Funciones y bibliotecas

- Funciones
- Funciones con argumentos predeterminados
- Bibliotecas

Funciones

En PHP se pueden definir funciones, de manera que no haga falta escribir el mismo código varias veces. Las funciones deben definirse antes de utilizarlas (aunque no sea en el mismo fragmento de código php). Las funciones pueden no tener argumento o tener argumentos por valor o por referencia.

Los nombres de las funciones siguen las mismas reglas de los identificadores de PHP, es decir, deben comenzar por una letra o un guión bajo (_) y el resto de caracteres pueden ser letras, números o guiones bajos (se pueden utilizar caracteres no ingleses como acentos, eñes, etc), pero los nombres de funciones no distinguen entre mayúsculas o minúsculas.

La guía de estilo PEAR para PHP recomienda que los nombres de las funciones sigan el estilo camelCase (es decir, sin espacios ni guiones, con la primera palabra en minúsculas y el resto con la primera letra en mayúsculas).

Ejemplo de función y de su uso:

```php
<?php

// ESTA ES LA DEFINICIÓN DE LA FUNCIÓN calculaHipotenusa
function calculaHipotenusa($arg1, $arg2)
{
    $hipotenusa = sqrt($arg1 * $arg1 + $arg2 * $arg2);
    return $hipotenusa;
}

// ESTO ES UN EJEMPLO DE USO DE LA FUNCIÓN calculaHipotenusa
$cateto1 = 12;
$cateto2 = 16;
$hipotenusa = calculaHipotenusa($cateto1, $cateto2);

print "<p>El triángulo de lados $cateto1, $cateto2 y $hipotenusa es rectángulo.</p>\n";
?>
```

```
<p>El triángulo de lados 12, 16 y 20 es rectángulo.</p>
```

PHP no distingue entre mayúsculas y minúsculas en los nombres de las funciones.

```php
<?php

// ESTA ES LA DEFINICIÓN DE LA FUNCIÓN calculaHipotenusa
function calculaHipotenusa($arg1, $arg2)
{
    $hipotenusa = sqrt($arg1*$arg1+$arg2*$arg2);
    return $hipotenusa;
}

// ESTO ES UN EJEMPLO DE USO DE LA FUNCIÓN calculaHipotenusa
$cateto1 = 12;
$cateto2 = 16;
$hipotenusa = CALCULAHIPOTENUSA($cateto1, $cateto2);

print "<p>El triángulo de lados $cateto1, $cateto2 y $hipotenusa es rectángulo.</p>\n";
?>
```

```
<p>El triángulo de lados 12, 16 y 20 es rectángulo.</p>
```

Funciones con argumentos predeterminados

En PHP se pueden definir funciones con argumentos predeterminados, de manera que si en la llamada a la función no se envía un argumento, la función asume un valor predeterminado. Lógicamente, los argumentos predeterminados deben ser los últimos en la lista de argumentos, para evitar ambigüedades.

Los argumentos predeterminados se establecen en la definición de la función, igualando el nombre del argumento a su valor predeterminado.

El ejemplo siguiente muestra una función que calcula diferentes tipos de media (aritmética, geométrica, armónica). Los argumentos de la función son los números cuya media se debe calcular y el tipo de media a calcular. En el ejemplo, el tipo de media predeterminado es la media aritmética.

```php
<?php

// ESTA ES LA DEFINICIÓN DE LA FUNCIÓN calculaMedia
function calculaMedia($arg1, $arg2, $arg3 = "aritmética")
{
    if ($arg3 == "aritmética") {
        $media = ($arg1 + $arg2) / 2;
    } elseif ($arg3 == "geométrica") {
        $media = sqrt($arg1 * $arg2) / 2;
    } elseif ($arg3 == "armónica") {
        $media = 2 * ($arg1 * $arg2) / ($arg1 + $arg2);
    }
    return $media;
}

// ESTO SON EJEMPLOS DE USO DE LA FUNCIÓN calculaMedia
$dato1 = 12;
$dato2 = 16;

// EL TERCER ARGUMENTO INDICA EL TIPO DE MEDIA A CALCULAR
$media = calculaMedia($dato1, $dato2, "geométrica");
print "<p>La media geométrica de $dato1 y $dato2 es $media.</p>\n";

// AL NO PONER EL TERCER ARGUMENTO, LA FUNCIÓN DEVUELVE LA MEDIA ARITMÉTICA
$media = calculaMedia($dato1, $dato2);
print "<p>La media aritmética de $dato1 y $dato2 es $media.</p>\n";
?>
```

```
<p>La media geométrica de 12 y 16 es 6.9282032302755.</p>
<p>La media aritmética de 12 y 16 es 14.</p>
```

Bibliotecas

Las bibliotecas son archivos php que se pueden incluir en cualquier otro archivo php. Las bibliotecas se suelen utilizar para centralizar fragmentos de código que se utilizan en varias páginas. De esa manera, si se quiere hacer alguna modificación, no es necesario hacer el cambio en todos las páginas si no únicamente en la biblioteca.

Por ejemplo, si definimos en la biblioteca una función que imprima la cabecera de las páginas, desde cualquier página se puede incluir la biblioteca mediante la construcción `include` y llamar a la función como si se hubiera definido en la propia página:

- biblioteca.php

```
<?php
function cabecera($titulo)
{
    print "<?xml version=\"1.0\" encoding=\"iso-8859-1\"?.">
<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0 Strict//EN\"
\"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd\">
<html xmlns=\"http://www.w3.org/1999/xhtml\">
<head>
  <meta http-equiv=\"Content-Type\" content=\"text/html; charset=iso-8859-1\" />
  <title>$titulo</title>
  <link href=\"estilo.css\" rel=\"stylesheet\" type=\"text/css\" />
</head>

<body>
<h1>$titulo</h1>\n";
}
?>
```

- pagina_1.php

```
<?php
include "biblioteca.php";
cabecera("Página de ejemplo");
print "<p>Esta página es válida</p>";
?>
</body>
</html>
```

```
<?xml version=\"1.0\" encoding=\"iso-8859-1\"?>
<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0 Strict//EN\"
\"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd\">
<html xmlns=\"http://www.w3.org/1999/xhtml\">
<head>
  <meta http-equiv=\"Content-Type\" content=\"text/html; charset=iso-8859-1\" />
  <title>Página de ejemplo</title>
  <link href=\"estilo.css\" rel=\"stylesheet\" type=\"text/css\" />
</head>

<body>
<h1>Página de ejemplo</h1>
<p>Esta página es válida</p>
</body>
</html>
```

- pagina_2.php

```php
<?php
include "biblioteca.php";
cabecera("Otra página de ejemplo");
print "<p>Esta página también es válida</p>";
?>
</body>
</html>
```

```
<?xml version=\"1.0\" encoding=\"iso-8859-1\"?>
<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0 Strict//EN\"
\"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd\">
<html xmlns=\"http://www.w3.org/1999/xhtml\">
<head>
  <meta http-equiv=\"Content-Type\" content=\"text/html; charset=iso-8859-1\" />
  <title>Otra página de ejemplo</title>
  <link href=\"estilo.css\" rel=\"stylesheet\" type=\"text/css\" />
</head>

<body>
<h1>Otra página de ejemplo</h1>
<p>Esta página también es válida</p>
</body>
</html>
```

Se pueden crear todas las bibliotecas que se necesiten e incluir cualquier número de bibliotecas en cualquier punto de un programa. Las bibliotecas pueden a su vez incluir otras bibliotecas.

Normalmente, las bibliotecas suelen contener funciones, definiciones de constantes o inicialización de variables, pero en realidad pueden incluir cualquier tipo de código php, que se ejecutará en la posición en la que se incluya la biblioteca.

En el ejemplo siguiente, las bibliotecas modifican variables, lo que afecta a su valor.

- biblioteca_1.php

```php
<?php
$i = 1;
?>
```

- biblioteca_2.php

```php
<?php
$i = $i + 10;
?>
```

- Programa:

```php
<?php
include "biblioteca_1.php";
print "<p>Ahora \$i vale $i</p>\n";
include "biblioteca_2.php";
print "<p>Ahora \$i vale $i</p>\n";
include "biblioteca_2.php";
print "<p>Ahora \$i vale $i</p>\n";
?>
```

```
<p>Ahora $i vale 1</p>
<p>Ahora $i vale 11</p>
<p>Ahora $i vale 21</p>
```

Existe una construcción similar a `include`, la construcción `require`. La diferencia con respecto a `include` es que `require` produce un error si no se encuentra el archivo (y no se procesa el resto de la página), mientras que `include` sólo produce un aviso (y se procesa el resto de la página).

En un mismo archivo php se pueden incluir varias construcciones `include` o `require`, pero si las bibliotecas incluidas contienen definiciones de funciones, al incluir de nuevo la definición de la función se genera un error.

Para que no ocurra este problema se pueden utilizar las funciones `include_once` o `require_once`, que también incluyen los ficheros pero que, en caso de que los ficheros ya se hayan incluido, entonces no los incluyen.

Las cuatro construcciones (`include`, `require`, `include_once` y `require_once`) pueden utilizarse escribiendo entre paréntesis el nombre de los ficheros o sin escribir paréntesis.

La guía de estilo del proyecto PEAR prescribe el uso de la construcción `require_once` en el caso de bibliotecas cuya inclusión no dependa de ninguna condición y que no se utilicen paréntesis alrededor de los nombres de las bibliotecas.

include

(PHP 4, PHP 5)
La sentencia *include* incluye y evalúa el archivo especificado.

La siguiente documentación también se aplica a require.

Los archivos son incluidos con base en la ruta de acceso dada o, si ninguna es dada, el include_path especificado. Si el archivo no se encuentra en el include_path, *include* finalmente verificará en el propio directorio del script que hace el llamado y en el directorio de trabajo actual, antes de fallar. El constructor *include* emitirá una advertencia si no puede encontrar un archivo, éste es un comportamiento diferente al de require, el cual emitirá un error fatal..

Si una ruta es definida — ya sea absoluta (comenzando con una letra de unidad o \ en Windows o / en sistemas Unix/Linux) o relativa al directorio actual (comenzando con . o ..) — el include_path será ignorado por completo. Por ejemplo, si un nombre de archivo comienza con ../, el interprete buscará en el directorio padre para encontrar el archivo solicitado.

Para más información sobre como PHP maneja la inclusión de archivos y la ruta de accesos para incluir, ver la documentación de include_path.

Cuando se incluye un archivo, el código que contiene hereda el ámbito de las variables de la línea en la cual ocurre la inclusión. Cualquier variable disponible en esa línea del archivo que hace el llamado, estará disponible en el archivo llamado, desde ese punto en adelante. Sin embargo, todas las funciones y clases definidas en el archivo incluido tienen el ámbito global.

vars.php

```php
<?php

$color = 'verde';
$fruta = 'manzana';

?>
```

test.php

```php
<?php

echo "Una $fruta $color"; // Una

include 'vars.php';

echo "Una $fruta $color"; // Una manzana verde

?>
```