Clase 04

Análisis de Sistemas

Materia:
Programación Web II

Docente contenidista: ROLDÁN, Hernán

Revisión: Coordinación

Contenido

Clase 4

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

En esta clase vamos a ver los siguientes temas:

- Constructores Include y Require.
- Constructores Include_Once y Require_Once.
- Funciones.
- Funciones built-in.
- Arrays (introducción).

Constructores Include y Require

include y **require** son dos constructos en PHP que se utilizan para incluir archivos dentro de otro archivo PHP. Estos constructos permiten reutilizar código y dividir la lógica del programa en diferentes archivos, lo que mejora la organización y mantenibilidad del código.

Constructor include

include es una declaración en PHP que permite incluir y ejecutar el contenido de un archivo en otro archivo PHP. Si el archivo especificado en include no se encuentra, PHP mostrará un aviso (warning) y el script continuará ejecutándose.

Ejemplo de uso del constructor include:

Supongamos que tenemos dos archivos PHP: "saludo.php" e "index.php"

Archivo "saludo.php"

```php
<?php

  $mensaje = "¡Hola, mundo!";

?>
```

Archivo "index.php"

```php
<?php

  // Incluimos el archivo saludo.php
  include 'saludo.php';

  // Ahora podemos usar la variable $mensaje que está definida en saludo.php
  echo $mensaje;

?>
```

En este ejemplo, al ejecutar "index.php" se incluirá el contenido del archivo "saludo.php" en el lugar donde se encuentra la instrucción ***include***. Como resultado, se mostrará el mensaje "¡Hola, mundo!".

Constructor require

require es similar a include, pero con una diferencia clave: si el archivo especificado en require no se encuentra, PHP mostrará un error fatal y detendrá la ejecución del script.

Ejemplo de uso del constructor require:

Supongamos que tenemos el siguiente archivo PHP llamado "config.php":

```php
<?php

  $db_host = 'localhost';
  $db_usuario = 'usuario';
  $db_clave = 'clave';
  $db_nombre = 'davinci'

?>
```

Y otro archivo PHP llamado "conexion.php":

```php
<?php

  // Requerimos el archivo config.php para obtener los datos de
conexión
  require 'config.php';

  // Aquí se establecería la conexión a la base de datos usando las
variables definidas en config.php
  // Por simplicidad, en este ejemplo no realizamos la conexión real.

?>
```

En este ejemplo, al ejecutar "conexion.php", el archivo "config.php" se requerirá con ***require***, y si "config.php" no existe o contiene algún error, se mostrará un error fatal que detendrá la ejecución del script.

Usos y Consideraciones:

- ***include*** y ***require*** son especialmente útiles cuando tienes código que deseas reutilizar en diferentes partes de tu proyecto.

- ***include*** se utiliza cuando quieres incluir un archivo y que el script continúe su ejecución, aunque el archivo no se encuentre.

- ***require*** se utiliza cuando es esencial que el archivo se incluya y su ausencia se considera un error crítico, deteniendo la ejecución del script.

- Es recomendable utilizar require cuando estés incluyendo archivos esenciales para el funcionamiento del programa, como archivos de configuración o bibliotecas externas.

- Es importante tener en cuenta que ambos constructos se pueden utilizar con rutas relativas o absolutas para especificar la ubicación del archivo a incluir.

En general, ***include*** y *require* son herramientas poderosas para mejorar la modularidad y reutilización del código en PHP.

Al usarlos adecuadamente, podrás dividir tu código en archivos más pequeños y fáciles de mantener, lo que facilitará el desarrollo y la escalabilidad de tus proyectos.

Constructores Include_Once y Require_Once

En PHP, tanto include_once como require_once se utilizan para incluir un archivo en otro archivo PHP. Ambas construcciones tienen similitudes, pero también hay diferencias importantes en su comportamiento. Aquí te explicaré las diferencias entre include_once y require_once:

include_once:

- include_once se utiliza para incluir un archivo en otro archivo PHP.
- Si el archivo incluido ya ha sido incluido previamente en el mismo script o en otro archivo mediante include_once o include, PHP no lo volverá a incluir. En otras palabras, evita la inclusión duplicada del mismo archivo.
- Si el archivo no se encuentra, PHP mostrará una advertencia, pero el script continuará ejecutándose sin interrupciones.

Ejemplo de uso de include_once.

Supongamos que tenemos el siguiente archivo PHP llamado "archivo_incluido.php":

```php
<?php

  echo "Este es un archivo incluido.";

?>
```

Y por otra parte el archivo llamado “archivo2.php”

```php
<?php

  include_once 'archivo_incluido.php';
  include_once 'archivo_incluido.php'; // No se volverá a incluir

?>
```

require_once:

- require_once funciona de manera similar a include_once, pero con una diferencia crucial: si el archivo no se encuentra, PHP mostrará un error fatal y detendrá la ejecución del script.
- Se utiliza cuando se necesita asegurar que el archivo se incluya correctamente, y su ausencia impide que el script funcione adecuadamente.

Ejemplo de uso de require_once.

Usaremos el mismo ejemplo anterior, pero cambiando solo parte del archivo llamado "archivo2".

```php
<?php

  include_once 'archivo_incluido.php';
  include_once 'archivo_incluido.php'; // No se volverá a incluir
  echo "Esta línea nunca se imprimirá debido al error anterior.";

?>
```

En resumen, la principal diferencia entre include_once y require_once radica en cómo manejan los errores cuando el archivo no se encuentra. *include_once* solo muestra una advertencia y permite que el script continúe su ejecución, mientras que *require_once* muestra un error fatal y detiene la ejecución del script.

Por lo tanto, la elección entre ambas depende de la importancia que tenga el archivo a incluir en el funcionamiento del script y si se desea una inclusión más estricta y segura.

Funciones

En programación, una función definida por el usuario (también conocida como función definida por el programador) es un bloque de código que realiza una tarea específica cuando es llamada por el programa.

Las funciones definidas por el programador son útiles porque permiten encapsular bloques de código que pueden ser llamados desde diferentes partes del programa, lo que ayuda a modularizar y organizar el código. También permiten evitar la duplicación de código, ya que una función puede ser reutilizada en diferentes partes del programa.

Una función definida por el usuario típicamente tiene un nombre que describe su propósito, y puede tomar uno o varios parámetros de entrada y puede producir una salida. El cuerpo de la función contiene las instrucciones que se ejecutan cuando la función es llamada.

En PHP, una función definida por el programador es un bloque de código reutilizable que realiza una tarea específica. Esta tarea puede ser tan simple o compleja como sea necesario.

La función se define utilizando la palabra clave "function", seguida de un nombre descriptivo y una lista de parámetros de entrada entre paréntesis.

El cuerpo de la función contiene las instrucciones que se ejecutarán cuando la función sea llamada. Estas instrucciones pueden realizar cálculos, procesar datos, interactuar con la base de datos, generar una salida en pantalla, entre otros.

La sintaxis en PHP para crear una función es la siguiente:

```
function nombreFuncion() {
  código a ejecutar;
}
```

Ejemplo de una función definida por el programador

```php
<?php

  function sumar($num1, $num2){
    $suma = $num1 + $num2;
    return $suma;
  }

  echo sumar(3, 20); // Salida: 23

?>
```

Funciones con argumentos variables

En PHP, los **"..."** se utilizan para definir una función que puede aceptar un número variable de argumentos. Esto se conoce como "argumentos variables" o "argumentos de longitud variable".

Cuando se utiliza "...", significa que la función puede aceptar cualquier cantidad de argumentos, y esos argumentos se pasarán a la función como un array.

Ejemplo de una función con argumentos variables

```php
<?php

  function sumar(...$num){
    $suma = 0;
    foreach($num as $valor){
      $suma += $valor;
    }
    return $suma;
  }

  echo sumar(14, 23, 45, 47); // Salida: 129

?>
```

En el ejemplo de arriba, los "..." indican que la función puede aceptar cualquier número de argumentos, y los argumentos se pasan como un array llamado $num.

Por lo tanto, los "..." en una función PHP son una forma de definir una función que pueda aceptar un número variable de argumentos, lo que puede ser útil en una amplia variedad de situaciones donde se

necesita flexibilidad en el número de argumentos que se pasan a una función.

Funciones built-in

Las funciones built-in, también conocidas como funciones integradas o funciones nativas, son un conjunto de funciones predefinidas y disponibles de forma predeterminada en un lenguaje de programación.

Estas funciones son parte del núcleo del lenguaje y se proporcionan como parte de su biblioteca estándar. En otras palabras, son funciones que no requieren importar librerías adicionales o realizar configuraciones especiales para utilizarlas; están listas para ser utilizadas directamente en cualquier momento.

En el caso de PHP, las funciones built-in son aquellas que vienen incorporadas en el lenguaje y se utilizan para realizar tareas comunes y fundamentales. Estas funciones son parte esencial del lenguaje y proporcionan una amplia variedad de características y utilidades que facilitan el desarrollo de aplicaciones web y otros proyectos en PHP.

Las funciones built-in de PHP se agrupan en diversas categorías según su funcionalidad, como manipulación de Strings, operaciones con Arrays, manejo de Fechas y Tiempos, conexión con Bases de Datos, validación de datos, entre otras.

Para ver todas las funciones built-in disponibles en PHP, puedes consultar la documentación oficial en el sitio web oficial de PHP (https://www.php.net/manual/es/funcref.php).
En esta sección de la documentación, encontrarás una lista completa de todas las funciones built-in de PHP, junto con una descripción detallada de su uso, los parámetros que aceptan y los valores que devuelven. Además, la documentación también incluye ejemplos de cómo utilizar cada función en diferentes contextos y escenarios.

La documentación oficial de PHP es una valiosa fuente de información para todos los programadores de PHP, ya que proporciona una guía completa y actualizada sobre el lenguaje y sus características. Es una buena práctica consultar la documentación siempre que tengas dudas sobre una función específica o cuando desees explorar nuevas funcionalidades y técnicas en PHP.

En resumen, las funciones built-in son funciones predefinidas que forman parte del núcleo del lenguaje y están disponibles de manera nativa en PHP. Proporcionan una amplia gama de herramientas para realizar tareas comunes en la programación, lo que facilita la creación de aplicaciones web y otros proyectos en PHP.

Las funciones built-in de PHP se agrupan en diferentes categorías según su funcionalidad, y algunas de las más utilizadas y relevantes son:

Funciones para manipular Strings (cadenas de texto):

- **strlen():** Devuelve la longitud de una cadena.
- **str_replace():** Reemplaza una parte de una cadena por otra.
- **substr():** Extrae una parte de una cadena.
- **strtolower():** Convierte una cadena a minúsculas.
- **strtoupper():** Convierte una cadena a mayúsculas.

Funciones para manipular Arrays:

- **count():** Cuenta el número de elementos en un array.
- **array_push():** Agrega uno o más elementos al final de un array.
- **array_pop():** Elimina y devuelve el último elemento de un array.
- **array_merge():** Combina dos o más arrays en uno solo.
- **array_search():** Busca un valor en un array y devuelve su clave correspondiente.

Funciones para trabajar con Fechas y Tiempos:

- **date():** Formatea una fecha y hora.
- **strtotime():** Convierte una fecha y hora en un timestamp Unix.
- **time():** Devuelve el timestamp Unix actual.

Funciones para la Manipulación de Archivos:

- **file_get_contents():** Lee un archivo y devuelve su contenido.
- **file_put_contents():** Escribe contenido en un archivo.
- **file_exists():** Verifica si un archivo existe.

Funciones para la Gestión de Cookies y Sesiones:

- **setcookie():** Establece una cookie para ser enviada al navegador.
- **session_start():** Inicia una nueva sesión o reanuda la sesión existente.

Funciones para la Conexión con Bases de Datos:

- **mysqli_connect():** Establece una conexión con una base de datos MySQL.
- **mysqli_query():** Ejecuta una consulta SQL en la base de datos.

Funciones para la Validación y Sanitización de Datos:

- **filter_var():** Filtra una variable con el filtro especificado.
- **htmlspecialchars():** Convierte caracteres especiales en entidades HTML.

Funciones para la Interacción con el Sistema:

- **echo():** Muestra uno o más elementos en el navegador.
- **print_r():** Muestra información legible sobre una variable.
- **die():** Detiene la ejecución del script y muestra un mensaje.

Es importante recordar que PHP es un lenguaje de programación en constante desarrollo, y nuevas funciones y mejoras se agregan con cada nueva versión.

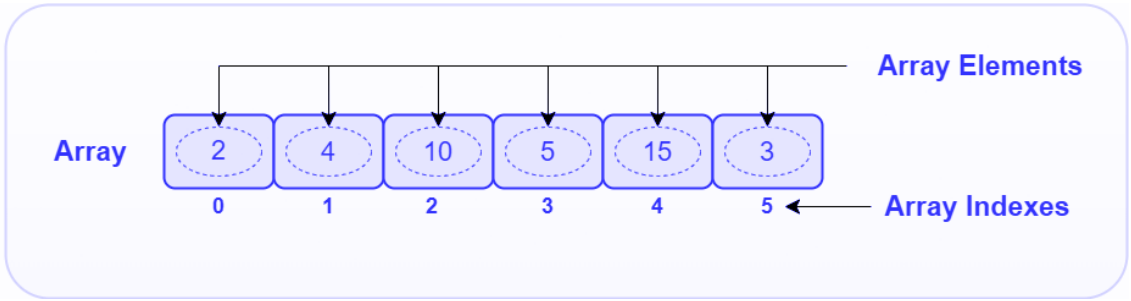
Por lo tanto, siempre es recomendable consultar la documentación oficial para obtener la información más actualizada sobre las funciones built-in de PHP y cómo utilizarlas correctamente en tus proyectos.

Arrays

En programación, un array es una estructura de datos que se utiliza para almacenar una colección de elementos del mismo tipo. Se trata de una forma eficiente de organizar y acceder a datos relacionados entre sí.

En un array, cada elemento se identifica por un índice numérico que indica su posición en la estructura. El primer elemento suele tener un índice de 0, y el último elemento se identifica por el número total de elementos menos uno.

Por ejemplo, si tuviéramos un array que almacenara los números enteros del 1 al 5, podríamos acceder al tercer elemento (el número 3) utilizando el índice 2, ya que el primer elemento tiene un índice de 0, el segundo un índice de 1, y así sucesivamente.



Imágenes extraídas de la web

Los arrays pueden tener una longitud fija o variable, dependiendo del lenguaje de programación utilizado. Además, algunos lenguajes permiten la creación de arrays multidimensionales, lo que significa que cada elemento del array puede contener a su vez otro array.

En PHP, los arrays son un tipo de datos que pueden almacenar múltiples valores en un solo objeto.

En PHP, la función array() se usa para crear una matriz.

Sintaxis:

array();

Hay tres tipos principales de arrays en PHP:

1. Arrays numéricos:

Son arrays que utilizan índices numéricos para acceder a sus elementos. Por defecto, los índices empiezan en 0 y se incrementan en 1 para cada elemento agregado.

Ejemplo de un array numérico:

```php
<?php

   $simpsons = array('Homer', 'Marge', 'Bart', 'Lisa', 'Maggie');

   echo "\n";

   echo $simpsons[2]; // Salida: Bart

   echo "\n";

   foreach($simpsons as $variable_temporal){
       echo $variable_temporal . '.' . "\n"; // Salida: Homer, Marge,
Bart, Lisa, Maggie
   }

   // Salida:
   // Homer
   // Marge
   // Bart
   // Lisa
   // Maggie

   echo "\n";

   foreach($simpsons as $posicion => $valor){
       echo 'En la posición ' . $posicion . ' del array está: ' . $valor
. '.' . "\n";
   }

   // Salida:
   // En la posición 0 del array está: Homer.
   // En la posición 1 del array está: Marge.
   // En la posición 2 del array está: Bart.
   // En la posición 3 del array está: Lisa.
   // En la posición 4 del array está: Maggie.

?>
```

2. Arrays asociativos:

Son arrays que utilizan índices alfanuméricos (en lugar de números) para acceder a sus elementos, se accede a través de su clave.

Ejemplo de un array asociativo:

```php
<?php

$avengers = array('IM' => 'Iron Man', 'SM' => 'Spider-Man', 'BW' =>
'Black Widow', 'HK' => 'Hulk', 'SW' => 'Scarlet Witch');

   echo "\n";

   // array asociativo (se accede a los elementos mediante la clave)
   foreach($avengers as $clave => $valor){
       echo 'La clave: ' . $clave . ' corresponde al Avenger: ' . $valor
. '.' . "\n";
   }

   // Salida:
   // La clave: IM corresponde al Avenger: Iron Man.
   // La clave: SM corresponde al Avenger: Spider-Man.
   // La clave: BW corresponde al Avenger: Black Widow.
   // La clave: HK corresponde al Avenger: Hulk.
   // La clave: SW corresponde al Avenger: Scarlet Witch.

?>
```

3. Arrays multidimensionales:

Son arrays que contienen arrays como elementos, permitiendo almacenar información en forma de matrices o tablas.

Ejemplo de un array multidimensional:

```php
<?php

   $simpsons = array(
       array('Homer', array(36)),
       array('Marge', array(36)),
       array('Bart', array(10)),
       array('Lisa', array(8)),
       array('Maggie', array(1))
   );

   foreach($simpsons as $c => $v){ // primera dimensión del array
       foreach($v as $c1 => $v1){ // segunda dimensión del array
           if(is_array($v1)){
               foreach($v1 as $c2 => $v2){ // tercera dimensión del array
                   echo 'Personaje: ' . $v[0] . ' ' . 'Edad: ' . $v2 .
"\n";
               }
           }
       }
   }

   // Salida:
   // Personaje: Homer Edad: 36
   // Personaje: Marge Edad: 36
   // Personaje: Bart Edad: 10
   // Personaje: Lisa Edad: 8
   // Personaje: Maggie Edad: 1

?>
```

El tema arrays lo volveremos a tratar más adelante y en profundidad, donde además estaremos viendo los métodos más usados para este tipo de dato como así también las diferentes maneras que PHP tiene para recorrerlos.

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

- Constructores Include y Require.
- Constructores Include_Once y Require_Once.
- Funciones.
- Funciones built-in.
- Arrays (introducción).

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: Include

PHP: Require

PHP: Include_Once

PHP: Require_Once

PHP: Funciones definidas por el usuario

PHP: Referencia de funciones

PHP: Arrays

Guía de HTML

Todo sobre PHP

PHP: The Right Way

PHP Programming Language Tutorial - Full Course

PHP Full Course for non-haters

CURSO de php desde cero