Clase 07

Análisis de Sistemas

Materia:
Programación Web II

Docente contenidista: ROLDÁN, Hernán

Revisión: Coordinación

Contenido

Clase 7

¡Te damos la bienvenida a la materia **Programación Web II**!

En esta clase vamos a ver los siguientes temas:

- Arrays.
- Tipos de arrays.
- Formas de recorrer un array.
- Algunos métodos más usados en arrays.

Arrays

Introducción

Los arrays son una estructura de datos fundamental en la programación. En PHP, los arrays nos permiten almacenar y organizar múltiples valores relacionados bajo un mismo nombre.

Podemos pensar en los arrays como una caja que contiene diferentes compartimentos, y cada compartimento guarda un valor específico. Esta capacidad de almacenar varios valores en una sola variable hace que los arrays sean muy útiles y versátiles.

Ejemplo del mundo real

Imaginemos que estamos organizando una lista de invitados para una fiesta. Podríamos tener un array llamado "invitados" donde cada elemento del array representa el nombre de un invitado.

De esta manera, podemos acceder rápidamente a cualquier invitado en particular y realizar diferentes operaciones con la lista de invitados.

Tipos de arrays

En PHP, existen (tres) tipos de arrays que nos permiten almacenar y organizar datos de manera eficiente, estos son:

Arrays indexados

Los arrays indexados son la forma más básica de array en PHP.

En este tipo de array, cada elemento se almacena en una posición numérica llamada índice.

Los índices comienzan desde cero y se incrementan en uno para cada elemento adicional.

Ejemplo de array indexado (también llamado numérico):

```php
<?php


  $frutas = array("manzana", "banana", "naranja");


?>
```

En este ejemplo, **$frutas** es un array indexado que contiene tres elementos. Podemos acceder a cada elemento utilizando su índice, como **$frutas[0]** para "manzana", **$frutas[1]** para "banana" y **$frutas[2]** para "naranja".

Arrays asociativos

Los arrays asociativos utilizan claves personalizadas en lugar de índices numéricos para acceder a los elementos.

Cada elemento se almacena en una clave específica, que puede ser una cadena o un número.

Ejemplo de array asociativo:

```php
<?php

  $persona = array("nombre" => "Juan", "edad" => 25, "ciudad" => "Madrid");

?>
```

En este ejemplo, **$persona** es un array asociativo donde cada elemento se almacena con una clave específica.

Podemos acceder a los elementos utilizando las claves, como **$persona["nombre"]** para "Juan", **$persona["edad"]** para 25 y **$persona["ciudad"]** para "Madrid".

Arrays multidimensionales:

Los arrays multidimensionales son aquellos que contienen otros arrays como elementos.

Esto permite organizar y estructurar datos de manera más compleja.

Los arrays multidimensionales pueden tener tantas dimensiones como sean necesarias.

Ejemplo de array multidimensional:

```php
<?php

  $matriz = array(
    array(1, 2, 3),
    array(4, 5, 6),
    array(7, 8, 9)
  );

?>
```

En este ejemplo, **$matriz** es un array multidimensional que contiene tres arrays, cada uno con tres elementos.

Podemos acceder a los elementos utilizando índices para cada dimensión, como **$matriz[0][1]** para el valor 2, **$matriz[1][2]** para el valor 6 y **$matriz[2][0]** para el valor 7.

Arrays con claves mixtas:

PHP también permite crear arrays con claves mixtas, donde se combinan índices numéricos y claves personalizadas.

Estos arrays pueden contener tanto elementos indexados como elementos asociativos.

Ejemplo de array con claves mixtas:

```php
<?php

  $mezclado = array("manzana", "color" => "rojo", "banana",
"amarillo");

?>
```

Estos son los tipos de arrays más comunes en PHP. Cada tipo tiene sus características y se utiliza según las necesidades específicas de cada situación.

Los arrays nos permiten organizar y manipular datos de manera eficiente, brindando flexibilidad y potencia en el desarrollo de aplicaciones web en PHP.

Recordá que PHP ofrece una variedad de funciones y métodos para trabajar con arrays y realizar operaciones tales como: agregar, eliminar, ordenar y buscar elementos en ellos.
Explorar y familiarizarse con estas funciones ampliará tus habilidades en el manejo de arrays en PHP.

Formas de recorrer un array

Bucle for

El bucle for es una estructura de control que permite recorrer un array utilizando un índice numérico.

Es útil cuando se conoce el tamaño del array o se desea acceder a elementos específicos utilizando los índices.

Ejemplo de un array recorrido con un bucle for:

```php
<?php


  $frutas = array("manzana", "banana", "naranja");

  for($i=0; $i<count($frutas); $i++){

      echo $frutas[$i] . "<br />";

  }

?>
```

En este ejemplo, el bucle for recorre el array $frutas utilizando el índice $i para acceder a cada elemento y mostrarlo en pantalla.

Bucle foreach

El bucle foreach es una forma más sencilla y conveniente de recorrer un array.

Recorre automáticamente cada elemento del array sin necesidad de utilizar índices.

Es especialmente útil cuando no se necesita conocer el tamaño del array o no se requiere un acceso directo a los índices.

Ejemplo de un array recorrido con un bucle foreach:

```php
<?php

  $frutas = array("manzana", "banana", "naranja");

  foreach ($frutas as $fruta){

      echo $fruta . "<br />";

  }

?>
```

En este ejemplo, el bucle foreach recorre el array $frutas y asigna cada valor a la variable $fruta en cada iteración, lo que nos permite trabajar con cada elemento directamente.

Bucle while con list()

El bucle while junto con la función *list()* permite recorrer un array utilizando los valores y sus respectivas claves.

Esto es especialmente útil cuando se desea acceder a las claves y valores simultáneamente.

Ejemplo de un array recorrido con un bucle while:

```php
<?php

  $frutas = array("manzana", "banana", "naranja");

  foreach($frutas as &$fruta){

    $fruta = strtoupper($fruta);

  }

  print_r($frutas);

?>
```

En este ejemplo, el bucle while con *list()* recorre el array $frutas y asigna cada clave a la variable $nombre y cada valor a la variable $color en cada iteración.

Recomendación:

La forma más recomendable de recorrer un array en PHP es utilizando el bucle foreach.

Este método es más legible, fácil de usar y menos propenso a errores en comparación con otros bucles.

Además, el bucle foreach se adapta automáticamente al tamaño del array, lo que lo hace ideal cuando no se conoce el número de elementos.

El bucle foreach también proporciona acceso directo a los valores del array sin necesidad de utilizar índices.

Esto simplifica el código y lo hace más comprensible, especialmente en casos de arrays multidimensionales o asociativos.

En resumen, el bucle foreach es recomendable por su simplicidad, legibilidad y adaptabilidad al tamaño y tipo de arrays en PHP.

Recordá que el bucle foreach es recomendable por su simplicidad, legibilidad y adaptabilidad al tamaño y tipo de arrays en PHP.

Algunos métodos más usados en arrays

Los métodos para trabajar con arrays en PHP se pueden dividir en cuatro grandes grupos, y son los siguientes:

1. Métodos para agregar y eliminar elementos

Estos métodos permiten agregar nuevos elementos al array o eliminar elementos existentes.

- **array_push():** Agrega uno o más elementos al final del array.
- **array_pop():** Elimina y devuelve el último elemento del array.
- **array_unshift():** Agrega uno o más elementos al inicio del array.
- **array_shift():** Elimina y devuelve el primer elemento del array.
- **array_splice():** Agrega o elimina elementos desde una posición específica del array.

*Ejemplo del método **array_push()**: Este método permite agregar uno o más elementos al final de un array.*

```php
<?php

  $frutas = array("manzana", "banana");

  // Agregar elementos al final del array
  array_push($frutas, "naranja", "uva");

  print_r($frutas);

  /* Salida

    Array
  (
      [0] => manzana
      [1] => banana
      [2] => naranja
      [3] => uva
  )

  */

?>
```

*Ejemplo del método **array_pop()**: Este método elimina el último elemento del array y devuelve ese elemento.*

```php
<?php

  $frutas = array("manzana", "banana", "naranja", "uva");

  // Eliminar el último elemento del array
  $ultimaFruta = array_pop($frutas);

  echo "Última fruta eliminada: " . $ultimaFruta . "<br>";
  print_r($frutas);

  /* Salida

  Última fruta eliminada: uva

  Array
  (
      [0] => manzana
      [1] => banana
      [2] => naranja
  )

  */

?>
```

*Ejemplo del método **array_unshift()**: Este método permite agregar uno o más elementos al inicio de un array.*

```php
<?php

  $frutas = array("manzana", "banana");

  // Agregar elementos al final del array
  array_push($frutas, "naranja", "uva");

  print_r($frutas);

  /* Salida

  Array
  (
      [0] => naranja
      [1] => uva
      [2] => manzana
      [3] => banana
  )

  */
```

```
?>
```

*Ejemplo del método **array_shift()**: Este método elimina el primer elemento del array y devuelve ese elemento.*

```php
<?php

  $frutas = array("naranja", "uva", "manzana", "banana");

  // Eliminar el primer elemento del array
  $primeraFruta = array_shift($frutas);

  echo "Primera fruta eliminada: " . $primeraFruta . "<br>";

  print_r($frutas);

  /* Salida

  Primera fruta eliminada: naranja

  Array
  (
      [0] => uva
      [1] => manzana
      [2] => banana
  )

  */

?>
```

*Ejemplo del método **array_splice()**: Este método permite agregar o eliminar elementos desde una posición específica del array.*

```php
<?php

  $frutas = array("manzana", "banana", "naranja", "uva");

  // Eliminar elementos desde la posición 1 (banana) y agregar nuevos
elementos
  array_splice($frutas, 1, 2, "pera", "sandía");

  print_r($frutas);

  /* Salida

  Array
  (
      [0] => manzana
      [1] => pera
      [2] => sandía
      [3] => uva
  )

  */
```

```
?>
```

2. Métodos para ordenar y reorganizar elementos

Estos métodos permiten agregar nuevos elementos al array o eliminar elementos existentes.

- **sort():** Ordena un array en orden ascendente según su valor.
- **rsort():** Ordena un array en orden descendente según su valor.
- **asort():** Ordena un array en orden ascendente según su valor, manteniendo la asociación entre clave y valor.
- **ksort():** Ordena un array en orden ascendente según su clave.
- **arsort():** Ordena un array en orden descendente según su valor, manteniendo la asociación entre clave y valor.
- **krsort():** Ordena un array en orden descendente según su clave.

*Ejemplo del método **sort()**: Este método ordena el array en orden ascendente según los valores numéricos o alfabéticos.*

```php
<?php

  $frutas = array("banana", "manzana", "naranja", "uva");

  // Ordenar el array en orden ascendente
  sort($frutas);

  print_r($frutas);

  /* Salida

  Array
  (
      [0] => banana
      [1] => manzana
      [2] => naranja
      [3] => uva
  )

  */

?>
```

*Ejemplo del método **rsort()**: Este método ordena el array en orden descendente según los valores numéricos o alfabéticos.*

```php
<?php

  $frutas = array("banana", "manzana", "naranja", "uva");

  // Ordenar el array en orden descendente
  rsort($frutas);

  print_r($frutas);

  /* Salida

  Array
  (
      [0] => uva
      [1] => naranja
      [2] => manzana
      [3] => banana
  )

  */

?>
```

*Ejemplo del método **asort()**: Este método ordena el array asociativo en orden ascendente según los valores, manteniendo la asociación entre claves y valores.*

```php
<?php

  $frutas = array("manzana" => 2, "banana" => 4, "naranja" => 1, "uva"
=> 3);

  // Ordenar el array asociativo por valores en orden ascendente
  asort($frutas);

  print_r($frutas);

  /* Salida

  Array
  (
      [naranja] => 1
      [manzana] => 2
      [uva] => 3
      [banana] => 4
  )

  */
```

```
?>
```

*Ejemplo del método **ksort()**: Este método ordena el array asociativo en orden ascendente según las claves.*

```php
<?php

  $frutas = array("manzana" => 2, "banana" => 4, "naranja" => 1, "uva"
=> 3);

  // Ordenar el array asociativo por claves en orden ascendente
  ksort($frutas);

  print_r($frutas);

  /* Salida

  Array
  (
      [banana] => 4
      [manzana] => 2
      [naranja] => 1
      [uva] => 3
  )

  */

?>
```

*Ejemplo del método **shuffle()**: Este método mezcla aleatoriamente los elementos del array.*

```php
<?php

  $frutas = array("manzana", "banana", "naranja", "uva");

  // Mezclar los elementos del array en orden aleatorio
  shuffle($frutas);

  print_r($frutas);

  /* Salida

  Array
  (
      [0] => uva
      [1] => naranja
      [2] => banana
      [3] => manzana
  )

  */

?>
```

3. Métodos para buscar y acceder a elementos

Estos métodos permiten ordenar los elementos del array o reorganizarlos de diferentes formas.

- **in_array():** Verifica si un valor existe en el array.
- **array_search():** Busca un valor y devuelve su clave correspondiente.
- **array_key_exists():** Verifica si una clave existe en un array asociativo.
- **array_values():** Devuelve todos los valores de un array asociativo como un array indexado.
- **array_keys():** Devuelve todas las claves de un array asociativo como un array indexado.

*Ejemplo del método **in_array()**: Este método verifica si un valor dado existe dentro del array y devuelve true si se encuentra o false si no se encuentra.*

```php
<?php

  $frutas = array("manzana", "banana", "naranja", "uva");

  // Verificar si "banana" existe en el array
  if(in_array("banana", $frutas)){
      echo "La fruta 'banana' existe en el array.";
  }else{
      echo "La fruta 'banana' no existe en el array.";
  }

  /* Salida

  La fruta 'banana' existe en el array.

  */

?>
```

*Ejemplo del método **array_search()**: Este método busca un valor en el array y devuelve la clave correspondiente si se encuentra, o false si no se encuentra.*

```php
<?php

  $frutas = array("manzana", "banana", "naranja", "uva");

  // Buscar la clave correspondiente al valor "naranja"
  $clave = array_search("naranja", $frutas);

  if($clave !== false){
      echo "El valor 'naranja' se encuentra en la clave: " . $clave;
  }else{
      echo "El valor 'naranja' no se encuentra en el array.";
  }

  /* Salida

  El valor 'naranja' se encuentra en la clave: 2

  */

?>
```

*Ejemplo del método **array_key_exists()**: Este método verifica si una clave dada existe dentro del array asociativo y devuelve true si se encuentra o false si no se encuentra.*

```php
<?php

  $frutas = array("manzana" => 2, "banana" => 4, "naranja" => 1, "uva"
=> 3);

  // Verificar si la clave "manzana" existe en el array asociativo
  if(array_key_exists("manzana", $frutas)){
      echo "La clave 'manzana' existe en el array asociativo.";
  }else{
      echo "La clave 'manzana' no existe en el array asociativo.";
  }

  /* Salida

  La clave 'manzana' existe en el array asociativo.

  */
```

```
?>
```

*Ejemplo del método **array_values()**: Este método devuelve todos los valores de un array asociativo como un nuevo array indexado.*

```php
<?php

  $frutas = array("manzana" => 2, "banana" => 4, "naranja" => 1, "uva"
=> 3);

  // Obtener todos los valores del array asociativo
  $valores = array_values($frutas);

  print_r($valores);

  /* Salida

  Array
  (
      [0] => 2
      [1] => 4
      [2] => 1
      [3] => 3
  )

  */

?>
```

*Ejemplo del método **array_keys()**: Este método devuelve todas las claves de un array asociativo como un nuevo array indexado.*

```php
<?php

  $frutas = array("manzana" => 2, "banana" => 4, "naranja" => 1, "uva"
=> 3);

  // Obtener todas las claves del array asociativo
  $claves = array_keys($frutas);

  print_r($claves);

  /* Salida

  Array
  (
      [0] => manzana
      [1] => banana
      [2] => naranja
      [3] => uva
  )
```

```
   */

?>
```

4. Métodos para realizar operaciones sobre arrays

Estos métodos permiten realizar diversas operaciones o cálculos basados en los elementos del array.

- **array_sum():** Calcula la suma de los valores del array.
- **array_filter():** Filtra elementos del array según una función de devolución de llamada.
- **array_map():** Aplica una función a cada elemento del array y devuelve un nuevo array con los resultados.
- **array_reduce():** Reduce los valores del array a un solo valor utilizando una función de devolución de llamada.
- **array_merge():** Combina dos o más arrays en uno solo.

*Ejemplo del método **array_sum()**: Este método devuelve la suma de todos los valores numéricos en un array.*

```php
<?php

  $numeros = array(10, 20, 30, 40);

  // Calcular la suma de los valores del array
  $suma = array_sum($numeros);

  echo "La suma de los valores es: " . $suma;

  /* Salida

  La suma de los valores es: 100

  */

?>
```

*Ejemplo del método **array_filter()**: Este método filtra los elementos del array utilizando una función de devolución de llamada para definir el criterio de filtrado.*

```php
<?php

  $edades = array(25, 18, 35, 12, 30);

  // Filtrar edades mayores o iguales a 18
  $adultos = array_filter($edades, function ($edad){
      return $edad >= 18;
  });

  print_r($adultos);

  /* Salida

  Array
  (
      [0] => 25
      [2] => 35
      [4] => 30
  )

  */

?>
```

*Ejemplo del método **array_map()**: Este método aplica una función a cada elemento del array y devuelve un nuevo array con los resultados.*

```php
<?php

  $nums = array(1, 2, 3, 4);

  // Elevar al cuadrado cada número del array
  $resultados = array_map(function ($num) {
      return $num * $num;
  }, $nums);

  print_r($resultados);

  /* Salida

  Array
  (
      [0] => 1
      [1] => 4
      [2] => 9
      [3] => 16
  )
```

```
  */
?>
```

*Ejemplo del método **array_reduce()**: Este método reduce los valores del array a un solo valor utilizando una función de devolución de llamada.*

```php
<?php

  $nums = array(1, 2, 3, 4);

  // Sumar todos los valores del array
  $total = array_reduce($nums, function ($acumulado, $num) {
      return $acumulado + $num;
  }, 0);

  echo "La suma de todos los valores es: " . $total;

  /* Salida

  La suma de todos los valores es: 10

  */

?>
```

*Ejemplo del método **array_merge()**: Este método combina dos o más arrays en un solo array resultante.*

```php
<?php

  $frutas = array("manzana", "banana");
  $verduras = array("zanahoria", "espinaca");

  // Combinar los arrays de frutas y verduras
  $comida = array_merge($frutas, $verduras);

  print_r($comida);

  /* Salida

  Array
  (
      [0] => manzana
      [1] => banana
      [2] => zanahoria
      [3] => espinaca
  )
```

```
  */
?>
```

Recordá que los arrays son estructuras de datos fundamentales en la programación de PHP que te permiten almacenar y organizar múltiples valores bajo un mismo nombre.

Explorá los métodos disponibles para trabajar con arrays, como agregar, eliminar, ordenar y buscar elementos, y descubrí cómo utilizarlos para resolver problemas en tus proyectos de programación.

Hemos llegado así al final de esta clase en la que vimos:

- Arrays.
- Tipos de arrays.
- Formas de recorrer un array.
- Algunos métodos más usados en arrays.
-

Te esperamos en la **clase en vivo** de esta semana.
No olvides realizar el **desafío semanal**.

¡Hasta la próxima clase!

Bibliografía

Documentación Oficial de PHP:
https://www.php.net/manual/es/index.php

W3Schools: PHP Tutorial:
https://www.w3schools.com/php/default.asp

Cabezas Granado, L., González Lozano, F., (2018). Desarrollo web con PHP y MySQL. Anaya Multimedia.

Heurtel, O., (2016). Desarrolle un sitio web dinámico e interactivo. Eni Ediciones.

Welling, L., Thompson, L., (2017). Desarrollo Web con PHP y MySQL. Quinta Edición. Editorial Anaya.

Para ampliar la información

PHP: Funciones de Arrays

PHP: The Right Way

Todo sobre PHP

PHP Programming Language Tutorial - Full Course

PHP Full Course for non-haters

CURSO de php desde cero

MDN: Introducción al lado Servidor

Guía de HTML