Desarrollo Full Stack

Fundación Telefónica Movistar

CURSO DE DESARROLLO FULL STACK.

Sintaxis básica de PHP.

PHP es uno de los lenguajes más potentes a la hora de desarrollar páginas web dinámicas, aplicaciones web o servicios web.

Podemos escribir nuestras aplicaciones directamente en PHP, pegarle código HTML o podemos embeberlo en páginas web hechas en HTML.

Para indicarle al intérprete del servidor que el código que va a leer es PHP, lo hacemos de varias formas.

La más común y recomendada es mediante <?PHP ?> ya que es la que reconoce el servidor por defecto sin ningún tipo de configuración.

```php
<?php

   // Todo el código PHP va entre estas etiquetas.

?>
```

Dijimos que PHP es un lenguaje interpretado.

Cuando el servidor accede a un archivo de nuestra aplicación web, comienza a examinarlo desde la primera línea.

Cuando encuentra una etiqueta **<?php** sabe que a partir de ahí comienza una porción de código en PHP y debe interpretarla para generar su equivalente en HTML que será entregado al cliente que quiere ver nuestra aplicación.

El servidor seguirá interpretando el código hasta que le digamos que ya se ha terminado, mediante la etiqueta de cierre **?>**

Comentarios.

Los comentarios son muy importantes más allá del lenguaje que utilicemos, son la manera más práctica de documentar y nos permite identificar la función de cualquier parte del código sin importar cuanto hace que lo hayamos escrito.

Es aconsejable siempre comentar nuestro código, para beneficio propio y para facilitar la manera en que compartimos nuestro trabajo con el resto del equipo.

```php
<?php
   // Comentario de una sola línea.

   # Comentario de una sola línea.

   /* Este es un comentario
   de más de una línea.
   Es muy útil para hacer una descripción detallada
   dentro de un archivo PHP */
?>
```

Fin de línea.

¡IMPORTANTE!

En PHP, todas las líneas de código (excepto los comentarios) terminan con un punto y coma.

Función PRINT.

La función PRINT le indica al intérprete PHP que escriba todo lo que tiene entre paréntesis, en el código de la página que va a devolver al navegador.

En resumen, veremos por pantalla todo aquello que le indiquemos que muestre.

Estrictamente hablando, PRINT no es en realidad una función, es un constructor de lenguaje, algo que veremos más adelante cuando veamos programación orientada a objetos, por lo tanto, no es necesario que utilicemos los paréntesis.

Siempre que sea texto, lo pondremos entre comillas, pueden ser simples o dobles.

Si necesitamos intercalar código HTML, podemos hacerlo colocando las etiquetas dentro de las comillas.

```php
<?php
    // Print con paréntesis.
    print("Hola mundo con comillas dobles.");
    print('Hola mundo con comillas simples.');
?>
```

También podemos incluir código HTML dentro de las comillas, por ejemplo, un salto de línea:

```php
<?php
    // Print con paréntesis.
    print("Hola mundo con comillas dobles.<br>");
    print('Hola mundo con comillas simples.<br>');
?>
```

Otro ejemplo de cómo intercalar código HTML:

```php
<?php
    // Print con paréntesis.
    print("<h1>Hola mundo con comillas dobles.</h1>");
    print('<h2>Hola mundo con comillas simples.</h2>');
?>
```

Este es el resultado que vemos por pantalla:

Hola mundo con comillas dobles.

Hola mundo con comillas simples.

Probemos el código anterior, pero sin usar paréntesis, veremos que el resultado es el mismo.

¿Cómo hacemos si necesitamos mostrar comillas por pantalla desde PHP?

En caso de necesitar que se muestre por pantalla alguna comilla desde PHP, utilizamos el carácter de escape \.

```php
<?php
    print"<h1>Hola \"mundo\" mostrando comillas en pantalla.</h1>";
?>
```

Siempre que usemos este carácter de escape, será para “escapar” de las comillas en las que hayamos encerrado toda la frase.

Esto se podría evitar si hacemos uso de ambos tipos de comillas, simples y dobles, en una misma línea de código.

Por ejemplo, encerramos todo el texto en comillas dobles y mostramos por pantalla comillas simples o a la inversa.

```php
<?php
    print"<h1>Hola 'mundo' mostrando comillas simples.</h1>";
    print'<h1>Hola "mundo" mostrando comillas dobles.</h1>';
?>
```

Comando ECHO.

El comando echo es utilizado para devolver al navegador los datos que el servidor procesa. también puede utilizar ambos tipos de comillas, simples o dobles, para delimitar lo que va a imprimir.

Al igual que la función PRINT no es necesario colocar paréntesis para lo que vamos a mostrar y el código HTML en su interior puede estar distribuido en múltiples líneas.

```php
<?php
    echo '<h1>Título</h1>';
    echo '<h3>Sub título</h3>';
    echo '<p>Parrafo</p>';
?>
```

Título

Sub título

Párrafo

*** No solamente muestra cadenas de texto, sino también variables y constantes.

Variables.

El verdadero sentido de trabajar con PHP es usar datos, veamos los diferentes tipos de almacenes de datos con los que trabajaremos:

- Variables y matrices.
- Datos enviados por formulario o variable adosada a URL.
- Devoluciones de funciones.
- Datos almacenados por el navegador, como por ejemplo las cookies.
- Variables de sesión y variables de servidor.
- Datos escritos en un archivo de texto (txt, XML, etc.)
- Bases de datos.

Comenzaremos por el primero de la lista, las variables.

Una variable se define mediante dos elementos, su identificador (el nombre que le damos) y el valor que le asignemos.

Las variables pueden contener todos los tipos de datos con los que trabaja el lenguaje, incluidos los objetos.

- **El identificador deberá estar precedido por el signo $, no podrá iniciar con un número.**
- **El intérprete PHP distingue entre mayúsculas y minúsculas.**
- **Para concatenar dos variables, utilizamos el punto.**
- **No es necesario declarar el tipo.**

Veamos un ejemplo del uso de variables:

```php
<?php
    $a = 'Hola';
    $b = 'mundo';

    // Variables concatenadas.
    echo '<h3> Concatenamos variables y texto. </h3>';
    echo $a.' - '.$b;

    // Lista de variables.
    echo '<br><br><h3> Concatenamos variables y texto. </h3>';
    echo $a,$b;
?>
```

En el código anterior vemos dos variables cada una con un texto asignado, y vemos dos formas de mostrar ambas concatenadas.

El primer echo utiliza el punto para concatenar ambas variables con un guion en el medio.

El segundo echo muestra el contenido de ambas variables.

Es importante, sobre todo a la hora de trabajar en equipo, poner de antemano una convención para los nombres de variable y la forma en que vamos a escribir dichos nombres, en caso de contener más de una palabra.

***** ¿Qué diferencia hay entre encerrar las variables entre comillas simples o encerrarlas entre comillas dobles?**

Constantes.

Las constantes se utilizan por lo general, cuando necesitamos que un determinado valor permanezca inalterable y no se pueda cambiar ni por error.

Se definen mediante la función DEFINE que espera dos parámetros, el nombre de la constante y su valor.

Vemos un ejemplo:

```php
<?php
    define('IDIOMA','Castellano');


    echo IDIOMA;
?>
```

El identificador de la constante no lleva un signo $ adelante y por convención, se usa escribirlo con todas sus letras en mayúsculas.

Se las puede concatenar siguiendo lo visto con variables.

```php
<?php
    define('IDIOMA','Castellano');


    echo 'El lenguaje de la Web es: '.IDIOMA;
?>
```

Matrices.

Una matriz es, al igual que las variables, un almacén de datos, y a diferencia de estas, podemos guardar varios datos ordenados y no solamente uno.

Veamos cómo definir una matriz de tres elementos y como poder utilizarlos:

```php
<?php
    $paises[0] = 'Argentina';
    $paises[1] = 'Brasil';
    $paises[2] = 'Uruguay';


    echo $paises[1];
?>
```

Los elementos de nuestra matriz llamada países, se diferencian entre sí por su índice.

Este debe ser único para cada elemento, si le asignamos el mismo índice a dos elementos diferentes, persistirá el último valor que le hayamos asignado.

Tenemos diferentes formas de inicializar, podemos hacerlo de forma explícita, de forma implícita, mezclando ambas o usando la función ARRAY.

Declaración explícita de matrices.

Es el método visto en el punto anterior, declaramos explícitamente cada elemento de la matriz.

```php
<?php
    $numeros[0] = 12;
    $numeros[1] = 5;
    $numeros[2] = 3;

    echo 'Suma = ',$numeros[0]+$numeros[1]+$numeros[2];
?>
```

Declaración implícita de matrices.

La declaración implícita se hace omitiendo los índices, y el intérprete PHP irá asignando en forma automática un índica a cada valor.

Si bien no es necesario en este tipo de declaración indicar el número de índice, si es necesario indicarlo al momento de trabajar.

```php
<?php
    $numeros[] = 12;
    $numeros[] = 5;
    $numeros[] = 3;

    echo 'Suma = ',$numeros[0]+$numeros[1]+$numeros[2];
?>
```

Declaración mixta de matrices.

Otra opción que tenemos es indicar solo el valor del primer índice, y luego dejar que el intérprete asigne en forma automática los demás.

Nosotros definimos el primer índice, le indicamos al intérprete que mi matriz empiece a partir del índice 1 y a partir de ese, los asigne en forma automática.

```php
<?php
    $numeros[2] = 12;
    $numeros[] = 5;
    $numeros[] = 3;

    echo 'Suma = ',$numeros[2]+$numeros[3]+$numeros[4];
?>
```

La función ARRAY.

El uso de la función ARRAY es también una declaración implícita, pero mucho más simple si tenemos que declarar varias matrices con muchos valores.

Dentro de la función ARRAY ingresamos los valores de la matriz en formato lista y después los utilizamos como habíamos visto, haciendo referencia al índice.

```php
<?php
    $paises = array('Argentina','Brasil','Uruguay');

    echo $paises[2];
?>
```

Los datos que almacenemos en una matriz no tienen que ser todos del mismo tipo, podemos mezclar cadenas de texto con números:

```php
<?php
    $paises = array('Argentina',45,'Uruguay');

    echo $paises[1];
?>
```

En este tipo de declaración también podemos forzar el valor del primer índice, lo hacemos mediante el operador =>

```php
<?php
    $paises = array(3=>'Argentina',45,'Uruguay');

    echo $paises[3];
?>
```

Y no es necesario que sea el primer elemento al que le forzamos el valor:

```php
<?php
    $paises = array('Argentina',7=>45,'Uruguay');

    echo $paises[8];
?>
```

Los índices numéricos no tienen que ser consecutivos, tomemos por ejemplo el caso de una matriz que guarde productos y que usemos el índice para el dato del código de producto.

Veamos un ejemplo:

```php
<?php
    $productos[325] = 'Heladera';
    $productos[766] = 'Televisor';
    $productos[221] = 'Lavarropas';
?>
```

Índices alfanuméricos.

En muchas ocasiones, en especial cuando trabajemos con bases de datos, nos va a resultar especialmente útil designar los índices con valores alfanuméricos, por ejemplo, para guardar los datos de un cliente o un usuario:

```php
<?php
    $usuario['nombre'] = 'Guillermo';
    $usuario['apellido'] = 'Alfaro';
    $usuario['dni'] = 123456;
?>
```

Como se ve, es muy útil a la hora de recordar que valor de la matriz necesitamos lo que simplifica mucho el código.

Los índices alfanuméricos, por ser una cadena de texto, se deben colocar entre comillas.

Matrices definidas por el servidor.

Son un tipo especial de matrices con índice alfanumérico, son aquellas que el intérprete declara y carga los datos en forma automática, sin que tengamos que hacerlo nosotros.

Un ejemplo es la matriz SERVER, que contiene información tal como cabeceras, rutas y ubicaciones de script.

Cada servidor puede crear dicha matriz con los valores que crea necesarios, por lo que no tenemos garantía de que en todos vamos a encontrar los mismos valores.

Veamos un par de ejemplos de cómo utilizarla:

```php
<?php
    echo $_SERVER['HTTP_USER_AGENT'];
?>
```

Esta instrucción me devuelve una cadena de texto que me indica el navegador que está usando la persona que ingresó a mi web.

```
Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36
```

Otro ejemplo, es el nombre del servidor donde tenemos alojada nuestra aplicación:

```php
<?php
    echo $_SERVER['SERVER_NAME'];
?>
```

```
localhost
```


Tenemos más matrices creadas por el servidor a las cuales podemos utilizar para facilitar el desarrollo de nuestra aplicación y simplificar el código.

Por ejemplo, las matrices GET y POST tienen las variables enviadas al servidor mediante la URL (GET) o mediante un formulario (POST).

Es importante recordar que no todos los valores de estas matrices estarán disponibles, ya que dependen del administrador del servidor, esto es fundamental para no desarrollar nuestra aplicación sin antes verificar las variables disponibles en el servidor definitivo.

Incluir archivos externos.

Es normal que en desarrollos de aplicaciones web, varias de nuestras páginas contengan elementos en común, como por ejemplo un menú de navegación, una lista de mensajes de error personalizados, una lista de constantes o archivos de clases.

No tiene sentido repetir ese código en cada una de las páginas que lo utilizan, sería muy difícil y complicado de mantener y actualizar.

La solución indicada es guardar el código que utilizan estas páginas en un archivo y luego cada página incorpora lo que necesite.

Esto lo podremos hacer mediante las siguientes construcciones (no son funciones):

- **Include**
- **Require**
- **Include_once**
- **Require_once**

```php
<?php
    include('variables.php');


    echo $a;
?>
```


Las diferencias entre INCLUDE y REQUIRE es simple, solo se diferencian en el tipo de error que devuelven.

En el caso de INCLUDE, si no encuentra el archivo, solo devuelve un WARNING, pero no detiene la ejecución de la página, en cambio REQUIRE devolverá un FATAL ERROR que interrumpe definitivamente la ejecución del archivo en ese punto.

Decidiremos cual usar de los dos, de acuerdo con la necesidad que tengamos de que el archivo en cuestión sea incorporado.

```php
<?php
    require('frases.php');


    echo 'La carga de la página continua.';
?>
```

Agregando la palabra ONCE a uno u oro, lo que le indicamos es que, si el archivo ya fue incluido en esa página, no lo vuelva a hacer.

Es común en algunos sistemas o aplicaciones que la totalidad de la página sea recargada y esto evita que los archivos externos se vuelvan a cargar