PHP foreach

Una de las funciones más útiles que tenemos en PHP y en prácticamente todos los lenguajes de programación es el foreach. Esta simple función nos permite recorrer todos los elementos que contiene un array o un objeto.

La sintaxis de un bucle foreach es muy simple, básicamente definimos la variable desde la que obtendremos los elementos a recorrer y que variable asignaremos a cada elemento. Veamos un ejemplo sencillo:

```
<?php

//Array de animales que recorreremos.
$animals = array('gato', 'perro', 'hurón', 'ardilla');

//Recorremos el array con un foreach
foreach ($animals as $animal) {
   echo $animal . '<br>';
}
```

Como pueden ver el código es muy simple y básicamente lo que hará será imprimir en pantalla todos los animales con un salto de línea.

Pero, ¿Qué pasa si quisiéramos modificar los elementos del array para por ejemplo poner la primera letra en mayúscula?

En ese caso tendríamos que recurrir a una ligera modificación del foreach, como vemos a continuación:

```
<?php

//Array de animales que recorreremos.
$animals = array('gato', 'perro', 'hurón', 'ardilla');

//Recorremos el array con un foreach
foreach ($animals as $key => $animal) {
   $animals[$key] = ucfirst($animal);
```

Agencia de Aprendizaje a lo largo de la vida

}

print_r($animals);
La única diferencia que hay entre este foreach es que agregamos $key => delante de $animal, gracias al $key lo que sabemos es la posición en la que esta el elemento en nuestro array.

Por lo que sabiendo la posición, simplemente tenemos que coger nuestro array $animals y pasarle la posición (que está en $key) entre corchetes y asignarle un nuevo valor, que en este caso será básicamente el elemento actual con la primera letra en mayúscula.

PHP empty: cómo comprobar si una variable está vacia
Una de las características de PHP es que permite trabajar con diferentes tipos de variables, como números, cadenas de texto, booleanos, arrays u objetos.

Sin embargo, a veces necesitamos saber si una variable tiene algún valor asignado o no, es decir, si está vacía o no. Para ello, podemos usar PHP empty(), que nos devuelve un valor verdadero o falso según el caso.

En este capitulo te explicaremos qué es y cómo usar la función empty() en PHP para comprobar si una variable está vacía o no.

La función empty() en PHP se utiliza para comprobar si una variable está vacía o no. Una variable se considera vacía si no existe o si su valor es equivalente a alguno de los siguientes:

Un número cero (0)
Un número decimal cero (0.0)
Una cadena de texto vacía (“”)
Una cadena de texto que contiene solo el carácter cero (“0”)
Un valor NULL
Un valor FALSE
Un array vacío (array())
La sintaxis de la función empty() es la siguiente:

empty(variable);

Donde variable es la variable que queremos comprobar. La función devuelve un valor booleano: true si la variable está vacía o false si no lo está.

Ejemplos de uso PHP empty
Validar formularios
La función empty() en PHP es muy útil para validar datos de entrada, por ejemplo, cuando recibimos datos de un formulario web. Podemos usar la función empty() para comprobar si el usuario ha introducido algún valor en los campos obligatorios o si ha dejado alguno en blanco.

Por ejemplo, supongamos que tenemos un formulario con los siguientes campos: nombre, email y mensaje. Podemos usar la función empty() para validar los datos de la siguiente manera:

```
<?php
// Recibimos los datos del formulario
$nombre = $_POST[“nombre”];
 $email = $_POST[“email”];
$mensaje = $_POST[“mensaje”];

// Comprobamos si los campos están vacíos o no
if (empty($nombre))
{
   echo "Por favor, introduce tu nombre.";
} elseif (empty($email))
{
   echo "Por favor, introduce tu email.";
} elseif (empty($mensaje))
{
   echo "Por favor, escribe un mensaje.";
}
else {
   echo "Gracias por contactarnos. Te responderemos lo antes posible.";
}
```

Solo mostrar las variables que tengan contenido

Agencia de Aprendizaje a lo largo de la vida

Supongamos que tenemos un array con los nombres de algunos países y queremos mostrarlos en una lista ordenada. Sin embargo, puede que el array tenga algunos elementos vacíos que no queremos mostrar. Podemos usar la función empty() para filtrar los elementos vacíos del array y mostrar solo los que tienen algún valor. Por ejemplo:

```
<?php
// Definimos un array con los nombres de algunos países
$paises = array("España", "", "Francia", "Alemania", "", "Italia");

// Creamos una lista ordenada
echo "<ol>";

// Recorremos el array con un bucle foreach
foreach ($paises as $pais) {
    // Comprobamos si el elemento está vacío o no
    if (!empty($pais)) {
        // Si no está vacío, lo mostramos en la lista
        echo "<li>$pais</li>";
    }
}

// Cerramos la lista ordenada
echo "</ol>";
```

En este ejemplo, usamos la función empty() para comprobar si cada elemento del array $paises está vacío o no. Si no está vacío, lo mostramos en la lista ordenada con la etiqueta <li>. Si está vacío, lo ignoramos. El resultado sería algo así:

```
<ol>
    <li>España</li>
    <li>Francia</li>
    <li>Alemania</li>
    <li>Italia</li>
</ol>
```

Conectar Base de datos con PHP

Agencia de Aprendizaje a lo largo de la vida

¿Qué es MySQLi de PHP?
MySQLi es una extensión mejorada (la i final es de improved) de PHP para acceder a bases de datos MySQL. MySQL es, junto con Oracle y Microsoft SQL Server, uno de los sistemas de gestión de bases de datos (es decir, un Database Management System o DBMS) relacionales más populares a nivel mundial. Las bases de datos relacionales son un elemento central de Internet, ya que permiten procesar grandes cantidades de datos y guardarlos de forma permanente. Para hacerlo, dividen los complejos conjuntos de datos en partes y establecen luego las relaciones necesarias entre ellos.

Este software, desarrollado en 1994 por la compañía sueca MySQL AB, es distribuido en la actualidad por Oracle Corporation mediante un sistema dual de licencias: además de la licencia propietaria para empresas, Oracle ofrece también una versión con licencia GPL y de código abierto. Este doble sistema de licencias da a las empresas la oportunidad de desarrollar aplicaciones propias basadas en MySQL sin necesidad de recurrir a una licencia open source.

¿Qué abarca la extensión mysqli?
En PHP existen tres maneras de acceder a una base de datos MySQL. La más antigua es utilizar la extensión MySQL, la cual, sin embargo, está considerada desfasada o deprecated desde la versión PHP 5.5 y fue eliminada completamente con PHP 7. En esta última versión, la función mysql ya no funciona y ha sido reemplazada por mysqli.

Además de la anticuada extensión mysql, para acceder a una base de datos MySQL, PHP también ofrece los PHP Data Objects (PDO), cuya aplicación es particularmente flexible. La tercera opción es usar la

MySQL Improved Extension, es decir, la extensión mysqli, que ya desde PHP 5 permite acceder a bases de datos MySQL. El siguiente snippet o fragmento de código puede ayudarte a hacerte una idea de cómo funciona la extensión MySQLi de PHP.

Ejemplo simple forma procedimental con mysqli y php:

```php
<?php


$mysqli = mysqli_connect('localhost', 'root', '', 'nombreBD');

if(!$mysqli){

    die("fallo la conexion: ". mysqli_connect_error());

}else{

    var_dump($mysqli);
}


?>
```

¿Qué ventajas tiene MySQLi?

Al contrario que su predecesora, la extensión mysqli no solo puede usarse de forma procedural, sino también de forma orientada a objetos. Una ventaja de la programación orientada a objetos es que el código escrito puede corregirse y adaptarse fácilmente a posteriori. Esto puede ser útil, por ejemplo, para crear nuevas clases que puedan heredar el comportamiento y las propiedades de otras clases ya existentes. Así, se acorta considerablemente el tiempo de desarrollo y se facilita la adaptación del programa ante un entorno cambiante o nuevos requisitos.

Otra ventaja importante de MySQLi son los prepared statements o consultas preparadas. Se trata, por así decirlo, de instrucciones ya preparadas para el sistema de la base de datos. Mientras que los statements convencionales contienen valores de parámetros, los prepared statements contienen en su lugar los

llamados marcadores de posición o caracteres comodín. Cuando se ejecuta en el sistema de la base de datos un statement con distintos parámetros varias veces (en un bucle, por ejemplo), los prepared statements permiten aumentar la velocidad, puesto que las órdenes en sí ya se encuentran compiladas en la base de datos y simplemente han de ser ejecutadas con los nuevos parámetros. Además, los prepared statements son una medida efectiva de prevención contra las inyecciones SQL, ya que el sistema de la base de datos ha de comprobar la validez de los parámetros antes de procesarlos.
Ejemplo simple de conexión orientada a objetos con BD:

```php
$mysqliPOO = new mysqli('localhost', 'root', '', 'nombreBD');

if($mysqliPOO->connect_errno){
    die("falló la conexion: ". $mysqliPOO->connect_error);
}else{
    var_dump($mysqliPOO);
}
```

Agencia de Aprendizaje a lo largo de la vida