '0.0.1', true );
    }
}

new My_Plugin();
```

Añadir hojas de estilo alternativas


```
<?php wp_enqueue_style('theme-five', get_template_directory_uri() .  
'/path/to/additional/css');  
wp_style_add_data('theme-five', 'alt', true);  
wp_style_add_data('theme-five', 'title', __('theme-five.css', 'your-theme-name')); ?>
```

[wp_style_add_data](#)

Lea Estilos de puesta en cola en línea: <https://riptutorial.com/es/wordpress/topic/1247/estilos-de-puesta-en-cola>

Capítulo 39: Extractos personalizados con `excerpt_length` y `excerpt_more`

Examples

Limitar la longitud del extracto a 50 palabras

Ponga el siguiente código en **functions.php** :

```
function themify_custom_excerpt_length( $length ) {  
    return 50;  
}  
add_filter( 'excerpt_length', 'themify_custom_excerpt_length', 999 );
```

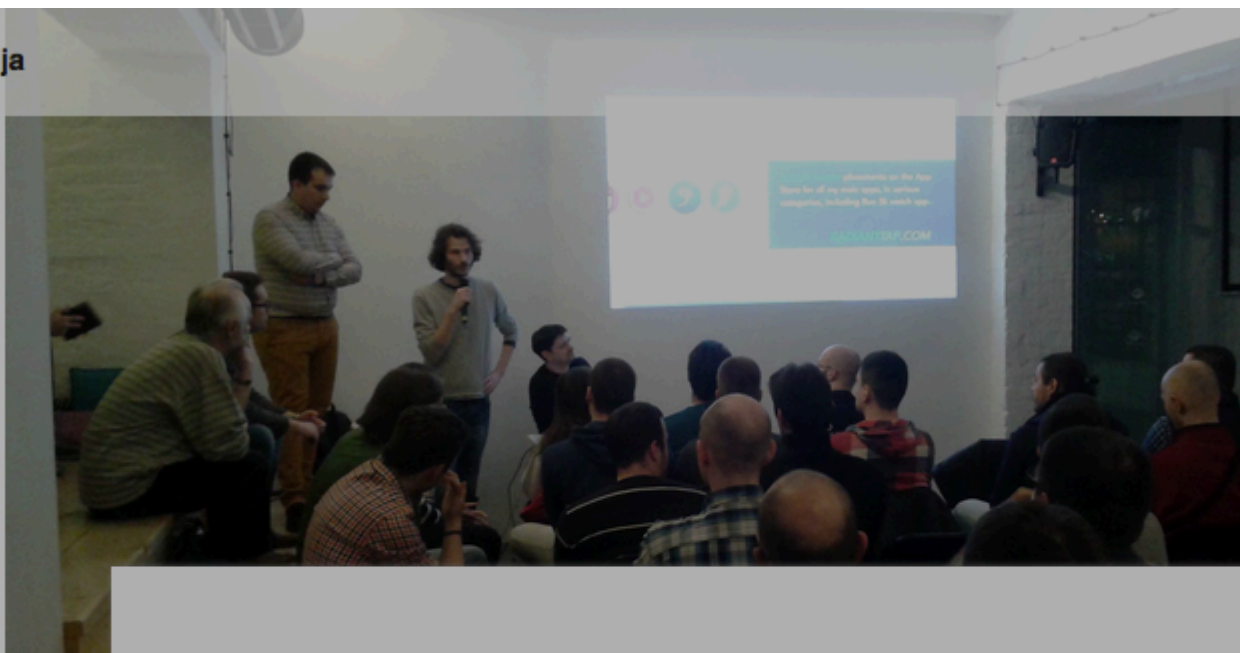
Use 999 como prioridad para asegurarse de que la función se ejecute después del filtro de WordPress predeterminado, de lo contrario anularía lo que se establece aquí.

Adición de un enlace Leer más al final del extracto

Para hacer esto, ponga el siguiente código en **functions.php** :

```
function custom_excerpt_more($more) {  
    return '<a href="'. get_permalink($post->ID) . '">Read More</a>';  
}  
add_filter('excerpt_more', 'custom_excerpt_more');
```

Los resultados deberían verse así:



Utisci sa prvog iOS meetupa

Mar 4, 2016

Utisci sa prvog iOS meetupa su fantastični, što bi u našem narodu rekli-prvo pa muško! Ovo okupljanje dokazalo je da iOS zajednica u Srbiji i te kako postoji i da uopšte nije mala kao što se mislilo. Na događaju je bilo nešto više od 100 ljudi, a predavanja su bila izuzetno zanimljiva i korisna. Organizatori [→ Read More](#)

Ostavi komentar



Agregando algunos puntos al final del extracto

En nuestras **funciones**.

```
function new_excerpt_more( $more ) {  
    return '.....';  
}  
add_filter('excerpt_more', 'new_excerpt_more');
```

Debemos obtener esto:



Utisci sa prvog iOS meetupa

Mar 4, 2016

Utisci sa prvog iOS meetupa su fantastični, što bi u našem narodu rekli-prvo pa muško! Ovo okupljanje dokazalo je da iOS zajednica u Srbiji i te kako postoji i da uopšte nije mala kao što se mislilo. Na događaju je bilo nešto više od 100 ljudi, a predavanja su bila izuzetno zanimljiva i korisna. Organizatori.....

Ostavi komentar



Lea Extractos personalizados con `excerpt_length` y `excerpt_more` en línea:

<https://riptutorial.com/es/wordpress/topic/6104/extractos-personalizados-con-excerpt-length-y-excerpt-more>

Capítulo 40: Función: add_action ()

Sintaxis

- add_action (\$ tag, \$ function_to_add)
- add_action (\$ tag, \$ function_to_add, \$ prioridad)
- add_action (\$ tag, \$ function_to_add, \$ prioridad, \$ aceptado en args)

Parámetros

Parámetro	Detalles
\$ etiqueta	(cadena) (Requerido) El nombre de la acción a la que se engancha \$function_to_add
\$ function_to_add	(callable) (Requerido) La función que se debe llamar cuando se ejecuta la acción indicada por \$tag
\$ prioridad	(int) (Opcional) Valor predeterminado: 10 Se utiliza para especificar el orden en que se ejecutan las funciones asociadas con una acción en particular. Los números más bajos se corresponden con la ejecución anterior y las funciones con la misma prioridad se ejecutan en el orden en que se agregaron a la acción.
\$ accept_args	(int) (Opcional) Valor predeterminado: 1 El número de argumentos que acepta la función.

Observaciones

La función `add_action()` crea un **Gancho de Acción** , asociando una función de PHP con una "etiqueta" o nombre de acción particular. Cuando la acción es "activada" por una llamada a `do_action()` (o `do_action_ref_array()`) con una etiqueta específica, todas las funciones "enganchadas" a esa etiqueta se ejecutarán.

En la mayoría de los casos, esta función debe usarse en el archivo `functions.php` un tema o en un archivo de complemento, u otro archivo de origen cargado por cualquiera.

Esta función es parte de la **API del complemento**

Examples

Gancho de accion basico

La aplicación más básica de `add_action()` es agregar código personalizado para que se ejecute en

una ubicación determinada en el código fuente de una instalación de WordPress, ya sea mediante acciones proporcionadas por el **Core** de WordPress o creadas por códigos de terceros, como complementos y temas

Para agregar contenido a la sección `<head></head>` del sitio, digamos a un elemento meta `<link>` para indicar dónde se puede encontrar la información de copyright del sitio - se puede usar `add_action()` para adjuntar una función que imprime el marcado apropiado para la acción `'wp_head'` (que se "dispara" cuando WordPress construye la sección `<head>`):

```
function add_copyright_meta_link() {
    echo( '<link rel="copyright" href="' . get_home_url() . '/copyright">' );
}

add_action( 'wp_head', 'add_copyright_meta_link' );
```

Prioridad de gancho de acción

Cualquier número de funciones puede estar "enganchado" a cualquier acción dada. En algunos casos, es importante que una función enganchada se ejecute antes o después de otras, que es donde el tercer parámetro de `add_action()`, `$priority` entra en juego.

Si se omite el argumento `$priority`, la función se adjuntará con la prioridad predeterminada de `10`. Cuando la acción se "desencadena", las funciones "enganchadas" se llamarán comenzando con las agregadas con la menor `$priority`, y avanzando a las funciones con la mayor `$priority`. Cualquier función enganchada que comparta la misma prioridad se llamará en el orden en que se agregaron (el orden en que se ejecutaron sus respectivas llamadas `add_action()`).

Por ejemplo, digamos que un complemento de terceros está utilizando una función enlazada a la acción `'template_redirect'` para reenviar a los visitantes a la página de `daily-deal` a un enlace de afiliado para un sitio de comercio electrónico externo, pero le gustaría la redirección solo para usuarios registrados. Necesitaría usar su propio `'template_redirect'` para enviar a los visitantes que han cerrado la sesión a la página de inicio de sesión. Después de determinar que el complemento de terceros adjunta su función con el valor predeterminado de `$priority` de `10`, puede conectar su función con una prioridad de `9` para asegurarse de que su comprobación de inicio de sesión se `$priority` primero:

```
function redirect_deal_visitors_to_login() {
    if( is_page( 'daily-deal' ) && !user_is_logged_in() ) {
        wp_redirect( wp_login_url() );
        exit();
    }
}

add_action( 'template_redirect', 'redirect_deal_visitors_to_login', 9 );
```

Enganchar métodos de clase y objeto a acciones

Las clases de PHP son una herramienta poderosa para mejorar la organización del código y minimizar las colisiones de nombres. En algún momento u otro, surge inevitablemente la cuestión

de cómo crear un gancho de acción para un método de clase.

El argumento `$function_to_add` menudo se muestra como una cadena que contiene el nombre de la función, sin embargo, el tipo de datos del argumento es en realidad " **callable** ", que para nuestros propósitos puede resumirse como "una referencia a una función o método".

Hay varios formatos que se pueden llamar que se pueden usar para hacer referencia a métodos en clases y objetos. Sin embargo, en todos los casos, el método al *que* se hace referencia *debe* ser **visible** públicamente. Un método es público cuando está prefijado con la palabra clave `public` o no tiene ninguna palabra clave de visibilidad (en cuyo caso, el método se establece de forma predeterminada como público).

Método del objeto Acción ganchos

Los métodos de objeto se ejecutan en una instancia particular de una clase.

```
class My_Class {
    // Constructor
    function My_Class() {
        // (Instantiation logic)
    }

    // Initialization function
    public function initialize() {
        // (Initialization logic)
    }
}
```

Después de instanciar la clase anterior de la siguiente manera,

```
$my_class_instance = new My_Class();
```

el método `initialize()` normalmente se invoca en el objeto llamando a `$my_class_instance->initialize()`; . Enganchar el método a la acción 'init' WordPress se hace pasando una matriz que contiene una referencia a la instancia y una cadena que contiene el nombre del método del objeto:

```
add_action( 'init', [ $my_class_instance, 'initialize' ] );
```

Si se llama a `add_action()` dentro de un método de objeto, también se puede usar la pseudo-variable `$this` :

```
class My_Class {
    // Constructor
    function My_Class() {
        // (Instantiation logic)
        add_action( 'init', [ $this, 'initialize' ] );
    }

    // Initialization function
```

```
public function initialize() {
    // (Initialization logic)
}
}
```

Método de clase de ganchos de acción

Los métodos de clase se ejecutan de forma estática en una clase en lugar de cualquier instancia particular. Dada la siguiente clase,

```
class My_Class {
    // Initialization function
    public static function initialize() {
        // (Initialization logic)
    }
}
```

el método `initialize()` normalmente se invocará utilizando el operador de `::` alcance-resolución, es decir, `My_Class::initialize()`; . Enganchar un método de clase estático a un WordPress se puede hacer de dos maneras diferentes:

- Usando una matriz compuesta de una cadena que contiene el nombre de la clase y una cadena que contiene el nombre del método:

```
add_action( 'init', [ 'My_Class', 'initialize' ] );
```

- Pasando una cadena que contiene una referencia completa al método, incluido el operador `::`:

```
add_action( 'init', 'My_Class::initialize' );
```

- Si se llama a `add_action()` dentro de un método de clase estático, la palabra clave `self` o `__CLASS__` magic-constant se pueden usar en lugar del nombre de la clase. Tenga en cuenta que esto generalmente no es aconsejable, ya que los valores de estos elementos se vuelven contraintuitivos en el caso de la herencia de clase.

```
class My_Class {
    // Setup function
    public static function setup_actions() {
        add_action( 'init', 'self::initialize' );
    }

    // Initialization function
    public static function initialize() {
        // (Initialization logic)
    }
}
```

Lea Función: `add_action()` en línea: <https://riptutorial.com/es/wordpress/topic/6561/funcion--add->

action---

Capítulo 41: Función: wp_trim_words ()

Sintaxis

- `<?php $trimmed_text = wp_trim_words($text, $num_words = 55, $more = null); ?>`

Parámetros

Parámetro	Detalles
<code>\$text</code>	(Cadena) (Requerido) Texto que se acortará o recortará.
<code>\$num_words</code>	(Entero) (Requerido) Número de palabras a las que se restringirá el texto.
<code>\$more</code>	(Cadena) (Opcional) Qué anexar si se debe recortar \$ text.

Examples

Recortar contenido de la publicación

Esta función acorta el texto a un número específico de palabras y devuelve el texto abreviado.

```
<?php echo wp_trim_words( get_the_content(), 40, '...' ); ?>
```

En el ejemplo anterior estamos pasando el contenido de la publicación a la función. Restringirá la longitud del contenido a 40 palabras y recortará el resto de las palabras.

Lea Función: wp_trim_words () en línea: <https://riptutorial.com/es/wordpress/topic/7866/funcion--wp-trim-words--->

Capítulo 42: Fundamentos del tema infantil

Sintaxis

- **template** — es el nombre del tema principal de WordPress, el principal.
- **child-theme** — es el paquete que anula la **plantilla** .

Observaciones

He estado anunciando que el uso de un tema infantil siempre es bueno, pero siempre hay un *Pero ...*

En nuestro ejemplo de sobrescritura de plantillas, imaginemos que el autor de un tema está agregando sus propias mejoras a la plantilla de la barra lateral y habrá una nueva en

`/themes/template/sidebar.php`

```
<?php
/**
 * The template for the sidebar containing the main widget area
 *
 * @link https://developer.wordpress.org/themes/basics/template-files/#template-partials
 */

if ( is_active_sidebar( 'sidebar-1' ) ) : ?>
    <aside id="secondary" class="sidebar widget-area" role="complementary">
        <?php dynamic_sidebar( 'sidebar-1' ); ?>
    </aside><!-- .sidebar .widget-area -->
<?php endif; ?>
```

Ahora nuestro sitio web no se beneficiará de la nueva especificación de `role="complementary"` porque nuestro tema secundario aún está sobrescribiendo la **plantilla** con su propio archivo en `/themes/child-theme/sidebar.php`

Es nuestro deber como mantenedores de sitios web hacer un seguimiento de las plantillas que sobrescribimos y, en el caso inminente de una actualización, observar atentamente el registro de cambios para que pueda actualizar los archivos de temas secundarios si es necesario.

Examples

2) El propósito

Los temas secundarios están diseñados para ser una forma segura de mantener las personalizaciones de la plantilla principal sin temor a perderlas en una actualización del tema.

Básicamente, cada vez que desee editar un archivo dentro de la plantilla activa de su sitio web, debe preguntarse " **¿Qué sucederá cuando actualizaré el tema?**

Y la respuesta es simple: los **pierdes porque la actualización traerá una carpeta de tema completamente nueva** .

Así que veamos un tema secundario como una carpeta con archivos que sobrescribirá los archivos con la misma ruta en el tema principal. Ahora vamos a traer algunos ejemplos reales:

Definición y requisitos.

WordPress identifica un tema secundario cuando hay un directorio (por ejemplo, `child-theme`) dentro de `/wp-content/themes/` con los siguientes archivos:

- `style.css`

Este archivo debe comenzar con una plantilla de comentarios como esta:

```
/*
Theme Name: Example Child
Author: Me
Author URI: https://example.com/
Template: example
Text Domain: example-child-theme
Domain Path: /languages/
*/
```

Las cosas más importantes a considerar aquí son:

- El nombre de la `Template` debe ser exactamente el nombre de la carpeta que contiene el tema principal (también conocido como la babosa del tema principal)
- Asigne un nombre al tema de su hijo de tal manera que pueda identificarlo fácilmente en el Tablero de instrumentos (normalmente solo agregue `Child` al nombre del padre, como `Example Child`)

- `index.php`
- `functions.php`

3) sobrescritura de plantillas

El uso más común de un tema secundario es anular partes de la plantilla. Por ejemplo, una barra lateral, si tenemos un tema con el siguiente archivo en

`/themes/template/sidebar.php`

```
<?php
/**
 * The sidebar containing the main widget area.
 *
 * @link https://developer.wordpress.org/themes/basics/template-files/#template-partials
 */

if ( ! is_active_sidebar( 'sidebar-1' ) ) {
    return;
}
```

```
}>
<div id="sidebar" class="widget-area">
    <?php dynamic_sidebar( 'sidebar-1' ); ?>
</div>
```

Definitivamente podemos agregar nuestro propio archivo en tema infantil (con las especificaciones del primer ejemplo) con el siguiente archivo

/themes/child-theme/sidebar.php

```
<?php
/**
 * The sidebar containing the main widget area.
 */

if ( ! is_active_sidebar( 'sidebar-1' ) ) {
    return;
}
>
<div id="my-sidebar" class="my-own-awesome-class widget-area">
    <div class="my-wrapper">
        <?php dynamic_sidebar( 'sidebar-1' ); ?>
    </div>
</div>
```

Ahora `my-own-awesome-class` está a salvo en el tema infantil y no se eliminará en ninguna actualización del tema y WordPress siempre elegirá una plantilla de los temas secundarios cuando encuentre una en la misma ruta.

Reemplazo de activos

Incluso si no es una buena práctica, a veces necesita reemplazar activos como archivos o bibliotecas CSS o JS.

Tenga en cuenta que el sistema de sobrescritura de plantillas de WordPress no funciona con nada más que los archivos `.php`, por lo que cuando hablamos de activos nos referimos a [activos registrados](#).

Un ejemplo podría ser el reemplazo de la biblioteca jQuery con su versión deseada. En nuestro archivo de `functions.php` temas secundarios `functions.php` debemos agregar una función que elimine la versión actual de jQuery y agregar la nuestra desde CDN (recuerde que es solo un ejemplo).

```
/**
 * Dequeue the jQuery script and add our own version.
 *
 * Hooked to the wp_print_scripts action, with a late priority (100),
 * so that it is after the script was enqueued.
 */
function my_own_theme_scripts() {
    // remove the current version
    wp_dequeue_script( 'jquery' );
    // register my desired version
    wp_register_script( 'jquery', 'https://code.jquery.com/jquery-3.1.0.min.js', false,
    '3.1.0' );
}
```

```
// load my version, here or somewhere else
wp_enqueue_script( 'jquery' );
}
add_action( 'wp_print_scripts', 'my_own_theme_scripts' );
```

Lea Fundamentos del tema infantil en línea:

<https://riptutorial.com/es/wordpress/topic/6238/fundamentos-del-tema-infantil>

Capítulo 43: get_bloginfo ()

Introducción

Recupera información sobre el sitio actual.

Sintaxis

- `get_bloginfo ($ show, $ filtro)`

Parámetros

Parámetro	Detalles
<code>\$ show</code>	(cadena) La información de configuración del sitio para recuperar.
<code>\$ filtro</code>	(cadena) La especificación sobre si devolver un valor filtrado o no.

Observaciones

\$ show

Valores	Descripción	Ejemplo
'nombre' (predeterminado)	Título del sitio	'Matt Mullenweg'
'descripción'	Lema del sitio	'Just another WordPress site'
'wpurl'	URL de la instalación de WordPress. Igual que la función <code>site_url()</code>	'http://example.com' , 'http://localhost/wordpress'
'url'	URL del sitio. Igual que la función <code>home_url()</code>	'http://example.com' , 'http://localhost/wordpress'
'admin_email'	Dirección de correo electrónico del administrador principal	'matt@mullenweg.com'
charset	Codificación de caracteres de las páginas y feeds.	'UTF-8'

Valores	Descripción	Ejemplo
'versión'	Versión actual de la instalación de WordPress.	'4.5'
'html_type'	valor de tipo de contenido del HTML	'text/html'
'dirección del texto'	Dirección del texto determinada por el idioma del sitio.	'ltr'
'idioma'	Código de idioma basado en ISO 639-1	'en-US'
'stylesheet_url'	URL de la hoja de estilo del tema activado. Prioridad de valor: Tema infantil » Tema principal.	'http://example.com/wp-content/themes/twenty-sixteen/style.css'
'stylesheet_directory'	Localización de recursos del tema activado. Prioridad de valor: Tema infantil » Tema principal.	'http://example.com/wp-content/themes/twenty-sixteen'
'template_url'	Directorio de URL del tema activado. Prioridad de valor: tema de los padres » Tema de los niños.	'http://example.com/wp-content/themes/twenty-sixteen'
'plantilla_directorio'	Igual que 'template_url'	
'pingback_url'	Archivo XML-RPC de Pingback	'http://example/xmlrpc.php'
'atom_url'	URL del feed Atom	'http://example/feed/atom/'
'rdf_url'	URL del feed RDF / RSS 1.0	'http://example/feed/rdf/'
'rss_url'	RSS de la alimentación de 0.92 URL	'http://example/feed/rss/'
'rss2_url'	URL de feed RSS 2.0	'http://example/feed/'
'comments_atom_url'	Comentarios Atom feed	'http://example/comments/feed/atom/'

Valores	Descripción	Ejemplo
	URL	
'Sitio URL'	(<i>en desuso</i>) Use 'url' en su lugar	
'casa'	(<i>en desuso</i>) Use 'url' en su lugar	

\$ filtro

Valores	Descripción	Ejemplo
'raw' (predeterminado)	No se aplicarán filtros.	<i>datos en bruto</i>
'monitor'	Los filtros se aplicarán al valor de retorno si <code>\$show</code> no es 'url', 'directory', 'home'	<i>datos filtrados</i>

Examples

Obtener el título del sitio

```
<?php echo get_bloginfo( 'name' ); ?>
```

o

```
<?php echo get_bloginfo(); ?>
```

Salida

```
Matt Mullenweg
```

Basado en estos ajustes de muestra

WordPress dashboard header: Matt Mullenweg 0 + New

Left sidebar menu: Dashboard, Posts, Media, Pages, Comments, Appearance, Plugins, Users, Tools, **Settings**, General, Writing, Reading, Discussion, Media, Permalinks, Collapse menu

General Settings

Site Title: Matt Mullenweg

Tagline: Just another WordPress site
In a few words, explain what this site is about.

WordPress Address (URL): http://localhost/wordpress

Site Address (URL): http://localhost/wordpress
Enter the address here if you [want your site home page](#)

Email Address: matt@mullenweg.com
This address is used for admin purposes, like new user n

Membership: Anyone can register

New User Default Role: Subscriber

Timezone: UTC+0

Obtener el lema del sitio

```
<?php echo get_bloginfo( 'description' ); ?>
```

Salida

```
Just another WordPress site
```

Basado en estos ajustes de muestra

WordPress dashboard header: Matt Mullenweg 0 + New

Left sidebar menu: Dashboard, Posts, Media, Pages, Comments, Appearance, Plugins, Users, Tools, **Settings**, General, Writing, Reading, Discussion, Media, Permalinks, Collapse menu

General Settings

Site Title: Matt Mullenweg

Tagline: Just another WordPress site
In a few words, explain what this site is about.

WordPress Address (URL): http://localhost/wordpress

Site Address (URL): http://localhost/wordpress
Enter the address here if you [want your site home page](#).

Email Address: matt@mullenweg.com
This address is used for admin purposes, like new user notifications.

Membership: Anyone can register

New User Default Role: Subscriber

Timezone: UTC+0

Obteniendo la URL del tema activo

```
<?php echo esc_url( get_bloginfo( 'stylesheet_directory' ) ); ?>
```

Salida

```
http://example.com/wp-content/themes/twenty sixteen
```

Alternativas

Internamente, `get_bloginfo('stylesheet_directory')` llama a `get_stylesheet_directory_uri()`, por lo que es posible que desee utilizar eso:

```
<?php echo esc_url( get_stylesheet_directory_uri() ); ?>
```

Muchos desarrolladores prefieren usar estas funciones dedicadas debido a las convenciones de nomenclatura inconsistentes entre ellos y `get_bloginfo()`. Por ejemplo, `get_stylesheet_directory()` devuelve la ruta del tema secundario; sin embargo, como ilustra nuestro ejemplo anterior, `get_bloginfo('stylesheet_directory')` devuelve la URL del tema secundario. Si usa `get_stylesheet_directory_uri()` lugar, hay menos posibilidades de confusión sobre si está recuperando una ruta o una URL.

Obtener url del sitio

```
<?php echo esc_url(get_bloginfo('url')); ?>
```

o si necesitabas enlazar a una subpágina

```
<?php echo esc_url(get_bloginfo('url') . '/some-sub-page'); ?>
```

Obtenga la dirección de correo electrónico del administrador del sitio

Podemos usar la función `get_bloginfo` para recuperar la dirección de correo electrónico del administrador del sitio.

```
<?php echo get_bloginfo('admin_email'); ?>
```

Lea `get_bloginfo()` en línea: <https://riptutorial.com/es/wordpress/topic/524/get-bloginfo--->

Capítulo 44: `get_home_path()`

Introducción

Obtenga la ruta absoluta del sistema de archivos a la raíz de la instalación de WordPress.

Parámetros

Parámetro	Detalles
<i>Ninguna</i>	Esta función no acepta ningún parámetro.

Observaciones

Diferencia importante entre `get_home_path()` y `ABSPATH`

Tenga en cuenta la diferencia entre `ABSPATH` y `get_home_path()` si tiene WordPress instalado en una subcarpeta.

La función `get_home_path()` siempre devolverá una ruta **sin** la subcarpeta:

- <http://www.example.com> - / var / www / htdocs / example
- <http://www.example.com/wp> - / var / www / htdocs / example

Así es como difiere de `ABSPATH`, que devolverá valores diferentes:

- <http://www.example.com> - / var / www / htdocs / example
- <http://www.example.com/wp> - / var / www / htdocs / example / wp

`ABSPATH` se define primero en `wp-load.php` que se ubicará en `/var/www/htdocs/example/wp/wp-load.php` por lo tanto, aquí es donde `ABSPATH` tomará su definición.

`get_home_path()` comprueba si `site_url` y `home_url` difieren, y elimina la subcadena de la ruta. De lo contrario, devuelve el valor `ABSPATH`:

```
function get_home_path() {
    $home      = set_url_scheme( get_option( 'home' ), 'http' );
    $siteurl   = set_url_scheme( get_option( 'siteurl' ), 'http' );
    if ( ! empty( $home ) && 0 !== strcasecmp( $home, $siteurl ) ) {
        $wp_path_rel_to_home = str_ireplace( $home, '', $siteurl ); /* $siteurl - $home */
        $pos = stripos( str_replace( '\\', '/', $_SERVER['SCRIPT_FILENAME'] ),
            trailingslashit( $wp_path_rel_to_home ) );
        $home_path = substr( $_SERVER['SCRIPT_FILENAME'], 0, $pos );
        $home_path = trailingslashit( $home_path );
    } else {
        $home_path = ABSPATH;
    }
}
```

```
return str_replace( '\\', '/', $home_path );  
}
```

Usandolo en tu codigo

La llamada a `get_home_path()` debe hacerse en un contexto donde ya se haya incluido `wp-admin/includes/file.php`.

Por ejemplo, usar `get_home_path()` dentro del `admin_init` está bien, pero usarlo dentro de `init` no lo es y dará lugar a un error fatal de PHP:

```
Call to undefined function get_home_path()
```

Este archivo solo se incluye desde el contexto de administración (panel de control), si lo necesita absolutamente fuera de este contexto, deberá incluir el archivo usted mismo antes de llamar a la función:

```
require_once(ABSPATH . 'wp-admin/includes/file.php');
```

Examples

Uso

```
$path = get_home_path();
```

Valor de retorno:

string

Ruta completa del sistema de archivos a la raíz de la instalación de WordPress, incluso si está instalada en una subcarpeta.

Ejemplo:

```
/var/www/htdocs/example
```

Lea `get_home_path ()` en línea: <https://riptutorial.com/es/wordpress/topic/9699/get-home-path--->

Capítulo 45: `get_option ()`

Introducción

Recupera un valor de opción basado en un nombre de opción.

Sintaxis

- `get_option ($ opción, $ por defecto)`

Parámetros

Parámetro	Detalles
\$ opción	(cadena) Nombre de la opción a recuperar. Se espera que no sea escapado de SQL.
\$ por defecto	(mixto) (Opcional) Valor predeterminado para devolver si la opción no existe.

Observaciones

Lista de argumentos para la opción \$

- 'admin_email'
- 'nombre de blog'
- 'blogdescripción'
- 'blog_charset'
- 'formato de fecha'
- 'default_category'
- 'casa'
- 'Sitio URL'
- 'modelo'
- 'start_of_week'
- 'upload_path'
- 'users_can_register'
- 'publicaciones por página'
- 'posts_per_rss'

Examples

Mostrar título del blog

Código

```
<h1><?php echo get_option( 'blogname' ); ?></h1>
```

Salida

Nombre del blog en estilo H1.

Mostrar conjunto de caracteres

Código

```
<p><?php echo esc_html( sprintf( __( 'Character set: %s', 'textdomain' ), get_option( 'blog_charset' ) ) ); ?></p>
```

Salida

Conjunto de caracteres: UTF-8

Manejo de opciones no existentes.

Código

```
<?php echo get_option( 'something_bla_bla_bla' ); ?>
```

Salida

falso

Código

```
<?php echo get_option( 'something_bla_bla_bla', 'Oh no!' ); ?>
```

Salida

¡Oh no!

Lea `get_option ()` en línea: <https://riptutorial.com/es/wordpress/topic/9194/get-option--->

Capítulo 46: get_permalink ()

Introducción

Esta función devuelve el paralelismo completo de la publicación actual o la publicación designada.

Sintaxis

- get_permalink (\$ post, \$ leavename)

Parámetros

Parámetro	Detalles
\$ post	(int) (opcional) Post ID u objeto post. El valor predeterminado es el ID de la publicación actual.
\$ leavename	(bool) (opcional) Si se debe mantener el nombre de la publicación o el nombre de la página.

Observaciones

Para el parámetro \$ leavename, es falso por defecto.

Examples

Uso simple de get_parmalink

Código

```
echo get_permalink();
```

Salida

El enlace de la página actual, por ejemplo: <http://website.com/category/name-of-post/>

Especificando la publicación para obtener el enlace.

Código

```
echo get_permalink( 44 );
```

Salida

El enlace de la ID de publicación: 44, por ejemplo: <http://website.com/category/name-of-post/>

Obtén el enlace sin el nombre del post.

Código

```
echo get_permalink( 44 , false );
```

Salida

El enlace de la ID de publicación: 44 sin el nombre de la publicación, por ejemplo:
<http://website.com/category/%postname%/>

Lea `get_permalink ()` en línea: <https://riptutorial.com/es/wordpress/topic/9209/get-permalink--->

Capítulo 47: `get_template_part ()`

Sintaxis

- `get_template_part ($ slug, $ name);`
- `get_template_part ($ slug);`

Parámetros

Parámetro	Detalles
<code>\$ babosa</code>	(cadena) nombre genérico de la barra de la plantilla
<code>\$ nombre</code>	(cadena) nombre de la plantilla especializada

Examples

Cargar una parte de la plantilla desde una subcarpeta

```
get_template_part( 'foo/bar', 'page');
```

requerirá 'bar-page.php' del directorio 'foo'.

Obtener un archivo específico

Puede obtener un archivo específico utilizando esta función.

```
get_template_part('template-parts/layout');
```

Incluya el archivo `layout.php` del subdirectorio de partes de la plantilla ubicado en la raíz de su carpeta de temas.

Lea `get_template_part ()` en línea: <https://riptutorial.com/es/wordpress/topic/5897/get-template-part--->

Capítulo 48: get_template_part ()

Introducción

El propósito de esta función es estandarizar la forma en que se importan parciales o componentes de un tema en la plantilla de tema principal. Podría usar un PHP estándar SSI (servidor incluye), sin embargo, hay algunos beneficios al usar `get_template_part ()`. El uso de esta función reduce los errores propensos a los desarrolladores menos experimentados que intentan identificar una ruta completa en el servidor. Además, falla cuando los archivos no existen y maneja un sistema de respaldo de jerarquía personalizado, también conocido como "búsqueda de plantillas difusas".

Sintaxis

- `get_template_part ($ slug)`
- `get_template_part ($ slug, $ name)`

Parámetros

Parámetro	Detalles
<code>\$ babosa</code>	(cadena) El nombre del slug de la plantilla personalizada.
<code>\$ nombre</code>	(cadena) El nombre de la plantilla especializada. Opcional

Examples

Incluyendo una plantilla personalizada

```
<?php get_template_part( 'foo' ); ?>
```

Incluye

```
../wp-content/themes/your-theme-slug/foo.php
```

Incluyendo una plantilla personalizada con un nombre de archivo separado por guión

```
<?php get_template_part( 'foo','bar' ); ?>
```

Incluye

```
../wp-content/themes/your-theme-slug/foo-bar.php
```

Incluyendo una plantilla personalizada desde dentro de un directorio

```
<?php get_template_part( 'dir/foo' ); ?>
```

Incluye

```
../wp-content/themes/your-theme-slug/dir/foo.php
```

Incluyendo una plantilla personalizada con un nombre de archivo separado por guión ubicado dentro de un directorio

```
<?php get_template_part( 'dir/foo', 'bar' ); ?>
```

Incluye

```
../wp-content/themes/your-theme-slug/dir/foo-bar.php
```

Pasando la variable al ámbito de la plantilla personalizada

```
<?php  
set_query_var( 'passed_var', $my_var );  
get_template_part( 'foo', 'bar' );  
?>
```

Accede a él en `foo-bar.php`

```
<?php echo $passed_var; ?>
```

Lea `get_template_part ()` en línea: <https://riptutorial.com/es/wordpress/topic/5993/get-template-part--->

Capítulo 49: get_template_part ()

Sintaxis

- `get_template_part ('file-name-no-extension');`

Parámetros

Parámetro	Descripción
nombre-archivo-sin-extensión	El nombre de la parte de la plantilla sin extensión. Por ejemplo, 'foo' en lugar de 'foo.php'

Examples

Cargando parte de la plantilla

Extrae el código de un determinado archivo especificado en otro archivo donde se realizó la llamada.

Por ejemplo, en `example.php`

```
<h1>Hello World!</h1>
```

Dentro de `page.php`

```
// header code  
get_template_part ('example');  
// rest of page code
```

Salida:

```
// header code  
<h1>Hello World!</h1>  
// rest of page code
```

Lea `get_template_part ()` en línea: <https://riptutorial.com/es/wordpress/topic/6267/get-template-part--->

Capítulo 50: `get_the_category ()`

Introducción

Esta función devuelve todas las categorías como una matriz de la publicación o página actual o la publicación o página designada.

Sintaxis

- `get_the_category ($ id)`

Parámetros

Parámetro	Detalles
\$ id	(int) (Opcional) predeterminado a la ID de la publicación actual. La identificación del correo.

Observaciones

Tenga en cuenta que `get_the_category ()` devuelve una matriz, lo que significa que no puede repetir directamente el valor devuelto.

Aquí hay una lista de objetos de cada categoría que puede imprimir:

- `term_id`
- `nombre`
- `babosa`
- `term_group`
- `term_taxonomy_id`
- `taxonomia`
- `descripción`
- `padre`
- `contar`
- `object_id`
- `filtrar`
- `cat_ID`
- `category_count`
- Descripción de categoría
- nombre de gato
- `nombre_categoria`
- `category_parent`

Examples

Consigue todos los nombres de las categorías de la publicación.

Código

```
$categories = get_the_category();  
foreach( $categories as $category ) {  
    echo $category->name . '<br />';  
}
```

Salida

Todos los nombres de las categorías de la página actual, uno en cada línea.

Consigue todas las identificaciones de las categorías de la publicación.

Código

```
$categories = get_the_category();  
foreach( $categories as $category ) {  
    echo $category->term_id . '<br />';  
}
```

Salida

Todos los identificadores de categorías de la página actual, uno en cada línea.

Lea `get_the_category ()` en línea: <https://riptutorial.com/es/wordpress/topic/9211/get-the-category->

--

Capítulo 51: get_the_title ()

Introducción

Esta función devuelve el título de la publicación actual o la publicación designada.

Sintaxis

- get_the_title (\$ post)

Parámetros

Parámetro	Detalles
\$ post	(int) (opcional) Post ID u objeto post. El valor predeterminado es el ID de la publicación actual.

Observaciones

Si planea obtener el título de una publicación o página utilizando un bucle de publicación, se sugiere utilizar the_title () en su lugar.

Examples

Uso simple de get_the_title

Código

```
get_the_title();
```

Salida

El título de la publicación o página actual.

Obtener el título de una ID de publicación especificada

Código

```
get_the_title( 44 );
```

Salida

El título del post id: 44.

Lea `get_the_title ()` en línea: <https://riptutorial.com/es/wordpress/topic/9214/get-the-title--->

Capítulo 52: Hacer solicitudes de red con API HTTP

Sintaxis

- \$ respuesta = wp_remote_get (\$ url, \$ args);
- \$ respuesta = wp_remote_post (\$ url, \$ args);
- \$ respuesta = wp_safe_remote_post (\$ url, \$ args);

Parámetros

Parámetro	Detalles
\$ url	(cadena) (Requerido) URL del sitio para recuperar.
\$ args	(array) (Opcional) Solicita argumentos.

Observaciones

Devoluciones

(*WP_Error* | *array*) La respuesta como una matriz, o *WP_Error* en caso de error.

Examples

OBTENER un recurso JSON remoto

Este fragmento tomará un recurso con formato JSON, lo decodificará e imprimirá en formato de matriz PHP.

```
// Fetch
$response = wp_remote_get( 'http://www.example.com/resource.json' );

if ( ! is_wp_error( $response ) ) {
    $headers = wp_remote_retrieve_headers( $response );

    if ( isset( $headers[ 'content-type' ] ) && 'application/json' === $headers[ 'content-type' ] ) {
        print_r( json_decode( wp_remote_retrieve_body( $response ) ) );
    }
}
```

Lea Hacer solicitudes de red con API HTTP en línea:

<https://riptutorial.com/es/wordpress/topic/1116/hacer-solicitudes-de-red-con-api-http>

Capítulo 53: home_url ()

Sintaxis

- home_url (\$ ruta, \$ esquema);

Parámetros

Parámetro	Detalles
\$ camino	(<i>Cadena</i> , <i>opcional</i>) Para agregar más segmentos después de la URL de inicio.
\$ esquema	(<i>Cadena</i> , <i>opcional</i>) Esquema para dar el contexto de la URL de inicio. Acepta 'http', 'https', 'relative', 'rest' o null.

Examples

Obteniendo la URL de inicio

`home_url()` se utiliza para recuperar la URL de inicio del sitio actual.

```
<?php echo esc_url( home_url( '/' ) ) ; ?>
```

Salida

```
http://www.example.com
```

Sitio URL

Devuelve la opción 'site_url' con el protocolo apropiado ([https: //](https://))

```
<?php echo site_url(); ?>
```

Salida

```
http://www.example.com
```

Lea `home_url ()` en línea: <https://riptutorial.com/es/wordpress/topic/1252/home-url--->

Capítulo 54: Instalacion y configuracion

Examples

WordPress en LAMP

He probado los siguientes pasos en Amazon EC2 con Ubuntu 14.04

Aquí hay un resumen de las líneas de comando para instalar la pila de tecnología LAMP (antes de instalar wordpress):

1: Instalar la pila de tecnología LAMP

Actualizar linux

#> sudo apt-get update -> actualizar la información de los repositorios de paquetes

#> sudo apt-get upgrade -> instalar actualizaciones de paquetes

Instalar apache2

#> sudo apt-get install apache2

Instalar php

#> sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt

Instalar mysql

#> sudo apt-get install mysql-server php5-mysql

Instalar phpmyadmin

#> sudo apt-get install phpmyadmin

En caso de que algo saliera mal, elimine el paquete instalado y su (s) archivo (s) de configuración

#> sudo apt-get purge package_name

Una vez que se instala la pila, aquí están los pasos para completar la instalación de wordpress:

2: instalar WordPress

1. `wget https://wordpress.org/latest.zip`
2. `sudo mkdir /var/www/wordpress`
3. Mueva el archivo zip a / var / www / wordpress y extraiga.
4. `cd /etc/apache2/sites-available/`
5. `sudo cp 000-default.conf wordpress.conf`

6. Modifique `wordpress.conf` para que `apache2` sepa enviar cualquier solicitud o su dominio a la carpeta de la aplicación `wordpress`.
7. `sudo a2ensite wordpress.conf`
8. `sudo service apache2 restart`
9. Vaya a `PhpMyAdmin` a través de su navegador por `youdomain.com/phpmyadmin`. Cree un nuevo usuario "wordpress" y marque para crear la base de datos correspondiente. `cd /var/www/wordpress`
10. `sudo cp wp-config-example.php wp-config.php`
11. Modifique el archivo de configuración para agregar su información de base de datos de `wordpress MySQL`.
12. `sudo chown -R www-data:www-data /var/www/wordpress`
13. `sudo chmod -R 755 /var/www/wordpress/`
14. Abra el navegador y escriba su dominio. Deberías ver la página de instalación de `WordPress`. Sigue las instrucciones y listo!

Instalación WP en MAMP

Es bastante sencillo instalar `wordpress` en `MAMP`.

Tienes que seguir unos sencillos pasos:

- 1: descargue `MAMP` desde [aquí](#) , es gratis y probablemente no necesite la versión pro.
- 2 - Instalar en tu PC o Mac.
- 3 - Ejecute el programa -> verá una pequeña ventana abierta, desde allí puede configurar `MAMP`. De momento, deje todos los valores preestablecidos, ¡para la primera instalación no necesita complicar su vida! En el futuro, recuerde que es una buena práctica cambiar su contraseña y nombre de usuario para la base de datos `MySQL`. Por defecto es `root`.
- 4 - Haga clic en "Abrir página webStart" - aquí puede ver su información de datos como contraseña, nombre de administrador y también información sobre `MAMP`.
- 5 - Haga clic en herramientas -> `phpMyAdmin` y será redirigido a la página de la base de datos.
- 6 - Cree una nueva base de datos, haga clic en Nuevo y dele el nombre que desee, necesitará este nombre más adelante.
- 7 - Ahora busque una carpeta llamada "htdocs", está dentro de la carpeta `MAMP` en su PC o Mac. Si usas una Mac, necesitas una aplicación en `Finder` y abrir la aplicación usando "mostrar contenido del paquete". Dentro encontrarás la carpeta `htdocs` .
- 8 - Tome la carpeta de `Wordpress` y cópiela dentro de los `htdocs`, debe colocar dentro de toda la carpeta, no colocar el archivo zip. Renombra la carpeta, puedes llamar con el nombre de tu

proyecto.

9 - Vuelva a la ventana de MAMP en su navegador y haga clic en "Mi sitio": esto abrirá una llamada a la URL `http://localhost:8888` (8888 es el puerto predeterminado, puede cambiarlo si lo desea). Aquí puede ver la carpeta que ha colocado dentro de la carpeta `htdocs` , haga clic en el nombre que le ha dado a la carpeta.

10 - Ahora inicie una instalación normal de WordPress, necesita usar el administrador y la contraseña. Por defecto, es `root` y `root` , y usa el nombre de la base de datos que creaste anteriormente.

11 - Ejecutar la instalación y terminar!

Verá su sitio web en `http://localhost:8888/namefolder` Por supuesto, necesita mantener MAMP en funcionamiento.

Lea Instalacion y configuracion en línea:

<https://riptutorial.com/es/wordpress/topic/6606/instalacion-y-configuracion>

Capítulo 55: Jerarquía de plantillas

Observaciones

Plugins para depuración en WordPress:

- <https://wordpress.org/plugins/query-monitor/>
- <https://wordpress.org/plugins/debug-bar/>
- <https://wordpress.org/plugins/debug-bar-console/>
- <https://wordpress.org/plugins/kint-debugger/>
- <https://wordpress.org/plugins/rest-api-console/>

Examples

Introducción

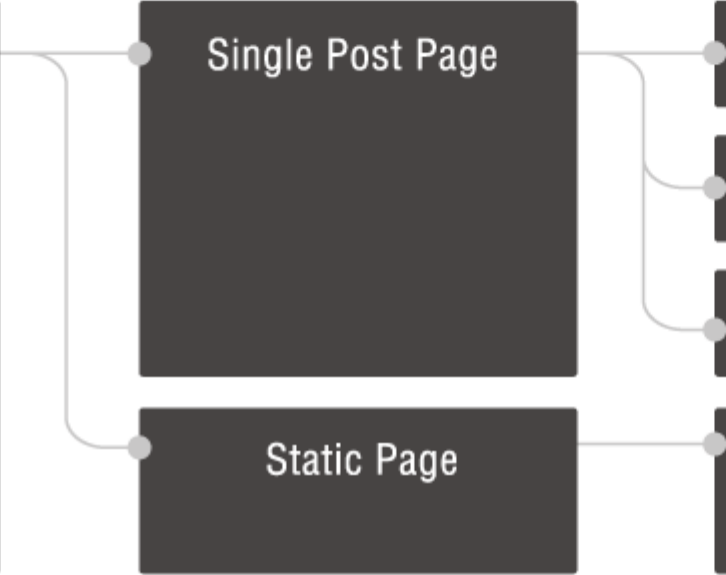
Una de las cosas más importantes que debe aprender cuando está creando un tema de WordPress es la jerarquía de plantillas de WordPress para los temas. La jerarquía de plantillas define qué archivo de plantilla se cargará para cada solicitud y en qué orden. Si la primera plantilla no existe en la jerarquía, WordPress intentará cargar la siguiente y así sucesivamente hasta que termine en `index.php`.

Para describir en detalle la jerarquía de plantillas, la mejor manera es, por supuesto, utilizar una imagen con la estructura completa:

Archive Page



Singular Page



Site Front Page



que solo se dirige a la categoría con un slug específico, por ejemplo `category-books.php` se usaría solo para la categoría con el `book` slug. Otro ejemplo es `page-$id.php` que solo se dirige a una página con una ID específica, por ejemplo, `page-41.php` solo apuntaría a la página con la ID 41.

Después de las plantillas dirigidas a tipos específicos o publicaciones, llegamos a las plantillas de tipo genérico, como `archive.php` para todas las páginas de archivo o `page.php` para todas las páginas. Pero recuerde, esos solo se utilizarán si la página actual no coincide con ninguna de las plantillas que están en una jerarquía superior.

Por último, si WordPress no pudo encontrar ninguna plantilla coincidente en el directorio de plantillas, la última alternativa alternativa es siempre `index.php` que es el único archivo de plantilla requerido en un tema de WordPress.

Depuración

Es fácil perderse mientras se depura la búsqueda. Puede utilizar el comando `debug_backtrace` .

Coloque el siguiente fragmento dentro de cualquier plantilla que desee depurar y vea la página generada:

```
<!--  
<?php print_r( debug_backtrace() ) ?>  
-->
```

Lea Jerarquía de plantillas en línea: <https://riptutorial.com/es/wordpress/topic/6116/jerarquia-de-plantillas>

Capítulo 56: La barra de administración (también conocida como "La barra de herramientas")

Observaciones

La barra de herramientas de administración de WordPress se agregó en la versión 3.1 y contiene enlaces a tareas administrativas comunes, así como enlaces al perfil del usuario y otra información de WordPress. Sin embargo, a muchos propietarios de sitios les disgusta mostrar la barra de herramientas de forma predeterminada a todos los usuarios que han iniciado sesión y / o desean agregar sus propias opciones.

Examples

Eliminar la barra de herramientas de administración de todos, excepto los administradores

Agregue el siguiente código a `functions.php` para eliminarlo de todos excepto el nivel de usuario Administrador:

```
add_action('after_setup_theme', 'no_admin_bar');

function no_admin_bar() {
    if (!current_user_can('administrator') && !is_admin()) {
        show_admin_bar(false);
    }
}
```

Eliminar la barra de herramientas de administración usando filtros

Otra forma de ocultar la barra de administración es agregar

```
if ( !current_user_can( 'manage_options' ) ) {
    add_filter( 'show_admin_bar', '__return_false' , 1000 );
}
```

Los usuarios que no tienen privilegios para acceder a la página de Configuración, no podrán ver la barra de administración.

Cómo quitar el logotipo de WordPress de la barra de administración

Los desarrolladores pueden usar la acción **admin_bar_menu** para eliminar elementos de la barra de administración o la barra de herramientas de WordPress.

```
add_action('admin_bar_menu', 'remove_wp_logo_from_admin_bar', 999);
function remove_wp_logo_from_admin_bar( $wp_admin_bar ) {
    $wp_admin_bar->remove_node('wp-logo');
}
```

El código anterior elimina el logotipo de WordPress de la barra de administración. Todo lo que necesita hacer es pegar el código dentro de su archivo functions.php.

El parámetro que se pasa al método remove_node es el ID del nodo que desea eliminar. Los ID se pueden encontrar en el código fuente HTML de la página de WordPress con una barra de herramientas. Por ejemplo, el ID del elemento li para el logotipo de WordPress a la izquierda en la barra de herramientas es "wp-admin-bar-wp-logo":

```
<li id="wp-admin-bar-wp-logo" class="menupop"> ... </li>
```

Elimine "wp-admin-bar-" de la ID de li para obtener la ID del nodo. De este ejemplo, el ID de nodo es "wp-logo".

Puede usar las herramientas del inspector del navegador para averiguar los ID de nodo de varios elementos o nodos en la barra de administración.

Agregue su logotipo personalizado y enlace personalizado en la página de inicio de sesión de administrador

Puede agregar los ganchos de abajo para agregar su propio logotipo y el enlace para reemplazar el logotipo de wordpress predeterminado.

Para agregar logo personalizado

```
function custom_login_logo() {
echo '<style type="text/css">
h1 a { background-image: url(.'.get_bloginfo('template_directory').'/images/custom-logo.png)
!important; background-size : 100% !important; width: 300px !important; height : 100px
!important;}
</style>';
}
add_action('login_head', 'custom_login_logo');
```

Para agregar enlace logo personalizado

```
add_filter( 'login_headerurl', 'custom_loginlogo_url' );
function custom_loginlogo_url($url) {
    return home_url();
}
```

Lea [La barra de administración \(también conocida como "La barra de herramientas"\) en línea: https://riptutorial.com/es/wordpress/topic/2932/la-barra-de-administracion--tambien-conocida-como--la-barra-de-herramientas--](https://riptutorial.com/es/wordpress/topic/2932/la-barra-de-administracion--tambien-conocida-como--la-barra-de-herramientas--)

Capítulo 57: Meta Box

Introducción

Uso simple de una Meta Box en los editores de contenido de wp-admin

Sintaxis

- `_x` ('Text', 'Description', 'textdomain') se usa para agregar una descripción para el servicio de traducción en lugar de `__` ('Text', 'textdomain') que es solo la traducción
- `_ex` ('Text', 'Description', 'textdomain') se usa para hacer eco del texto traducido con una descripción

Observaciones

El contenido dentro de la meta caja de render puede ser cualquier cosa. En lugar de que los valores se integren directamente, también puede usar un `include` con una plantilla de PHP y usar el método `set_query_var` para pasarle datos. La salvación funcionaría de la misma manera.

Examples

Un ejemplo simple con una entrada regular y una entrada de selección

```
/**
 * Add meta box to post types.
 *
 * @since 1.0.0
 */
function myplugin_add_meta_box() {
    // Set up the default post types/
    $types = array(
        'post',
    );

    // Optional filter for adding the meta box to more types. Uncomment to use.
    // $types = apply_filters( 'myplugin_meta_box_types', $types );

    // Add the meta box to the page
    add_meta_box(
        'myplugin-meta-box', // Meta Box Id. Can be anything.
        _x( 'Custom Meta', 'Custom Meta Box', 'myplugin' ), // The title of the meta box.
        // Translation is optional. Can just be string.
        'myplugin_render_meta_box', // The render meta box function.
        $types, // Add the post types to which to add the meta box.
        'side', // Show on the side of edit.
        'high' // Show at top of edit.
    );
}
```

```

add_action( 'add_meta_boxes', 'myplugin_add_meta_box' );

/**
 * Render the meta box.
 *
 * This shows examples of a basic input and a select inside a meta box. These can be anything.
 *
 * @since 1.0.0
 *
 * @param $post WP_Post The post being edited.
 */
function myplugin_render_meta_box( $post ) {
    // Get the current post meta values for our custom meta box.
    $city = get_post_meta( $post->ID, 'city', true ); // True is for returning a single
value.
    $country = get_post_meta( $post->ID, 'country', true ); // True is for returning a single
value.

    // Add the WP Nonce field for security.
    wp_nonce_field( plugin_basename( __FILE__ ), 'myplugin_meta_nonce' );
?>

<p>
<label for="city">
    <?php _ex( 'City', 'Custom Meta Box Template', 'myplugin' ); ?>
</label>
<input name="city" id="city" value="<?php echo $city; ?>" />
</p>
<p>
<label for="country">
    <?php _ex( 'Country', 'Custom Meta Box Template', 'myplugin' ); ?>
</label>
<select name="country" id="country">
    <option value="United States" <?php selected( $country, 'United States' ); ?><?php
_ex( 'United States', 'Custom Meta Box Template', 'myplugin' ); ?></option>
    <option value="Mexico" <?php selected( $country, 'Mexico' ); ?><?php _ex( 'Mexico',
'Custom Meta Box Template', 'myplugin' ); ?></option>
    <option value="Canada" <?php selected( $country, 'Canada' ); ?><?php _ex( 'Canada',
'Custom Meta Box Template', 'myplugin' ); ?></option>
</select>
</p>

<?php
}

/**
 * Save meta box data.
 *
 * @since 1.0.0
 *
 * @param $post_id int The Id of the Post being saved.
 */
function myplugin_save_meta_data( $post_id ) {
    // Verify this is not an auto save.
    if ( defined( 'DOING_AUTOSAVE' ) && DOING_AUTOSAVE ) {
        return;
    }

    // Validate the meta box nonce for security.
    if ( ! isset( $_POST['myplugin_meta_nonce'] ) || ! wp_verify_nonce(
$_POST['myplugin_meta_nonce'], plugin_basename( __FILE__ ) ) ) {

```

```

        return;
    }

    // Get the new values from the form.
    $city    = $_POST['city'];
    $country = $_POST['country'];

    // update_post_meta will add the value if it doesn't exist or update it if it does.
    update_post_meta( $post_id, 'city', $city );
    update_post_meta( $post_id, 'country', $country );

    /*
    * OPTIONAL ALTERNATIVE
    *
    * Instead of just using update_post_meta, you could also check the values and
    * issue create/update/delete on the post meta value.
    */
    // $current_city_value = get_post_meta( $post_id, 'city', true ); // True is for returning
a single value.
    //
    // // Add the post meta if it doesn't exist.
    // if ( $city && '' === $city ) {
    //     add_post_meta( $post_id, 'city', $city, true ); // True means the key is unique to
the post. False is default and more than one can be added.
    // }
    // // Edit the post meta if it does exist and there is a new value.
    // elseif ( $city && $city !== $current_city_value ) {
    //     update_post_meta( $post_id, 'city', $city );
    // }
    // // Delete the post meta if there is no new value.
    // elseif ( '' === $city && $current_city_value ) {
    //     delete_post_meta( $post_id, 'city', $current_city_value ); // $current_city_value
is optional and is used to differentiate between other values with the same key.
    // }
}

add_action( 'save_post', 'myplugin_save_meta_data' );

```

Lea Meta Box en línea: <https://riptutorial.com/es/wordpress/topic/9611/meta-box>

Capítulo 58: Migración del sitio

Sintaxis

- OLD_SITE: el sitio antiguo que se está migrando (por ejemplo, <http://localhost/example>)
- NEW_SITE - El nuevo sitio al que migrar (por ejemplo: <https://example.com>)

Examples

Actualización de las tablas con una nueva URL.

```
UPDATE wp_options SET option_value = replace(option_value, 'OLD_SITE', 'NEW_SITE') WHERE
option_name = 'home' OR option_name = 'siteurl';
UPDATE wp_posts SET guid = replace(guid, 'OLD_SITE','NEW_SITE');
UPDATE wp_posts SET post_content = replace(post_content, 'OLD_SITE', 'NEW_SITE');
```

Lea Migración del sitio en línea: <https://riptutorial.com/es/wordpress/topic/9610/migracion-del-sitio>

Capítulo 59: Personalizador Hello World

Parámetros

Parámetro	Detalles
mi tema	Un identificador único para su tema (o tema infantil). Este puede ser tu tema babosa

Examples

Hola mundo ejemplo

El concepto fundamental del personalizador es que los administradores pueden realizar una vista previa de los cambios en su sitio y luego agregarlos permanentemente.

Lo siguiente se puede copiar y pegar en el archivo `functions.php` un tema para

- Añadir una sección de personalización llamada `My First Section`
- Agregue una configuración de personalización llamada `Hello World Color` permita al administrador elegir un color
- Agregue una regla de css para `.hello-world` que corresponda con el color elegido y por defecto a `#000000` si no se elige nada. La configuración se colocará en una etiqueta `<style>` al final de `<head>` .

```
function mytheme_customize_register( $wp_customize ) {

    $wp_customize->add_section( 'my_first_section_id' , array(
        'title'      => __( 'My First Section', 'mytheme' ),
        'priority'   => 30,
    ) );

    $wp_customize->add_setting( 'hello_world_color' , array(
        'default'    => '#000000',
        'transport'  => 'refresh',
    ) );

    $wp_customize->add_control( new WP_Customize_Color_Control( $wp_customize, 'link_color',
array(
        'label'      => __( 'Hello World Color', 'mytheme' ),
        'section'    => 'my_first_section_id',
        'settings'   => 'hello_world_color',
    ) ) );

}
add_action( 'customize_register', 'mytheme_customize_register' );
```

```
function mytheme_customize_css()
{
    ?>
    <style type="text/css">
        .hello-world { color: #<?php echo get_theme_mod('hello_world_color', '000000'); ?>; }
    </style>
    <?php
}
add_action( 'wp_head', 'mytheme_customize_css');
```

Lea Personalizador Hello World en línea:

<https://riptutorial.com/es/wordpress/topic/2875/personalizador-hello-world>

Capítulo 60: Post Formatos

Observaciones

Los siguientes formatos de publicación están disponibles para que los usuarios elijan, si el tema permite su soporte.

Tenga en cuenta que si bien la entrada de contenido de la publicación real no cambiará, el tema puede usar esta opción del usuario para mostrar la publicación de manera diferente según el formato elegido. Por ejemplo, un tema podría dejar de mostrar el título para una publicación de "Estado". La forma en que se muestran las cosas depende completamente del tema, pero aquí hay algunas pautas generales.

- A un lado - Normalmente estilo sin título. Similar a una actualización de nota de Facebook.
- Galería - Una galería de imágenes. La publicación probablemente contendrá un código abreviado de la galería y tendrá archivos adjuntos de imágenes.
- enlace - Un enlace a otro sitio. Es posible que los temas deseen utilizar la primera etiqueta en el contenido de la publicación como enlace externo para esa publicación. Un enfoque alternativo podría ser si la publicación consiste solo en una URL, entonces esa será la URL y el título (`post_title`) será el nombre adjunto al ancla.
- imagen - una sola imagen. La primera etiqueta en la publicación podría ser considerada la imagen. Alternativamente, si la publicación consta solo de una URL, esa será la URL de la imagen y el título de la publicación (`post_title`) será el atributo del título de la imagen.
- cita - una cita. Probablemente contendrá un blockquote con el contenido de la cita. Alternativamente, la cita puede ser solo el contenido, con la fuente / autor como título.
- estado: una breve actualización de estado, similar a una actualización de estado de Twitter.
- video - Un solo video o lista de reproducción de video. La primera etiqueta u objeto / inserción en el contenido de la publicación podría considerarse el video. Alternativamente, si la publicación consiste solo en una URL, esa será la URL del video. También puede contener el video como un archivo adjunto a la publicación, si el soporte de video está habilitado en el blog (como a través de un complemento).
- audio - Un archivo de audio o lista de reproducción. Podría ser utilizado para Podcasting.
- chat - una transcripción de chat

Examples

Agregar tipo de publicación al tema

Añadir post-formatos a `post_type` 'página'

```
add_post_type_support( 'page', 'post-formats' );
```

El siguiente ejemplo registra el tipo de publicación personalizada `'my_custom_post_type'`, y agrega los formatos de publicación.

Registrar tipo de publicación personalizada 'my_custom_post_type'

```
add_action( 'init', 'create_my_post_type' );
function create_my_post_type() {
    register_post_type( 'my_custom_post_type',
        array(
            'labels' => array( 'name' => __( 'Products' ) ),
            'public' => true
        )
    );
}
```

Agregue post-formatos a post_type 'my_custom_post_type'

```
add_post_type_support( 'my_custom_post_type', 'post-formats' );
```

O en la función `register_post_type()`, agregue 'post-formats', en la matriz de parámetros 'admite'. El siguiente ejemplo es equivalente al anterior.

Registre el tipo de publicación personalizada 'my_custom_post_type' con el parámetro 'support'

```
add_action( 'init', 'create_my_post_type' );
function create_my_post_type() {
    register_post_type( 'my_custom_post_type',
        array(
            'labels' => array( 'name' => __( 'Products' ) ),
            'public' => true,
            'supports' => array('title', 'editor', 'post-formats')
        )
    );
}
```

Añadir soporte de tema para la publicación

Llamada de función

```
add_theme_support( 'post-formats' )
```

Lea Post Formatos en línea: <https://riptutorial.com/es/wordpress/topic/6075/post-formatos>

Capítulo 61: Seguridad en WordPress - Escape

Sintaxis

- `esc_html` (string \$ text)
- `esc_url` (string \$ url, array \$ protocolos, string \$ _context)
- `esc_js` (string \$ text)
- `wp_json_encode` (\$ data mixta, int \$ options, int \$ depth = 512)
- `esc_attr` (string \$ text)
- `esc_textarea` (string \$ text)

Observaciones

La seguridad debe estar siempre en mente cuando se desarrolla. Sin seguridad, una aplicación está abierta a varios ataques, como las inyecciones de SQL, XSS, CSRF, RFI, etc., que pueden provocar graves problemas.

Los datos que no son de confianza provienen de muchas fuentes (usuarios, sitios de terceros, su propia base de datos, ...) y todo debe validarse tanto en la entrada como en la salida. (Fuente: WordPress Codex)

Los datos deben validarse, desinfectarse o escaparse según el uso y el propósito.

Para validar es para asegurarse de que los datos que ha solicitado del usuario coincidan con lo que han enviado. (Fuente: WordPress Codex)

La desinfección es un enfoque un poco más liberal para aceptar datos de usuarios. Podemos recurrir al uso de estos métodos cuando hay un rango de información aceptable. (Fuente: WordPress Codex)

Escapar es tomar los datos que ya tiene y ayudar a protegerlos antes de procesarlos para el usuario final. (Fuente: WordPress Codex)

Examples

escape de datos en código HTML

`esc_html` se debe utilizar en cualquier momento en que estemos generando datos dentro del código HTML.

```
<h4><?php echo esc_html( $title ); ?></h4>
```

escapar de una url

```
<a href="<?php echo esc_url( home_url( '/' ) ); ?>">Home</a>


```

datos de escape en código js

`esc_js()` está destinado a ser utilizado para JS en línea, dentro de un atributo de etiqueta.

Para datos dentro de una etiqueta `<script>` use `wp_json_encode()` .

```
<input type="text" onfocus="if( this.value == '<?php echo esc_js( $fields['input_text'] ); ?>' ) { this.value = ''; }" name="name">
```

`wp_json_encode()` codifica una variable en JSON, con algunas comprobaciones de validez.

Tenga en cuenta que `wp_json_encode()` incluye las comillas de delimitación de cadenas automáticamente.

```
<?php
$book = array(
    "title" => "JavaScript: The Definitive Guide",
    "author" => "Stack Overflow",
);
?>
<script type="text/javascript">
var book = <?php echo wp_json_encode($book) ?>;
/* var book = {
    "title": "Security in WordPress",
    "author" => "Stack Overflow",
}; */
</script>
```

0

```
<script type="text/javascript">
    var title = <?php echo wp_json_encode( $title ); ?>;
    var content = <?php echo wp_json_encode( $content ); ?>;
    var comment_count = <?php echo wp_json_encode( $comment_count ); ?>;
</script>
```

atributos de escape

```
<input type="text" value="<?php echo esc_attr($_POST['username']); ?>" />
```

datos de escape en textarea

```
<textarea><?php echo esc_textarea( $text ); ?></textarea>
```

Lea Seguridad en WordPress - Escape en línea:

<https://riptutorial.com/es/wordpress/topic/6115/seguridad-en-wordpress---escape>

Capítulo 62: Seguridad en WordPress - Sanitización

Sintaxis

- `sanitize_text_field` (string \$ str)
- `sanitize_title` (string \$ title, string \$ fallback_title, string \$ context)
- `sanitize_email` (string \$ email)
- `sanitize_html_class` (string \$ class, string \$ fallback)
- `sanitize_file_name` (string \$ nombre)
- `sanitize_user` (string \$ username, boolean \$ strict)

Observaciones

La seguridad debe estar siempre en mente cuando se desarrolla. Sin seguridad, una aplicación está abierta a varios ataques, como las inyecciones de SQL, XSS, CSRF, RFI, etc., que pueden provocar graves problemas.

Los datos que no son de confianza provienen de muchas fuentes (usuarios, sitios de terceros, su propia base de datos, ...) y todo debe validarse tanto en la entrada como en la salida. (Source: WordPress Codex)

Los datos deben validarse, desinfectarse o escaparse según el uso y el propósito.

Para validar es para asegurarse de que los datos que ha solicitado del usuario coincidan con lo que han enviado. (Source: WordPress Codex)

La desinfección es un enfoque un poco más liberal para aceptar datos de usuarios. Podemos recurrir al uso de estos métodos cuando hay un rango de información aceptable. (Source: Wordpress Codex)

Escapar es tomar los datos que ya tiene y ayudar a protegerlos antes de procesarlos para el usuario final. (Source: WordPress Codex)

Examples

Desinfectar el campo de texto

```
$title = sanitize_text_field( $_POST['title'] );
```

Desinfectar título

El valor devuelto está destinado a ser adecuado para su uso en una URL, no como un título legible por humanos. Utilice `sanitize_text_field` en su lugar.

```
$new_url = sanitize_title($title);
```

Desinfectar el correo electrónico

```
$sanitized_email = sanitize_email(' admin@example.com! ');
```

Desinfectar clase html

```
$post_class = sanitize_html_class( $post->post_title );  
echo '<div class="' . $post_class . "'>';
```

Desinfectar nombre de archivo

```
$incfile = sanitize_file_name($_REQUEST["file"]);  
include($incfile . ".php");
```

Sin limpiar el nombre del archivo, un atacante podría simplemente pasar [http: // attacker_site / malicious_page](http://attacker_site/malicious_page) como entrada y ejecutar cualquier código en su servidor.

Desinfectar nombre de usuario

```
$user = sanitize_user("attacker username<script>console.log(document.cookie)</script>");
```

El valor de \$ user después de sanear es "nombre de usuario del atacante"

Lea Seguridad en WordPress - Sanitización en línea:

<https://riptutorial.com/es/wordpress/topic/6348/seguridad-en-wordpress---sanitizacion>

Capítulo 63: Taxonomías

Sintaxis

- register_taxonomy (\$ taxonomy, \$ object_type, \$ args);

Parámetros

Parámetro	Detalles
\$ taxonomy	(cadena) (requerido) El nombre de la taxonomía. El nombre solo debe contener letras minúsculas y el carácter de subrayado, y no debe tener más de 32 caracteres (restricción de la estructura de la base de datos).
\$ object_type	(matriz / cadena) (requerido) Nombre del tipo de objeto para el objeto de taxonomía. Los tipos de objeto pueden ser Tipo de publicación incorporado o cualquier Tipo de publicación personalizado que se pueda registrar.
\$ args	(matriz / cadena) (opcional) Una matriz de Argumentos.

Examples

Ejemplo de registro de una taxonomía para géneros.

```
<?php
// hook into the init action and call create_book_taxonomies when it fires
add_action( 'init', 'create_book_taxonomies', 0 );

// create taxonomy genres for the post type "book"
function create_book_taxonomies() {
    // Add new taxonomy, make it hierarchical (like categories)
    $labels = array(
        'name'          => _x( 'Genres', 'taxonomy general name' ),
        'singular_name' => _x( 'Genre', 'taxonomy singular name' ),
        'search_items'  => __( 'Search Genres' ),
        'all_items'     => __( 'All Genres' ),
        'parent_item'   => __( 'Parent Genre' ),
        'parent_item_colon' => __( 'Parent Genre:' ),
        'edit_item'     => __( 'Edit Genre' ),
        'update_item'   => __( 'Update Genre' ),
        'add_new_item'  => __( 'Add New Genre' ),
        'new_item_name' => __( 'New Genre Name' ),
        'menu_name'     => __( 'Genre' ),
    );

    $args = array(
        'hierarchical' => true,
        'labels'       => $labels,
        'show_ui'     => true,
```

```

        'show_admin_column' => true,
        'query_var'         => true,
        'rewrite'          => array( 'slug' => 'genre' ),
    );

    register_taxonomy( 'genre', array( 'book' ), $args );
}
?>

```

Puede definir taxonomías personalizadas en el archivo de plantilla de `functions.php` un tema:

Añadir categoría en la página

También puede agregar la misma taxonomía creada a la medida en la página de tipo de publicación utilizando el siguiente código.

```

function add_taxonomies_to_pages() {
    register_taxonomy_for_object_type( 'genre', 'page' );
}
add_action( 'init', 'add_taxonomies_to_pages' );

```

Agregue el código anterior en el archivo `functions.php` de su tema. Misma manera se puede añadir personalizada o predeterminada `post_tag` en la página Tipo de puesto.

Para obtener páginas utilizando una consulta de taxonomía personalizada, debe agregar el código a continuación en el mismo archivo.

```

if ( ! is_admin() ) {
    add_action( 'pre_get_posts', 'category_and_tag_archives' );
}

function category_and_tag_archives( $wp_query ) {
    $my_post_array = array('page');
    if ( $wp_query->get( 'category_name' ) || $wp_query->get( 'cat' ) )
        $wp_query->set( 'post_type', $my_post_array );
}

```

Agregue categorías y etiquetas a las páginas e insértelas como clase en

```

// add tags and categories to pages

function add_taxonomies_to_pages() {
    register_taxonomy_for_object_type( 'post_tag', 'page' );
    register_taxonomy_for_object_type( 'category', 'page' );
}
add_action( 'init', 'add_taxonomies_to_pages' );
if ( ! is_admin() ) {
    add_action( 'pre_get_posts', 'category_and_tag_archives' );
}

function category_and_tag_archives( $wp_query ) {
    $my_post_array = array('post','page');

```

```

if ( $wp_query->get( 'category_name' ) || $wp_query->get( 'cat' ) )
$wp_query->set( 'post_type', $my_post_array );

if ( $wp_query->get( 'tag' ) )
$wp_query->set( 'post_type', $my_post_array );
}

// add tags and categories as class to <body>

function add_categories_and_tags( $classes = '' ) {
    if( is_page() ) {
        $categories = get_the_category();
        foreach( $categories as $category ) {
            $classes[] = 'category-'. $category->slug;
        }
        $tags = get_the_tags();
        foreach( $tags as $tag ) {
            $classes[] = 'tag-'. $tag->slug;
        }
    }
    return $classes;
}
add_filter( 'body_class', 'add_categories_and_tags' );

```

Agregue categorías y etiquetas a las páginas e inserte como clase en

Puede agregar este código a su archivo functions.php personalizado:

```

// add tags and categories to pages

function add_taxonomies_to_pages() {
    register_taxonomy_for_object_type( 'post_tag', 'page' );
    register_taxonomy_for_object_type( 'category', 'page' );
}
add_action( 'init', 'add_taxonomies_to_pages' );

if ( ! is_admin() ) {
    add_action( 'pre_get_posts', 'category_and_tag_archives' );
}

function category_and_tag_archives( $wp_query ) {
    $my_post_array = array('post','page');

    if ( $wp_query->get( 'category_name' ) || $wp_query->get( 'cat' ) )
        $wp_query->set( 'post_type', $my_post_array );

    if ( $wp_query->get( 'tag' ) )
        $wp_query->set( 'post_type', $my_post_array );
}

// add tags and categories as class to <body>

function add_categories_and_tags( $classes = '' ) {
    if( is_page() ) {
        $categories = get_the_category();
        foreach( $categories as $category ) {
            $classes[] = 'category-'. $category->slug;
        }
    }
}

```

```
$tags = get_the_tags();
foreach( $tags as $tag ) {
    $classes[] = 'tag-'. $tag->slug;
}
}
return $classes;
}
add_filter( 'body_class', 'add_categories_and_tags' );
```

Funciona perfecto, probado en **WordPress 4.8**.

Lea **Taxonomias en línea**: <https://riptutorial.com/es/wordpress/topic/5943/taxonomias>

Capítulo 64: Tema de WordPress y desarrollo de temas infantiles.

Introducción

Wordpress es un CMS ampliamente utilizado para crear sitios web de información simple, pero también para crear sitios web más sofisticados e incluso pequeñas tiendas web.

Wordpress hace uso de temas. Estos temas se utilizan para crear la funcionalidad de diseño y contenido de un sitio web de Wordpress. Los temas se pueden encontrar en todo el internet.

Cada uno tiene su propia funcionalidad y diseño únicos, pero a veces es difícil encontrar el tema adecuado para un sitio web. Por suerte también somos capaces de crear nuestro propio tema.

Examples

Desarrollando tu propio tema

Un tema de wordpress consiste en dos tipos de archivos. Los archivos básicos que tiene cada tema y los archivos que definen el diseño y la funcionalidad del tema. Este segundo grupo voy a llamar a los archivos específicos del tema.

Los archivos de temas básicos.

Los archivos de temas básicos son los archivos que se utilizan para configurar y registrar un tema. En la siguiente lista, describiré brevemente cada archivo y su uso. Más adelante agregaré los archivos de ejemplo más básicos que se necesitan para configurar su propio tema de wordpress.

- `functions.php` : el archivo `functions.php` se usa para registrar todas las funciones, barras laterales, scripts e inclusiones del tema. En este archivo puede, por ejemplo, incluir archivos CSS, archivos JS, etc.
- `Header and footer` : los archivos de encabezado y pie de página (`header.php` y `footer.php`) son los archivos que se utilizan para llamar al encabezado y al pie de página. El archivo de encabezado y pie de página, por ejemplo, contiene el enlace al sistema de fondo de wordpress.
- `index.php` : el archivo `index.php` es el archivo que crea la plantilla de página predeterminada. En este archivo puede ver, editar y eliminar partes de este diseño de plantilla predeterminado.
- `single.php` : el archivo `single.php` es el archivo que crea la página de la plantilla de publicaciones individuales. Al igual que la plantilla predeterminada para las páginas, pero ahora para las páginas de una sola publicación.
- `format.php` El archivo `format.php` es el archivo que crea la plantilla de texto de contenido desde una página. Entonces, si tuviera una página de inicio y la editaría desde el back-end agregando un texto. Este archivo crea el marcado estándar de este texto.

- `404.php` El archivo `404.php` crea la plantilla 404. Este archivo consiste en el diseño básico de esta página.
- `archive.php` El archivo `archive.php` crea el diseño de la página de archivo.
- `style.css` El archivo básico de hojas de estilo.

Así que en esta lista puede ver todos los archivos **necesarios** para configurar su propio tema de Wordpress. Ahora echemos un vistazo a algunos archivos que puede crear si desea, pero **no** son archivos **necesarios** para un tema de wordpress. Estos archivos son en su mayoría archivos de plantilla y otras extensiones funcionales.

Plantillas de página personalizadas

`page-<your own name>.php` : en un tema de Wordpress puede crear varias plantillas de página. creando nuevos archivos de plantilla de página. Un archivo de plantilla de página estándar consta de los siguientes atributos de nombre. `name of the template` `page name of the template` y `.php` Si, por ejemplo, desea crear una nueva plantilla de página para la página de su blog, podría llamarlo `page-blog.php` Wordpress lee automáticamente el archivo y agrega el archivo al menú de elegir plantilla. Asegúrese de que al menos haya incluido las `get_header()` y `get_footer()` . También asegúrese de nombrar su plantilla en un comentario en la parte superior del archivo agregando el siguiente ejemplo.

```
<?php
/*
 * Template Name: Homepage Template
 */
get_header();
?>
```

Plantillas de página de entrada única personalizadas

`single-<your own name>.php` : en un tema de Wordpress como la plantilla de página descrita anteriormente, también puede crear sus propias plantillas de página de publicaciones individuales. Al igual que la plantilla de página, el archivo consta de tres partes `single` para declarar que es una página de publicación única `<your name of the template>` y la extensión de archivo `.php` . Al igual que la plantilla de la página, los requisitos mínimos para asegurarse de que Wordpress lea la nueva plantilla están agregando las funciones `get_header()` y `get_footer()` . Y, por supuesto, también agrega el nombre de su plantilla como el ejemplo a continuación.

```
<?php
/*
 * Template Name: Post Portfolio
 * Template Post Type: post, page
 */
?>
```

También indicamos el `Template post type:` que representa el tipo de plantilla que es, en este caso, publicación y página.

Plantillas de texto personalizadas

`format -<your own name>.php` : en un tema de Wordpress también puede crear plantillas de salida.

Estas plantillas de formato son el diseño y el contenido de una publicación. Por ejemplo, si en algunos casos desea que la publicación muestre solo el contenido o el título de la publicación, puede usar estas plantillas para crear ese tipo de ajustes. Dado que este tipo de plantillas solo dan formato al contenido de back-ends de publicaciones creado por un usuario, no es necesario que `get_header()` y `get_footer()` ya que estas ya están definidas en las plantillas de páginas. Asegúrese de que su plantilla sea capaz de reconocer una publicación utilizando el siguiente ejemplo básico.

```
<div>
  <article id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
  </article>
</div>
```

Ahora que sabemos algo sobre los archivos básicos y algunos de los muchos archivos de plantillas específicas, es hora de comenzar a hablar de barras laterales y widgets. En el futuro, esto se agregará junto con un tutorial de inicio en el paso a paso sobre cómo crear un tema de Wordpress muy propio.

Lea Tema de WordPress y desarrollo de temas infantiles. en línea:

<https://riptutorial.com/es/wordpress/topic/9940/tema-de-wordpress-y-desarrollo-de-temas-infantiles->

Capítulo 65: Temas

Introducción

Los temas de WordPress son el front-end de su sitio web. Son lo que la gente ve cuando visitan el sitio. Hay miles de temas para elegir, versiones pagas y gratuitas. Incluso puede crear su propio tema personalizado con solo un par de archivos necesarios.

Examples

Temas de WordPress

Cómo elegir un tema

Cada instalación de WordPress viene con un tema preinstalado. Manejas tus temas desde el Dashboard. Vaya a Apariencia > Temas para instalar, previsualizar, eliminar, activar y actualizar temas. El tema actual se encuentra en la esquina superior izquierda de este menú.

Al pasar el cursor sobre la imagen del tema, aparece el botón "Detalles del tema". Este botón proporciona información sobre el tema, como la versión y la descripción. Al hacer clic en la imagen del tema actual, podrá acceder a la configuración de temas específicos, como el título.

Actualización disponible

Si hay actualizaciones disponibles para los temas instalados, encontrará un mensaje que le informa que hay una nueva versión disponible. Debería poder ver los detalles de la nueva versión o actualizar ahora.

- Ver detalles de la versión

Al hacer clic en el enlace de detalles de la versión, lo llevará a una página del Directorio de temas de WordPress. Aquí encontrarás los detalles de la versión de actualización.

- Actualizar ahora

Al hacer clic en el enlace Actualizar ahora se instalará la actualización del tema. Los temas también se pueden actualizar desde la pantalla Administración > Panel de control > Actualizaciones.

Además de su tema actual, la pantalla Administrar temas también muestra los otros temas que están instalados pero que actualmente están inactivos. Cada tema está representado por una pequeña captura de pantalla. Al pasar el cursor sobre estas imágenes, se muestran los botones 'Detalles del tema', 'Activar' y 'Vista previa en vivo'. También podrás actualizar o eliminar temas inactivos de esta página. Cada página en esta pantalla mostrará hasta 15 capturas de pantalla de Temas a la vez.

- Activar

Al hacer clic en este enlace, este es el tema actual.

- Vista previa en vivo

Al hacer clic en este enlace, se muestra una vista previa del blog con esta versión específica del tema.

- Borrar

Al hacer clic en este enlace, se elimina completamente este tema e incluye todos los archivos y carpetas del tema. Todo lo que no esté respaldado se perderá para siempre.

- Actualización disponible

Consulte la sección de actualización disponible más arriba.

Instalar temas

A continuación se enumeran varias formas de instalar Temas:

- Instalador automatizado de temas

Esto se puede usar para instalar Temas del Directorio de Temas de WordPress. Vaya a Administración> Aspecto> Temas para encontrar la pantalla de Temas de aspecto. Haga clic en el botón Agregar nuevo. Desde aquí encontrarás Temas para usar que están libres de cambio. En la parte superior de esta pantalla hay una función de búsqueda con tres métodos disponibles para encontrar un nuevo tema; Búsqueda de filtros, palabras clave y atributos.

- Usando el método de carga

El método de carga instala un tema a través de un archivo ZIP. Todos los temas en el directorio de temas de WordPress se pueden instalar de esta manera. Después de descargar el archivo ZIP, visite Administración> Aspecto> Temas y haga clic en el botón Agregar nuevo. A continuación, haga clic en el enlace Cargar tema. Busque el archivo ZIP y haga clic en Instalar ahora. Para terminar de hacer de este el tema actual, haga clic en el enlace Activar.

- Usando El Método FTP

Para instalar un tema con el método FTP, primero debe descargar los archivos del tema en su computadora local. Extraiga el contenido del archivo ZIP, conservando la estructura del archivo y agréguelos a una nueva carpeta. Si hay instrucciones del autor del tema, asegúrese de seguirlas.

Use un cliente FTP para acceder al servidor web de su sitio. Agregue los archivos de tema cargados a su directorio wp-content / themes proporcionado por WordPress. Si es

necesario, cree una carpeta para contener su nuevo Tema dentro del directorio wp-content / themes. Un ejemplo de esto sería, si su tema se llama Prueba, debería estar en wp-content / themes / test.

Vaya a Administración> Aspecto> Temas y haga clic en el enlace Activar para seleccionar el tema como su tema actual.

- Instalación con cPanel

Los paneles de control de cPanel ofrecen otro método para instalar Temas con archivos ZIP o GZ. En el Administrador de cPanel, si WordPress está instalado, vaya a la carpeta Temas. La ruta sería similar a 'public_html / wp-content / themes'. Haga clic en Cargar archivo (s) y cargue el archivo ZIP. Seleccione el archivo ZIP en cPanel y haga clic en Extraer contenido del archivo en el panel a la derecha para descomprimir ese archivo.

Vaya a Administración> Aspecto> Temas y haga clic en el enlace Activar para seleccionar el tema como su tema actual.

Toda la información enumerada anteriormente es de acuerdo con el Códice de WordPress. Se ha acortado por brevedad. El material fuente original se puede encontrar [aquí](#) . O para más información visite codex.wordpress.org .

Creación de un tema personalizado

Estas instrucciones crean un tema de WordPress muy básico y que cumple con los estándares mínimos.

El primer paso es crear una nueva carpeta de temas dentro de su directorio de temas de WordPress. La ruta correcta será:> wp-content> temas> Para crear un tema válido, los temas de WordPress requieren que al menos estos dos archivos se encuentren allí:

- index.php
- style.css

Su hoja de estilo debe contener un comentario que avise a WordPress de que existe un tema aquí.

```
/*
Theme Name: <theme name>
Author: <author name>
Description: <description goes here>
Version: <theme version #>
Tags: <tag to id theme>
*/
```

Tu tema ya ha sido creado. Vaya al panel de WordPress y haga clic en Apariencia> Temas, para activarlo.

Lea Temas en línea: <https://riptutorial.com/es/wordpress/topic/8967/temas>

Capítulo 66: `template_include`

Parámetros

Parámetro	Explicación
<code>\$template</code>	Pasa un parámetro al filtro, <code>\$template</code> es la ruta actual al archivo apropiado para el tipo de publicación tal como se encuentra en el tema secundario activo o en el tema principal (si no hay un tema secundario en su lugar o el tema secundario tiene plantillas de rango inferior. Ver jerarquía de plantillas de WordPress para más detalles).

Observaciones

Debe devolver `$template` incluso si no está modificando. Si esto te confunde, mira los ejemplos donde se ha utilizado `apply_filter()` en el código

No debe configurar variables aquí para usarlas más adelante, hay mejores ganchos para esto.

Un flujo de programa útil para este filtro es:

1. Check `$template` incluye nuestro nombre de tipo de publicación personalizado -> jerarquía de plantillas !!
2. si no, busque en nuestro complemento los archivos adecuados -> Es mejor apuntar a archivos específicos en lugar de buscar archivos en las carpetas. Más eficiente. Pero completamente a la altura del desarrollador.
3. devuelve la plantilla.

Examples

Ejemplo simple

Este filtro es muy útil. Uno de los problemas comunes para los desarrolladores es cómo incluir plantillas en los complementos que desarrollan.

El filtro se aplica inmediatamente después de que wordpress ubique la plantilla apropiada en el tema secundario / padre activo usando la jerarquía wp.

Tenga cuidado de definir cuándo desea modificar la ruta de la plantilla. En el siguiente ejemplo, el código verifica si la página actual es la vista única de nuestro tipo de publicación personalizada `cpt`.

Ejemplo suficientemente simple para empezar!

```

add_filter('template_include', 'custom_function');

function custom_function($template){

    //change a single post template...

    if( is_singular('cpt') ){
        $template= 'path/to/another/template_file';
    }

    return $template;

}

```

Más ejemplo de Adv

```

add_filter('template_include', 'custom_function');

function custom_function($template){

    /*
    *   This example is a little more advanced.
    *   It will check to see if $template contains our post-type in the path.
    *   If it does, the theme contains a high level template e.g. single-cpt.php
    *   If not we look in the plugin parent folder for the file. e.g. single-cpt.php
    */

    //check to see if the post type is in the filename (high priority file)
    //return template if it is!

    global $post;

    if( strpos($template, 'single-'. $post->post_type.'.php' ) !== false && strpos($template,
'archive-'. $post->post_type.'.php' ) !== false ){
        return $template;
    }

    $plugin_path = 'var/etc/wp-content/plugins/plugin'; //include own logic here...

    if( is_singular($post->post_type) && file_exists($plugin_path.'/single-'. $post->
post_type.'.php' ) ){
        $template= $plugin_path.'/single-'. $post->post_type.'.php';
    } elseif ( is_archive($post->post_type) && file_exists($plugin_path.'/archive-'. $post->
post_type.'.php' ) ) {
        $template= $plugin_path.'/archive-'. $post->post_type.'.php';
    }

    return $template;

}

```

Lea `template_include` en línea: <https://riptutorial.com/es/wordpress/topic/1439/template-include>

Capítulo 67: Tipos de correos personalizados

Sintaxis

- `register_post_type` (\$ post_type, \$ args);

Parámetros

Parámetro	Detalles
\$ post_type	(cadena) (Requerido)
\$ args	(array / string) (Opcional)

Examples

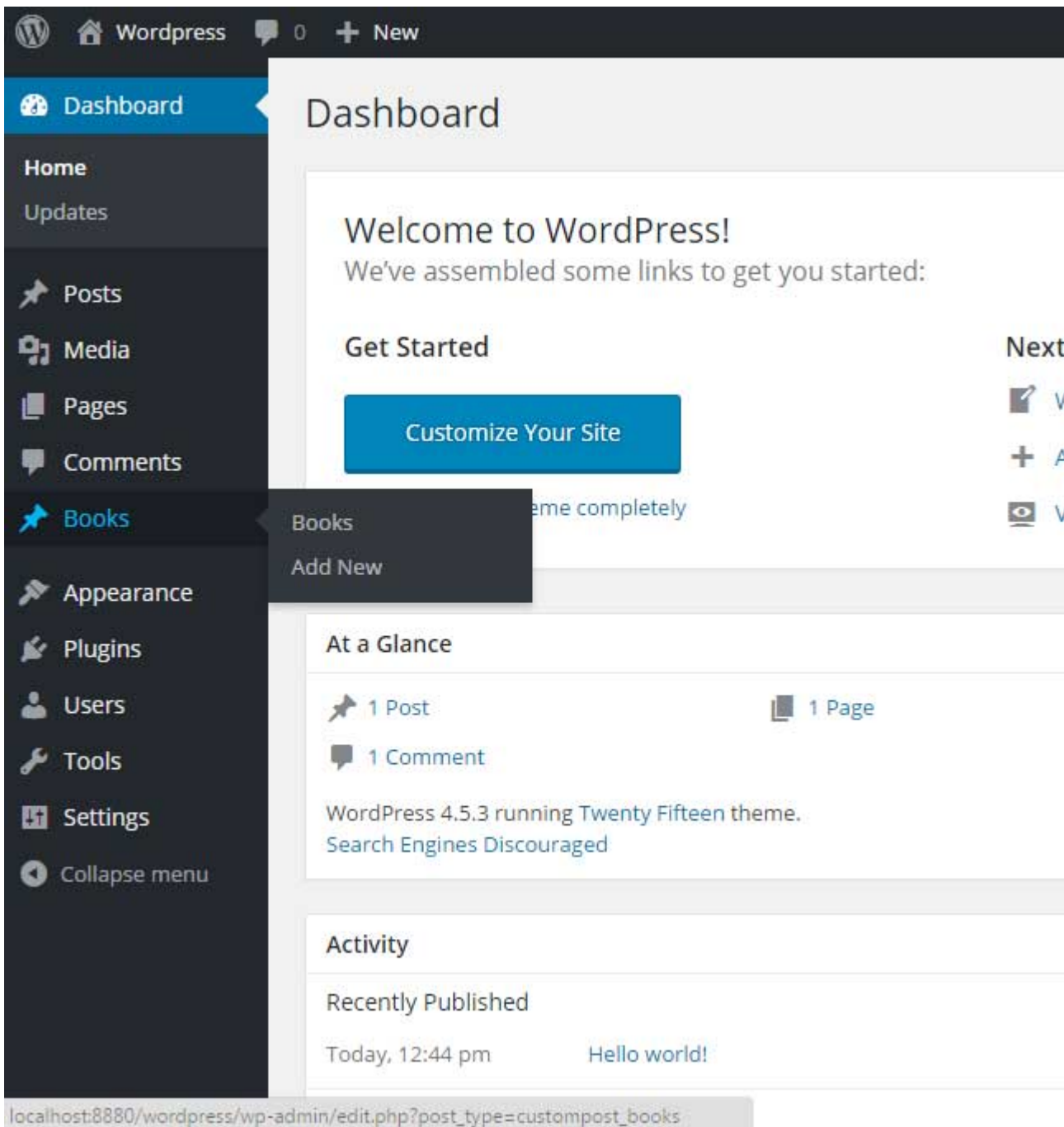
Registro de un tipo de mensaje personalizado

Supongamos que tiene un sitio web de biblioteca y desea tener un tipo de publicación personalizado llamado *Libros* . Se puede registrar como

```
function create_bookposttype() {
    $args = array(
        'public' => true,
        'labels' => array(
            'name' => __( 'Books' ),
            'singular_name' => __( 'Book' )
        ),
    );
    register_post_type( 'custompost_books', $args );
}

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

y, por más simple que sea, ahora tiene un tipo de publicación personalizado registrado.



Este fragmento de código se puede colocar en su archivo de `functions.php` tema.php, o dentro de una estructura de complemento.

Agregar tipos de publicaciones personalizados a la consulta principal

Registrar un tipo de publicación personalizada no significa que se agregue a la consulta principal automáticamente. `pre_get_posts` usar el filtro `pre_get_posts` para agregar tipos de publicaciones personalizadas a la consulta principal.

```
// Show posts of 'post' and 'book' custom post types on home page
add_action( 'pre_get_posts', 'add_my_post_types_to_query' );

function add_my_post_types_to_query( $query ) {
    if ( is_home() && $query->is_main_query() )
        $query->set( 'post_type', array( 'post', 'book' ) );
}
```

```
return $query;
}
```

Agregar tipos de publicaciones personalizadas a la fuente RSS principal

Registrar un tipo de publicación personalizada no significa que se agregue a la fuente RSS principal automáticamente. Debe usar el filtro de `request` para agregar tipos de publicación personalizados a la fuente RSS principal.

```
// Add 'books' custom post types on main RSS feed
function add_book_post_types_to_rss($qv) {
    if (isset($qv['feed']) && !isset($qv['post_type']))
        $qv['post_type'] = array('post', 'books', );
    return $qv;
}
add_filter('request', 'add_book_post_types_to_rss');
```

Registrar tipo de mensaje personalizado

```
if ( ! function_exists('products_post_type') ) {

function products_post_type() {

    $labels = array(
        'name'                => _x( 'Products', 'Post Type General Name', 'text_domain' ),
        'singular_name'       => _x( 'Product', 'Post Type Singular Name', 'text_domain' ),
        'menu_name'          => __( 'Products', 'text_domain' ),
        'name_admin_bar'     => __( 'Product', 'text_domain' ),
        'archives'           => __( 'Item Archives', 'text_domain' ),
        'attributes'         => __( 'Item Attributes', 'text_domain' ),
        'parent_item_colon'  => __( 'Parent Product:', 'text_domain' ),
        'all_items'          => __( 'All Products', 'text_domain' ),
        'add_new_item'       => __( 'Add New Product', 'text_domain' ),
        'add_new'            => __( 'New Product', 'text_domain' ),
        'new_item'           => __( 'New Item', 'text_domain' ),
        'edit_item'          => __( 'Edit Product', 'text_domain' ),
        'update_item'        => __( 'Update Product', 'text_domain' ),
        'view_item'          => __( 'View Product', 'text_domain' ),
        'view_items'         => __( 'View Items', 'text_domain' ),
        'search_items'       => __( 'Search products', 'text_domain' ),
        'not_found'          => __( 'No products found', 'text_domain' ),
        'not_found_in_trash' => __( 'No products found in Trash', 'text_domain' ),
        'featured_image'     => __( 'Featured Image', 'text_domain' ),
        'set_featured_image' => __( 'Set featured image', 'text_domain' ),
        'remove_featured_image' => __( 'Remove featured image', 'text_domain' ),
        'use_featured_image' => __( 'Use as featured image', 'text_domain' ),
        'insert_into_item'   => __( 'Insert into item', 'text_domain' ),
        'uploaded_to_this_item' => __( 'Uploaded to this item', 'text_domain' ),
        'items_list'         => __( 'Items list', 'text_domain' ),
        'items_list_navigation' => __( 'Items list navigation', 'text_domain' ),
        'filter_items_list' => __( 'Filter items list', 'text_domain' ),
    );

    $args = array(
        'label'                => __( 'Product', 'text_domain' ),
        'description'          => __( 'Product information pages.', 'text_domain' ),
        'labels'               => $labels,
    );
}
```

```

        'supports'                => array( 'title', 'editor', 'excerpt', 'author', 'thumbnail',
'comments', 'custom-fields', ),
        'taxonomies'              => array( 'category', 'post_tag' ),
        'hierarchical'            => false,
        'public'                  => true,
        'show_ui'                  => true,
        'show_in_menu'             => true,
        'menu_position'           => 5,
        'menu_icon'                => 'dashicons-products',
        'show_in_admin_bar'        => true,
        'show_in_nav_menus'        => true,
        'can_export'               => true,
        'has_archive'              => true,
        'exclude_from_search'      => false,
        'publicly_queryable'       => true,
        'capability_type'          => 'page',
        'show_in_rest'             => true,
    );
    register_post_type( 'product', $args );

}
add_action( 'init', 'products_post_type', 0 );

}

```

Tipo de publicación personalizada utilizando Twenty Fifteen WordPress Theme

Puedes usar cualquier nombre para la función.

```

function custom_posttype(){
    register_post_type('cus_post',array(

        'labels'=>array(
            'name'=>'khaiyam'// Use any name you want to show in menu for your users
        ),
        'public'=>true,// **Must required
        'supports'=>array('title','editor','thumbnail')// Features you want to provide on your
posts
    ));
}
add_action('after_setup_theme','custom_postytp');

```

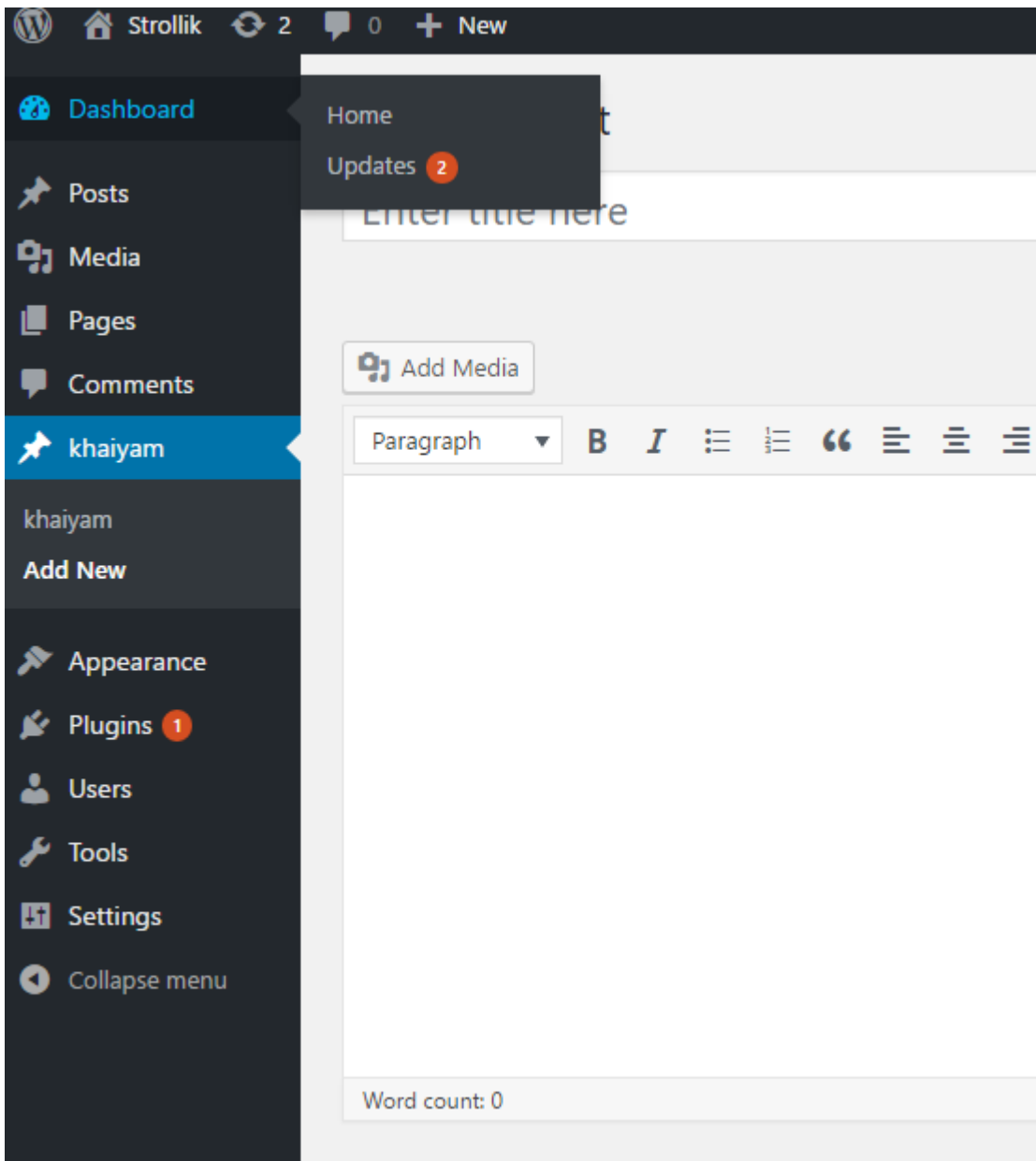
0

```

add_action('init','custom_postytp');

```

Puede usar cualquiera de los ganchos que desee pero, por supuesto, tienen diferentes significados y usos.



Tipo de publicación personalizada en la búsqueda por defecto

Puede agregar publicaciones de tipo de publicación personalizadas en la búsqueda de wordpress predeterminada. Agregue el siguiente código en las funciones del tema.php

```
function my_search_filter($query) {
    if ( !is_admin() && $query->is_main_query() ) {
        if ($query->is_search) {
            $query->set('post_type', array( 'news','post','article' ) );
        }
    }
}
add_action('pre_get_posts','my_search_filter');
```

Lea Tipos de correos personalizados en línea:

<https://riptutorial.com/es/wordpress/topic/1374/tipos-de-correos-personalizados>

Capítulo 68: wp_get_current_user ()

Sintaxis

- wp_get_current_user ()

Examples

Obteniendo el usuario actual

Obtención de toda la información del usuario actual en wordpress usando la función predefinida wp_get_current_user ();

```
<?php

$current_user = wp_get_current_user();

echo "Username : ".$current_user->user_login;
echo "Username : ".$current_user->ID;
echo "Username : ".$current_user->user_pass;
echo "Username : ".$current_user->user_nicename;
echo "Username : ".$current_user->user_email;
echo "Username : ".$current_user->user_url;
echo "Username : ".$current_user->user_registered;
echo "Username : ".$current_user->user_activation_key;
echo "Username : ".$current_user->user_status;
echo "Username : ".$current_user->display_name;

?>
```

Utilice el bucle foreach para obtener información de usuario de wp_get_current_user ()

```
$user = wp_get_current_user();

foreach($user->data as $key=>$user_data){
    if($key == 'user_pass' || $key == 'user_activation_key' || $key=='user_status'){
    }
    else{
        $nice_key = ucfirst(str_replace('_', ' ', $key));

        if($key == 'user_registered'){
            $user_data = date_i18n(get_option('date_format'), strtotime($user_data));
        }

        echo $nice_key . ' : ' . $user_data . '<br />';
    }
}
```

Lea wp_get_current_user () en línea: <https://riptutorial.com/es/wordpress/topic/2693/wp-get-current-user--->

Capítulo 69: wp_get_current_user ()

Examples

Obtener información actual de usuario registrado

Recupera la información correspondiente al usuario que ha iniciado sesión actualmente y la coloca en la variable global \$ current_user

Esta función no acepta ningún parámetro.

Uso:

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

Ejemplo:

```
<?php
$current_user = wp_get_current_user();

echo 'Username: ' . $current_user->user_login . "\n";
echo 'User email: ' . $current_user->user_email . "\n";
echo 'User level: ' . $current_user->user_level . "\n";
echo 'User first name: ' . $current_user->user_firstname . "\n";
echo 'User last name: ' . $current_user->user_lastname . "\n";
echo 'User display name: ' . $current_user->display_name . "\n";
echo 'User ID: ' . $current_user->ID . "\n";
?>
```

Para determinar si un visitante ha iniciado sesión o no, puede usar is_user_logged_in () antes, luego obtenga la información del usuario actual si el visitante ha iniciado sesión:

```
<?php
if ( is_user_logged_in() ) {
    $current_user = wp_get_current_user();

    echo 'Welcome, ' . $current_user->user_login . '!';
} else {
    echo 'Welcome, visitor!';
}
?>
```

Lea wp_get_current_user () en línea: <https://riptutorial.com/es/wordpress/topic/6649/wp-get-current-user--->

Capítulo 70: WP_Query () Loop

Introducción

WP_Query para consultar publicaciones, páginas y tipos de publicaciones personalizadas. Obtendrá una lista para publicaciones y páginas específicas o tipos de publicaciones personalizadas. WP_Query le permite extraer publicaciones de la base de datos según sus criterios.

Examples

Recuperar las últimas 10 publicaciones

```
$args = array(
    'post_type'=>'post',
    'posts_per_page' =>'10'
);
$latest_posts_query = new WP_Query( $args );
if($latest_posts_query->have_posts()) :
while ( $latest_posts_query-> have_posts() ) : $latest_posts_query->the_post();
//Get post details here
endwhile;
endif;
```

Lea WP_Query () Loop en línea: <https://riptutorial.com/es/wordpress/topic/8301/wp-query----loop>

Capítulo 71: WP-CLI

Introducción

WP-CLI es un conjunto de herramientas de línea de comandos para administrar instalaciones de WordPress. Puede actualizar complementos, configurar instalaciones en múltiples sitios y mucho más, sin utilizar un navegador web.

Examples

Gestionar temas

Obtener una lista de temas.

```
$ wp theme list
```

Instala la última versión de wordpress.org y activa

```
$ wp theme install twentysixteen --activate
```

Instalar desde un archivo zip local

```
$ wp theme install ../my-theme.zip
```

Instalar desde un archivo zip remoto

```
$ wp theme install http://s3.amazonaws.com/bucketname/my-theme.zip?AWSAccessKeyId=123&Expires=456&Signature=abcdef
```

Obtener detalles de un tema instalado

```
$ wp theme get twentysixteen --fields=name,title,version
```

Obtener el estado del tema

```
$ wp theme status twentysixteen
```

Administrar complementos

Obtener una lista de complementos

```
$ wp plugin list
```

Lista de complementos activos en el sitio.

```
$ wp plugin list --status=active --format=json
```

Lista de complementos en cada sitio en una red.

```
$ wp site list --field=url | xargs -I % wp plugin list --url=%
```

Activar plugin

```
$ wp plugin activate hello-dolly
```

Desactivar el plugin

```
$ wp plugin deactivate hello-dolly
```

Eliminar plugin

```
$ wp plugin delete hello-dolly
```

Instala la última versión de wordpress.org y activa

```
$ wp plugin install bbpress --activate
```

Administrar el propio WP-CLI

Muestra la versión actualmente instalada.

```
$ wp cli version
```

Compruebe las actualizaciones de WP-CLI.

```
$ wp cli check-update
```

Actualice WP-CLI a la última versión estable.

```
$ wp cli update
```

Lista todos los alias disponibles.

```
$ wp cli alias
```

Imprima varios detalles sobre el entorno WP-CLI.

```
$ wp cli info
```

Volcar la lista de comandos instalados, como JSON.

```
$ wp cli cmd-dump
```

Descarga, instala, actualiza y gestiona una instalación de WordPress.

Descargar WordPress Core

```
$ wp core download --locale=nl_NL
```

Instala WordPress

```
$ wp core install --url=example.com --title=Example --admin_user=supervisor --  
admin_password=strongpassword --admin_email=info@example.com
```

Mostrar la versión de WordPress

```
$ wp core version
```

Transformar una instalación de un solo sitio en una instalación multisitio de WordPress.

```
$ wp core multisite-convert
```

Instalar WordPress multisite desde cero.

```
$ wp core multisite-install
```

Gestionar usuarios

Lista de ID de usuario

```
$ wp user list --field=ID
```

Crea un nuevo usuario.

```
$ wp user create bob bob@example.com --role=author
```

Actualizar un usuario existente.

```
$ wp user update 123 --display_name=Mary --user_pass=marypass
```

Eliminar usuario 123 y reasignar publicaciones al usuario 567

```
$ wp user delete 123 --reassign=567
```

Realice operaciones básicas de la base de datos utilizando las credenciales almacenadas en wp-config.php

Crear una nueva base de datos.

```
$ wp db create
```

Soltar una base de datos existente.

```
$ wp db drop --yes
```

Restablecer la base de datos actual.

```
$ wp db reset --yes
```

Ejecutar una consulta SQL almacenada en un archivo.

```
$ wp db query < debug.sql
```

Lea WP-CLI en línea: <https://riptutorial.com/es/wordpress/topic/9169/wp-cli>

Capítulo 72: WP-Cron

Examples

Ejemplo de wp_schedule_event ()

```
// register activation hook
register_activation_hook( __FILE__, 'example_activation' );

// function for activation hook
function example_activation() {
    // check if scheduled hook exists
    if ( !wp_next_scheduled( 'my_event' ) ) {
        // Schedules a hook
        // time() - the first time of an event to run ( UNIX timestamp format )
        // 'hourly' - recurrence ('hourly', 'twicedaily', 'daily' )
        // 'my_event' - the name of an action hook to execute.
        wp_schedule_event( time(), 'hourly', 'my_event' );
    }
}

add_action( 'my_event', 'do_this_hourly' );

// the code of your hourly event
function do_this_hourly() {
    // put your code here
}

// register deactivation hook
register_deactivation_hook( __FILE__, 'example_deactivation' );

// function for deactivation hook
function example_deactivation() {
    // clear scheduled hook
    wp_clear_scheduled_hook( 'my_event' );
}
```

Importante: el cron de WordPress se ejecuta solo cuando se llega a alguna página de su sitio web. Por lo tanto, para un sitio web con poco tráfico, debe configurar el cron en su alojamiento para llegar a las páginas.

intervalo de repetición personalizado en wp_schedule_event ()

```
// this function add custom interval (5 minutes) to the $schedules
function five_minutes_interval( $schedules ) {
    $schedules['five_minutes'] = array(
        'interval' => 60 * 5,
        'display'  => '5 minutes';
    );
    return $schedules;
}

// add a custom interval filter
add_filter( 'cron_schedules', 'five_minutes_interval' );
```

```
// Schedules a hook  
wp_schedule_event( time(), 'five_minutes', 'my_event' );
```

Lea WP-Cron en línea: <https://riptutorial.com/es/wordpress/topic/6783/wp-cron>

Creditos

S. No	Capítulos	Contributors
1	Empezando con WordPress	4444 , A. Raza , Andrew , animuson , Anupam , Chris Fletcher , Ciprian , Community , Florida , James Jones , JonasCz , Leo F , Marc St Raymond , Mayank Gupta , Milap , nus , Panda , rap-2-h , Seth C. , Shubham , Trevor Clarke , vajrasar
2	¿Cómo puedo integrar el editor de Markdown con el complemento del repetidor del campo personalizado avanzado?	Fatbit
3	Acciones y Filtros	David , Ihor Vorotnov , Mrinal Haque , Trying Tobemyself
4	Actualizar WordPress manualmente	KnightHawk
5	<code>add_action ()</code>	Abel Melquiades Callejo , Waqas Bukhary
6	<code>add_editor_style ()</code>	Gabriel Chi Hong Lee
7	<code>add_menu_page ()</code>	brasofilo , Gabriel Chi Hong Lee
8	<code>add_submenu_page ()</code>	Gabriel Chi Hong Lee , theoretisch
9	<code>add_theme_support ()</code>	Gabriel Chi Hong Lee
10	Admin Dashboard Widgets	theoretisch
11	Agregar / eliminar información de contacto para usuarios con gancho de filtro <code>user_contactmethods</code>	Petar Popovic , RamenChef
12	AJAX	Andy , Digvijayad , Gaurav Srivastava , GreatBlakes , Nisarg Patel

		, Ruslan Murarov , stweb
13	Añadir Shortcode	purvik7373 , RamenChef
14	API de opciones	Harshal Limaye , Pat J , RamenChef
15	API REST	Picard
16	Asegure su instalación	James Jones
17	Barras laterales	dingo_d , Kushal Shah , theoretisch
18	Bucle principal alternante (filtro pre_get_posts)	Dawid Urbanski , Petar Popovic
19	Código corto	Ad Wicks , Adam Genshaft , brasofilo , John Slegers , Kylar , Shashank Agarwal
20	Código corto con atributo	Digvijayad , Firefog , RamenChef
21	Códigos cortos	Pelmered
22	Conceptos básicos del personalizador (Agregar panel, Sección, Configuración, Control)	4444 , Ahmad Awais , RamenChef
23	Consulta de mensajes	dingo_d
24	Creación de plugins de WordPress	Seth C.
25	Creando una plantilla personalizada	Petar Popovic
26	Crear plantilla para tipo de publicación personalizada	Ashok G , Egnaro , Joe Dooley , mnoronha
27	Crear una publicación programáticamente	RamenChef , Roel Magdaleno , RRikesh
28	Depuración	barbocc , dingo_d , jgraup

29	Desarrollo de plugins	Angle.R , Ping.Chen
30	Ejecutar WordPress local con XAMPP	Pierre.Vriens , theoretisch
31	El Loop (bucle principal de WordPress)	anik4e , Dawid Urbanski
32	El objeto \$wpdb	Kushal Shah , mcon , stweb
33	el título()	Gabriel Chi Hong Lee
34	Eliminar saltos de línea automáticos del contenido y extracto	Austin Winstanley
35	Eliminar versión de Wordpress y hojas de estilo	jay.jivani , mnoronha , theoretisch
36	en eso	Abel Melquiades Callejo , barbocc
37	Encolando guiones	dingo_d , Harshal Limaye , J.D. , mbacon40 , montrealist , Pelmered , Petar Popovic
38	Estilos de puesta en cola	dingo_d , Harshal Limaye , Laxmana , mnoronha , montrealist , Petar Popovic , RamenChef , Ruslan Murarov , virtualLast
39	Extractos personalizados con excerpt_length y excerpt_more	inkista , Petar Popovic , RamenChef
40	Función: add_action ()	bosco , RamenChef
41	Función: wp_trim_words ()	Harshal Limaye
42	Fundamentos del tema infantil	Andrei , Razvan Onofrei , Vlad Olaru
43	get_bloginfo ()	Abel Melquiades Callejo , Harshal Limaye , HeyCameron , KenB , Nate Beers , RamenChef , Tom J Nowell , virtualLast , Wes Moberly
44	get_home_path ()	Ihor Vorotnov
45	get_option ()	Gabriel Chi Hong Lee

46	<code>get_permalink ()</code>	Gabriel Chi Hong Lee
47	<code>get_template_part ()</code>	Dan Devine , Kushal Shah
48	<code>get_the_category ()</code>	Gabriel Chi Hong Lee
49	<code>get_the_title ()</code>	Gabriel Chi Hong Lee
50	Hacer solicitudes de red con API HTTP	Jordan , mjangda , Rarst
51	<code>home_url ()</code>	dingo_d , Kushal Shah , matthew , Mr. Developer
52	Instalacion y configuracion	Kenyon , Marco Romano , Ping.Chen , S.L. Barth , stig-js , theoretisch , Yuan Lung Luo
53	Jerarquía de plantillas	jgraup , MarZab , Pelmered , theoretisch
54	La barra de administración (también conocida como "La barra de herramientas")	dingo_d , Harshal Limaye , JCL1178 , Kushal Shah
55	Meta Box	Austin Winstanley
56	Migración del sitio	Austin Winstanley
57	Personalizador Hello World	Dan Green-Leipciger
58	Post Formatos	Shashank Agarwal
59	Seguridad en WordPress - Escape	Laxmana , the4kman
60	Seguridad en WordPress - Sanitización	Laxmana
61	Taxonomias	adifatz , Kushal Shah , purvik7373
62	Tema de WordPress y desarrollo de temas infantiles.	Deathstorm
63	Temas	Jef
64	<code>template_include</code>	Abel Melquiades Callejo , David , RamenChef

65	Tipos de correos personalizados	Caio Felipe Pereira , Dan Devine, J.D. , janw , jgraup , Kushal Shah , Omar Khaiyam , Ranuka , theoretisch
66	<code>wp_get_current_user()</code>	Abel Melquiades Callejo , Benoti , Muhammad Farrukh Faizy
67	WP_Query () Loop	vrajesh
68	WP-CLI	jgraup
69	WP-Cron	stweb