#1

UNDERCODE

TALLER DE PHP

TEMAS

- INTRODUCCIÓN
- FUNCIONAMIENTO
- INSTALACIÓN
- EDITORES DE TEXTO
- ESTRUCTURAS Y FUNCIONES
- COMILLAS
- VARIABLES
- Y MUCHO MÁS..!

TUTOR

ZFAC3R

Introducción:

PHP es un lenguaje de programación **del lado del servidor (back-end)**, lo que significa que todas las peticiones son procesadas en modo "background".A diferencia de los lenguajes que se ejecutan del lado del cliente (front-end) (como JavaScript) el intérprete PHP se aloja en el servidor y éste es quien se encarga de que todo funcione correctamente.

PHP ha sido desarrollado para manejar contenido de forma dinámica, puede trabajar de forma "amigable" con otros lenguajes en un mismo archivo, de esta forma hablamos que podemos incluirlo junto con javascript, html, y demás.

Hay una frase entre los programadores que debes tener en cuenta:

" La efectividad de un lenguaje depende de la imaginación del programador ".

Con esto nos referimos a que, con php puedes hacer infinidad de acciones, pero siempre estarás limitado a tú imaginación. Entre algunos ejemplos que puedo citar son: mailers, encoder/decoder, uploaders, foros, websites, blogs, manejo de bases de datos, sistemas de ficheros, conexiones por sockets, etc, etc y un largo etcétera...

Como habrás deducido ya, para entender al máximo este lenguaje debes tener conocimientos básicos de HTML.

INTRODUCCIÓN AL FUNCIONAMIENTO

Antes de continuar, quiero explicar un poco mejor el concepto de back-end y front-end.
En cualquier página web (ya sea Facebook, twitter, underc0de, lo que sea) veremos siempre el front-end, lo que vemos nosotros al acceder a una página web es html dibujado (renderizado), cuando nosotros escribimos una dirección

web por ejemplo http://www.google.com nosotros estamos realizando una conexión al servidor que tiene almacenado todos los datos, y lo que pedimos es el archivo index, es que el servidor interpreta normalmente que si no pedimos un archivo en concreto, estamos pidiendo el index, nosotros podríamos pedir un hola.html accediendo a este link, http://www.google.com/hola.html al realizar la conexión con google, pedirá el archivo hola.html. el servidor analizará dicho archivo, y dependiendo de que sea, si es php por ejemplo ejecutará el programa (ejecutable php_cgi.exe en caso de plataformas Windows) que interpreta o entiende el php, cuando se interpreta todo el código que sea, este devuelve un html que se entrega al navegador, y el navegador se encarga de renderizarlo, para que nosotros lo podamos ver, en el caso de que fuere simple html lo devolvería sin interpretarlo, todo este trabajo lo realiza normalmente apache (aunque hay otros programas gestores del protocolo http, como cherooke server). El html como dijimos lo renderiza nuestro navegador, todo lo que se ejecute o realice nuestro navegador es denominado Front-end, o lenguaje de cliente, javascript no se ejecuta en el servidor por ejemplo, se devuelve tal cual está y el navegador se encarga de pasárselo al intérprete de js (javascript), en cambio php es ejecutado en el servidor y se envía el resultado, eso sería el Back-end.

INSTALACIÓN (Windows y Linux).

Ya que php es un lenguaje de código libre y nivel web (por lo tanto multiplataforma) podemos ejecutarlo desde un sistema Windows o Linux, como este taller está orientado a gente no-experimentada, haremos uso de los programas más simples para dejar todo Full (Completo) sin mucho esfuerzo.

Para Windows tenemos WAMP Server (o Xampp) y para Linux XAMPP.

En ambos casos la instalación es bastante sencilla y es el típico “siguiente --> siguiente”, así que no veremos eso.

EDITORES DE TEXTO

Existen distintos editores de texto que podemos usar para llevar a cabo nuestros scripts, les dejo una lista de los más conocidos/utilizados:

- DreamWeaver
- Aptana Studio 3 (el que uso yo :P)
- Eclipse
- Notepad++
- Gedit (para los linuxeros)
- Block de notas (si, el de windows)
- Pspad es muy bueno también (porque tiene listador de funciones y bloques)

A excepción del block de notas, los editores nombrados anteriormente nos ayudan a tener una mejor "visión" del código, ya que nos hace un highlight (Coloreado) de nuestro código, lo que nos ayuda a una mejor programación.

Les recomiendo iniciar con el Notepad++, y después que ya lleven un tiempo programando usen Aptana Studio o Eclipse, que nos facilitan un poco la vida ;) .

ESTRUCTURA Y FUNCIONES BÁSICAS

PHP nos da la posibilidad de iniciar un script de diferentes maneras.

```
<?
        # Forma simple
echo'Hola mundo!';
```

```
?>

<?='Hola Mundo!' // Forma rápida, solo de una línea?>

<?php
    /*
        Esta seria la forma más normal
        para empezar a programar en php
    */
echo'Hola Mundo!';
?>
```

Lo guardamos como **hola.php** en */var/www* (en casos de linux) y en */wamp/www* o */wamp/htdocs*, dependiendo de como lo hayas instalado, vamos a nuestro navegador (les recomiendo Chrome o Firefox) e ingresamos al servidor:

127.0.0.1/hola.php

localhost/hola.php

Cualquiera de las 2 formas es válida.

El típico ejemplo de un "*Hola mundo!*" no puede faltar, en los ejemplos, he coloreado de distintos tipos para una mejor visualización, eso es lo que nos hará un editor de textos para programación como notepad++.

<?php – Le decimos a php que trabaje

echo – Nos sirve para mostrar algo en pantalla

Comentarios; nos sirve para dejar mensajes sin que sean ejecutados.

– Comentario de una línea

// – Otra forma de comentar de una línea.

/*

Comentario de

variaslineas...

*/

?> – Cerramos php

Nota: la función***echo*** *es intercambiable con* ***print,*** *en nuestro caso siempre usaremos* ***echo.***

Tenga en cuenta que la mejor forma de comentar un código es documentándolo con PHPDocumentor.

Cuando queramos poner más funciones en un mismo script, php nos obliga a usar un "separador" que es ";" (punto y coma), veamos un ejemplo:

```
<?php
echo 'Probando ejemplo <br>';
print'Taller php para Underc0de';
?>
```

Como vemos usamos las funciones **echo** y **print** con un separador cuando

terminamos una linea, podemos ejecutar código HTML/JS dentro de una instrucción echo/print, en este caso un *break o
para hacer un salto de línea*.

COMILLAS SIMPLES Y DOBLES

Como habrás notado ya, cuando imprimimos en pantalla podemos hacerlo con comillas simples '' o dobles “”, así que vamos a ver las diferencias entre usar unas u otras.

comillas.php:

```
<?php
$site = 'Underc0de';
$leng = 'PHP';
echo “Bienvenido a $site ! <br>”;
echo 'Este es el taller de '.$leng;
?>
```

comillas2.php

```
<?php
$site = 'Underc0de';
echo'El contenido de la variable $site es '.$site;
?>
```

Las comillas dobles ejecutan el contenido de la variable (proceso denominado parseo), en cambio con comillas simples se imprimirá el nombre de la variable en sí (porque es un literal), para poder ver el contenido de la variable hay que

dejarla fuera y utilizar un concatenador que unirá el texto con el valor de la variable como vemos en el ejemplo anterior.

VARIABLES

Una variable es un **contenedor** en donde podemos guardar datos, por ejemplo, en clase de matemáticas (geometría, por ejemplo) hablamos de un valor pi que equivale a 3.1416... (varía dependiendo del caso), y cuando vamos a usar ese valor, en vez de poner 3.1416... simplemente nos referimos a él como “pi” (algebra).

En php nos referimos a las variables con el signo de dollar$.

$PI = 3.1416;

$variable = valor;

TIPOS DE VARIABLES

- Int/Entero = Valoresenteros.
- str/Cadena = Cadenas de texto.
- bool/Booleanos = Valores lógicos, TRUE o FALSE.
- Float/Flotante = Numéricos con decimales.

Más información.

Variables.php

```
<?php
```

```php
$edad = 17;

$nombre = '2Fac3R';

$cool = TRUE;

$float = 2.6;


echo 'Bienvenido '.$nombre.' !, asi que tienes '.$edad.' años eehe.e ';


	if($cool == TRUE){

		// En caso de ser TRUE

		echo 'Bienvenido a Underc0de!! en su version'.$float;

	}else{

		// Sino...

		echo 'Solo los cools entran a Uc0de :P ';

	}


?>
```

Nota: El punto (**.**) nos sirve para concatenar.

En este pequeño script mostramos una nueva función, los condicionales, pero antes de irnos a ellos, vamos a ver los operadores matemáticos que disponemos:

Operadores aritméticos

Ejemplo	Nombre	Resultado
-$a	Negación	Opuesto de $a.
$a + $b	Adición	Suma de $a y $b.
$a - $b	Sustracción	Diferencia de $a y $b.
$a * $b	Multiplicación	Producto de $a y $b.
$a / $b	División	Cociente de $a y $b.
$a % $b	Módulo	Resto de $a dividido por $b.

Operadores.php

```php
<?php
$a = 10;
$b = 4;
$resta = $a - $b;
$suma = $a + $b;
echo "La resta es $resta<br>";
echo "La suma es $suma<br>";
?>
```

CONSTANTES

Una constante es un contenedor, donde podemos establecer datos, la diferencia con las variables es que estos datos serán así por el resto del código, no se pueden alterar, pero tienen mucha utilidad en códigos grandes donde por ejemplo tenemos un porcentaje de impuesto a calcular y este porcentaje puede variar con los años, pero en el código hay que usarlo mucho, para q

cuando cambie no tengamos que cambiarlo en todas partes se puede definir ese valor y si cambia solo cambiamos donde se definió y en el resto del código cambiará.

Otra opción que se me ocurre es declarar una constante con la versión del código, cuando hagamos cambios podemos cambiar ese valor en todo el código cambiando solo la declaración.

Una constante se declara:

Define('NOMBRE_CONSTANTE','valor');

Y luego en el código donde pongamos NOMBRE_CONSTANTE será como una variable, tendrá el valor que hallamos puesto, pero no se le puede guardar contenido.

Mi recomendación es que todas las constantes se declaren y utilizen en mayúsculas para identificarlas rápido.

Una constante predefinida como TRUE que es igual a 1 es aquella constante que viene sin necesidad de declararla, y se puede escribir tanto en minúscula como en mayúsculas pero recomiendo que sea mayusculas

CONDICIONALES

Los condicionales son la base de la programación y PHP no es la excepción, así que vamos a ver su sintaxis, su función y uso.

Así como en expresiones lógicas, se dice que si sólo si se cumple una condición entonces se realiza la acción.

Pseudocódigo:

```
Si (condición) { entonces...
        acciones...
} sino {
        acciones...
}
```

Si la condición se cumple, va a hacer algo, sino realizará otras acciones.

La condición se cumple cuando el resultado es Verdadero, en caso de ser falos ejecuta la segunda parte del if, el sino.

PHP dispone de una gran variedad de este tipo de condicionales, if(), Switch(), while(), do{}while(), for().

if, elseif, else

if.php

```
<?php


                # Condicional IF else
$num = 17;
$num2 = 15;


if($num>=$num2){ // si num es mayor o igual a num2
```

```
        // En caso de cumplirse la condición
        echo 'Es mayor o igual! <br>';
}else{ // Si la condición no se cumple
        echo 'No es mayor ni igual! <br>';
}
?>
```

Los operadores de comparación que podemos usar son:

<	Menorque
>	Mayor que
<=	Menor o igual
>=	Mayor o igual
==	Igual
===	Identico
!=	Distinto

Antes de continuar describiré la diferencia entre igual e idéntico, verán, igual se refiere a si el valor es igual, cuando se refiere a constantes por ejemplo podemos decir que 1 es = a TRUE, y 0 es igual a FALSE, por lo que si yo pongo

If(1==TRUE)

La condición se cumple, en cambio si pongo idéntico (===) entonces se requerirá que el tipo es el mismo, y un Integer (numero) no es lo mismo que un BOOLEAN (true o false) y así con otras cosas, por lo que el mito de que en php no hay tipos de datos, es falso, en realidad si los hay pero la gente no acostumbra a utilizarlos. Pero por esta razón también se crearon los

forzadores de tipos (int) (float) etc. Para que al guardar un valor en una variable se establezca el tipo para poder usar más sencillamente el triple símbolo, (=== !=== >== <==) etc.

Podemos hacer más condicionales dentro de otro (con elseif) por ejemplo:

Elseif.php

```
<?php

// Puedes ir cambiando el valor para probar
$edad = 20;

if ($edad<18) {
        echo 'Bienvenido a la secci&oacute;n de ni&ntilde;os!';
} elseif ($edad>18&&$edad>60) {
        echo 'Eres mayor de edad!';
}else{
        echo 'Ya eres viejo :P';
}

?>
```

Operadores lógicos

Ejemplo	Nombre	Resultado

Ejemplo	Nombre	Resultado
$a and $b	And (y)	TRUE si tanto $a como $b son TRUE.
$a or $b	Or (o inclusivo)	TRUE si cualquiera de $a o $b es TRUE.
$a xor $b	Xor (o exclusivo)	TRUE si $a o $b es TRUE, pero no ambos.
! $a	Not (no)	TRUE si $a no es TRUE.
$a && $b	And (y)	TRUE si tanto $a como $b son TRUE.
$a || $b	Or (o inclusivo)	TRUE si cualquiera de $a o $b es TRUE.

Mas información!.

Como podrás ver, podemos usar varios operadores que a la vez son intercambiables, por ejemplo “&&” es intercambiable con “and”.

WHILE

La estructura de un bucle while (un bucle, es un trozo de código que se ejecuta tantas veces mientras una condición se cumpla) es bastante sencilla, veamos un ejemplo:

```
while(condicion){
        acciones...
}
```

while.php

```
<?php
        # Contador simple en php con while()
```

```php
$contador = 1;


while ($contador<=10) { // Mientras sea menor o igual a 10
        echo "Valor: $contador<br>";
        $contador++; // Incrementamos 1 a la variable
}


?>
```

Declaramos una variable de inicio para el contador, entramos en la condición del while y mientras se cumpla mostraremos el valor de la variable luego la incrementamos en 1 para no entrar en un bluce infinito, fácil no? ;).

En el caso de do while es “casi igual”, primero ejecuta la sentencia y después comprueba que se cumpla la condición:

do_while.php

```php
<?php


$contador = 1;


do {
        echo "Valor: $contador<br>";
        $contador++;
} while ($contador<=10)
```

```
?>
```

Cuando tenemos un código largo con muchos condicionales, en vez de usar tanto *elseif* en un *if* tenemos la opción de *switch*, que es más profesional y de un programador real.

sintaxis:

```
switch(valor){
        casevalor:
                acciones...
                break;
        caseotro:
                acciones...
                break;
        default:
                break;
}
```

switch.php

```
<?php
$color = 'red';
switch ($color) {
        case 'red':
                echo "<p style='color:$color'>Probando switch </p>";
```

```
            break;
        case 'green':
            echo "<p style='color:$color'>Probando switch </p>";
            break;
        default:
            echo 'No se ha seleccionado un color valido';
            break;
}

?>
```

La función *break* nos sirve para terminar las instrucciones del *case*, puedes probar con *elseif* para que compruebes la flexibilidad de la función switch :).

FOR

Otra forma de llevar un bucle de forma más profesional, es el famoso *for()*, veamos su sintáxis:

```
for(inicio;condición;iteracción){
    acciones...
}
```

for.php

```
<?php
```

```php
// Contador con for
for ($i =1; $i<= 10 ; $i++) {
        echo "Valor : $i<br>";
}

?>
```

Primero asignamos a la variable "**$i**" el valor inicial (1), separamos cada con punto y coma (**;**), después agregamos nuestra condición, y por último una acción que se ejecute en cada iteracción (en cada vuelta), aumentamos la variable.

Como vemos, el ejemplo del contador con for, es más profesional y compacto, por lo tanto una mejor forma para programar.

RECOMENDACIONES:

- En el nombre de tus variables, pon un nombre acorde a al script.
- Siempre usa ";" a pesar de que no sea necesario.
- Trata de optimizar tu código al máximo.
- Practica lo más que puedas.
- Busca por tu cuenta, no esperes que todo venga en algún manual.