Capítulo 4
261

1. Funciones

1.1 Introducción

Al igual que en otros lenguajes de programación, PHP ofrece la posibilidad de definir sus propias funciones (llamadas funciones del "usuario") con todas las ventajas asociadas (modularidad, uso de mayúsculas...). Una función es un conjunto de instrucciones identificadas por un nombre, cuya ejecución devuelve un valor y cuya llamada se puede utilizar como operando en una expresión. Un procedimiento es un conjunto de instrucciones identificadas por un nombre que puede ser llamado como una instrucción.

1.2 Declaración y llamada

La palabra clave `función` permite introducir la definición de una función.

Sintaxis

```
función nombre_función([parámetro]) [: tipo]{
  instrucciones;
}
```

`nombre_función` Nombre de la función (debe respetar las reglas de denominación presentes en el capítulo Introducción a PHP - Estructura básica de una página PHP). En este nombre no se diferencian mayúsculas y minúsculas (para PHP, las funciones `unafunción` y `UnaFunción` son las mismas).

`parámetro` Parámetros posibles de la función expresados como una lista de variables (véase la sección Parámetros): `$parámetro1, $parámetro2, ...`

`tipo` Declaración del tipo de datos devuelto por la función. Valores posibles: `int`, `float`, `string`, `bool`, `array`, `callable`, `iterable`, `object`, `mixed`, `void`, un nombre de clase o de interfaz (véase en este capítulo la sección Clases) o una unión de tipos. El nombre del tipo se puede preceder por un punto de interrogación (`?`) que indica que la función puede devolver un valor `NULL`. En el capítulo Introducción a PHP puede encontrar la definición de los tipos de datos (apartado Las bases del lenguaje PHP - Tipos de datos).

`instrucciones` Conjunto de instrucciones que componen la función.

El nombre de la función no debe ser una palabra reservada de PHP (nombre de función nativa, de instrucción) ni ser igual al nombre de otra función definida de antemano.

Una función de usuario se puede llamar como una función nativa de PHP: en una asignación, en una comparación, etc.

Si la función devuelve un valor, es posible utilizar la instrucción `return` para definir el valor de retorno de la función.

Sintaxis

```
return expresión;
```

`expresión` Expresión cuyo resultado constituye el valor de retorno de la función (`NULL` por defecto).

El resultado de una función puede ser de cualquier tipo (cadena, número, matriz, etc.).

La instrucción `return` detiene la ejecución de la función y devuelve el resultado de `expresión` a quien realiza la llamada. Si hay varias instrucciones `return` en la función, la primera que se encuentre en el desarrollo de las instrucciones es la que define el valor de retorno y provoca la interrupción de la función. Si la función no incluye ninguna instrucción `return` (o si no se ha ejecutado ninguna instrucción `return`), el valor de retorno de la función es `NULL`.

Ejemplo

```
<?php
// Función sin parámetro que muestra "¡Hola!"
// Sin valor de retorno.
function mostrar_hola() {
  echo '¡Hola!<br />';
}
// Función con dos parámetros que devuelve el producto
// de los dos parámetros.
function producto($valor1,$valor2) {
  return $valor1 * $valor2;
}
// Llamada de la función mostrar_hola.
mostrar_hola();
// Usos de la función producto:
// - en una asignación
$resultado = producto(2,4);
echo "2 x 4 = $resultado<br />";
// - en una comparación
if (producto(10,12) > 100) {
  echo '10 x 12 es superior a 100.<br />';
}
?>
```

Resultado

```
¡Hola!
2 x 4 = 8
10 x 12 es superior a 100.
```

Observación

En el lenguaje PHP, no existe un procedimiento real. Para definir algo equivalente a un procedimiento, basta con definir una función que no devuelva ningún valor y llamar a la función como si se tratara de una instrucción (como la función `mostrar_hola`, por ejemplo). Una función que no devuelve nada, se puede declarar de manera explícita con el tipo de retorno `void`.

Como ya mencionamos anteriormente, el contenido de una matriz se puede transformar en una lista de parámetros dentro de una llamada de función gracias al operador `...` (tres puntos suspensivos).

Ejemplo

```
<?php
// Función con tres parámetros que devuelve la suma
// de los tres parámetros.
function suma($valor1,$valor2,$valor3) {
  return $valor1 + $valor2 + $valor3;
}
// Transformación del contenido de una matriz en
// lista de parámetros.
$valores = [1,2,3];
echo '1 + 2 + 3 = ',suma(...$valores),'<br />';
// Lo mismo para una parte solamente de los parámetros
// con una matriz definida directamente en la llamada.
echo '1 + 2 + 4 = ',suma(1,...[2,4]),'<br />';
?>
```

Resultado

```
1 + 2 + 3 = 6
1 + 2 + 4 = 7
```

Cuando una función devuelve una matriz, es posible acceder directamente a un elemento de la matriz llamando a la función con una sintaxis de tipo `función(...)[clave]`.

Ejemplo

```
<?php
// Definición de una función que devuelve una matriz.
function quien() {
  return ['Olivier','Heurtel'];
}
// Llamada a la función y recuperación directa del nombre almacenado
// en el índice 0 de la matriz devuelta.
$nombre = quien()[0];
echo "quien()[0] = $nombre<br />";
?>
```

Resultado

```
quien()[0] = Olivier
```

Esta técnica funciona también cuando la función devuelve una matriz multidimensional con una sintaxis de tipo `función(...)[clave1][clave2]`.

Es posible utilizar una función antes de definirla.

Ejemplo

```
<?php
// Utilización de la función producto.
echo producto(5,5);
// Definición de la función producto.
function producto($valor1,$valor2) {
  return $valor1 * $valor2;
}
?>
```

Resultado

```
25
```

Por lo tanto, no hay ningún problema para definir funciones que se llamen entre ellas.

Observación

Una función se puede utilizar solo en el script donde se define. Para utilizarla en varios scripts, es necesario o bien copiar su definición en los diferentes scripts (se pierde el interés por definir una función) o bien definirla en un archivo incluido donde la función sea necesaria.

Ejemplo

– Archivo `funciones.inc` que contiene la definición de funciones:

```
<?php
// Definición de la función producto.
function producto($valor1,$valor2) {
  return $valor1 * $valor2;
}
?>
```

– Script que utiliza las funciones definidas en `funciones.inc`:

```
<?php
// Inclusión del archivo contenedor de la definición de las funciones.
include('funciones.inc');
// Utilización de la función producto.
echo producto(5,5);
?>
```

Declaración del tipo de retorno

Es posible definir el tipo de datos devuelto por una función.

Cuando es el caso, en el modo de funcionamiento por defecto (a diferencia de modo estricto que se presenta a continuación), PHP realiza si es necesario una conversión automática del valor devuelto al tipo de datos declarado.

Ejemplo

```
<?php
// Declaración de dos funciones que devuelven el producto
// de dos parámetros, el segundo especifica un tipo
// de datos "entero" para el valor de retorno.
function producto1($valor1,$valor2) {
  return $valor1 * $valor2;
}
function producto2($valor1,$valor2) : int {
  return $valor1 * $valor2;
}
// Llamada de dos funciones con los mismos parámetros
echo 'producto1(20,1/7) => ',var_dump(producto1(20,1/7)),'<br />';
echo 'producto2(20,1/7) => <b>',var_dump(producto2(20,1/7)),'</b><br />';
?>
```

Resultado

```
producto1(20,1/7) => float(2.8571428571428568)
producto2(20,1/7) => int(2)
```

En este ejemplo, podemos ver claramente que PHP ha convertido el valor devuelto por la segunda función a un valor entero (con las reglas de conversión mencionadas en el capítulo Introducción a PHP - Las bases del lenguaje PHP - Tipos de datos).

Si PHP no es capaz de realizar la conversión (tipos de datos no convertibles entre ellos), se produce una excepción `TypeError`. Esta excepción detiene el script si no se maneja (véase en este capítulo la sección Clases - Excepciones).

Ejemplo

```
<?php
// Declaración y llamada de una función que debe devolver una
// matriz pero que devuelve una cadena de caracteres.
function quien() : array {
  return 'Olivier Heurtel';
}
echo 'quien()[0] = ',quien()[0];
?>
```

Resultado

```
quien()[0] =
Fatal error: Uncaught TypeError: Return value of quien() must be of the
type array, string returned in /app/scripts/index.php:5 Stack trace: #0
/ app/scripts/index.php(7): quien() #1 {main} thrown in/app/scripts/
index.php on
line 5
```

Una función declarada con un tipo de retorno diferente de `void`, debe devolver un valor no `NULL`. Si este no es el caso, se devuelve un error, diferente según el caso:

Ausencia de instrucción `return`

```
Fatal error: Uncaught TypeError: Return value of MiFuncion() must be of
the type int, none returned in ...
```

Instrucción `return` vacía

```
Fatal error: A function with return type must return a value in ...
```

Instrucción `return NULL`

```
Fatal error: Uncaught TypeError: Return value of MiFuncion() must be of
type int, null returned in ...
```

Para autorizar una función para que devuelva un valor `NULL`, hay que utilizar un punto de interrogación (`?`), delante del nombre del tipo (diferente de `void`).

```
<?php
// Declaración y llamada de una función que especifica un
// tipo de datos de retorno que puede ser NULL
function cubo($valor) : ?int {
  if (is_null($valor)) {
    return NULL;
  } else {
    return $valor ** 3 ;
  }
}
echo 'cubo(2) => <b>',var_dump(cubo(2)),'</b><br />';
echo 'cubo(NULL) => <b>',var_dump(cubo(NULL)),'</b><br />';
?>
```

Resultado

```
cubo(2) => int(8)
cubo(NULL) => NULL
```

Incluso con esta opción, la función debe tener una instrucción `return` no vacía. Si no es el caso, se devuelve un error, diferente según el caso:

Ausencia de instrucción `return`

```
Fatal error: Uncaught TypeError: Return value of MiFuncion() must be of
type int, none returned in ...
```

Instrucción `return` vacía

```
Fatal error: A function with return type must return a value (did you
mean "return null;" instead of "return;"?) in ...
```

Capítulo 3
WordPress y PHP

1. Introducción

WordPress es un CMS diseñado íntegramente en PHP que es un lenguaje orientado a objetos. Por lo tanto, es normal que WordPress haya desarrollado sus propias APIs, clases, métodos, funciones, etc. para simplificar la vida de los desarrolladores. Eso implica que como desarrollador, deberá familiarizarse con el funcionamiento de WordPress para poder aprovechar todo su potencial.

Este capítulo describe la preponderancia de PHP en WordPress, la estructura y funciones recurrentes de WordPress. Sirve como referencia para todo el libro. Si lo desea, puede omitirlo y consultarlo más adelante cuando personalice el archivo functions.php, cree temas avanzados o cuando cree extensiones, etc.

Este capítulo es un resumen de las posibilidades que ofrece WordPress. Se centra en lo esencial, así que no dude en consultar la documentación oficial, el códex o los numerosos tutoriales gratuitos disponibles en la Web, para profundizar en sus conocimientos sobre un tema específico.

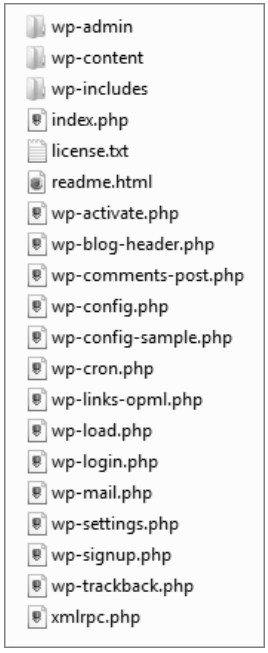
Todas las palabras clave y términos técnicos de cada capítulo le permiten realizar su propia investigación en la Red. Con una comunidad muy activa, sería una pena no aprovechar la experiencia de todas estas personas, dispuestas a ayudarle de forma gratuita. Encontrará muchos recursos, con muchos ejemplos de código.

2. La estructura de los archivos y carpetas WordPress

Es importante saber cómo se organizan los archivos en WordPress. La estructura es siempre la misma, tanto para un sitio web pequeño o para uno de mayor tamaño: el núcleo o corazón de WordPress no cambia.

2.1 Los archivos y carpetas en la raíz del sitio web

Todos los sitios de WordPress tienen la misma estructura. En la raíz, encontrará las carpetas wp-admin, wp-content, wp-includes y los archivos de WordPress.



Archivos y carpetas en la raíz del sitio web WordPress

No debe modificar o eliminar ningún archivo o carpeta nativo de WordPress. En el caso de aquellos que forman parte del core, es posible que el sitio web deje de funcionar. Además, la actualización de WordPress sobrescribe todos los cambios.

La carpeta **wp-admin** contiene todos los archivos relacionados con la administración del sitio web. No debe modificar archivos de esta carpeta.

La carpeta **wp-content** contiene todos los archivos por temas, extensiones y medios, idiomas, etc. En esta carpeta es donde se realizan las principales modificaciones y personalizaciones del sitio web.

La carpeta **wp-includes** contiene todos los archivos principales de WordPress (clases, funciones, scripts, etc.). No debe editar archivos de esta carpeta.

El archivo **wp-config.php** es un archivo generado durante la instalación de WordPress, a partir del archivo wp-config-sample.php. La información utilizada para configurar el sitio web se encuentra en este archivo: el nombre de la base de datos, la contraseña, la URL de la base de datos, el prefijo de las tablas, las claves de seguridad, la configuración del idioma principal del sitio, etc. Este archivo se modifica durante el cambio de host, la optimización del sitio web, la depuración, un cambio de idioma, etc. La eliminación del archivo wp -config.php vuelve a mostrar la página de configuración.

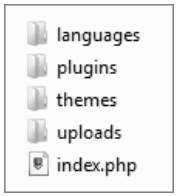
El resto de archivos .php se utilizan para el núcleo de WordPress y no se deben modificar ni eliminar en absoluto. Sin embargo, puede encontrar archivos en la raíz del sitio que hayan sido generados por un complemento, por WordPress o agregados por usted mismo. Por tanto, se pueden modificar algunos archivos: .htaccess, robots.txt o sitemap.xml.

2.2 La carpeta wp-content

La carpeta **wp-content** es la carpeta que se utiliza para crear temas, extensiones y funciones. Contiene las carpetas **languages**, **plugins**, **themes** y el archivo **index.php**.

Aparecen otras carpetas al agregar imágenes, como la carpeta **uploads** y durante las actualizaciones, como la carpeta **upgrade**.

Es posible que sigan apareciendo otras carpetas según la instalación de determinadas extensiones, como la carpeta **cache**.



Carpetas y archivos en el directorio wp-content, al instalar WordPress

La carpeta **languages** contiene los archivos de idioma, temas y complementos de WordPress. La instalación manual se realiza descargando por FTP los archivos de idioma a esta carpeta y las subcarpetas relevantes. Los archivos de idioma tienen las extensiones .mo y .po/.pot.

La carpeta **plugins** contiene todas las extensiones instaladas. De base, la carpeta contiene dos extensiones: Akismet y Hello Dolly. La instalación manual se lleva a cabo descargando por FTP la carpeta que contiene los archivos de la extensión en la carpeta plugins. Todos los plugins se muestran en la administración, dentro de la pestaña **Plugins**.

La carpeta **themes** contiene todos los temas instalados. Hay tres temas básicos: Twenty Seventeen, Twenty Sixteen y Twenty Fifteen. La instalación manual se realiza descargando por FTP la carpeta que contiene los archivos y carpetas del tema en la carpeta themes. Todos los temas se muestran en la administración, dentro de la pestaña **Apariencia** - **temas**.

La carpeta **uploads** contiene todos los medios cargados. Para las imágenes, WordPress crea tres formatos de imagen y los clasifica según la configuración indicada en la administración, en la pestaña **Ajustes** - **Medios**. De forma predeterminada, WordPress clasifica los medios en carpetas con el nombre del año, conteniendo a su vez carpetas con el número de los meses (01, 02,…, 12). Algunas veces, ciertas extensiones crean carpetas para almacenar varios archivos. Todos los medios se muestran en la administración, pestaña **Medios**.

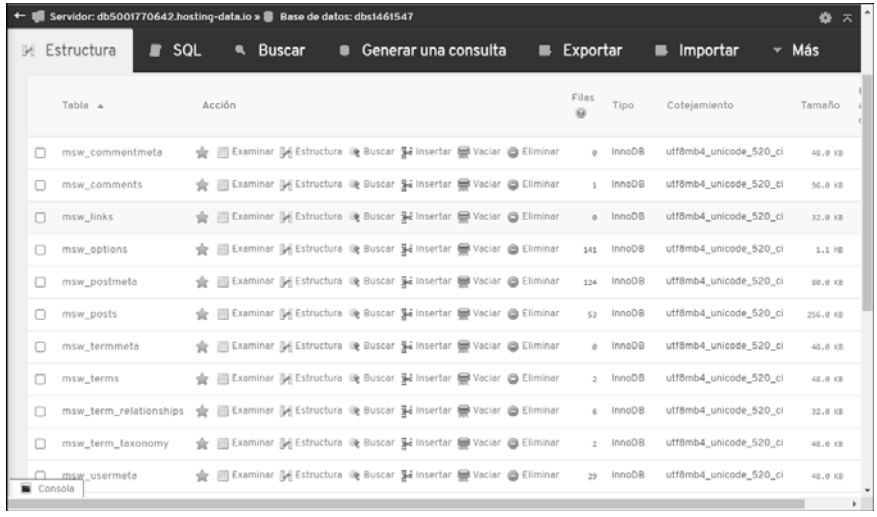
La carpeta **upgrade** aloja archivos comprimidos para actualizaciones de temas o extensiones. Antes de descomprimirlos, WordPress los descarga en esta carpeta. Si la actualización sale mal, encontrará el archivo comprimido en esta carpeta. Esta carpeta aparecerá cuando realice una actualización.

El archivo **index.php** solo se usa para la seguridad de la carpeta **wp-content**. Este archivo vacío se utiliza para mostrar una página en blanco en lugar del árbol del sitio web, en caso de un mal funcionamiento. Lo encontrará en diferentes carpetas.

3. La base de datos WordPress

WordPress instala doce tablas en la base de datos. Para ver esto, conéctese a su base de datos. Aquí es donde se almacena toda la información sobre su sitio (contenido de artículos, páginas, categorías, etiquetas, configuración de WordPress, complementos, temas, información sobre los usuarios, etc.).

En este libro, los ejemplos se ilustran con el prefijo `msw_` (por defecto, WordPress usa el prefijo `wp_`). Normalmente, el prefijo se elige durante la instalación, por lo que podría ser diferente si lo cambia por razones de seguridad. Recuerde cambiar el prefijo cuando utilice los ejemplos.



Tablas en la base de datos durante la instalación de WordPress, en phpMyAdmin

Hay otras tablas que aparecen según los temas, extensiones y el uso del modo multisitio.

Observación

Referencia al códex: https://codex.wordpress.org/Database_Description

Aquí están las tablas nativas de WordPress, con el prefijo base wp:

– La tabla **wp_commentmeta** contiene información adicional sobre los comentarios. La utiliza la extensión **Akismet**. Se ha convertido en una extensión nativa de WordPress, imprescindible para evitar el spam. La encontrará en la administración en la pestaña **Plugins** y para la configuración, en la pestaña **Ajustes** - **Akismet Anti-Spam**.

– La tabla **wp_comments** contiene todos los comentarios de los artículos y las páginas.

– La tabla **wp_links**(opcional) agrupa todos los enlaces registrados a través de la pestaña **Enlaces** de la administración. La pestaña **Enlaces** ya no existe desde la versión 3.5 y requiere el uso de la extensión Link Manager, pero la tabla aún está presente para que las personas que la usaban antes no pierdan sus datos en las versiones recientes de WordPress. También puede habilitar el administrador de enlaces agregando el siguiente código al archivo functions.php:

```
add_filter('pre_option_link_manager_enabled',
'__return_true');
```

– La tabla **wp_options** contiene las configuraciones generales del sitio (introducidas durante la instalación del sitio) y extensiones, entre las más importantes. Durante la creación de temas avanzados, esta tabla también se usa para almacenar información, gracias a las funciones de WordPress.

– La tabla **wp_postmeta** contiene información adicional relacionada con los artículos o las páginas. Esta tabla está directamente relacionada con la tabla wp_posts.

– La tabla **wp_posts** es la más importante, ya que tiene todo el contenido del sitio. Encontrará toda la información sobre los artículos, páginas, fotos, archivos PDF y otros medios, o sobre los productos en el caso de las extensiones de comercio electrónico, así como el contenido de sus publicaciones personalizadas (custom post type), que aprenderemos a crear.

– La tabla **wp_termmeta** contiene los metadatos agregados a las categorías. Esta nueva tabla está disponible desde la versión 4.4.

- La tabla **wp_terms** contiene categorías y etiquetas. Esta tabla está directamente relacionada con las tablas wp_term_relationships y wp_term_taxonomy.
- La tabla **wp_term_relationships** vincula categorías y etiquetas a los diferentes artículos y páginas. Esta tabla está directamente relacionada con las tablas wp_terms y wp_term_taxonomy.
- La tabla **wp_term_taxonomy** se utiliza para diferenciar las categorías y las etiquetas. Allí encontrará la información adicional sobre las categorías y las etiquetas. Esta tabla está directamente relacionada con las tablas: wp_terms y wp_term_relationships.
- La tabla **wp_usermeta** contiene información adicional de todos los usuarios, así como su rol.
- La tabla **wp_users** contiene todos los usuarios, su contraseña, dirección de correo electrónico, etc.

Si olvida la contraseña, puede cambiarla directamente en su base de datos modificando la entrada después debe asignar al campo `user_pass` el valor de su nueva contraseña. No olvide poner el campo en `md5` antes de ejecutar la petición, lo que permite encriptar la contraseña por razones de seguridad. De esta forma, nadie podrá conocer la contraseña, ni siquiera una persona con acceso a la base de datos.

Observación

MD5(Message Digest 5) es una función de hash criptográfico que calcula un código digital único (huella digital), a partir de un archivo digital.

También puede ejecutar directamente la siguiente consulta SQL, cambiando primero los términos entre comillas (no olvide reemplazar el prefijo de la tabla si es diferente):

```
UPDATE wp_users SET user_pass=MD5('nuevacontraseña') WHERE
user_login='nombre';
```

En esta solicitud, se cambia "nuevacontraseña" por la contraseña que ha elegido y cambia "nombre" por el nombre de usuario.