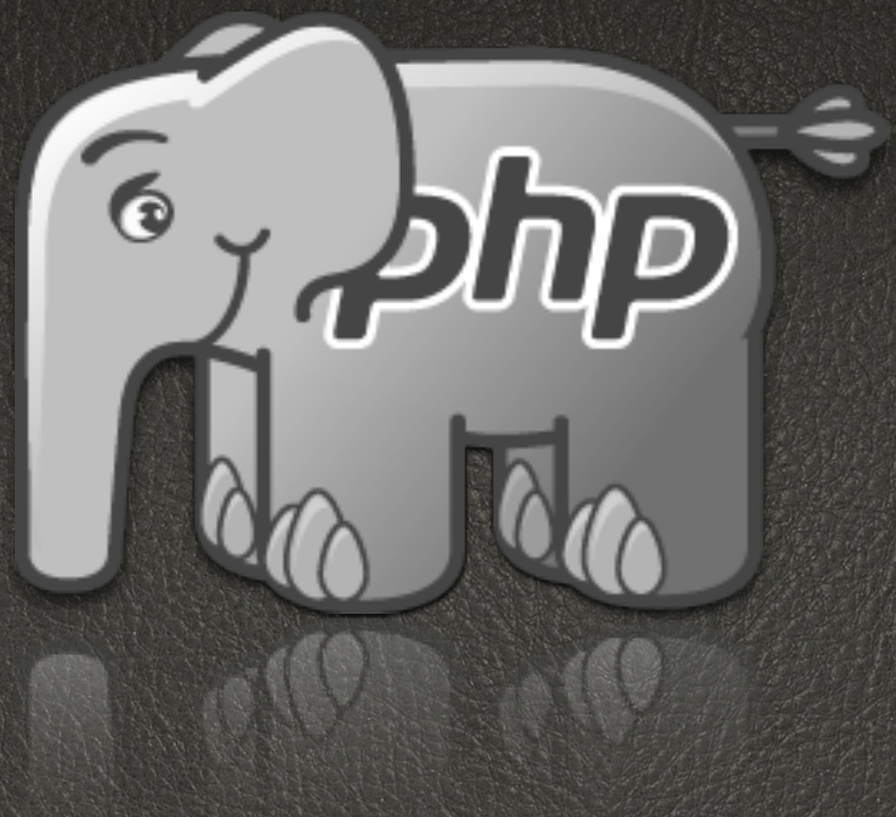


PHP

(CONCEPTOS BÁSICOS/AVANZADOS)

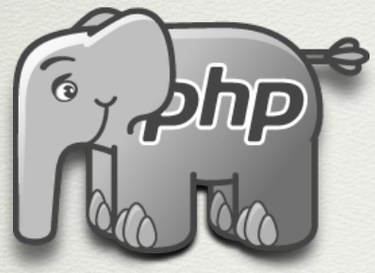
Oscar Fernando Aristizábal Cardona
ofaczero@gmail.com

Introducción

- Lenguaje de programación tipo script para entornos Web, que se ejecuta del lado del servidor.

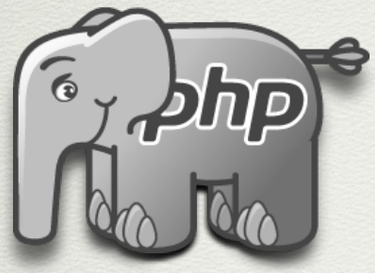
- PHP es un poderoso instrumento para hacer páginas web dinámicas e interactivas.

- PHP es ampliamente utilizado, libre y eficiente alternativa a los competidores como ASP de Microsoft.



Qué es?

- PHP significa: Hypertext Preprocessor
- PHP es un lenguaje de script del lado del servidor, como ASP
- Scripts PHP se ejecutan en el servidor
- PHP soporta muchas bases de datos (MySQL, Informix, Oracle, Sybase, PostgreSQL, Interbase, ODBC, etc)
- PHP es un software de código abierto
- PHP es libre de descargar y utilizar



Qué es?

¿Qué es un archivo PHP?

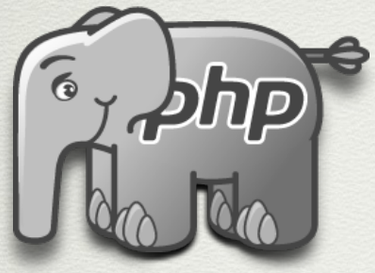
- Los archivos de PHP pueden contener texto, etiquetas HTML y scripts.
- Los archivos PHP son devueltos al navegador Web como codigo HTML.
- Los archivos de PHP tiene una extensión de archivo ". php”.

¿Qué es MySQL?

- MySQL es un servidor de base de datos
- MySQL es ideal para aplicaciones pequeñas y grandes
- MySQL soporta el estándar SQL
- MySQL se compila tanto en plataformas Windows como Linux
- MySQL es gratuito para descargar y utilizar

PHP + MySQL

- PHP en combinación con MySQL son multiplataforma (se puede desarrollar en Windows y servir en una plataforma Unix)



Por qué?

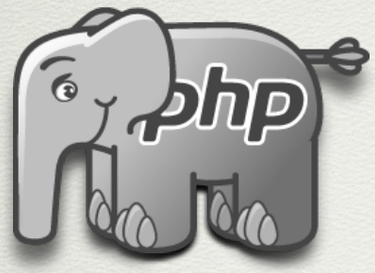
¿Por qué PHP?

- PHP corre en diferentes plataformas (Windows, Linux, Unix, etc)
- PHP es compatible con casi todos los servidores que se utilizan hoy en día (Apache, IIS, etc)
- PHP es gratis.
- PHP es fácil de aprender y se ejecuta de manera eficiente en el lado del servidor.

Por dónde empezar?

Para obtener acceso a un servidor web con soporte PHP, usted puede:

- Instalación de Apache (o IIS) en su propio servidor, instalar PHP y MySQL "Local".
- O encontrar un plan de alojamiento web con PHP y MySQL "Remoto".



Instalar

¿Qué necesito?

- Si su servidor soporta PHP que no necesita hacer nada.
- Basta con crear algunos archivos .php en tu directorio web, y el servidor los interpretará para usted. Porque es libre, la mayoría de servidores web ofrecen soporte para PHP.

Sin embargo, si su servidor no soporta PHP, debe instalar PHP.

Descargar PHP gratis aquí:
http://www.php.net/downloads.php

Descargar Base de Datos MySQL gratis aquí:
http://www.mysql.com/downloads/

Descargar Servidor Apache de forma gratuita aquí:
http://httpd.apache.org/download.cgi

Sintaxis

Sintaxis básica de PHP

Un bloque de programación PHP siempre comienza con **<?php** y termina con **?>**. Un bloque de secuencias de comandos PHP se pueden colocar en cualquier parte del documento.

En los servidores con modo abreviado (shorthand) activado, usted puede iniciar un bloque de secuencias de comandos con **<?** y terminan con **?>**.

Para la máxima compatibilidad, es recomendable que utilice la forma estándar (**<?php**) en vez de la forma abreviada.

```
<?php
?>
```

Sintaxis

Un archivo PHP normalmente contiene etiquetas HTML, como un archivo HTML, y algo de código PHP.

A continuación, tenemos un ejemplo de un simple script PHP que envía el texto “Hola mundo ADSI :)" al navegador:

```
<html>
<body>
<?php
    echo “Hola mundo ADSI :)”;
?>
</body>
</html>
```

Cada línea de código en PHP debe terminar con un punto y coma. El punto y coma es un separador y se utiliza para distinguir un conjunto de instrucciones de otro.

Hay dos estados básicos de texto de salida con PHP: **echo** y **print**.

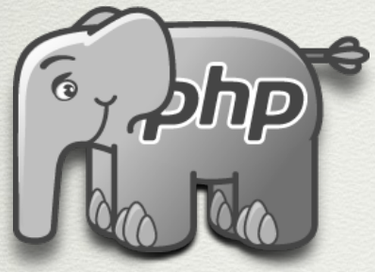
Nota: El archivo debe tener una extensión de PHP. Si el archivo tiene una extensión. html, el código PHP no será ejecutada.

Sintaxis

Comentarios en PHP:

En PHP, usamos **//** para hacer un comentario de una sola línea o **/*** y ***/** para hacer un bloque de comentario general.

```
<html>
<body>
<?php
    // Esto es un comentario de una linea
    /*
       Esto es un
       bloque de
       comentarios.
    */
?>
</body>
</html>
```



Variables

Variables en PHP:

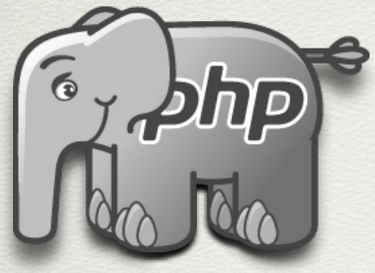
- Las variables se utilizan para almacenar valores, como cadenas de texto, números o matrices.
- Cuando se declara una variable, puede ser utilizada una y otra vez en el script.
- Todas las variables en PHP comienzan con un símbolo de signo **$**.
- La forma correcta de declarar una variable en PHP:

```
$nombre_variable = valor;
```

En PHP a menudo se olvida el signo **$** al principio de la variable. En ese caso no va a funcionar.

Vamos a intentar crear una variable que contiene una cadena y una variable que contiene un número:

```
<?php
    $txt = “Hola mundo!”;
    $num = 5;
?>
```



Variables

PHP es un lenguaje flexible

- En PHP, una variable no tiene que ser declarada antes de agregar un valor a la misma.
- En el ejemplo anterior, usted no tiene que decirle a PHP que tipo de datos es la variable, PHP automáticamente convierte la variable con el tipo de datos correcto, dependiendo de su valor.
- En un lenguaje de programación fuertemente tipado, lo que tienes que declarar (definir) el tipo y el nombre de la variable antes de usarla.
- En PHP, la variable se declara de forma automática cuando se utiliza.

Normas para la denominación de las variables:

- Un nombre de variable deben comenzar con una letra o un guión bajo "_"
- A nombre de la variable sólo puede contener caracteres alfanuméricos y guiones bajos (az, AZ, 0-9, y _)
- Un nombre de variable no debe contener espacios. Si un nombre de variable es más que una palabra, deben ser separados con un guión bajo (**$mi_cadena**), o con la capitalización (**$miCadena**)

Cadenas

Las variables de cadena en PHP:

Las variables de cadena se utilizan para los valores que contiene caracteres.

Después de crear una cadena que puede manipular. Una cadena se puede utilizar directamente en una función o puede ser almacenado en una variable.

A continuación, el script PHP asigna el texto “Hola Mundo!” a una variable de cadena llamada **$texto**:

```
<?php
    $texto = “Hola mundo!”;
    echo $texto;
?>
```

Cadenas

El operador de concatenación:

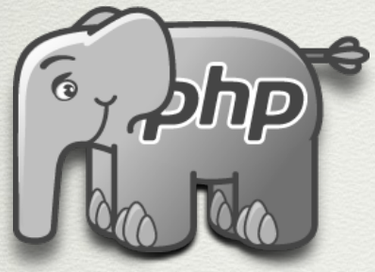
Sólo hay un operador de cadenas en PHP.

El operador de concatenación (.) Se utiliza para poner dos valores de cadena juntos.

Para concatenar dos variables de cadena en conjunto, utilice el operador de concatenación:

```php
<?php
    $texto1 = “Hola mundo!”;
    $texto2 = “Te saluda ADSI…”;
    echo $texto1 . “ ” . $texto2;
?>
```

Si nos fijamos en el código anterior se ve que hemos utilizado el operador de concatenación dos veces. Esto se debe a que hemos tenido que insertar una tercera cadena (un carácter de espacio), para separar las dos cadenas.



Cadenas

La función strlen():

La función **strlen()** se utiliza para devolver la longitud de una cadena.

Vamos a encontrar la longitud de una cadena:

```
<?php
    echo strlen.(“Hola Mundo”);
?>
```

La funcion strpos():

La función **strpos()** se utiliza para buscar caracteres en una cadena.

Si se encuentra una coincidencia, esta función devolverá la posición de la primera coincidencia. Si no hay coincidencias, devolverá FALSO.

```
<?php
    echo strpos.(“Hola Mundo”, “Mundo”);
?>
```

Operadores

- **Operadores Aritméticos:**

Operador	Descripción	Ejemplo	Resultado
+	Adición	x=2 x+2	4
-	Substracción	x=2 5-x	3
*	Producto	x=4 x*5	20
/	División	15/5	3
%	Residuo	5%2	1
++	Incremento	x=5 x++	x=5
- -	Decremento	x=5 x- -	x = 4

Operadores

- **Operadores Asignación:**

Operador	Ejemplo	Es igual que
=	x = y	x = y
+=	x += y	x = x+y
-=	x -= y	x = x-y
*=	x *= y	x = x*y
/=	x /= y	x = x/y
.=	x .= y	x = x.y
%=	x %= y	x = x%y

Operadores

- **Operadores Comparación:**

Operador	Descripción	Ejemplo
==	Es igual	5 == 8, falso
!=	No es igual	5 != 8, verdadero
<>	No es Igual	5 <> 8, verdadero
>	Es mayor que	5 > 8, falso
<	Es menor que	5 < 8, verdadero
>=	Es mayor o igual que	5 >= 8, falso
<=	Es menor o igual que	5 <= 8, verdadero

Operadores

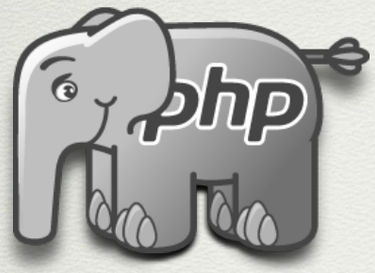
- **Operadores Lógicos:**

Operador	Descripción	Ejemplo

&&	“Y”	x = 6;
		y = 3;
		(x < 10 && y > 1), verdadero

||	“O"	x = 6;
		y = 3;
		(x == 5 || y == 5), falso

!	No	x = 6;
		y = 3;
		!(x == y), verdadero



1. Condicionales

Instrucciones condicionales:

Muy a menudo cuando se escribe código, se desea llevar a cabo diferentes acciones para diferentes decisiones, puede utilizar instrucciones condicionales en el código para hacer esto.

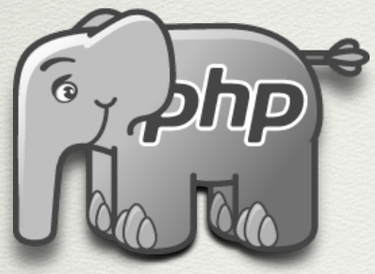
En PHP tenemos las instrucciones condicionales siguientes:

>> if - Utilizar esta instrucción para ejecutar un código si una condición especificada es verdadera.

>> else - Utilizar esta instrucción para ejecutar un código si una condición especificada es falsa.

>> elseif - Utilizar esta instrucción para seleccionar uno de varios bloques de código que se ejecutará.

>> switch - Utilizar esta instrucción para seleccionar uno de los muchos bloques de código que se han ejecutado.



2. Condicionales

La declaración if:

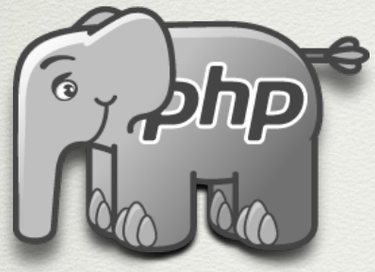
Utilice la sentencia **if** para ejecutar código si la condición especificada es verdadera.

Sintaxis:

```
<?php
    if(condición)
    // Código a ejecutar si la condición es verdadera
?>
```

El siguiente ejemplo imprime “Tenga un buen fin de semana!" si el día de hoy es el viernes:

```
<?php
    $dia = date(“D”);
    if($dia == “Fri”)
        echo “Tenga un buen fin de semana!”;
?>
```



3. Condicionales

La declaración else:

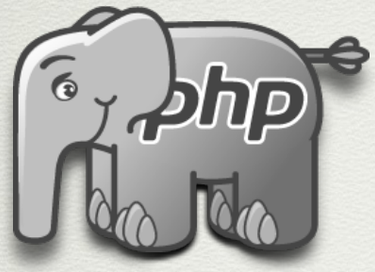
Utilice la instrucción **else** para ejecutar un código si una condición es verdadera y otro código si una condición es falsa.

Sintaxis:

```
<?php
    if(condición)
        // Código a ejecutar si la condición es verdadera
    else
        // Código a ejecutar si la condición es falsa
?>
```

El siguiente ejemplo imprime “Tenga un buen fin de semana!" si el día de hoy es el viernes, de lo contrario, imprime “Tenga un buen día!":

```
<?php
    $dia = date(“D”);
    if($dia == “Fri”)
        echo “Tenga un buen fin de semana!”;
    else
        echo “Tenga un buen día!”;
?>
```



4. Condicionales

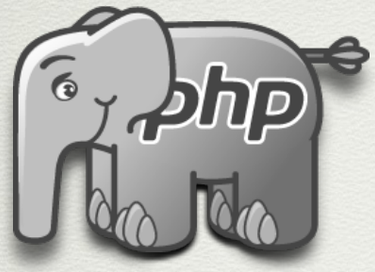
La declaración elseif:

Utilice la instrucción **elseif** para seleccionar uno de varios bloques de código que se ejecutará.

Sintaxis:

```php
<?php
     if(condición)
          // Código a ejecutar si la condición es verdadera
     elseif
          // Código a ejecutar si la condición es verdadera
     else
          // Código a ejecutar si la condición es falsa
?>
```

```php
<?php
     $dia = date(“D”);
     if($dia == “Fri”)
          echo “Tenga un buen fin de semana!”;
     elseif($dia == “Sun”)
          echo “Tenga un buen Domingo!”;
     else
          echo “Tenga un buen día!”;
?>
```



5. Condicionales

La declaración switch:

Usar la sentencia **switch** para seleccionar uno de los muchos bloques de código que se han ejecutado.

Sintaxis:

```php
<?php
     switch(num) {
          case 1:
               // Código a ejecutar si num = 1;
               break;
          case 2:
               // Código a ejecutar si num = 2;
               break;
          default:
               // Código a ejecutar si num != 1 y num != 2;
     }

?>
```

1. Matrices

¿Qué es una matriz?

- Una variable puede almacenar un número o texto. El problema es, que una variable tendrá un solo valor.

- Una matriz es una variable especial, que puede almacenar varios valores en una sola variable.

- Si usted tiene una lista de elementos (por ejemplo una lista de marcas de autos), el almacenamiento de las marcas de autos en una sola variable, seria de la siguiente forma:

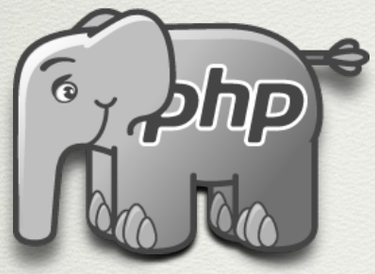
```php
<?php
     $carro1 = “BMW”;
     $carro2 = “VolksWagen”;
     $carro3 = “Mercedez Benz”;
?>
```

- Sin embargo, si usted desea almacenar 500 marcas de autos?

- La mejor solución es utilizar una matriz!

- Una matriz puede contener todos los valores de las variables bajo un solo nombre. - Usted puede acceder a los valores por referencia al nombre de la matriz.

- Cada elemento de la matriz tiene su propio índice para que pueda acceder fácilmente.



2. Matrices

Matrices numéricas:

Una matriz numérica almacena cada elemento de la matriz con un índice numérico.

Hay dos métodos para crear una matriz numérica.

1. El índice se asignan automáticamente (el índice comienza en 0):

```php
<?php
     $carros = array(“BMW”, “VolksWagen”, “Mercedez Benz”);
?>
```

2. El índice se asigna de forma manual:

```php
<?php
     $carros[0] = “BMW”;
     $carros[1] = “VolksWagen”;
     $carros[2] = “Mercedez Benz”;
?>
```

3. Matrices

Matrices asociativas:

En una matriz asociativa, cada clave de identificación se asocia a un valor.

Con arreglos asociativos podemos utilizar los valores como claves y asignar valores a ellos.

En este ejemplo se utiliza una matriz para asignar edades a las diferentes personas:

```php
<?php
    $edades = array("Hugo"=> 32, "Paco"=> 50, "Luis"=> 20);

    $edades["Hugo"] = 32;
    $edades["Paco"] = 50;
    $edades["Luis"] = 20;
?>
```

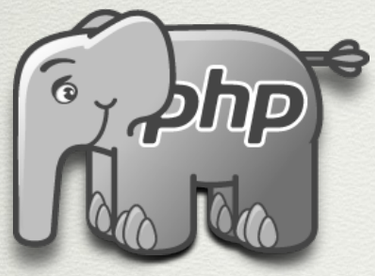
4. Matrices

Matrices multidimensionales:

En una matriz multidimensional, cada elemento de la matriz principal también puede ser una matriz y cada elemento en la sub-matriz puede ser una matriz, y así sucesivamente.

En este ejemplo vamos a crear una matriz multidimensional, con claves de identificación asignado automáticamente:

```
<?php
     $familias = array(
                          “Goku”=> array(
                                        “Gohan”,
                                        “Goten”),
                          “Vegeta”=> array(
                                          “Trunks”,
                                          “Bura”)
     );
?>
```



1. Bucles

A menudo, cuando se escribe código, desea que el mismo bloque de código se ejecute una y otra vez. En lugar de agregar varias líneas casi iguales en una secuencia de comandos.

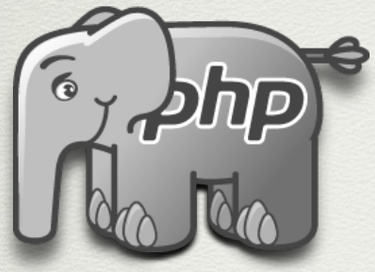
En PHP, tenemos las siguientes declaraciones de bucle:

>> while: - Bucles a través de un bloque de código mientras una condición especificada sea verdadero.

>> do while: - Bucles a través de un bloque de código una vez, y luego repite el bucle mientras una condición especificada sea verdadera.

>> for: - Bucles a través de un bloque de código un número determinado de veces.

>> foreach: - Bucle a través de un bloque de código para cada elemento de una matriz.



2. Bucles

Bucle while:

El bucle **while** ejecuta un bloque de código mientras una condición es verdadera.

Sintaxis:

```
<?php
     while (condición) {
          // Código que será ejecutado.
     }
?>
```

```
<?php
     $i = 1;
     while ($i <= 10) {
          echo “El número es:”. $i . “<br>”;
          $i++;
     }
?>
```

3. Bucles

Bucle do while:

El bucle **do while** siempre ejecutara un bloque de código una vez, y se encargara de verificar la condición, y repite el bucle mientras la condición sea verdadera.

Sintaxis:

```php
<?php
     do {
          // Código que será ejecutado.
    }
    while (condición);
?>
```

```php
<?php
     $i = 1;
     do {
          $i++;
          echo “El número es:”. $i . “<br>”;
    }
    while ($i <= 10);
?>
```

4. Bucles

Bucle for:

El bucle **for** se utiliza cuando se sabe de antemano cuántas veces la secuencia de comandos se debe ejecutar.

Sintaxis:

```php
<?php
    for (inicializar; condición; incremento) {
        // Código que será ejecutado.
    }
?>
```

Parámetros:

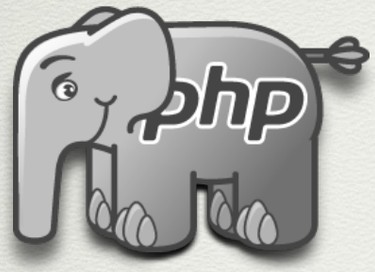
- **inicializar:** Generalmente se utiliza para establecer un contador.
- **condición:** Evaluado por cada iteración del bucle. Si se evalúa como TRUE, el bucle continúa. Si se evalúa como FALSE, el bucle termina.
- **incremento:** Generalmente se utiliza para incrementar un contador.

5. Bucles

Bucle for:

El ejemplo siguiente define un ciclo que comienza con i = 1. El bucle continuará ejecutándose mientras i es menor o igual a 5. i se incrementará en 1 cada vez que el bucle se ejecuta:

```
<?php
     for ($i=1; $i<=5; $i++) {
          echo “El número es:”. $i . “<br>”;
     }
?>
```



6. Bucles

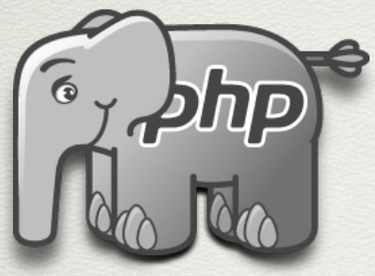
Bucle foreach:

El bucle **foreach** se utiliza para recorrer matrices.

Sintaxis:

```php
<?php
     foreach ($arreglo as $valor) {
          // Código que será ejecutado.
     }
?>
```

```php
<?php
     $x = array("Uno", "Dos", "Tres");
     foreach ($x as $valor) {
          echo "El valor es:". $valor . "<br>";
     }
?>
```



1. Funciones

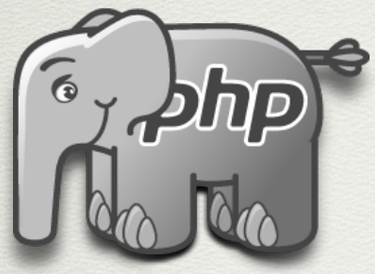
Funciones PHP:

- El verdadero poder de PHP viene de sus funciones.
- Una función será ejecutado por una llamada a la función.
- Usted puede llamar a una función desde cualquier lugar dentro de una página.

Sintaxis:

```php
<?php
     function nombreFuncion() {
          // Código que será ejecutado.
     }
?>
```

- Recomendable dar a la función un nombre que refleja lo que hace la función.
- El nombre de la función puede comenzar con una letra o un guión bajo (no un número).



2. Funciones

Ejemplo:

```
<?php
     function escribirNombre() {
          echo “Jeremias Sprindfield”;
     }
     echo “Mi nombre es: ”;
     escribirNombre();
?>
```

Salida:

```
Mi Nombre es: Jeremias Sprindfield
```

3. Funciones

Adición de parámetros:

- Para agregar más funcionalidad a una función, podemos agregar parámetros. Un parámetro es igual que una variable.
- Los parámetros se especifican después del nombre de la función, dentro de los paréntesis.

```php
<?php
     function escribirNombre($primerNombre) {
          echo $primerNombre . “ Flanders”;
    }
     echo “Mi nombre es: ” . escribirNombre(“Ned”);
     echo “El nombre de mi Esposa es: ” . escribirNombre(“Maude”);
     echo “Los nombres de mis Hijos son: ” . escribirNombre(“Rod y Todd”);
?>
```

4. Funciones

Retornar Valores:

Para que una función devuelve un valor, utilice la instrucción return.

Ejemplo:

```
<?php
     function sumar($n1, $n2) {
          $total = $n1 + $n2”;
          return $total;
     }
     echo “2 + 18: ” . sumar(2,18);
?>
```

Salida:

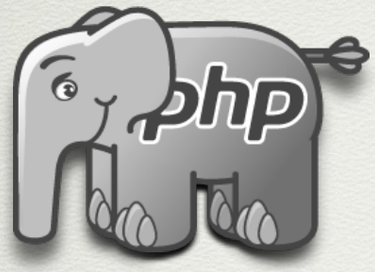
```
2 + 18 = 20
```

1. Formularios

Manejo de Formularios en PHP:

- Lo más importante tener en cuenta cuando se trata de formularios HTML y PHP es que cualquier elemento de formulario en una página HTML de forma automática estará disponible para los scripts PHP.
- En PHP las variables **$_GET** y **$_POST** se utilizan para recuperar información de formularios, como los campos de texto "inputs".

```
<form action=“” method=“post”>
     Nombre: <input type=“text” name=“nombre”>
     Edad: <input type=“number” name=“edad”>
     <input type=“submit” value=“Enviar”>
</form>
<?php
     if($_POST) {
          echo “Bienvenido: ”.$_POST[‘nombre’].”<br>”;
          echo “Su edad es: ”.$_POST[‘edad’].” años.”;
     }
?>
```



2. Formularios

La función $_GET:

- La función **$_GET** se utiliza para recoger los valores de un formulario enviado con **method="get"**.
- La información enviada desde un formulario con el método GET es visible para todos (Se muestra en la barra de direcciones del navegador) y tiene límites en el cantidad de información a enviar (máx. 100 caracteres).
- El archivo PHP ahora puede utilizar la función **$ _GET** para recopilar datos del formulario (los nombres de los campos del formulario será automáticamente las claves en el array **$ _GET**

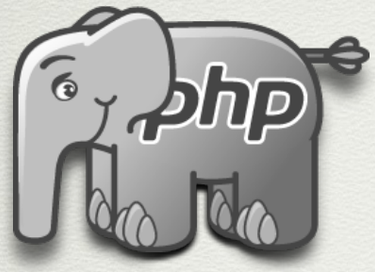
```
<form action=“” method=“get”>
     Nombre: <input type=“text” name=“nombre”>
     Edad: <input type=“number” name=“edad”>
     <input type=“submit” value=“Enviar”>
</form>
<?php
     if($_GET) {
          echo “Bienvenido: ”.$_GET[‘nombre’].”<br>”;
          echo “Su edad es: ”.$_GET[‘edad’].” años.”;
     }
?>
```

3. Formularios

La función $_POST:

- La función **$_POST** se utiliza para recoger los valores de un formulario enviado con **method="post"**.
- La información enviada desde un formulario con el método POST es invisible a los demás y no tiene límites en la cantidad de información a enviar.
- **Nota:** Sin embargo, hay un tamaño máximo de 8 Mb para el método POST, por defecto (puede ser cambiado mediante el establecimiento de la post_max_size en el archivo php.ini).
- El archivo PHP ahora puede utilizar la función **$_POST** para recopilar datos del formulario (los nombres de los campos del formulario será automáticamente las claves de la matriz **$_POST**)

```
<form action=“” method=“post”>
     Nombre: <input type=“text” name=“nombre”>
     Edad: <input type=“number” name=“edad”>
     <input type=“submit” value=“Enviar”>
</form>
<?php
     if($_POST) {
          echo “Bienvenido: ”.$_POST[‘nombre’].”<br>”;
          echo “Su edad es: ”.$_POST[‘edad’].” años.”;
     }
?>
```



4. Formularios

La función $_REQUEST:

- La función **$_REQUEST** permite el manejo de los contenidos de ambas funciones **$_GET**, **$_POST** y **$_COOKIE**.
- La función **$_REQUEST** se puede utilizar para recopilar datos de formularios enviados con los dos métodos GET y POST.

```
<form action=“” method=“post”>
     Nombre: <input type=“text” name=“nombre”>
     Edad: <input type=“number” name=“edad”>
     <input type=“submit” value=“Enviar”>
</form>
<?php
     if($_POST) {
          echo “Bienvenido: ”.$_REQUEST[‘nombre’].”<br>”;
          echo “Su edad es: ”.$_REQUEST[‘edad’].” años.”;
     }
?>
```

1. Fecha y Hora

La función date() de PHP:

- La función **date()** es utilizada para indicar la fecha y/o el tiempo en la que un determinado evento ha ocurrido.

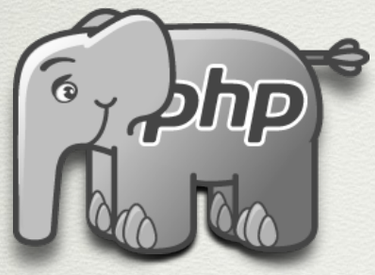
Sintaxis:

```
date(format, timestamp);
```

Formato de la Fecha

El parámetro **format** requerido en la función date() especifica cómo dar formato a la fecha y hora.

d - Representa el día del mes (01 a 31)
m - Representa un mes (01 a 12)
Y - Representa un año (en cuatro dígitos)



2. Fecha y Hora

Ejemplo:

```
<?php
     echo date(“d/m/Y”).”<br>”;
     echo date(“d.m.Y”).”<br>”;
     echo date(“d-m-Y”);
?>
```

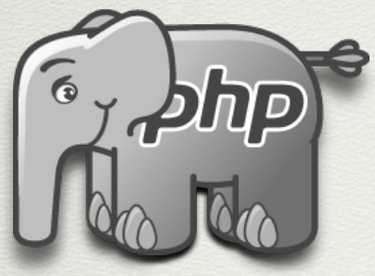
Agregar timestamp

El parámetro **timestamp** en la función **date()** especifica una fecha y hora. Si no se especifica una fecha y hora, se utilizaran las actuales.

Sintaxis para mktime()

```
mktime(hour, minute, second, month, day, year);
```

```
<?php
     $manana = mktime(0, 0, 0, date(‘m’), date(‘d’)+1, date(‘Y’));
     echo “Mañana es: ”.date(“d/m/Y”, $manana);
?>
```



1. SSI

Server Side Includes (SSI)

Insertar el contenido de un archivo PHP en otro archivo PHP es posible gracias a las funciones **include()** o **require()**.

Las dos funciones son idénticas en todos los sentidos, excepto la forma en que manejan errores:

- **include()** genera una advertencia, pero el script seguirá la ejecución
- **require()** genera un error fatal, y el script se detendrá

Estas dos funciones se utilizan para crear funciones, encabezados, pies de página, o elementos que se pueden reutilizar en varias páginas.

La verdadera utilidad de incluir código PHP es evitar repetir código, y hacer mas administrable nuestras aplicaciones.

Ejemplo: Podríamos tener un encabezado con un menú de opciones, este encabezado podría estar presente en 100 páginas, ósea que tendríamos que modificar este encabezado 100 veces, pero si lo incluimos en un archivo PHP solo lo modificaríamos una sola vez.

2. SSI

Función include():

La función include() toma todo el contenido en un archivo especificado y lo incluye en el archivo actual.

Si se produce un error, la función include () genera una advertencia, pero el script seguirá la ejecución.

Ejemplo:

```
<?php include(“encabezado.php”); ?>

<h1>Bienvenido a mi página Web!</h1>
<p>Lorem Ipsum Dolor Sit Amet</p>
```

3. SSI

Función require():

La función **require()** es idéntica a **include()**, excepto que controla los errores de forma diferente.

El require() genera un error fatal, y el script se detendrá.

Ejemplo:

```
<?php
     require(“noexiste.php”);

     echo “<h1> Bienvenidos a mi página Web </h1>”;
?>
```

1. Archivos

- **Abrir un archivo:**

La función **fopen()** se utiliza para abrir archivos en PHP.

El primer parámetro de esta función contiene el nombre del archivo que se abrirá y el segundo parámetro especifica el modo en que el archivo debe ser abierto:

```
<?php
     $archivo = fopen(“contenido.txt”, “r”);
?>
```

- **Cerrar un Archivo:**

El **fclose()** se utiliza para cerrar un archivo abierto:

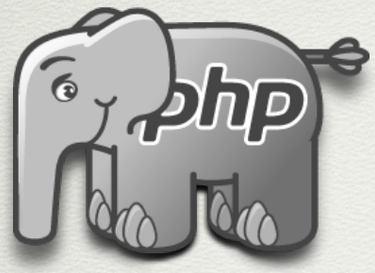
```
<?php
     $archivo = fopen(“contenido.txt”, “r”);

     fclose($archivo);
?>
```

2. Archivos

- **El archivo puede ser abierto en uno de los siguientes modos:**

Modos	Descripción
r	Sólo lectura, Comienza al inicio del archivo.
r+	Lectura/Escritura, Comienza al inicio del archivo.
w	Sólo Escritura, Borra el contenido del archivo o crea un nuevo archivo si no existe.
w+	Lectura/Escritura, Borra el contenido del archivo o crea un nuevo archivo si no existe.
a	Abre un archivo de sólo escritura. Los datos existentes en el archivo se conservan.
a+	Abre un archivo de sólo lectura/escritura. Los datos existentes en el archivo se conservan.
x	Crea un nuevo archivo de sólo escritura. Retorna FALSE y un error si ya existe el archivo.
x+	Crea un nuevo archivo de sólo lectura/escritura. Retorna FALSE y un error si ya existe el archivo.

```php
<?php
    $archivo = fopen(“contenido.txt”, “r”) or exit(“No se pudo abrir!”);
?>
```



3. Archivos

- **Comprobar al final de su archivo:**

El **feof()** comprueba si el "fin de archivo (EOF) se ha alcanzado.

El **feof()** es útil para recorrer a través de datos de longitud desconocida.

Nota: Usted no puede leer los archivos abiertos en modo w, a, y x.

```
<?php
     if(feof($archivo)) echo “Final del archivo!”;
?>
```

- **Lectura de un archivo línea por línea:**

El **fgets()** se utiliza para leer una sola línea de un archivo.

Nota: Después de una llamada a esta función el puntero de archivo se ha movido a la siguiente línea.

Ejemplo:

El siguiente ejemplo lee un archivo línea por línea, hasta el final del archivo:

```
<?php
     $archivo = fopen(“contenido.txt”, “r”) or exit(“No se pudo abrir!”);
     while(!feof($archivo)) {
          echo fgets($archivo).“<br>”;
     }
     fclose($archivo);
?>
```

4. Archivos

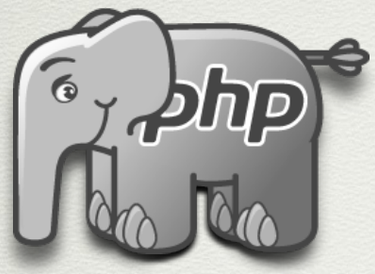
- **Lectura de un archivo carácter por carácter:**

El **fgetc()** se utiliza para leer un solo carácter de un archivo.

Nota: Después de una llamada a esta función se mueve el puntero de archivo al siguiente carácter.

Ejemplo:
El siguiente ejemplo lee un archivo carácter por carácter, hasta el final del archivo:

```
<?php
     $archivo = fopen(“contenido.txt”, “r”) or exit(“No se pudo abrir!”);
     while(!feof($archivo)) {
          echo fgetc($archivo).“-”;
     }
     fclose($archivo);
?>
```



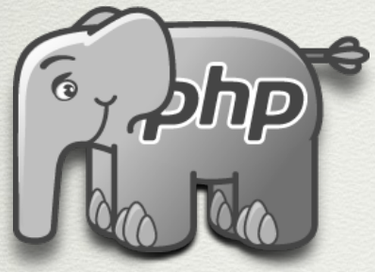
1. Subir Archivos

- **Permite a los usuarios subir archivos al servidor:**

```
<form action="subir.php" method="post" enctype="multipart/form-data">
    <label for="archivo">Archivo:</label>
    <input type="file" name="archivo" id="archivo">
    <br>
    <input type="submit" name="submit" value="Subir">
</form>
```

- El atributo **enctype** de la etiqueta **<form>** especifica qué tipo de contenido va a enviar el formulario. "**multipart/form-data**" se utiliza cuando el formulario requiere datos binarios, como el contenido de un archivo, para ser subido.

- El **type="file"** atributo de la etiqueta **<input>** especifica que la entrada debe ser procesada como un archivo.



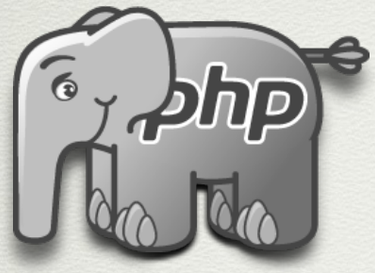
2. Subir Archivos

- **Crear el script de carga:**

El archivo "subir.php" contiene el código para cargar un archivo:

```
<?php
     if($_FILES[“file”][“error”] > 0) {
          echo “Error: ”.$_FILES[“file”][“error”].“<br>”;
     }
     else {
          echo “Archivo: ”.$_FILES[“file”][“name”].“<br>”;
          echo “Tipo: ”.$_FILES[“file”][“type”].“<br>”;
          echo “Tamaño: ”.$_FILES[“file”][“size”] / 1024.“ Kb <br>”;
          echo “Almacenado en: ”.$_FILES[“file”][“tmp_name”];
     }
?>
```

$_FILES ["file"] ["name"] - El nombre del archivo subido.
$_FILES ["file"] ["type"] - El tipo de archivo subido.
$_FILES ["file"] ["size"] - El tamaño en bytes del archivo subido.
$_FILES ["file"] ["tmp_name"] - El nombre de la copia temporal del archivo.
$_FILES ["file"] ["error"] - El código de error resultante de la carga de archivos.

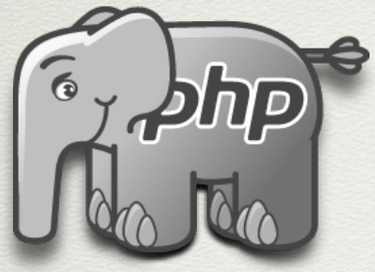


3. Subir Archivos

- **Restricciones en la Carga:**

El usuario sólo puede cargar archivos gif o jpeg y el tamaño del archivo debe ser de menos de 20 kb.

```php
<?php
     if((($_FILES[“file”][“type”] == “image/gif”)
     ||  ($_FILES[“file”][“type”] == “image/jpg”))
     &&  ($_FILES[“file”][“size”] < 20000)) {
          if($_FILES[“file”][“error”] > 0) {
               echo “Error: ”.$_FILES[“file”][“error”].“<br>”;
          }
          else {
               echo “Archivo: ”.$_FILES[“file”][“name”].“<br>”;
               echo “Tipo: ”.$_FILES[“file”][“type”].“<br>”;
               echo “Tamaño: ”.$_FILES[“file”][“size”] / 1024.“ Kb <br>”;
               echo “Almacenado en: ”.$_FILES[“file”][“tmp_name”];
          }
     }
     else {
          echo “Archivo Invalido:”;
     }
?>
```

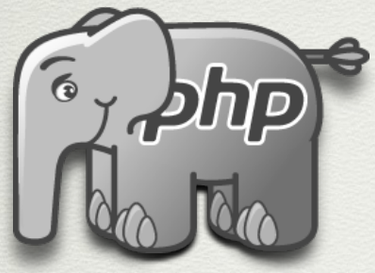


4. Subir Archivos

- **Guardar el archivo Subido 1:**

Para guardar el archivo subido es necesario copiarlo en una ubicación diferente.

```
<?php
     if(($_FILES[“file”][“type”] == “image/gif”)
     ||  ($_FILES[“file”][“type”] == “image/jpg”))
     &&  ($_FILES[“file”][“size”] < 20000)) {
          if($_FILES[“file”][“error”] > 0) {
               echo “Error: ”.$_FILES[“file”][“error”].“<br>”;
          }
          else {
               echo “Archivo: ”.$_FILES[“file”][“name”].“<br>”;
               echo “Tipo: ”.$_FILES[“file”][“type”].“<br>”;
               echo “Tamaño: ”.$_FILES[“file”][“size”] / 1024.“ Kb <br>”;
               echo “Almacenado en: ”.$_FILES[“file”][“tmp_name”];

               if(file_exist(“subir/”.$_FILES[“file”][“name”])) {
                    echo $_FILES[“file”][“name”].“El archivo ya existe.”;
               }
?>
```

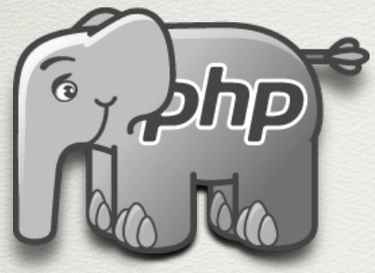


5. Subir Archivos

- **Guardar el archivo Subido:**

Para guardar el archivo subido es necesario copiarlo en una ubicación diferente.

```
<?php
     if(($_FILES[“file”][“type”] == “image/gif”)
     ||  ($_FILES[“file”][“type”] == “image/jpg”))
     &&  ($_FILES[“file”][“size”] < 20000)) {
          if($_FILES[“file”][“error”] > 0) {
               echo “Error: ”.$_FILES[“file”][“error”].“<br>”;
          }
          else {
               if(file_exist(“subir/”.$_FILES[“file”][“name”])) {
                    echo $_FILES[“file”][“name”].“El archivo ya existe.”;
               }
               else {
                    move_uploaded_file($_FILES[“file”][“tmp_name”],
                    “subir/”.$_FILES[“file”][“name”]);
                    echo “Archivo almacenado en: subir/”.$_FILES[“file”][“name”];
               }
          }
     }
     else {
          echo “Archivo Invalido:”;
     }
?>
```



1. Cookies

- **¿Qué es una cookie?**

Una "**cookie**" se usa frecuentemente para identificar a un usuario.

Una **cookie** es un pequeño archivo que el servidor incrusta en el ordenador del usuario.

Cada vez que el mismo equipo solicita una página con un navegador, enviará la **cookie** también. Con PHP, usted puede crear y recuperar los valores de **cookie**.

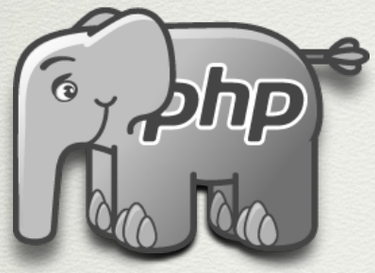
- **Cómo crear una cookie?**

La función **setcookie()** se utiliza para establecer una cookie.

Nota: La función **setcookie()** debe aparecer antes de la etiqueta **<html>**.

Sintaxis:

```
<?php
     setcookie(nombre, valor, expirará, ruta, dominio);
?>
```



2. Cookies

- **Ejemplo 1:**

Crearemos una cookie denominada "usuario" y le asignaremos el valor “Jeremias Sprinfield" a ella. También especificaremos que la cookie debe expirar después de una hora:

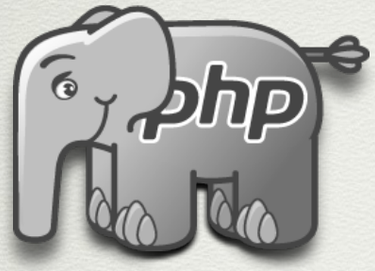
```
<?php
     setcookie(“usuario”,”Jeremias Sprinfield”, time()+3600);
?>
<html>
```

- **Ejemplo 2:**

También se puede establecer el tiempo de expiración de la cookie de otra manera. Puede ser más fácil de usar que en segundos.

La fecha de caducidad se establece en un mes (60 seg * 60 min * 24 horas * 30 días).

```
<?php
     $vence = time()+60*60*24*30;
     setcookie(“usuario”,”Jeremias Sprinfield”, $vence);
?>
<html>
```



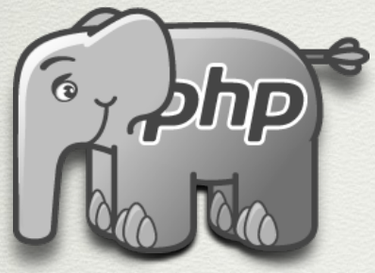
3. Cookies

- **Cómo Recuperar un Valor de una Cookie?**

La variable de PHP **$_COOKIE** se utiliza para recuperar un valor de la cookie.

```php
<?php
    // Imprimir una cookie
    echo $_COOKIE[“usuario”];

    // Imprimir todas las cookies
    print_r[$_COOKIE];
?>
```

```php
<?php
    if(isset($_COOKIE[“usuario”])) {
        echo “Bienvenido: ”.$_COOKIE[“usuario”].“<br>”;
    }
    else {
        echo “Bienvenido Visitante! <br>”;
    }
?>
```



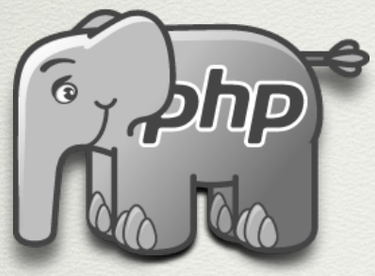
4. Cookies

- **Cómo eliminar una cookie?**

Al eliminar una cookie usted debe asegurarse de que la fecha de vencimiento está en el pasado.

Ejemplo:

```
<?php
     // La cookie venció hace una hora
     setcookie(“usuario”,””, time()-3600);
?>
```



1. Sesiones

- **Las variables de sesión:**

Una variable de sesión de PHP se utiliza para almacenar información acerca de un usuario. Las variables de sesión poseen información sobre un único usuario, y están disponibles para todas las páginas en una sola aplicación.

Las variables de tipo sesión permiten que usted almacene la información del usuario en el servidor para su uso posterior. Sin embargo, la información de sesión es temporal y se eliminará después de que el usuario ha abandonado el sitio web.

Si usted necesita un almacenamiento permanente es posible que desee almacenar los datos en una base de datos.

- **Inicio de una sesiónen PHP:**

Antes de que usted puede almacenar información del usuario en su sesión de PHP, primero debe iniciar la sesión.

Nota: El **session_start()** debe aparecer antes de la etiqueta <html>:

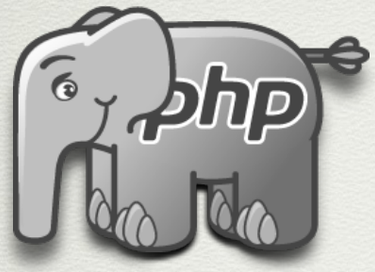
```
<?php session_start(); ?>
<html>
```

2. Sesiones

- **Almacenamiento de una variable de sesión:**

La forma correcta de almacenar y recuperar variables de sesión es utilizar la variable de PHP **$_SESSION**:

```php
<?php
     session_start();

     if(isset($_SESSION[“visitas”])) {
          $_SESSION[“visitas”]++;
     }
     else {
          $_SESSION[“visitas”] = 1;
     }
     echo “Número de Visitas: ”.$_SESSION[“visitas”];
?>
```



3. Sesiones

- **Destrucción de una sesión:**

Si desea borrar algunos datos de la sesión, puede utilizar el **unset()** o la función **session_destroy()**.

La función **unset()** se utiliza para liberar la variable de sesión especificada:

```
<?php
    unset($_SESSION[“visitas”]);
?>
```

También puede destruir completamente la sesión llamando a la función **session_destroy()**:

```
<?php
    session_destroy();
?>
```

Nota: session_destroy() eliminará los datos de la sesión y se perderán todos los datos almacenados.

1. Correo Electrónico

- **La función mail():**

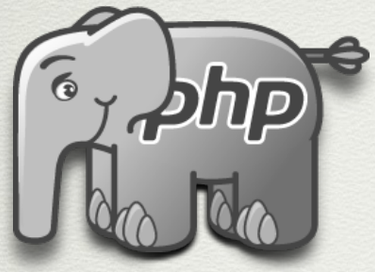
El PHP **mail ()** se utiliza para enviar mensajes de correo electrónico desde el interior de un script.

Sintaxis:

```
mail(para, asunto, mensaje, encabezados, parámetros);
```

La forma más sencilla de enviar un correo electrónico con PHP es enviar un correo electrónico de texto.

```
<?php
    $para        = “alguien@ejemplo.com”;
    $asunto      = “Prueba de Email”;
    $mensaje     = “Hola, esto es un simple mensaje de email”;
    $de          = “alguienmas@ejemplo.com”;
    $encabezados = “De: $de”;

    mail($para, $asunto, $mensaje, $encabezados);
    echo “El Correo fue enviado.”;
?>
```



2. Correo Electrónico

- **Formulario para el envío de correo electrónico:**

```
<form method=“post” action=“mail.php”>
     <strong>Correo:</strong><input type=“text” name=“email”><br>
     <strong>Asunto:</strong><input type=“text” name=“asunto”><br>
     <strong>Mensaje:</strong><textarea name=“mensaje”></textarea><br>
     <input type=“submit” value=“Enviar”>
</form>
```

```
<?php
     if(isset($_REQUEST[“email”])) {
          $email   = $_REQUEST[“email”];
          $asunto  = $_REQUEST[“asunto”];
          $mensaje = $_REQUEST[“mensaje”];

          mail(“alguien@ejemplo.com”, “Asunto: $asunto”, $mensaje, “De: $email”);
          echo “Gracias, El Correo fue enviado.”;
     }
?>
```

1. Excepciones

- **Manejo de excepciones:**

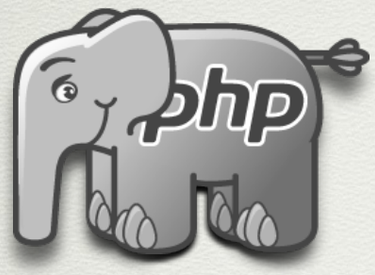
Las excepciones se utilizan para cambiar el flujo normal de un script si se produce un error especifico.

Nota: Las excepciones sólo se deben utilizar para condiciones de error, y no debe ser utilizado para ir a otro lugar en el código (punto especifico).

Uso Básico de Excepciones:

Si una excepción no se detecta, un error grave se publicará con un mensaje "excepción no detectada".

```php
<?php
     function verificarNum($numero) {
          if($numero > 1) {
               throw new Exception(“El número debe ser 1 o menor”);
          }
          return true;
     }
     verificarNum(2);
?>
```



2. Excepciones

- **Try, throw & catch (Intentar, lanzar y capturar)**

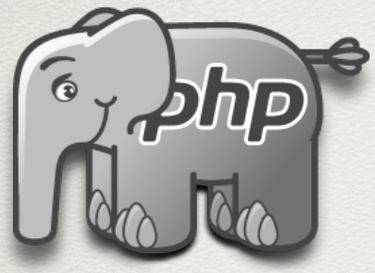
Para evitar el error en el ejemplo anterior, tenemos que crear el código apropiado para controlar una excepción.

El código correcto para una excepción debe incluir:

Try (Intentar): Una función con una excepción deben estar en un bloque "intentar". Si la excepción no se activa, el código seguirá con normalidad. Sin embargo, si la excepción es desencadenada, una excepción es "lanzada".

Throw (Lanzar): Esta es la forma de provocar una excepción. Cada "lanzar" debe tener al menos una "captura".

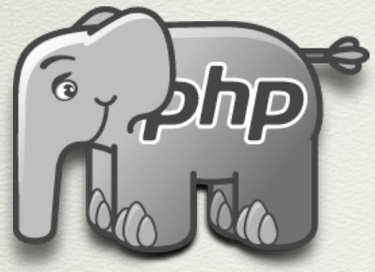
Catch (Captura): Un bloque "captura" recupera una excepción y se crea un objeto que contiene la información de la excepción.



3. Excepciones

- **Una excepción con un código válido:**

```php
<?php
     function verificarNum($numero) {
          if($numero > 1) {
               throw new Exception(“El número debe ser 1 o menor”);
          }
          return true;
     }
     try {
          verificarNum(2);
          echo “Si usted ve esto, el numero es 1 o menor”;
     }
     catch(Exception $e) {
          echo “Mensaje: ”.$e->getMessage();
     }

?>
```



1. Filtros

- **¿Qué es un filtro de PHP?**

Los filtros se utilizan para validar y filtrar datos provenientes de fuentes inseguras, como un campo de formulario.

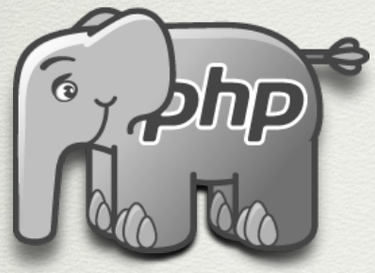
Los filtros están diseñados para hacer el filtrado de datos más fácil y más rápido.

- **¿Por qué utilizar un filtro?**

Usted siempre debe filtrar todos los datos externos!

Filtrar el ingreso de datos es uno de los problemas de seguridad más importantes de una aplicación web.

- **¿Qué son los datos externos?**
  - Entrada de datos de un formulario.
  - Cookies.
  - Datos de servicios Web.
  - Las variables del servidor.
  - Resultados de consulta a Bases de Datos.



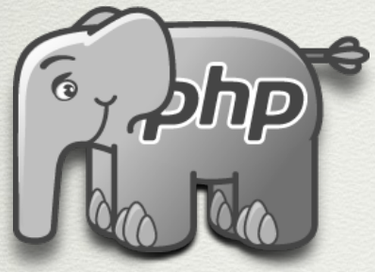
2. Filtros

- **Funciones y Filtros**

Para filtrar una variable, utilice una de las funciones de filtro:

- **filter_var():** Filtro de una sola variable.
- **filter_var_array():** Filtro para varias variables.
- **filter_input:** Obtener una variable de entrada y filtrarla.
- **filter_input_array:** Obtener varias variables de entrada y filtrarlas con los mismos filtros o diferentes.

```php
<?php
     $entero = 123;
     if(!filter_var($entero, FILTER_VALIDATE_INT)) {
          echo “El número no es un Entero”;
     }
     else {
          echo “El número es un Entero”;
     }
?>
```



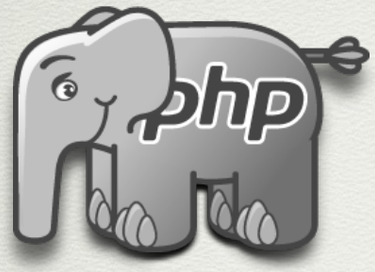
3. Filtros

- **Validación y desinfección:**

Hay dos tipos de filtros:

- **Filtros de Validación:**
  - Se utilizan para validar las entradas de los usuarios.
  - Reglas estrictas de formato (como dirección URL o validación de E-Mail)
  - Devuelve el tipo de datos si es exitoso o en caso contrario retorna falso.

- **Filtros de Desinfección:**
  - Se utilizan para permitir o no permitir caracteres especificados en una cadena.
  - No hay reglas de formato de datos
  - Siempre devuelve una cadena



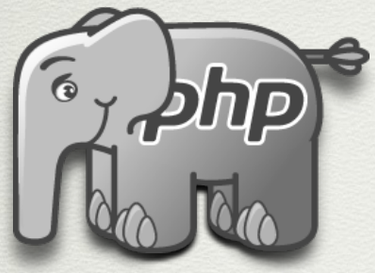
4. Filtros

- **Opciones y Banderas:**

Opciones y banderas se utilizan para añadir más opciones de filtrado a los filtros especificados.

En el ejemplo siguiente, se valida un número entero utilizando el **filter_var()** y las opciones "**min_range**" y "**max_range**":

```
<?php
	$variable = 400;
	$opciones = array(
	“options”=>array
		(
			“min_range”=>0,
			“max_range”=>256
		)
	);

	if(!filter_var($variable, FILTER_VALIDATE_INT, $opciones)) {
		echo “El número no es un Entero”;
	}
	else {
		echo “El número es un Entero”;
	}
?>
```



5. Filtros

- **Validar una entrada de texto:**

Vamos a tratar de validar la entrada de un formulario.

```
<?php
     if(!filter_has_var(INPUT_GET, “email”)) {
          echo “El Campo Correo Electrónico no Existe”;
     }
     else {

          if(!filter_input(INPUT_GET, “email”, FILTER_VALIDATE_EMAIL)) {
               echo “El Correo Electrónico no es valido”;
          }
          else {
               echo “El Correo Electrónico es valido”;
          }
     }
?>
```

6. Filtros

- **Desinfección de una campo de texto:**

Vamos a tratar de limpiar una dirección URL enviada desde un formulario.

```php
<?php
     if(!filter_has_var(INPUT_POST, “url”)) {
          echo “El Campo URL no Existe”;
     }
     else {
          $url = filter_input(INPUT_POST, “url”, FILTER_SANITIZE_URL);
     }
?>
```