Clase 01

Análisis de Sistemas

Materia:
Programación Web II

Docente contenidista: ROLDÁN, Hernán

Revisión: Coordinación

Contenido

Clase 1

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

En esta clase vamos a ver los siguientes temas:

- Introducción a PHP: breve historia.
- ¿Qué puede hacerse con PHP?
- Sitios estáticos vs sitios dinámicos.
- Lenguajes de programación del lado del servidor.
- Instalación de PHP (instalación independiente).
- Instalación de entornos locales que soportan PHP.
- Editores online que soportan PHP.
- Sintaxis Básica PHP.
- Comentarios.
- Variables.

Programación PHP

Introducción a PHP: breve historia

PHP fue creado en 1994 por Rasmus Lerdorf como un conjunto de herramientas para el seguimiento de visitantes de su sitio web personal, de ahí las siglas iniciales de "Personal Home Page Tools". En 1995, Andi Gutmans y Zeev Suraski reescribieron el núcleo de PHP, lo que lo convirtió en un lenguaje más eficiente y robusto.

Desde entonces, PHP se ha utilizado ampliamente en la creación de sitios web dinámicos y ha sido adoptado por muchas empresas y organizaciones de todo el mundo.
A pesar de su evolución, el nombre "PHP" se ha mantenido como un recordatorio de sus orígenes, pero haciendo referencia al acrónimo recursivo de “PHP Hypertext Preprocessor”.

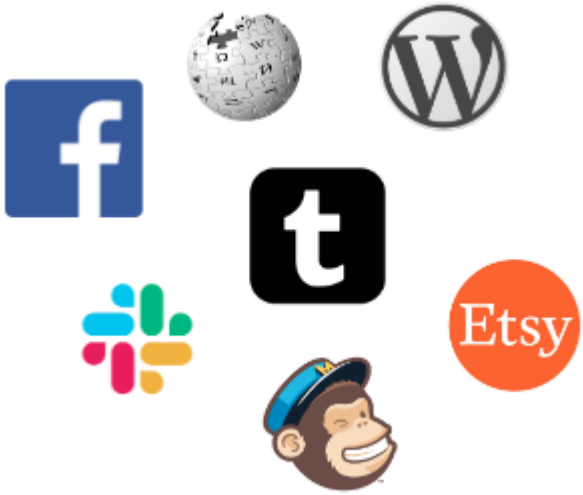
PHP es conocido por su facilidad de uso y su capacidad para integrarse con otros lenguajes y sistemas, como HTML, CSS, JavaScript y bases de datos como MySQL. Además, existe una gran comunidad de usuarios y desarrolladores de PHP, lo que significa que hay muchos recursos disponibles para aprender y resolver problemas.

Algunos de los usos más comunes de PHP incluyen:

- Desarrollo de sitios web dinámicos.
- Creación de aplicaciones web.
- Integración con bases de datos.
- Generación de contenido dinámico en tiempo real.
- Procesamiento de formularios y envío de correos electrónicos.

Algunas de las empresas conocidas que utilizan PHP son:

- Facebook
- Wikipedia
- Yahoo
- Etsy
- Tumblr
- WordPress
- Netflix
- Lyft
- Slack
- Square



Imágenes extraídas de la web

Estas empresas han adoptado PHP debido a su facilidad de uso, flexibilidad y capacidad para integrarse con otros sistemas y lenguajes.

Además, dada la gran comunidad de usuarios y desarrolladores de PHP implica que hay muchos recursos disponibles para aprender y resolver problemas.

En resumen, PHP es un lenguaje de programación versátil y popular que ha demostrado ser una herramienta valiosa para la creación de sitios web dinámicos y aplicaciones web.

¿Qué puede hacerse con PHP?

Con PHP, se pueden hacer muchas cosas, incluyendo:

- Desarrollo de sitios web dinámicos.
- Procesamiento de formularios.
- Conexión a bases de datos.
- Creación de scripts de administración de contenido.
- Generación de páginas dinámicas con contenido almacenado en una base de datos.
- Creación de aplicaciones web como tiendas en línea, foros, blogs, etc.
- Procesamiento de imágenes y archivos multimedia.

Estas son solo algunas de las muchas cosas que se pueden hacer con PHP. Es uno de los lenguajes de programación más populares y ampliamente utilizados en el mundo para el desarrollo de aplicaciones web.

Sitios estáticos vs sitios dinámicos

Un sitio web estático es aquel cuyo contenido se muestra de la misma manera para todos los usuarios y no cambia con el tiempo.

Es un conjunto de archivos HTML, CSS y JavaScript que se sirven directamente al cliente desde un servidor web. Los sitios web estáticos son simples de crear y mantener, ya que no requieren una interacción compleja con el usuario ni una base de datos.

Sin embargo, su contenido no se puede personalizar para cada usuario y no admiten actualizaciones en tiempo real.

Por otro lado, un sitio web dinámico es aquel cuyo contenido puede cambiar en función de las interacciones del usuario y otros datos variables. Utiliza tecnologías de programación en el lado del servidor, como PHP, Python o Ruby, para generar contenido dinámicamente antes de enviarlo al cliente.

Los sitios web dinámicos pueden acceder a bases de datos para recuperar y mostrar información actualizada, permitiendo a los usuarios interactuar con formularios, iniciar sesión, realizar compras en línea y más.

También pueden adaptarse a las preferencias del usuario y personalizar el contenido según sus necesidades.

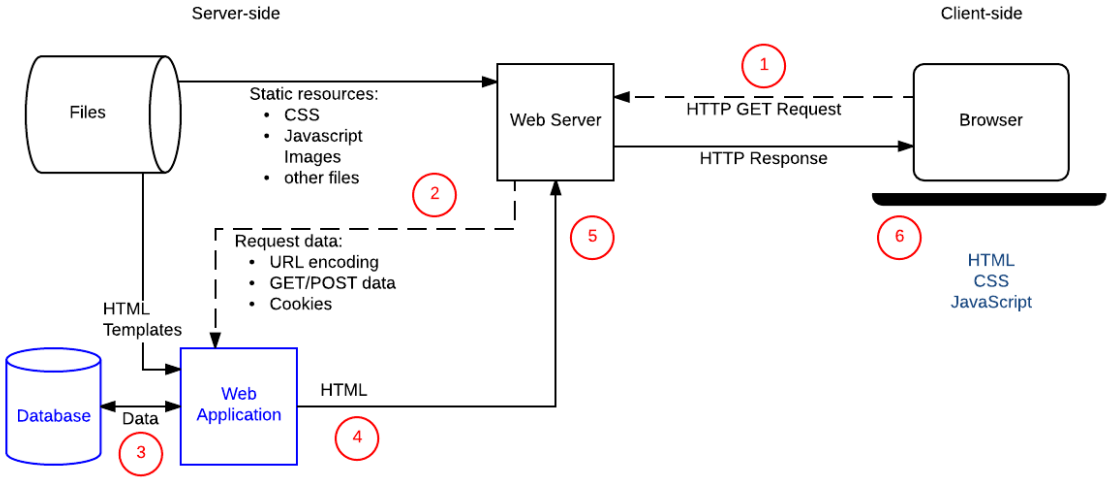


Imagen extraída de la web

Lenguajes de programación del lado del servidor

Un lenguaje de programación server-side (del inglés, "lado del servidor") es aquel que se utiliza para crear aplicaciones web que se ejecutan en el servidor. Esto significa que el código escrito en este lenguaje se ejecuta en el servidor, en lugar de en el navegador web del usuario.

Cuando un usuario solicita una página web que utiliza un lenguaje server-side, el servidor ejecuta el código y envía al usuario una versión "renderizada" de la página web. Por ejemplo, si el lenguaje server-side utilizado es PHP, el servidor ejecutará el código PHP y enviará al usuario una página HTML generada por el servidor.

El proceso en etapas que realiza un lenguaje server-side básicamente es el siguiente:

1. El cliente realiza una solicitud (por ejemplo, haciendo click en un enlace o enviando un formulario) a una página web alojada en un servidor.
2. El servidor recibe la solicitud y ejecuta el código server-side correspondiente al archivo que se ha solicitado (por ejemplo, un archivo PHP).
3. El código server-side procesa la solicitud, accede a la base de datos, manipula los datos, realiza cálculos y, en general, genera una respuesta dinámica que se ajusta a las necesidades del cliente.
4. La respuesta generada por el código server-side se envía de vuelta al cliente en forma de una página web renderizada que contiene HTML, CSS y JavaScript, pero que ha sido generada dinámicamente por el servidor a partir del código server-side.
5. El cliente recibe la página web renderizada y la muestra en el navegador web.

El siguiente código es un ejemplo sencillo de cómo se podría programar en PHP lo anteriormente citado:

Este proceso se repite cada vez que el cliente realiza una nueva solicitud a una página web alojada en el servidor, permitiendo que la página web sea actualizada y modificada dinámicamente en función de las acciones del usuario y de los datos que se almacenan en el servidor.

Los lenguajes de programación server-side se utilizan para desarrollar aplicaciones web dinámicas, donde el contenido de la página se genera en tiempo real en función de la interacción del usuario o de la información almacenada en una base de datos.
Los lenguajes de programación server-side más comunes son PHP, Python, Ruby, Java, y JavaScript (utilizado a menudo en su forma de Node.js).

Un ejemplo de programación del lado del servidor con PHP sería la creación de un formulario de registro de usuarios que, al enviar la información al servidor, éste la procesa y almacena en una base de datos.

Ejemplo de un formulario HTML:

```
<html>
 <head>
   <title>Formulario PHP</title>
 </head>
 <body>
   <h1>Formulario de Registro</h1>
   <form action="registro.php" method="post">
     <label for="nombre">Nombre:</label>
     <input type="text" name="nombre" id="nombre" required>
     <br>
     <label for="email">Email:</label>
     <input type="email" name="correo" id="correo" required>
     <br>
     <label for="clave">Contraseña:</label>
     <input type="password" name="clave" id="clave" required>
     <br>
     <input type="submit" value="Registrar">
   </form>
 </body>
</html>
```

Creación del archivo "registro.php" que procesa la información enviada por el formulario:

```php
<?php

   // Recupera los datos enviados en el formulario
   $nombre = $_POST['nombre'];
   $email = $_POST['correo'];
   $clave = $_POST['clave'];

   // Establece la conexión con la base de datos
   $conexion = mysqli_connect('localhost', 'usuario_bd', 'clave_bd',
'basedatos');

   // Si se genera un error durante la conexión muestra un mensaje y
detiene la ejecución del programa
   if(mysqli_connect_errno()){
        echo 'Se produjo un error durante la conexión a la base de
datos.' . mysqli_connect_error();
        exit();
   }

   // Si no, inserta los datos del usuario en la base de datos
   else{

  // Consulta SQL
        $query = "INSERT INTO usuarios (nombre, email, clave) VALUES
('$nombre', '$correo', '$clave')";

   // Ejecuta la consulta SQL contra la base de datos
   mysqli_query($conexion, $query);

   // Cierra la conexión a la base de datos
   mysqli_close($conexion);

   // Redirecciona al usuario a una página de confirmación
   header('Location: confirmacion.php');

   }

?>
```

Este es solo un ejemplo básico de programación del lado del servidor con PHP. En aplicaciones más complejas se utilizan múltiples archivos PHP para manejar diferentes funciones y procesos, y se pueden integrar otros lenguajes de programación y tecnologías como bases de datos, frameworks, APIs, etc.

Instalación de PHP (instalación independiente)

Para usar PHP, se necesita tener instalado lo siguiente:

1. **Un servidor web:**
   PHP se ejecuta en un servidor web, como Apache o Nginx, y es el servidor web quien se encarga de procesar los scripts PHP y enviar la respuesta al navegador del usuario.

2. **PHP:**
   La versión más reciente de PHP puede ser descargada desde la página oficial de descarga: https://windows.php.net/download/.

3. **Un navegador web:**
   Para visualizar las páginas generadas por PHP, se necesita un navegador web, como Google Chrome, Mozilla Firefox o Microsoft Edge.

4. **Un editor de texto o entorno de desarrollo integrado (IDE):**
   Para escribir y editar los scripts PHP. Algunos ejemplos de editores de texto son Notepad++ en Windows o Sublime Text en Windows, macOS y Linux. Algunos ejemplos de IDEs para PHP son PhpStorm o Visual Studio Code.

Además, si se desea trabajar con bases de datos, se necesitará tener instalado un gestor de bases de datos, como MySQL o MariaDB, y configurar PHP para trabajar con el gestor de base de datos elegido.

Instalación de entornos locales que soportan PHP

Hay varios paquetes de instalación de entornos locales que incluyen PHP y todas las herramientas necesarias para el desarrollo de aplicaciones web:

- **XAMPP:**
  Es un paquete de instalación de código abierto que incluye Apache, PHP, MySQL, y otros componentes necesarios para el desarrollo de aplicaciones web. Está disponible para Windows, macOS y Linux.

- **WAMP:**
  Es un paquete de instalación para Windows que incluye Apache, PHP y MySQL.

- **LAMP:**
  Es un paquete de instalación para sistemas operativos basados en Unix, como Linux, que incluye Apache, PHP y MySQL.

- **MAMP:**
  Es un paquete de instalación para macOS que incluye Apache, PHP y MySQL.

- **Vagrant:**
  Es una herramienta para la creación y gestión de entornos virtuales que incluye la posibilidad de instalar y configurar Apache, PHP, MySQL y otros componentes necesarios para el desarrollo de aplicaciones web.

Estos paquetes de instalación son muy útiles para ahorrar tiempo y esfuerzo en la configuración de un entorno de desarrollo local, ya que incluyen todos los componentes necesarios y la configuración necesaria para poner en marcha un servidor web local en poco tiempo.

Editores online que soportan PHP

Otra opción al momento de programar en PHP es buscar editores en línea que soportan este lenguaje de programación, en Internet existen varios, y algunos de estos son:

- **CodePen:** Una plataforma en línea para compartir y probar pequeños fragmentos de código. Tiene una amplia comunidad de usuarios y permite la colaboración en tiempo real.

- **Repl.it:** Un editor de código en línea que soporta una amplia variedad de lenguajes, incluido PHP. Ofrece un entorno de desarrollo en línea intuitivo y fácil de usar.

- **JSFiddle:** Una herramienta en línea para probar y compartir fragmentos de código JavaScript, HTML y CSS. Permite la ejecución en tiempo real y la colaboración con otros usuarios.

- **PHPFiddle:** Un editor en línea especializado en PHP con soporte para diversas versiones del lenguaje. Ofrece una interfaz intuitiva y una variedad de características para ayudar a los desarrolladores a escribir y probar su código.

- **GitHub Codespaces:** Un editor de código en línea que permite a los usuarios trabajar en proyectos de forma colaborativa. Soporta PHP y una amplia variedad de otros lenguajes.

- **OnlineGDB:** Un editor en línea que permite a los usuarios escribir, compilar y ejecutar su código en un entorno de desarrollo en línea. Además de PHP, también soporta otros lenguajes como C, C++, Java y más.

- **Bitbucket Cloud:** Una plataforma en línea para alojar y colaborar en proyectos de software. Ofrece un editor en línea integrado para PHP y otros lenguajes.

Estos son solo algunos ejemplos de los mejores editores de PHP en línea gratuitos disponibles. Hay muchas otras opciones disponibles, por lo que es importante investigar y elegir el que mejor se adapte a sus necesidades y preferencias.

Sintaxis Básica PHP

La sintaxis básica de PHP incluye los siguientes componentes:

1. Etiquetas de apertura y cierre: el código PHP se escribe entre las etiquetas <?php y ?>.

```php
<?php

   // Código PHP aquí

?>
```

2. **Comentarios:** se pueden agregar comentarios con // para una sola línea o con /* y */ para múltiples líneas.

```php
<?php

   // Este es un comentario de una sola línea

   /*
   Este es un comentario de
   múltiples líneas
   */

?>
```

3. **Variables:** se declaran con el signo $ seguido por el nombre de la variable.

```php
<?php

   // Declaración de una variable
   $nombre = "John Doe";
   $edad = 30;
   $salario = 3500.50;

?>
```

4. **Sentencias:** se utilizan para controlar el flujo del programa, como if, else, for, while, etc.

```php
<?php

    $edad = 30;

   // Sentencia if-else
   if($edad >= 18){
       echo "Eres mayor de edad";
   }
   else{
       echo "Eres menor de edad";
   }

   // Sentencia for
   for($i = 0; $i < 5; $i++){
       echo "Iteración " . $i . "<br>";
   }

   // Sentencia while
   $contador = 0;
   while($contador < 5){
       echo "Iteración " . $contador . "<br>";
       $contador++;
   }

?>
```

5. **Funciones:** son bloques de código que se pueden llamar para realizar una tarea específica.

```php
<?php

   // Definición de una función
   function saludar($nombre){
       echo "Hola, " . $nombre;
   }

   // Llamada a la función
   saludar("John Doe");

?>
```

6. **Operadores:** se utilizan para realizar operaciones matemáticas y comparaciones, como +, -, *, /, ==, etc.

```php
<?php

  // Asignación
   $a = 10;
   $b = 5;
   $a = $b;
   echo $a; // 5

   // Suma
   echo $a + $b; // 15

   // Resta
   echo $a - $b; // 5

   // Multiplicación
   echo $a * $b; // 50

   // División
   echo $a / $b; // 2

   // Módulo (resto de la división)
   echo $a % $b; // 0

   // Exponenciación
   echo $a ** $b; // 100000

  // Igual a
   echo $a == $b; // false
```

```php
    // No igual a
    echo $a != $b; // true

    // Mayor que
    echo $a > $b; // true

    // Menor que
    echo $a < $b; // false

    // Mayor o igual que
    echo $a >= $b; // true

    // Menor o igual que
    echo $a <= $b; // false

    // Idéntico
    echo $a === $b; // false

    // No idéntico
    echo $a !== $b; // true

?>
```

Comentarios

Los comentarios en PHP son líneas de texto que no son interpretadas como código. Son utilizados para agregar explicaciones y notas dentro del código para facilitar su comprensión y mantenimiento.

Hay dos tipos de comentarios en PHP:

1. **Comentarios de una línea:**
   Son representados por dos barras diagonales (//) al comienzo de una línea, o por el símbolo de gato (#).
2. **Comentarios de varias líneas:**
   Son representados por /* y */.

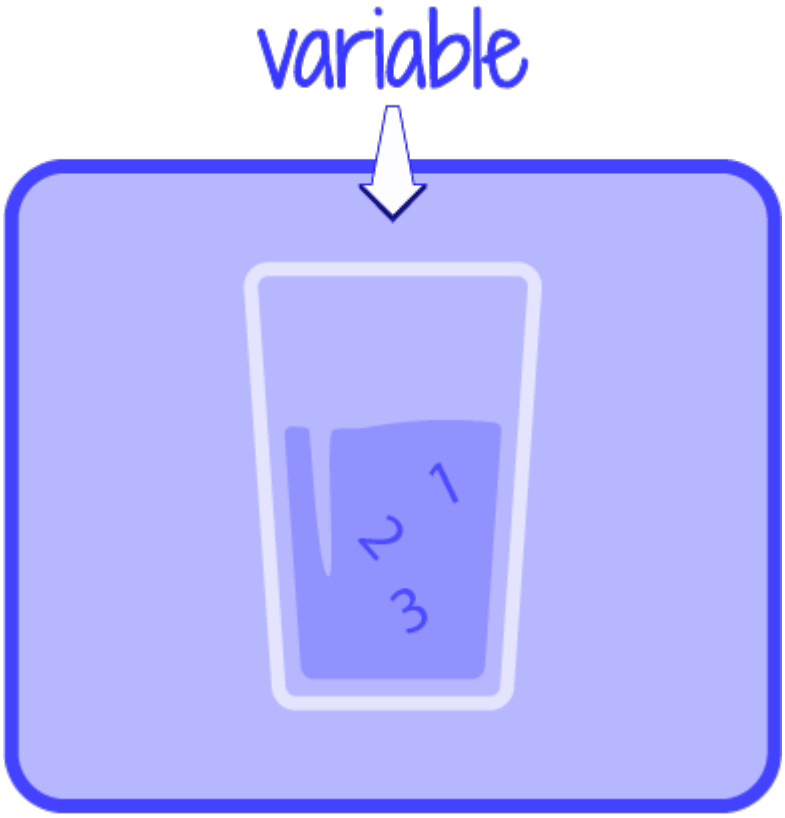
La finalidad de los comentarios es mejorar la legibilidad y documentación del código, facilitando su mantenimiento y actualización.

```php
<?php

   // Este es un comentario de una única línea.
   # Este es otro comentario de una única línea.

   /*
   Este es un
   comentario
   de múltiples líneas.
   */

?>
```

Variables

Las variables en PHP son espacios de almacenamiento que pueden contener valores de diferentes tipos, como números, cadenas de texto, arreglos, objetos, etc.

Pensá en las variables como contenedores para almacenar datos.



Imágenes extraídas de la web

En PHP, el nombre de una variable debe comenzar con un signo de dólar ($), seguido por una o más letras, números o guiones bajos.

Una variable puede tener un nombre corto como “x” e “y” o un nombre más descriptivo como “edad”, “modelo_auto”, “nombre”, “apellido”, “total”, etc).

Las reglas de las variables PHP son las siguientes:

- **Una variable comienza con el signo dólar ($), seguido del nombre de la variable.**
- **Una variable debe comenzar con una letra o guión bajo.**
- **Una variable no puede comenzar con un número.**
- **Una variable solo puede contener caracteres alfanuméricos y guión bajo (A-z, 0-9, y _).**
- **Los nombres de las variables son sensibles a mayúsculas ($edad y $EDAD son dos variables diferentes).**

Aquí, hemos creado tres variables: $nombre, $edad y $sueldo, y le hemos asignado valores.
Por ejemplo:

```php
<?php

   $nombre = "Juan";
   $edad = 30;
   $sueldo = 1500.50;

?>
```

El valor de una variable puede ser modificado en cualquier momento durante la ejecución de un script.
Por ejemplo:

```php
<?php

  $nombre = "Juan";
  $edad = 30;

  $nombre = "María";
  $edad = $edad + 5;

  echo $nombre; // Salida: María
  echo $edad; // Salida: 35

?>
```

Además de los tipos básicos como números y cadenas, PHP también permite trabajar con arreglos y objetos. Un arreglo es una colección de valores indexados o asociativos.
Por ejemplo:

```php
<?php

   $colores = array("rojo", "verde", "azul");

   echo $colores[0]; // Salida: rojo;

?>
```

Un objeto es una instancia de una clase y puede tener propiedades y métodos.
Por ejemplo:

```php
<?php

   class Coche{
       public $marca;
       public $modelo;

       function arrancar() {
         echo "El coche está arrancando.";
       }
     }

     $miCoche = new Coche();
     $miCoche->marca = "Toyota";
     $miCoche->modelo = "Corolla";

     echo $miCoche->marca; // Salida: Toyota
     $miCoche->arrancar(); // Salida: El coche está arrancando.

?>
```

En resumen, las variables son una parte fundamental del lenguaje PHP y son ampliamente utilizadas para almacenar y manipular datos.

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

- Introducción a PHP: breve historia.
- ¿Qué puede hacerse con PHP?
- Sitios estáticos vs sitios dinámicos.
- Lenguajes de programación del lado del servidor.
- Instalación de PHP (instalación independiente).
- Instalación de entornos locales que soportan PHP.
- Editores online que soportan PHP.
- Sintaxis Básica PHP.
- Comentarios.
- Variables.

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

MDN: Introducción al lado Servidor:
https://developer.mozilla.org/es/docs/Learn/Server-side/First_steps/Introduction

Guía de HTML:
https://developer.mozilla.org/es/docs/Learn/HTML

Todo sobre PHP:
https://desarrolloweb.com/home/php#track22

PHP: The Right Way:
https://phptherightway.com/

PHP Programming Language Tutorial - Full Course:
https://www.youtube.com/watch?v=OK_JCtrrv-c&t=6s

PHP Full Course for non-haters:
https://www.youtube.com/watch?v=zZ6vybT1HQs

CURSO de php desde cero:
https://www.youtube.com/watch?v=nCB1gEkRZ1g

Clase 02

Análisis de Sistemas

Materia:
Programación Web II

Docente contenidista: ROLDÁN, Hernán

Revisión: Coordinación

Contenido

Clase 2

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

En esta clase vamos a ver los siguientes temas:

- Alcance o ámbito de las variables.
- Constructores del lenguaje echo y print.
- Tipos de datos.
- Strings.
- Funciones de strings.
- Números.
- Casteo.

Alcance o ámbito de las variables

El ámbito de una variable en PHP se refiere al contexto en el que una variable está definida y en el que es accesible.
El ámbito de una variable es muy similar a su alcance, y en algunos casos los términos son intercambiables.

El ámbito de una variable puede ser global o local, dependiendo de dónde se define. Las variables definidas fuera de cualquier función o método tienen un ámbito global, lo que significa que pueden ser accedidas y modificadas en cualquier parte del script.

Por otro lado, las variables definidas dentro de una función o método tienen un ámbito local, lo que significa que solo pueden ser accedidas y modificadas dentro de esa función o método.

Es importante tener en cuenta el ámbito de las variables al diseñar y escribir código en PHP, ya que puede afectar el comportamiento y la legibilidad del mismo.

Al trabajar con variables de ámbito global, es importante utilizarlas de manera responsable y evitar la sobreescritura accidental de valores. Por otro lado, al trabajar con variables de ámbito local, es importante tener en cuenta que no se comparten entre diferentes funciones o métodos y que pueden tener diferentes valores en diferentes partes del script.

Alcance global:
Una variable con alcance global puede ser accedida y modificada en cualquier parte del script, incluyendo dentro de funciones y métodos. Para hacer que una variable tenga alcance global, debe ser declarada fuera de cualquier función o método con la palabra clave "global".

Ejemplo de una variable con alcance global

```php
<?php

   $numero = 10;

   function multiplicarPorDos(){
   global $numero;
   $numero *= 2;
   }

   multiplicarPorDos();
   echo $numero; // Salida: 20

?>
```

Alcance local:
Una variable con alcance local solo puede ser accedida y modificada dentro de la función o método en la que fue declarada. Por defecto, todas las variables son de alcance local dentro de una función o método.

Ejemplo de una variable con alcance local

```php
<?php

   function mostrarNumero(){
       $numero = 5;
       echo $numero; // 5
   }

   mostrarNumero();
   echo $numero; // Salida: Notice: Undefined variable

?>
```

Constructores del lenguaje echo y print

Las funciones echo y print son dos maneras de imprimir o mostrar información en una página web usando PHP.

"echo" es una función más rápida y permite imprimir múltiples argumentos separados por comas.

Ejemplo de uso del constructor echo:

```php
<?php

   echo "Hola";
   echo " mundo";

   // Salida: Hola mundo

?>
```

Ejemplo de uso del constructor echo con múltiples argumentos:

```php
<?php

   echo "Hola,", " mundo";

  // Salida: Esto imprimirá: "Hola, mundo".

?>
```

Es importante destacar que los argumentos deben separarse con una coma (,) y no con un punto y coma (;).

“print” es una función más lenta y solo permite imprimir un argumento. Además, print devuelve 1 como resultado, lo que lo hace útil para usar en expresiones.

Ejemplo de uso del constructor print:

```php
<?php

   print "Hola mundo"; // Salida: Hola mundo

?>
```

En general, se recomienda usar echo ya que es más rápido y flexible que print.

Tipos de datos

PHP soporta varios tipos de datos, incluyendo:

- **Booleanos:** valores lógicos verdadero (true) o falso (false).
- **Números enteros:** valores numéricos sin parte decimal.
- **Números de punto flotante:** valores numéricos con parte decimal.
- **Cadenas de caracteres:** secuencias de uno o más caracteres entre comillas simples o dobles.
- **Arreglos:** estructuras de datos que contienen una colección de valores, cada uno accesible a través de un índice o clave.
- **Objetos:** estructuras de datos que representan entidades con propiedades y métodos.
- **Recursos:** valores especiales que representan una conexión externa, como un archivo o una conexión a una base de datos.
- **Null:** un valor especial que representa la ausencia de un valor.

Ejemplo de tipo de dato booleano

```php
<?php

   $es_verdadero = true;
   $es_falso = false;

?>
```

Ejemplo de tipo de dato números enteros

```php
<?php

   $entero_positivo = 42;
   $entero_negativo = -42;

?>
```

Ejemplo de tipo de dato números de punto flotante

```php
<?php

   $flotante_positivo = 3.14;
   $flotante_negativo = -3.14;

?>
```

Ejemplo de tipo de dato cadena de caracteres

```php
<?php

   $cadena_simple = 'Hola mundo';
   $cadena_doble = "Hola mundo";

?>
```

Ejemplo de tipo de dato arreglos

```php
<?php

   $colores = array("rojo", "verde", "azul");
   $personas = array("nombre" => "Juan", "edad" => 30);

?>
```

Ejemplo de tipo de dato objetos

```php
<?php

   class Persona {
       public $nombre;
       public $edad;

       function __construct($nombre, $edad){
           $this->nombre = $nombre;
           $this->edad = $edad;
       }
   }

   $persona = new Persona("Juan", 30);

?>
```

Ejemplo de tipo de dato recursos

```php
<?php

   $archivo = fopen("datos.txt", "r");
   $conexion_db = mysqli_connect("localhost", "usuario", "clave",
"basededatos");

?>
```

Ejemplo de tipo de dato null

```php
<?php

   $valor_nulo = null;

?>
```

En resumen, PHP soporta varios tipos de datos, incluyendo booleanos, números enteros y de punto flotante, cadenas de caracteres, arreglos, objetos, recursos y null, lo que lo hace versátil y útil para una amplia gama de aplicaciones web.

Algo importante a saber es que PHP es un lenguaje de programación débilmente tipado (Loosely Typed Language). Esto significa que PHP asocia automáticamente un tipo de datos a la variable, dependiendo de su valor. Dado que los tipos de datos no se establecen en un sentido estricto, puede hacer cosas como agregar una cadena a un número entero sin causar un error.

En otras palabras, las variables en PHP pueden almacenar cualquier tipo de datos, sin tener que ser declaradas previamente con un tipo específico.

Por ejemplo, en un lenguaje de programación fuertemente tipado, como Java o C#, una variable debe ser declarada con un tipo específico antes de poder ser usada:

```
public class DatosPersonales {
   public static void main(String[] args) {
      String nombre = "Juan";
      int edad = 30;

      System.out.println("Mi nombre es " + nombre + " y tengo " + edad
+ " años.");
   }
}
```

En PHP, sin embargo, las variables no necesitan ser declaradas previamente con un tipo específico y pueden contener cualquier tipo de datos:

```php
<?php

   $numero = 42;
   $mensaje = "Hola mundo";

?>
```

Esta característica puede ser conveniente en algunos casos, pero también puede llevar a errores y confusiones si los programadores no prestan atención al tipo de datos que están manejando. Por lo tanto, aunque PHP es un lenguaje de programación débilmente tipado, es importante que los programadores comprendan y utilicen adecuadamente los tipos de datos en sus aplicaciones.

Strings

En programación, una "cadena" o "string" es una secuencia de caracteres, que pueden ser letras, números, símbolos o una combinación de ellos. Los strings se utilizan para representar información de texto, como nombres, palabras o frases. Las cadenas se delimitan entre comillas simples ('...') o comillas dobles ("...") en muchos lenguajes de programación, incluido PHP.

Funciones de strings

Las funciones de cadenas en PHP son un conjunto de funciones que se utilizan para manipular y trabajar con cadenas de texto. Estas funciones permiten realizar acciones como concatenar dos o más cadenas, buscar una subcadena dentro de una cadena, convertir una cadena a mayúsculas o minúsculas, recortar una parte de una cadena, calcular la longitud de una cadena, entre otras acciones.

Estas funciones son útiles en muchas situaciones, como cuando se trabaja con formularios de entrada de datos, se manipulan nombres de usuario, se procesan direcciones de correo electrónico, entre otros. Al utilizar estas funciones, es posible realizar operaciones complejas con cadenas de forma más sencilla y eficiente.

Algunos ejemplos de funciones de strings son:

- **strlen():** Devuelve la longitud de una cadena.
- **str_word_count():** Devuelve el número de palabras en una cadena.
- **strrev():** Invierte el orden de los caracteres en una cadena.
- **strpos():** Busca la primera aparición de una subcadena en una cadena y devuelve su posición.
- **str_replace():** Reemplaza una parte de una cadena con otra.
- **substr():** Devuelve una parte de una cadena.
- **strtolower():** Convierte una cadena a minúsculas.
- **strtoupper():** Convierte una cadena a mayúsculas.

- **trim():** Elimina los espacios en blanco al principio y al final de una cadena.
- **ltrim():** Elimina los espacios en blanco al principio de una cadena.
- **rtrim():** Elimina los espacios en blanco al final de una cadena.
- **explode():** Divide una cadena en un array en función de un delimitador.
- **implode():** Une los elementos de un array en una cadena, utilizando un delimitador especificado.
- **str_split():** Divide una cadena en un array de caracteres.
- **nl2br():** Inserta una etiqueta HTML
 antes de cada salto de línea en una cadena.
- **htmlentities():** Convierte todos los caracteres aplicables a entidades HTML.

*Ejemplo de uso de la función **strlen**:*

```php
<?php

   $str = "Hola mundo";
   echo strlen($str); // Salida: 11

?>
```

*Ejemplo de uso de la función **str_word_count**:*

```php
<?php

   $str = "Hola mundo";
   echo str_word_count($str); // Salida: 2

?>
```

*Ejemplo de uso de la función **strrev**:*

```php
<?php

   $str = "Hola mundo";
   echo strrev($str); // Salida: odnum aloH

?>
```

*Ejemplo de uso de la función **strpos**:*

```php
<?php

   $str = "Hola mundo";
   echo strpos($str, "mundo"); // Salida: 5

?>
```

*Ejemplo de uso de la función **str_replace**:*

```php
<?php

   $str = "Hola mundo";
   echo str_replace("mundo", "PHP", $str); // Salida: Hola PHP

?>
```

*Ejemplo de uso de la función **substr**:*

```php
<?php

   $str = "Hola mundo";
   echo substr($str, 5); // Salida: mundo

?>
```

*Ejemplo de uso de la función **strtolower**:*

```php
<?php

   $str = "HOLA MUNDO";
   echo strtolower($str); // Salida: hola mundo

?>
```

*Ejemplo de uso de la función **strtoupper**:*

```php
<?php

   $str = "hola mundo";
   echo strtoupper($str); // Salida: HOLA MUNDO

?>
```

*Ejemplo de uso de la función **trim:***

```php
<?php

   $str = "   Hola mundo   ";
   echo trim($str); // Salida: Hola mundo

?>
```

** En este ejemplo, se eliminaron los espacios de ambos lados.*

*Ejemplo de uso de la función **ltrim:***

```php
<?php

   $str = "   Hola mundo   ";
   echo ltrim($str); // Salida: Hola mundo

?>
```

** En este ejemplo, solo se eliminaron los espacios de la izquierda.*

*Ejemplo de uso de la función **rtrim:***

```php
<?php

   $str = "   Hola mundo   ";
   echo rtrim($str); // Salida: Hola mundo

?>
```

** En este ejemplo, solo se eliminaron los espacios de la derecha.*

*Ejemplo de uso de la función **explode:***

```php
<?php

   $str = "Hola mundo";
   $arr = explode(" ", $str);
   print_r($arr); // Salida: Array ( [0] => Hola [1] => mundo )

?>
```

*Ejemplo de uso de la función **implode:***

```php
<?php

   $arr = array("Hola", "mundo");
   $str = implode(" ", $arr);
   echo $str; // Salida: Hola mundo

?>
```

*Ejemplo de uso de la función **str_split:***

```php
<?php

   $str = "Hola mundo";
   $arr = str_split($str);
   print_r($arr);
   /* Salida:
    Array
    (
      [0] => H
      [1] => o
      [2] => l
      [3] => a
      [4] =>
      [5] => m
      [6] => u
      [7] => n
      [8] => d
      [9] => o
    )
  */

?>
```

*Ejemplo de uso de la función **nl2br:***

```php
<?php

   $str = "Hola mundo\nAdiós mundo";
   echo nl2br($str);

   /* Salida:
    Hola mundo
    Adiós mundo
   */

?>
```

Ejemplo de uso de la función ***htmlentities:***

```php
<?php

   $str = "<b>Hola mundo</b>";
   echo htmlentities($str); // Salida: &lt;b&gt;Hello World&lt;/b&gt;

?>
```

Estos son solo algunos de los métodos más comunes para el manejo de strings en PHP, hay muchos más disponibles y que se pueden encontrar en la sección PHP: Funciones de Strings en la documentación oficial del lenguaje.

Números

En programación, existen dos tipos principales de números: **enteros (integers) y flotantes (floats).**

1. **Números enteros:**
   un número entero es un número sin decimales, como -3, 0, 1, 100, etc. En la mayoría de los lenguajes de programación, los números enteros se representan mediante una secuencia de dígitos sin decimales. En PHP, se pueden representar tanto mediante decimales como hexadecimales.

2. **Números flotantes:**
   un número flotante es un número con decimales, como -3.14, 0.0, 1.23, 100.01, etc. Los números flotantes se representan mediante una secuencia de dígitos con un punto decimal, y se utilizan para representar números con una precisión más fina que los números enteros.

*Ejemplo de números enteros**:***

```php
<?php

   $int1 = 10;
   $int2 = -20;

   echo "Int1: $int1"; // Salida: Int1: 10
   echo "<br>Int2: $int2"; // Salida: Int2: -20

?>
```

*Ejemplo de números flotantes**:***

```php
<?php

   $float1 = 10.5;
   $float2 = -20.25;

   echo "Float1: $float1"; // Salida: Float1: 10.5
   echo "<br>Float2: $float2"; // Salida: Float2: -20.25

?>
```

Función is_numeric:
La función is_numeric en PHP se utiliza para determinar si un valor dado es un número o no. Devuelve TRUE si el valor es un número, y FALSE en caso contrario.

*Ejemplo de uso función **is_numeric:***

```php
<?php

   $num1 = "100";
   $num2 = 200;
   $num3 = "Hello";

   echo "Num1 is numeric: " . is_numeric($num1);
   // Salida: Num1 is numeric: 1

   echo "<br>Num2 is numeric: " . is_numeric($num2);
   // Salida: Num2 is numeric: 1

   echo "<br>Num3 is numeric: " . is_numeric($num3);
   // Salida: Num3 is numeric:

?>
```

Sin embargo, en el ejemplo anterior la variable $num1 contiene una cadena (“100”) pero de todas formas la función devuelve TRUE, esto se debe a que para que PHP pueda interpretar a esa variable como numérica es necesario antes convertirla a integer.

Entonces, aplicando la conversión de tipos de datos al ejemplo anterior, quedaría así:

```php
<?php

   $num1 = "100";

   if (is_numeric($num1)){
       $num1 = (int)$num1;
       echo "Num1 después de casteo: " . $num1 . "\n";
       // Salida: Num1 después de casteo: 100
   }else{
       echo "Num1 no es un número.\n";
   }
```

```
?>
```

Casteo

El casteo o casting en programación es la acción de convertir un valor de un tipo de datos a otro tipo de datos. En algunos lenguajes de programación, como PHP, el tipo de datos de una variable puede cambiar automáticamente en función de su contenido. Sin embargo, en otros casos, es necesario realizar una conversión explícita (o casteo) para cambiar el tipo de datos de una variable.

El casteo se puede realizar de forma implícita (cuando el lenguaje de programación lo hace automáticamente) o explícita (cuando se utiliza una función o un operador para realizar la conversión).

Algunos ejemplos de casteo en PHP son (int), (float), (string), (array), etc. Estos operadores se colocan antes de la variable que se quiere convertir y cambian su tipo de datos a la representación correspondiente (entero, flotante, cadena o array, respectivamente).

Ejemplo de casteo a entero:

```php
<?php

   $num1 = "100";
   $num2 = 200;
   $num3 = "Hello";

   $int1 = (int)$num1;
   $int2 = (int)$num2;
   $int3 = (int)$num3;

   echo "int1: " . $int1; // Salida: int1: 100
   echo "<br>int2: " . $int2; // Salida: int2: 200
   echo "<br>int3: " . $int3; // Salida: int3: 0

?>
```

Ejemplo de casteo a flotante:

```php
<?php

   $num1 = "100.5";
   $num2 = 200;
   $num3 = "Hello";

   $float1 = (float)$num1;
   $float2 = (float)$num2;
   $float3 = (float)$num3;

   echo "float1: " . $float1; // Salida: float1: 100.5
   echo "<br>float2: " . $float2; // Salida: float2: 200
   echo "<br>float3: " . $float3; // Salida: float3: 0

?>
```

Ejemplo de casteo a string:

```php
<?php

   $num1 = 100;
   $num2 = 200.5;
   $num3 = true;

   $string1 = (string)$num1;
   $string2 = (string)$num2;
   $string3 = (string)$num3;

   echo "string1: " . $string1; // Salida: string1: 100
   echo "<br>string2: " . $string2; // Salida: string2: 200.5
   echo "<br>string3: " . $string3; // Salida: string3: 1

?>
```

Vamos a explicar lo que sucedió.

En PHP, la palabra clave **true** se trata como un valor booleano, que se considera equivalente a 1 en comparaciones numéricas. Por lo tanto, al hacer un casteo explícito a string, el valor true se convierte en una cadena que representa el número 1. Esto se aplica a todos los valores booleanos en PHP.

En PHP, la palabra clave **false** se trata como un valor booleano que se considera equivalente a 0 en comparaciones numéricas. Sin embargo, al hacer un casteo explícito a string, el valor false se convierte en una cadena vacía (``), no en una cadena que representa el número 0. Esto se aplica a todos los valores booleanos en PHP.

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

- Alcance o ámbito de las variables.
- Constructores del lenguaje echo y print.
- Tipos de datos.
- Strings.
- Funciones de strings.
- Números.
- Casteo.

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: Ámbito de las variables

PHP: Echo

PHP: Print

PHP: Tipos

PHP: Funciones de strings

PHP: Números enteros

PHP: Manipulación de tipos

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

Clase 03

Análisis de Sistemas

Materia:
Programación Web II

Docente contenidista: ROLDÁN, Hernán

Revisión: Coordinación

Contenido

Clase 3

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

En esta clase vamos a ver los siguientes temas:

- Constantes.
- Operadores.
- If...Else...Elseif.
- Switch.
- Loops.
- Break y Continue.

Constantes

Una **constante** en PHP es una variable que no puede ser reasignada o redefinida a lo largo del script. Una vez que se ha definido una constante, su valor no puede ser cambiado. Las constantes se definen con la función define().

Las constantes se usan en PHP para almacenar valores que deben ser accedidos globalmente y no cambiados a lo largo del tiempo. Algunos ejemplos de uso de constantes son: valores de configuración, valores constantes utilizados en múltiples partes del código, etc.

Sintaxis

define(name, value, case-insensitive)

```php
<?php

   define('BD_SERVIDOR', 'localhost');
   define('BD_USUARIO', 'elon');
   define('BD_CLAVE', 'argentinacampeon2022');
   define('BD_NOMBRE', 'cursos');

  $conn = mysqli_connect(BD_SERVIDOR, BD_USUARIO, BD_CLAVE,
BD_NOMBRE);

   if(!$conn){
       die("Error de conexión: " . mysqli_connect_error());
   }
   echo "Conexión exitosa";
   mysqli_close($conn);

?>
```

Las constantes **BD_SERVIDOR**, **BD_USUARIO**, **BD_CLAVE** y **BD_NOMBRE** se definen para almacenar información sobre la conexión a la base de datos. Luego, se utiliza la función mysqli_connect para conectarse a la base de datos utilizando estas constantes como argumentos. Si la conexión es exitosa, se imprimirá "Conexión exitosa", de lo contrario, se mostrará un mensaje de error. Finalmente, se cierra la conexión utilizando mysqli_close.

Operadores

En programación, los **operadores** son símbolos o palabras clave que representan una acción o una operación matemática que se realiza en datos o variables. Por ejemplo, el operador + es utilizado para sumar dos números, el operador - para restar, * para multiplicar, y / para dividir.

PHP tiene varios tipos de operadores, incluyendo:

- **Aritméticos:** +, -, *, /, % (módulo), ** (potencia).
- **de Asignación:** =, +=, -=, *=, /=, %=, .= (concatenación).
- **de Comparación:** == (igual), != (diferente), < (menor que), > (mayor que), <= (menor o igual que), >= (mayor o igual que), === (idéntico), !== (no idéntico).
- **Lógicos:** && (y lógico), || (o lógico), ! (negación lógica).
- **de Incremento/Decremento:** ++$x (incremento), --$x (decremento).
- **de Cadena:** . (concatenación).
- **de Array:** + (unión de arrays), == (igualdad de arrays), === (identidad de arrays), != (desigualdad de arrays), <> (desigualdad de arrays), !== (desidentidad de arrays).
- **de Control de errores:** @ (supresión de errores).
- **de Tipos:** instanceof (verificación de tipo de objeto).
- **de Expresión ternaria:** ?: (operador ternario).

Estos son los tipos de operadores más comunes en PHP, y cada uno de ellos tiene una función específica y una sintaxis particular para su uso en el código.

Para más información, podés consultar la sección PHP: Operadores en la documentación oficial del lenguaje.

A continuación, algunos ejemplos prácticos de los operadores anteriormente citados.

Operadores aritméticos:
Los operadores aritméticos son símbolos matemáticos utilizados en programación para realizar operaciones aritméticas básicas como la suma, resta, multiplicación, división y módulo. Estos operadores permiten realizar cálculos y modificar variables en el código.

Ejemplo de uso del operador de adición

```php
<?php

   $x = 10;
   $y = 20;
   $z = $x + $y;

   // $z ahora es igual a 30

?>
```

Ejemplo de uso del operador de sustracción

```php
<?php

   $x = 10;
   $y = 20;
   $z = $y - $x;

   // $z ahora es igual a 10

?>
```

Ejemplo de uso del operador de multiplicación

```php
<?php

   $x = 10;
   $y = 20;
   $z = $x * $y;

   // $z ahora es igual a 200

?>
```

Ejemplo de uso del operador de división

```php
<?php

   $x = 10;
   $y = 20;
   $z = $y / $x;

   // $z ahora es igual a 2

?>
```

Ejemplo de uso del operador de módulo

```php
<?php

   $x = 10;
   $y = 20;
   $z = $y % $x;

   // $z ahora es igual a 0

?>
```

Ejemplo de uso del operador de potencia

```php
<?php

   $x = 2;
   $y = 10;
   $z = $x ** $y;

   // $z ahora es igual a 1024

?>
```

Operadores de asignación:
Los operadores de asignación son símbolos utilizados en programación para asignar valores a variables.

```php
<?php

   // Asignación (=)
   $x = 5;
   echo $x; // 5

   // Asignación suma (+=)
   $x = 5;
   $x += 3;
   echo $x; // 8

   // Asignación resta (-=)
   $x = 5;
   $x -= 3;
   echo $x; // 2

   // Asignación multiplicación (*=)
   $x = 5;
   $x *= 3;
   echo $x; // 15

   // Asignación división (/=)
   $x = 6;
   $x /= 2;
   echo $x; // 3

   // Asignación módulo (%)
   $x = 7;
   $x %= 3;
   echo $x; // 1

   // Asignación de concatenación (.=)
   $x = "Hola";
   $x .= " mundo";
   echo $x; // Hola mundo

?>
```

Operadores de comparación:
Los operadores de comparación en programación son símbolos utilizados para comparar dos valores y determinar si se cumplen o no ciertas condiciones. Estos operadores devuelven un valor booleano (verdadero o falso) dependiendo del resultado de la comparación.

```php
<?php

    $a = 5;
    $b = 10;

    // Operador igual a
    if($a == $b){
        echo "$a es igual a $b\n";
    }
    else{
        echo "$a no es igual a $b\n"; // 5 no es igual a 10
    }

    // Operador no igual a
    if($a != $b){
        echo "$a no es igual a $b\n"; // 5 no es igual a 10
    }
    else{
        echo "$a es igual a $b\n";
    }

    // Operador mayor que
    if($a > $b){
        echo "$a es mayor que $b\n";
    }
    else{
        echo "$a no es mayor que $b\n"; // 5 no es mayor que 10
    }
```

```php
    // Operador menor que
    if($a < $b){
        echo "$a es menor que $b\n"; // 5 es menor que 10
    }
    else{
        echo "$a no es menor que $b\n";
    }

    // Operador mayor o igual que
    if($a >= $b){
        echo "$a es mayor o igual que $b\n";
    }
    else{
        echo "$a no es mayor o igual que $b\n"; // 5 no es mayor o
igual que 10
    }

    // Operador menor o igual que
    if($a <= $b){
        echo "$a es menor o igual que $b\n"; // 5 es menor o igual que
10
    }
    else{
        echo "$a no es menor o igual que $b\n";
    }

?>
```

Operadores lógicos:
Los operadores lógicos son simbolos que se utilizan para realizar operaciones lógicas y comparaciones en un programa de computadora. Estos operadores permiten evaluar expresiones lógicas y determinar si son verdaderas o falsas, su uso más común es en estructuras de control de flujo, como if-else, para tomar decisiones en el código basadas en el resultado de una comparación o operación lógica.

En PHP, los operadores lógicos son similares a los de otros lenguajes de programación, estos operadores incluyen:

- **AND (&&):** Devuelve verdadero si ambas expresiones son verdaderas.

```php
<?php

   if($a == 5 && $b == 10){
       // hacer algo
   }

?>
```

- **OR (||):** Devuelve verdadero si al menos una de las expresiones es verdadera.

```php
<?php

   if($a == 5 || $b == 10){
       // hacer algo
   }

?>
```

- **NOT (!):** Invierte el valor booleano de la expresión.

```php
<?php

   if(!($a == 5)){
       // hacer algo
   }

?>
```

Operadores de incremento/decremento:
Los operadores de incremento y decremento son operadores unarios que se utilizan en la programación para aumentar o disminuir el valor de una variable en una unidad. Hay dos operadores de incremento y dos operadores de decremento en la mayoría de los lenguajes de programación, y PHP no es la excepción:

- **Incremento (++):** Aumenta el valor de una variable en una unidad.

- **Decremento (--):** Decrementa el valor de una variable en una unidad.

- **Pre-incremento (++x):** Aumenta el valor de una variable antes de ser utilizado en una expresión.

- **Pre-decremento (--x):** Decrementa el valor de una variable antes de ser utilizado en una expresión.

Operador pre-incremento:

```php
<?php

   $x = 5;
   ++$x;
   echo $x; // Salida: 6

?>
```

Operador post-incremento:

```php
<?php

   $x = 5;
   $y = $x++;
   echo $y; // Salida: 5
   echo $x; // Salida: 6

?>
```

Operador post decremento:

```php
<?php

   $x = 5;
   --$x;
   echo $x; // Salida: 4

?>
```

Operador post decremento:

```php
<?php

   $x = 5;
   $y = $x--;
   echo $y; // Salida: 5
   echo $x; // Salida: 4

?>
```

Los operadores de incremento y decremento se utilizan a menudo en bucles for y while para controlar el número de iteraciones. Por ejemplo, en un bucle for, se puede utilizar el operador de incremento para aumentar el valor de un contador en una unidad en cada iteración:

```php
<?php

   $x = 15;

   for($i=1; $i<=$x; $i++){
       echo $i . "\n"; // imprime los números 1 al 15
    }

?>
```

Operadores de cadena:
Los operadores de cadena son operadores que permiten unir o concatenar dos o más cadenas de texto en una sola. En muchos lenguajes de programación, incluyendo PHP, se utiliza el operador de concatenación "." para unir dos o más cadenas de texto.

```php
<?php

   $nombre = "Bart";
   $apellido = "Simpson";
   $nombre_completo = $nombre . " " . $apellido;

   echo $nombre_completo; // Salida: "Bart Simpson"

?>
```

Operadores de array:
Los operadores de array en programación son operadores específicos que se utilizan para trabajar con arreglos, que son estructuras de datos que almacenan una secuencia ordenada de valores. Algunos ejemplos de operadores de array incluyen:

Ejemplo de operador de unión de arrays:

```php
<?php

   $array1 = array(1, 2, 3);
   $array2 = array(4, 5, 6);
   $union = $array1 + $array2;
   // $union será [1, 2, 3, 4, 5, 6]

?>
```

Ejemplo de operador de igualdad de arrays:

```php
<?php

   $array1 = array(1, 2, 3);
   $array2 = array(1, 2, 3);
   $equal = ($array1 == $array2);
   // $equal será true

?>
```

Ejemplo de operador de identidad de arrays:

```php
<?php

   $array1 = array(1, 2, 3);
   $array2 = array(1, 2, 3);
   $identical = ($array1 === $array2);
   // $identical será true

?>
```

Ejemplo de operador de desigualdad de arrays:

```php
<?php

   $array1 = array(1, 2, 3);
   $array2 = array(4, 5, 6);
   $not_equal = ($array1 != $array2);
   // $not_equal será true

   $not_equal = ($array1 <> $array2);
   // $not_equal será true

?>
```

Ejemplo de operador de desidentidad de arrays:

```php
<?php

   $array1 = array(1, 2, 3);
   $array2 = array(4, 5, 6);
   $not_identical = ($array1 !== $array2);
   // $not_identical será true

?>
```

Operadores de control de errores:
Los operadores de control de errores en PHP son un conjunto de símbolos que permiten controlar los errores que ocurren en un script PHP.

En PHP, el operador '@' es conocido como el operador de supresión de errores y permite suprimir la visualización de un error en tiempo de ejecución. Al colocar el operador @ antes de una instrucción en un script PHP, se evita que cualquier mensaje de error generado por esa instrucción se muestre en la salida. Este operador es útil cuando se quiere ocultar errores no críticos o para evitar que los errores públicos estén disponibles para los usuarios finales. Sin embargo, suprimir los errores no es la forma recomendada de manejar errores en PHP, ya que puede dificultar la depuración y el seguimiento de problemas en la aplicación. En su lugar, se recomienda usar una técnica de manejo de errores adecuada, como la función set_error_handler() o la gestión de excepciones.

```php
<?php

   @$archivo = fopen('alta_de_usuarios.txt', 'r');

   if(!$archivo){
       echo 'Error al abrir el archivo "alta_de_usuarios.txt".';
   }
   else{
       echo 'Lectura del archivo exitosa.';
       fclose($archivo);
   }

?>
```

Operadores de Tipos:
Los operadores de tipo se utilizan para verificar el tipo de una variable. Hay dos operadores de tipo en PHP: instanceof y gettype.

- **instanceof:** El operador instanceof en PHP se utiliza para verificar si un objeto es una instancia de una determinada clase o de una clase que hereda de ella.

Ejemplo de uso del operador instanceof:

```php
<?php

   Class Animal{}
   Class Perro extends Animal{}

   $obj_perro = new Perro();

   if($obj_perro instanceof Animal){
       echo '"obj_perro" es instancia de la clase Animal.';
   }
   // Salida: La instancia "obj_perro" es instancia de la clase
Animal.


    echo "\n";

   if($obj_perro instanceof Perro){
       echo '"obj_perro" es instancia de la clase Perro.';
   }
   // Salida: La instancia "obj_perro" es instancia de la clase
Animal.

?>
```

- **gettype:** Este operador se utiliza para obtener el tipo de una variable en forma de cadena.

Ejemplo de uso del operador gettype:

```php
<?php

   Class Animal{}
   $obj_animal = new Animal();
   echo "\n";
   echo gettype($obj_animal);

   $nombre = 'John Wick';
   echo "\n";
   echo gettype($nombre);

   $edad = 10;
   echo "\n";
   echo gettype($edad);

   $precio = 100.5;
   echo "\n";
   echo gettype($precio);

   $luces_encendidas = true;
   echo "\n";
   echo gettype($luces_encendidas);

?>
```

Operadores ternarios:
Un operador ternario es un tipo de operador en programación que permite evaluar una expresión y elegir un valor o bloque de código en función de si la expresión es verdadera o falsa. Estos operadores son una forma abreviada de escribir una sentencia if-else y se representan mediante el símbolo ? seguido de :. En su forma más básica, la sintaxis de un operador ternario es la siguiente:

expresión ? valor_si_verdadera : valor_si_falsa

Donde expresión es cualquier expresión que puede evaluarse como verdadera o falsa, y valor_si_verdadera y valor_si_falsa son los valores o bloques de código que se deben elegir en función del resultado de la evaluación de expresión.

Como dijimos, el operador ternario en PHP es una forma abreviada de escribir una sentencia if-else en una sola línea. Aquí hay algunos ejemplos de su uso:

```php
<?php

   $a = 42;
   $b = 23;

   // Uso básico
   echo ($a > $b) ? '$a es mayor que $b' : '$a no es mayor que $b';
   // Salida: $a es mayor que $b

   // Asignación de valores
   $result = ($a > $b) ? $a : $b;
   echo $result;
   // Salida: 42

   // Uso con funciones
   $array = [1, 2, 3];
   $count = count($array);
   echo ($count > 0) ? "Hay $count elementos en el array" : 'El
array está vacío';
   // Salida: Hay 3

?>
```

If...Else...Elseif

Un bloque if-else-elseif es un tipo de estructura de control en programación que permite ejecutar diferentes bloques de código en función de si una o más condiciones son verdaderas o falsas. La sintaxis básica de un bloque if-else-elseif es la siguiente:

```
if(condición){
  // Código a ejecutar si la condición es verdadera
}
elseif(otra_condición){
  // Código a ejecutar si la primera condición es falsa y la segunda es verdadera
}
else{
  // Código a ejecutar si ninguna de las condiciones anteriores es verdadera
}
```

El bloque if es el bloque principal y contiene la condición que se debe evaluar.

Si la condición es verdadera, se ejecutará el bloque de código correspondiente.

Si la condición es falsa, se evaluarán las condiciones en los bloques elseif adicionales, si existen.

Si ninguna de las condiciones es verdadera, se ejecutará el bloque else, si existe.

Ejemplo de uso del bloque if

```php
<?php

   $num = 5;

   if($num > 10){
       echo "El número es mayor a 10.";
   }
   elseif($num < 10){
       echo "El número es menor a 10.";
   }
   else{
       echo "El número es igual a 10.";
   }

   // Salida: El número es menor a 10.

?>
```

El ejemplo de arriba evalúa la variable $num y determina si es mayor que 10, menor que 10 o igual a 10. En función de eso, se imprime un mensaje correspondiente.

Switch

En programación, un bloque switch es una estructura de control que permite la ejecución de un bloque de código específico de entre varios posibles, basado en el valor de una expresión o variable. Es similar a una serie de sentencias if-else, pero a menudo resulta más legible y eficiente para grandes conjuntos de casos.

La sintaxis básica de un bloque switch en algunos lenguajes de programación populares incluye la palabra clave "switch" seguida por la expresión o variable que se va a evaluar y varios bloques "case" que especifican los posibles valores y el código correspondiente a ejecutar en cada caso. También se puede utilizar un bloque "default" para especificar un comportamiento por defecto en caso de que ningún otro "case" coincida con la expresión evaluada.

Sintaxis

```
switch(expression){

   case value1:

     // código a ejecutar si expression es igual a value1
     break;

   case value2:

     // código a ejecutar si expression es igual a value2
     break;
   ...
   default:
     // código a ejecutar si expression no coincide con ningún
otro case
     break;

 }
```

La sentencia switch en PHP está compuesta por las siguientes partes:

1. La palabra clave ‘switch’, seguida de una expresión que se evalúa y su valor se compara con los valores especificados en los casos.

2. Un bloque de casos definidos con la palabra clave ‘case’, seguidos del valor que se desea comparar con la expresión evaluada.

3. Un bloque de código que se ejecutará si la expresión coincide con el valor especificado en algún caso.

4. Una sentencia ‘break’ para detener la ejecución después de ejecutar el bloque de código correspondiente.

5. Una sentencia opcional ‘default’ que se ejecutará si ninguno de los casos coincide con la expresión evaluada.

Debajo algunos ejemplos de la sentencia switch en PHP:

Ejemplo 1

```php
<?php

   $dia = "lunes";

   switch($dia){
      case "lunes":
        echo "Hoy es lunes.";
        break;
      case "martes":
        echo "Hoy es martes.";
        break;
      case "miércoles":
        echo "Hoy es miércoles.";
        break;
      default:
        echo "Hoy no es ni lunes, ni martes, ni miércoles.";
        break;
      }

      // Salida: Hoy es lunes.

?>
```

Ejemplo 2

```php
<?php

   $personajes_simpsons = ['Homer', 'Marge', 'Bart', 'Lisa',
'Maggie'];

   // print_r($personajes_simpsons);

   $personaje_actual = 'Lisa';

   echo "\n";

   switch($personajes_simpsons){

    case $personaje_actual == $personajes_simpsons[0]:
      echo 'Homer';
      break;

    case $personaje_actual == $personajes_simpsons[1]:
      echo 'Marge';
      break;

    case $personaje_actual == $personajes_simpsons[2]:
      echo 'Bart';
      break;

    case $personaje_actual == $personajes_simpsons[3]:
      echo 'Lisa';
      break;

    case $personaje_actual == $personajes_simpsons[4]:
      echo 'Maggie';
      break;

    default:
      echo 'No es un personaje de la familia Simpson.';

   }

   // Salida: Lisa

?>
```

Es importante mencionar que, si no se coloca una sentencia break después de cada caso, la ejecución continuará hasta encontrar una sentencia break o hasta el final de la sentencia switch. Esto puede ser útil para ejecutar varios bloques de código seguidos en el mismo caso, pero también puede ser una fuente de errores y debes ser cuidadoso al usar esta técnica.

Loops

Los loops en programación son estructuras de control que permiten ejecutar un bloque de código varias veces, hasta que se cumpla una condición específica. En otras palabras, los loops te permiten repetir un proceso hasta que se cumpla una determinada condición.

Hay dos tipos principales de loops en programación:

1. **For loops:** Este tipo de loop se utiliza para repetir un bloque de código un número determinado de veces. Por ejemplo, puedes utilizar un for loop para imprimir los números del 1 al 10.

2. **While loops:** Este tipo de loop se utiliza para repetir un bloque de código mientras se cumpla una condición determinada. Por ejemplo, puedes utilizar un while loop para imprimir los números mientras una variable sea menor que 10.

Los loops son una herramienta muy útil en programación y te permiten automatizar tareas repetitivas y realizar cálculos complejos de manera más eficiente.

En PHP hay 4 tipos de loops: for, while, do...while y foreach.

1. **For Loop:** El for loop se utiliza para repetir un bloque de código un número determinado de veces. Tiene las siguientes partes:
    - La primera parte, es donde se inicializa la variable del contador.
    - La segunda parte, es la condición que debe ser cumplida para que el loop continúe ejecutándose.
    - La tercera parte, es donde se incrementa o decrementa la variable del contador en cada iteración del loop.

Sintaxis

for(init counter; test counter; increment counter){

código a ejecutar por cada iteración;

}

Ejemplo:

```php
<?php

  for ($i = 1; $i <= 5; $i++){
    echo $i . "\n";
  }

  /*

    Salida:
    1
    2
    3
    4
    5

  */

?>
```

2. **While Loop:** El while loop se utiliza para repetir un bloque de código mientras se cumpla una determinada condición. Tiene solo una parte:
    - La parte, es la condición que debe ser cumplida para que el loop continúe ejecutándose.

Sintaxis

while(condition is true){

código a ejecutar;

}

Ejemplo:

```php
<?php

  $i = 1;

  while($i <= 5){

    echo $i . "\n";

    $i++;

  }

  /*

  Salida:
  1
  2
  3
  4
  5

  */

?>
```

3. ***Do...While Loop:*** El do...while loop es similar al while loop, pero se asegura de que el bloque de código se ejecute al menos una vez antes de comprobar la condición. Tiene las siguientes partes:
    - El bloque de código que se ejecuta en cada iteración del loop.
    - La parte, es la condición que debe ser cumplida para que el loop continúe ejecutándose.

Sintaxis

do{

código a ejecutar;

}while(condition is true);

Ejemplo

```php
<?php

  $i = 1;

  do{
    echo $i . "\n";
    $i++;
  }while($i <= 5);

  /* Salida:

    1
    2
    3
    4
    5

  */

?>
```

4. **Foreach:** El foreach loop en PHP es un tipo de loop diseñado específicamente para recorrer arrays y objetos iterables. Es muy útil para realizar tareas repetitivas con los elementos de un array, como imprimirlos en pantalla o modificarlos. El foreach loop consta de dos partes:
    - La primera parte, es el array u objeto que se quiere recorrer.
    - La segunda parte, es la variable que se utilizará para acceder a los elementos del array u objeto en cada iteración.

Sintaxis

foreach($array as $valor){

code a ejecutar;

}

Ejemplo:

```php
<?php

   $escritores = array('Edgar Allan Poe', 'Howard Phillips Lovecraft',
'Julio Cortázar');

   foreach($escritores as $escritor){
       echo $escritor .  '.' . "\n";
   }

   // Salida
   // Edgar Allan Poe
   // Howard Phillip Lovecraft
   // Julio Cortázar

   echo "\n";

   echo $escritores[0] . '.'; // Salida: Edgar Allan Poe.
   echo "\n";
   echo $escritores[1] . '.'; // Salida: Howard Phillips Lovecraft.
   echo "\n";
   echo $escritores[2] . '.'; // Salida: Julio Cortázar.

?>
```

Break y Continue

Las sentencias "break" y "continue" en PHP son instrucciones de control de flujo que permiten modificar el comportamiento de los bucles y estructuras de control.

La sentencia "break" se utiliza para salir de un bucle o una estructura de control antes de que se complete su ejecución normal. Cuando se encuentra la instrucción "break", el flujo de control se mueve fuera del bucle o estructura de control, y continúa ejecutando la siguiente instrucción después del bucle o estructura de control.

Ejemplo de la sentencia "break" en un bucle "while":

```php
<?php

  $i = 0;

  while ($i < 10){
    echo $i . "\n";
    $i++;
    if($i == 5){
      break;
    }
  }

   /*

    Salida:
    0
    1
    2
    3
    4

  */

?>
```

En este ejemplo, el bucle "while" imprimirá los números del 0 al 4, y luego se detendrá debido a la instrucción "break".

La sentencia "continue" se utiliza para omitir la ejecución de una iteración particular de un bucle o estructura de control, y pasar a la siguiente iteración. Cuando se encuentra la instrucción "continue", el flujo de control salta a la próxima iteración del bucle o estructura de control.

Ejemplo de la sentencia "continue" en un bucle "for":

```php
<?php

  for($i = 0; $i < 10; $i++){
    if ($i % 2 == 0){
      continue;
    }
    echo $i . "\n";
  }

?>
```

En este ejemplo, el bucle "for" imprimirá los números impares del 1 al 9, ya que la instrucción "continue" omitirá la ejecución de las iteraciones donde el valor de "$i" es par.

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

- Constantes.
- Operadores.
- If...Else...Elseif.
- Switch.
- Loops.
- Break y Continue.

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: Constantes

PHP: Operadores

PHP: If... else... elseif

PHP: Switch

PHP: Estructuras de control

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

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.

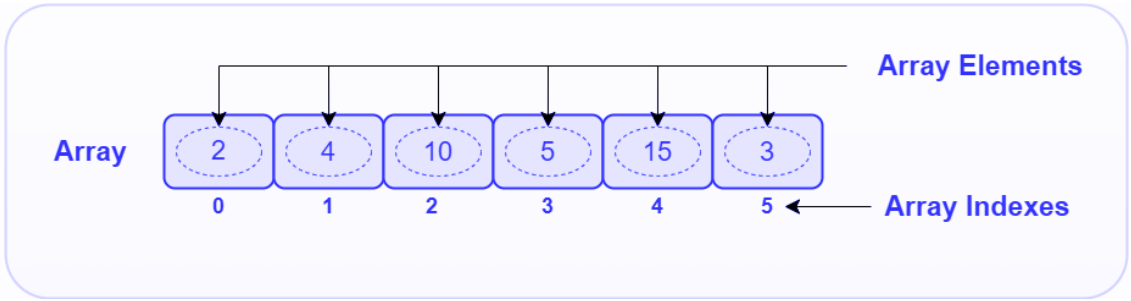
Por lo tanto, siempre es recomendable consultar la documentación oficial para obtener la información más actualizada sobre las funciones built-in de PHP y cómo utilizarlas correctamente en tus proyectos.

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