

# Curso - PHP Básico

Instructor: Enrique Areyán

Curso preparado para el Diario El Impulso.

Barquisimeto, Marzo 2011

Este curso forma parte de una serie de cursos cuyo objetivo principal es el de enseñar el lenguaje de programación PHP a personas con conocimientos básicos de programación, y prepararlos para llevar a cabo proyectos de gran envergadura. Para ello se enseñará PHP básico (este curso), PHP orientado a Objetos, las mejores y más novedosas prácticas en PHP, Patrones de Diseño y el Zend Framework.

## Acerca del Instructor

- Enrique Areyán
- Lic. Computación UCV
- Experiencia de aprox. 10 años (desde 2001) trabajando en aplicaciones web
- Certificado “Building PHP Applications using Zend Framework”
- Co-fundador [www.estoesweb.com](http://www.estoesweb.com)

Pueden contactarme a través del correo electrónico [enrique3@gmail.com](mailto:enrique3@gmail.com)

## Antes de Comenzar

- Bajar (o usar pendrive) WAMP
- Instalarlo
- Mover las siguientes carpetas del pendrive a c:  
  \wamp\www
  - Ejemplos
  - Ejercicios
  - Ejercicio Integral

## Contenido

- *¿Qué es PHP?*
- *¿Cómo funciona PHP?*
- *Instalación y Configuración*
- *Sintaxis*
- *Constantes*
- *Variables*
- *Operadores*
- *Strings*
- *Funciones*
- *Estructuras de Control*
- *Recibir parámetros por parte del cliente*
- *Tipos de datos estructurados: arreglos*
- *PHP y MySQL básico*
- *Ejercicio Integral*

Este curso básico de PHP pretender enseñar las herramientas básicas para comenzar a desarrollar en este lenguaje. Aquí se cubrirán los aspectos básicos más importantes tanto desde el punto de vista teórico como práctico.

Se asume que el participante tiene conocimientos básicos de programación, no necesariamente para la Web.

## Información General

PHP y documentación

Una revisión general.

## Información General

- Para documentación sobre PHP(*muchas y de calidad*) siempre tener presente:
  - <http://www.php.net/>
- A mi me resulta más fácil buscar en php.net a través de google, por ejemplo, escribiendo:
  - date site:php.net
- Esto busca todo lo relacionado con “date” en el sitio php.net. Date es una función en PHP.

La documentación en línea sobre PHP es vasta y de calidad. Si algo no se consigue en el sitio de php, www.php.net, utilizando google se tienen altas probabilidades de solventar esta deficiencia.

## Información General (cont.)

- PHP tiene más de 700 funciones nativas
- PHP tiene soporte incluído para:
  - Base de Datos
  - Conexiones FTP
  - Envio de correo electrónico
  - Manipulación de archivos
  - Encriptación (seguridad)
  - Manipulación XML
  - Manejo de Sesiones y Cookies

Una función nativa es aquélla que ya viene incluída como parte de las librerías internas del lenguaje y por lo tanto no debe ser implementada por el programador. Esto nos ahorra tiempo.

## Información General (cont.)

- Manejo de archivos subidos por HTTP
  - Colector de Basura
  - Codificación de Caracteres
  - etc
- 
- Sin embargo, lo que no se consiga como una función predeterminada del lenguaje siempre se puede implementar

PHP se caracteriza por ser un lenguaje flexible. Esta característica tiene sus ventajas y desventajas, como veremos más adelante.

## Glosario

- Usuario
  - Tu, el programador
- Cliente
  - La persona que visita la página web
- Interpretador o intérprete
  - El programa de PHP que se encarga de interpretar el código escrito en este lenguaje

## ¿Qué es PHP y cómo funciona?

Una introducción teórica y práctica al lenguaje.

En este apartado se tocará el tema de PHP desde un punto de vista conceptual. También se hablará en concreto cómo es que funciona y sus características principales.

## ¿Qué es PHP?

- PHP (Hypertext Preprocessor)
- Es un lenguaje de script muy utilizado, de código abierto (open source), y de propósito general pero que es particularmente útil como lenguaje de desarrollo para la web
- Por lenguaje de script se entiende que el código será ejecutado por medio de un intérprete, en contraste con los lenguajes compilados

Fuente: <http://www.php.net/manual/en/intro-whatis.php>

Esto implica que un archivo PHP no es más que un archivo de texto plano alojado en el servidor. El intérprete PHP recibe como entrada ese archivo y lo “interpreta” según sus reglas particulares.

En contraste, un lenguaje compilado se implementa mediante un compilador. Esto implica que una vez escrito el programa, éste se traduce a partir de su código fuente por medio de un compilador en un archivo ejecutable para una determinada plataforma.

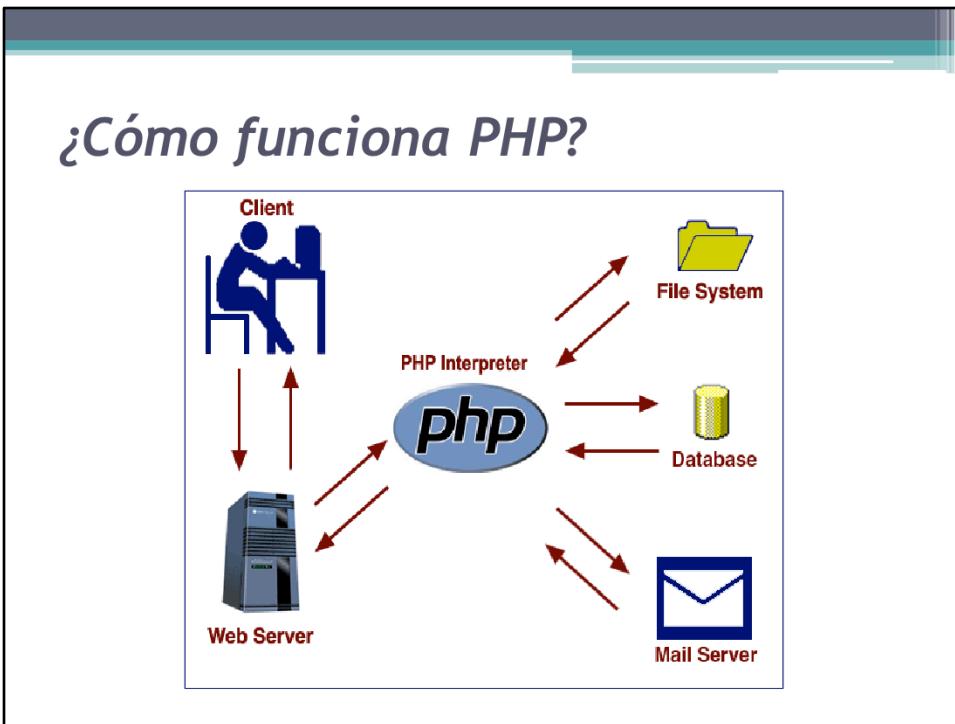
Los lenguajes compilados son lenguajes de alto nivel en los que las instrucciones se traducen del lenguaje utilizado a código máquina para su ejecución. Por el contrario un lenguaje interpretado es aquel en el que las instrucciones se traducen o interpretan una a una.

## *¿Qué es PHP? (cont.)*

- En un principio PHP se utiliza como un lenguaje “encrustado” dentro del HTML
- El objetivo es crear páginas Web dinámicas, es decir, páginas que se generan al momento de la petición por parte del usuario
- En este caso, el encargado de generar el contenido de estas páginas es el interpretador de PHP que se encuentra en el servidor Web

Según Wikipedia: “Las páginas dinámicas que se generan al momento de la visualización se hacen a través de lenguajes interpretados, generalmente JavaScript, y la aplicación encargada de visualizar el contenido es la que debe generarlo. La páginas dinámicas que se generan al ser solicitadas son creadas por una aplicación en el servidor Web que alberga las mismas.”

Fuente: [http://es.wikipedia.org/wiki/P%C3%A1gina\\_web](http://es.wikipedia.org/wiki/P%C3%A1gina_web)



Cuando un usuario navega la Internet a una página que termine en la extensión .php, la petición (request) se envía a un servidor Web el cual redirecciona dicha petición al interpretador de PHP.

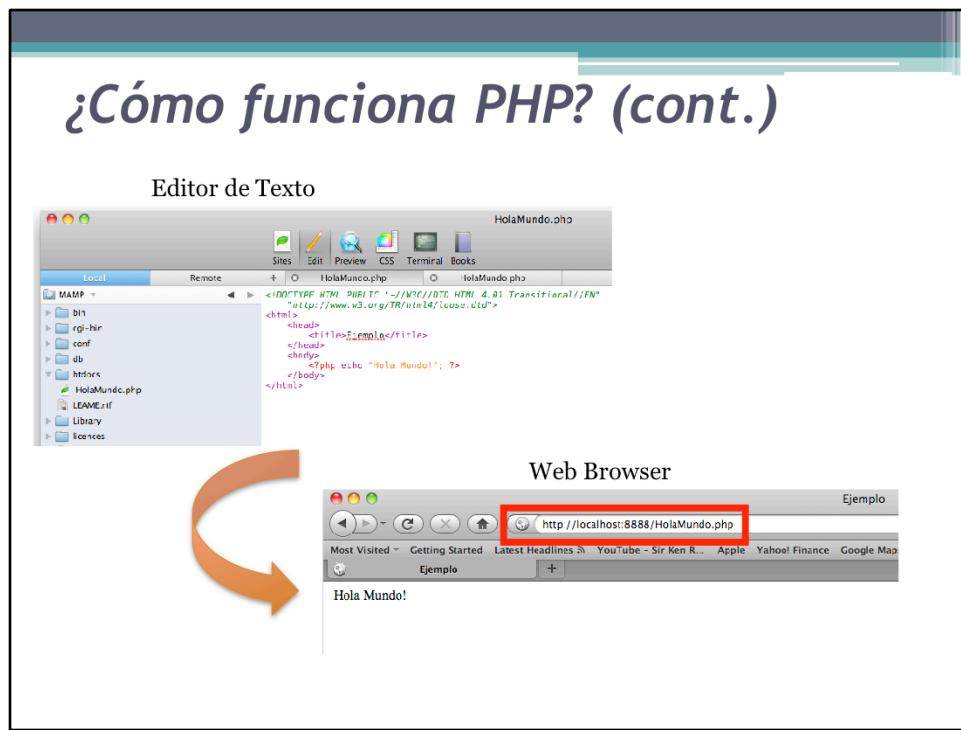
Como se muestra en el diagrama, el interpretador de PHP procesa la página que llegó como petición. Para este fin, el intérprete se comunicará con el sistema de archivos, base de datos, servidor de correo o cualquier otro componente del sistema del cual se requiera algún servicio para completar la petición.

Fuente: <http://www.learnphp-tutorial.com/PHPBasics.cfm>

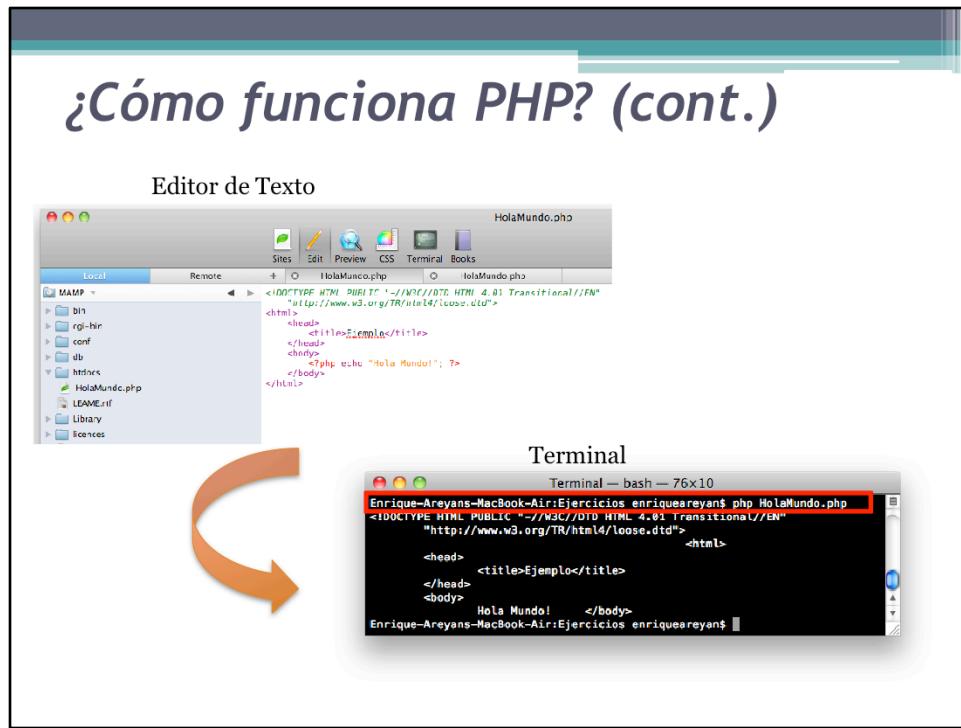
## ¿Cómo funciona PHP? (cont.)

```
<!DOCTYPE HTML PUBLIC "-//W3C//  
DTD HTML 4.01 Transitional//EN"  
    "http://www.w3.org/TR/html4/loose.dtd">  
<html>  
    <head>  
        <title>Ejemplo</title>  
    </head>  
    <body>  
        <?php echo "Hola Mundo!";?>  
    </body>  
</html>
```

La función echo en PHP sirve para imprimir expresiones.



Ejercicio: Subir el archivo de este ejemplo al servidor, correrlo desde el browser y ver el código fuente generado (clic derecho, ver código fuente)



PHP también puede correr desde la línea de comandos o terminal. Esto es útil en ciertas ocasiones como por ejemplo para hacer “debugging”.

## Instalación y Configuración

Apache + PHP + MySQL

Una breve introducción a la instalación y configuración (tunning) del lenguaje.

## Instalación

- PHP tiene tres usos básicos
  - Codificación para linea de comandos
  - Aplicaciones Desktop (GUI)
  - **Websites y aplicaciones web (*server-side scripting o codificación del lado del servidor*)**
    - PHP
    - Servidor Web (Apache)
    - Web Browser

PHP tiene tres usos básicos: (1) para generar scripts a nivel de línea de comandos, (2) para crear aplicaciones desktop (posiblemente PHP no sea la mejor herramienta para esto) y )(3) para páginas y aplicaciones web, que es el uso más común. Aquí nos enfocamos en PHP para la web.

Por tanto, para instalar PHP para la web se necesitan tres cosas: (1) PHP, (2) un servidor web, el cual es típicamente Apache para sistema tipo \*nix o IIS para Windows, y (3) un web browser (IE o Firefox usualmente)

## *Instalación (cont.)*

- PHP viene preinstalado en la mayoría de los servidores hoy en día
- En este curso no vamos a hacer mucho énfasis en la instalación de PHP.
- Sin embargo, se debe mencionar que en la mayoría de los casos PHP se instala como un plugin o módulo al servidor web, quién después se encarga de llamar a PHP para ejecutar una tarea

Para más información sobre la instalación de PHP, visite: <http://www.php.net/manual/en/install.general.php>

## *Configuración*

- Una vez instalado PHP, este se debe ajustar a las necesidades de la empresa.
- PHP se configura a través de varios mecanismos que dependen de la modalidad en la que se esté ejecutando:
  - Al momento de ejecución -> Funciones Nativas
  - Una vez a la inicialización -> Archivo **php.ini**
  - Si PHP corre como módulo de Apache -> archivos httpd.conf y htaccess

La configuración del lenguaje se puede parametrizar durante la ejecución del mismo, por cada script, o globalmente al momento de iniciar PHP.

## ***Configuración (cont.)***

- El archivo de configuración php.ini
  - Se lee cuando arranca php
  - Típicamente se ubica en la carpeta de configuración del sistema (varía dependiendo de la plataforma)
  - Aquí se puede ajustar una gran cantidad de parámetros bajo los cuales se regirá el funcionamiento de PHP

Cualquier cambio que se quiera que tenga efecto sobre todo el ambiente de desarrollo se debe hacer en el archivo de configuración php.ini

Este archivo es leído una vez al momento de arrancar el sistema y aplica para todos los scripts que vayan a ejecutarse.

El archivo php.ini es muy amplio y otorga una gran flexibilidad al usuario.

## Configuración (cont.)

```
;;;;;;;;;;;;;;;;;;
; Resource limits ;
;;;;;;;;;;;;;;;;;;

max_execution_time = 30 ; Maximum execution time of each script, in seconds
max_input_time = 60 ; Maximum amount of time each script may spend parsing request data
memory_limit = 32M ; Maximum amount of memory a script may consume (8MB)

;;;;;;;;;;;;;;;;;;
; Error handling and logging ;
;;;;;;;;;;;;;;;;;;

; error_reporting is a bit-field. Or each number up to get desired error
; reporting level
; E_ALL           - All errors and warnings
; E_ERROR          - fatal run-time errors
; E_WARNING        - run-time warnings (non-fatal errors)
; E_PARSE          - compile-time parse errors
; E_NOTICE         - run-time notices (these are warnings which often result
;                     from a bug in your code, but it's possible that it was
;                     intentional (e.g., using an uninitialized variable and
;                     relying on the fact it's automatically initialized to an
;                     empty string)
; E_CORE_ERROR     - fatal errors that occur during PHP's initial startup
; E_CORE_WARNING   - warnings (non-fatal errors) that occur during PHP's
;                     initial startup
; E_COMPILE_ERROR  - fatal compile-time errors
; E_COMPILE_WARNING - compile-time warnings (non-fatal errors)
; E_USER_ERROR     - user-generated error message
; E_USER_WARNING   - user-generated warning message
; E_USER_NOTICE    - user-generated notice message
;

; Examples:
; - Show all errors, except for notices
;error_reporting = E_ALL & ~E_NOTICE
;
; - Show only errors
;
```

Un ejemplo de una pequeña porción del archivo php.ini. Se muestran varios de los parámetros más importantes del lenguaje.

## *Configuración (cont.)*

- Un error en PHP es aquélla condición bajo la cuál el script no puede seguir ejecutándose
- Una notificación en PHP es un “error” más suave, en donde el interpretador genera un mensaje pero la ejecución continúa
  - En el ambiente de desarrollo se desea ver todo (errores o notificaciones)
  - En el ambiente de producción no se desea que el cliente vea ningún mensaje de error. Mejor es guardarlos en un archivo para su posterior “autopsia”

Desde el punto de vista del cliente, cualquier mensaje foráneo a la aplicación se considera como un error y levanta sospechas sobre el funcionamiento de la misma. Muchos mensajes de error pueden llevar a que el usuario desconfíe plenamente de la aplicación y deje de utilizarla.

## *Configuración (cont.)*

- Para manipular la forma de mandar mensajes se puede
  - Modificar el archivo php.ini
  - O insertar el siguiente código dentro del script en cuestión:

```
<?php  
//Mostrar todos los errores  
error_reporting(E_ALL);  
//No mostrar ningún error  
error_reporting(0);  
?>
```

Se muestra como cambiar el comportamiento de PHP al momento de ejecución.

## Sintaxis

```
<?php  
    if($valor){  
        echo "Hola....";  
    }
```

Para manejar cualquier lenguaje hay que empezar por el a,b,c; en este caso, la sintaxis. En general, la sintaxis de un lenguaje (hablado y escrito, o de computador) establece las reglas de cómo se deben formar expresiones válidas en el mismo.

Sintaxis - Etiquetas PHP	
<pre>&lt;?php     AQUI VA EL CÓDIGO PHP ?&gt;</pre>	Esta es la forma más común y recomendada para utilizar PHP. Se conoce como la forma XML porque puede ser incluida en un archivo XML sin corromperlo.
<pre>&lt;script language="php"&gt;     AQUI VA EL CÓDIGO PHP &lt;/script&gt;</pre>	Etiqueta tipo HTML
<pre>&lt;?     PHP CODE GOES HERE ?&gt;</pre>	Etiqueta corta
<pre>&lt;%     PHP CODE GOES HERE %&gt;</pre>	Etiqueta tipo ASP

A lo largo de este curso emplearemos la primera forma de las etiquetas para insertar código PHP.

## Sintaxis - Sentencias PHP

- Las sentencias PHP deben estar dentro de las etiquetas para poder ser procesadas
- Cada sentencia PHP debe terminar en punto y coma “;”, lo cual le indica al interpretador que la sentencia terminó
- Si no encuentra un “;” el interpretador supone que la sentencia continua en la próxima línea
- En PHP todos los espacios en blanco continuos son condensados en un sólo espacio en blanco. Esto permite al programador estructurar el código sin preocuparse de los efectos de saltos de línea o tabs.

PHP es un lenguaje muy sencillo que presenta pocas restricciones al momento de usarlo.

## Sintaxis - Comentarios

- Se puede comentar una sola línea utilizando doble slash (//).
- O se pueden comentar múltiples líneas comenzando con /\*" y terminando con "\*/.

```
<?php
    //Este es un comentario en una linea
    echo "Hola Mundo!";
    /*
        Este es un
        comentario en
        varias lineas.
    */
?>
```

Una de las reglas fundamentales para llevar a cabo proyectos de gran envergadura, es la intradocumentación del código. En esta lámina se muestra como comentar el código del lenguaje PHP.

## Sintaxis - Regla de nombrado

- Las siguientes reglas aplican para todas aquellas sentencias a las cuales el usuario (programador) necesite darles un nombre. Por ejemplo: constantes, variables, funciones,...
  - El nombre debe comenzar con una letra o “underscore” (\_)
  - Seguido de cualquier cantidad de letras, números o “underscores”
  - Visto como expresión regular:  
 $[a-zA-Z_][x7f-|xff][a-zA-Z0-9_][x7f-|xff]^*$

PHP reserva algunas palabras claves para su uso interno. Éstas comienzan con doble underscore. De resto hay plena libertad en el nombrado de objetos.

## Sintaxis - Regla de nombrado

- Ejemplos:

```
<?php  
//Nombre de variable válido  
$ejemplo = 5;  
//Nombre de variable inválido  
$5ejemplo = 5;  
//Nombre de variable válido, pero debe evitarse  
$__ejemplo__ = 5;  
?>
```

El último nombre de variable es válido y va a ejecutarse en su programa sin inconveniente técnico. No obstante, los nombres que comienzan con “underscores” son nombres utilizados por el equipo de desarrollo de PHP, y si por casualidad coincide el nombre que usted le está dando, por ejemplo, a una variable, con el nombre que ellos en el futuro utilicen para una variable u otro objeto, entonces el de ellos tendrá la prioridad.

Como consecuencia, su script dejará de funcionar o presentará errores que no presentaba con versiones anteriores las cuales no incorporaban de forma nativa el nombre de dicha variable. En pocas palabras, habrá una colisión entre el nombre del objeto que ellos definen y el suyo.

## Constantes y Variables

```
define("constante","valor");
$variable = "valor";
```

Los bloques fundamentales de cualquier lenguaje de programación son las variables. Sin ellas no se alcanzaría dinamismo y la utilidad de los programas sería escasa. Por su parte, las constates ayudan a establecer reglas uniformes al momento de llevar a cabo proyectos. En este apartado vemos en detalles como funcionan constantes y variables en PHP.

## Constantes

- Una vez definida una constante, esta no puede cambiar de valor
- Se crean usando la función **define()**
- Por convención se nombran todas en **MAYÚSCULAS**
- Las constantes pueden ser accedidas desde cualquier parte de la página

Reglas para la creación y lectura de constantes. Importante: el valor de las constantes no puede cambiar en el momento de ejecución de un programa.

## Constantes - Ejemplos

```
<?php
//Nombres de constantes válidos
define("EJEMPLO", 5);
define("EJEMPLO_5", 5);
define("EJEMPLO_CINCO", 5);
//Nombre de constante inválido
define("5EJEMPLO", 5);
//Nombre de constante válido, pero debe
evitarse
define("__EJEMPLO__", 5);
?>
```

El nombre de constante inválido no mostrará error hasta que se use la constante en otra sentencia. Por ejemplo, si deseas imprimir el valor de la constante: echo 5EJEMPLO;, el interpretador de PHP arrojará un error. Esto solo ocurre con constantes, ya que con otro tipo de objetos el error se arroja inmediatamente.

## Variables

- Las variables en PHP comienzan con el signo de dólar (\$)

```
<?php  
//Este es un comentario en una línea  
$nombre_var = "Hola Mundo!";  
?>
```

Las variables forman la estructura básica de cualquier lenguaje de programación. En PHP una variable se declara prefijando el símbolo de dólar (\$) y siguiendo las reglas de nombrado anteriormente descritas.

## Variables - Tipos

Tipo de Variable	Descripción
Integer (Entero)	Números enteros
Double (doble)	Números reales
String (cadena)	Cadena de caracteres (texto)
Boolean (variable lógicas)	True or false (verdadero o falso)
Array (arreglo)	Lista de datos
Objetc (objeto)	Instancia de una clase

Más sobre objetos en el próximo curso.

## Variables - Tipos

- PHP es un lenguaje **débilmente tipado**
  - A las variables no es necesarios asignarles un tipo de datos (por ejemplo, *Integer* o Entero), al momento en que son declaradas.
  - El tipo de variable es determinado por el interpretador PHP al momento de ejecución en función del valor que se le asigne y como se utilice



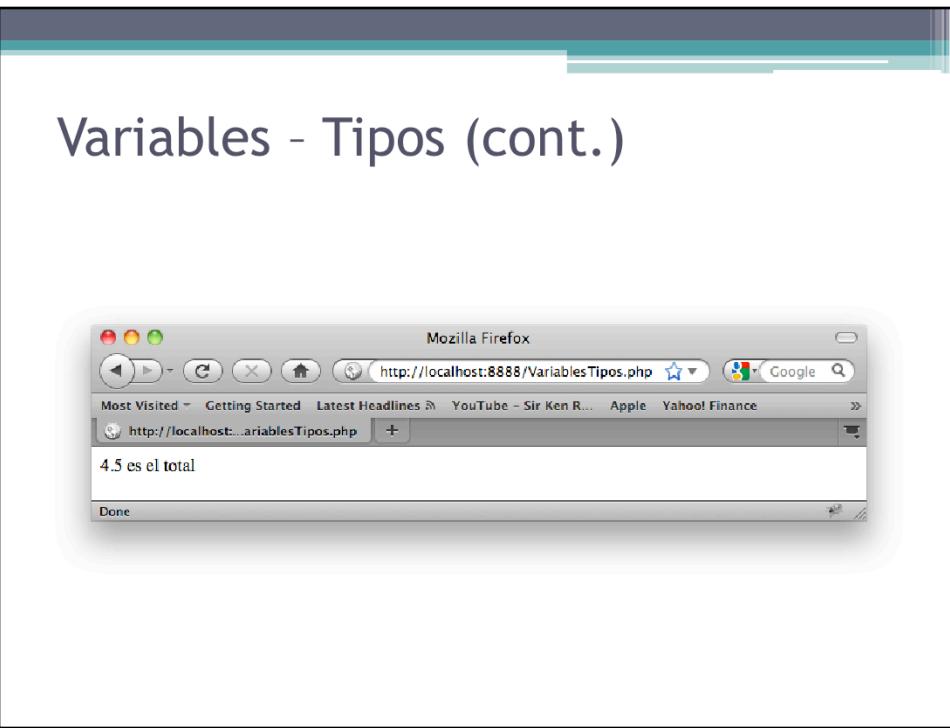
En próximas lecciones se verá claramente porque el hecho de que PHP sea débilmente tipado es una de las fuentes de errores más comunes al momento de llevar a cabo un proyecto de grandes dimensiones.

Aunque es cierto que esta característica hace de PHP un lenguaje sencillo de aprender y muy flexible, también es cierto que introduce desorden desde el inicio del proyecto. Dicho desorden se acumula hasta el punto en que resulta muy costoso mantener el código fuente.

## Variables - Tipos (cont.)

```
<?php  
//variable tipo string  
$var = "0";  
//var ahora cambia a tipo double  
$var += 2.5;  
//var es ahora entero  
$var += 2;  
//var es ahora un string  
$var .= " es el total";  
echo $var;  
?>
```

Este ejemplo demuestra como los tipos de variables en PHP cambian sin hacer referencia explícita a este cambio.



Resultado de ejecutar el código del ejemplo anterior. El resultado es un string.

## Variables - Una revisión a “Hola Mundo”

```
<?php $mensaje = "Hola Mundo!... Desde variable"; ?>
<!DOCTYPE HTML PUBLIC "-//W3C//"
DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Ejemplo</title>
  </head>
  <body>
    <?php echo $mensaje;?>
  </body>
</html>
```

Una nueva versión del clásico “Hola Mundo!” utilizando variables.

## Ejercicio 1 - Fecha y Hora

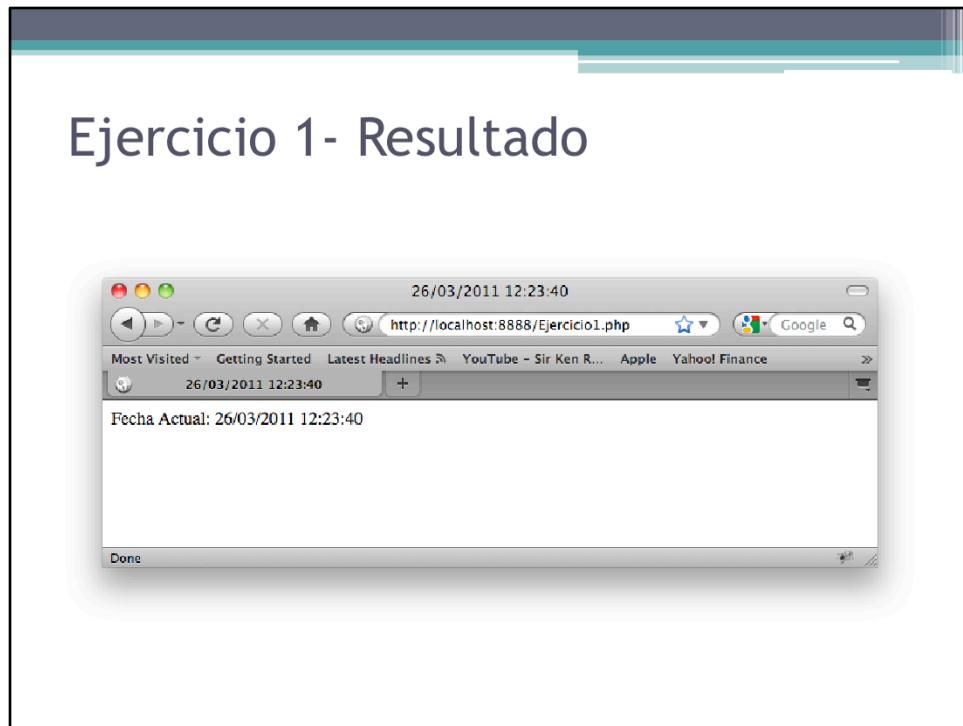
- Duración: de 5 a 10 minutos
- Objetivos:
  - Escribir un programa en PHP desde cero.
  - Utilizar la documentación en línea de PHP
- Especificaciones:
  - Declarar una variable **\$FechaActual** que contenga la fecha y hora actuales
  - Imprimir la fecha y hora actuales en el *titulo* (*title*) y *cuerpo* (*body*) del documento HTML

Este es un primer ejercicio para familiarizarse con la sintaxis de PHP, el uso de variables y la documentación en línea.

## Ejercicio 1 - Fecha y Hora (cont.)

- Instrucciones:
  - Crear un nuevo archivo **Ejercicio1.php**
  - Salvarlo en la carpeta  
*/htdocs/CursoPHPBasico/Ejercicios*
  - Buscar en la documentación de PHP (php.net) la función **date**
  - Asignar a la variable \$FechaActual el resultado de la función **date** con los parámetros adecuados
  - Imprimir \$FechaActual en el *title* y en el *body* del documento HTML

Seguir las instrucciones.



Resultado del ejercicio.

## Ejercicio 1- Solución

```
<?php $FechaActual = date("d/m/Y h:i:s");?>
<html>
  <head>
    <title><?=$FechaActual?></title>
  </head>
  <body>
    Fecha Actual: <?=$FechaActual?>
  </body>
</html>
```

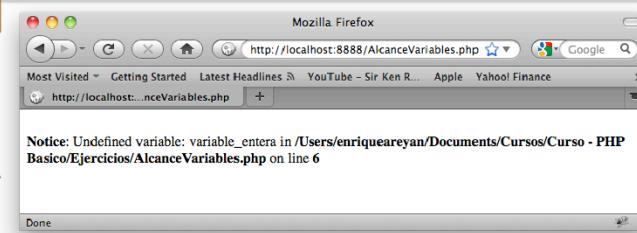
El parámetro "d/m/Y h:i:s" dentro de la función date, resulta en que la fecha se devuelva con el formato dia/mes/año hora:minutos:segundos.

Variables - Alcance ( <i>Scope</i> )	
Alcance	Descripción
Superglobal	Las variables superglobales son arreglos predefinidos, los cuales se pueden acceder desde cualquier parte del código. Ejemplos de estos son: <code>\$_POST</code> y <code>\$_GET</code>
Global	Las variables globales pueden ser accedidas a lo largo de cualquier parte del código donde fueron definidas. No se pueden acceder desde adentro de funciones sin antes redeclararlas dentro de la función como variables globales
Local	Las variables locales son aquéllas sólo accesibles dentro de una función, es decir, son locales a la función en cuestión donde fueron declaradas.

El alcance (scope) de una variable determina desde donde esta puede ser accedida.

## Variables - Alcance (Scope) (cont.)

```
<?php  
$variable_entera= 3;  
function imprimir_var(){  
    echo $variable_entera;  
}  
imprimir_var();  
?>
```

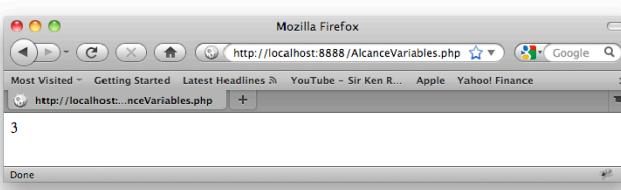


The screenshot shows a Mozilla Firefox window with the URL `http://localhost:8888/AlcanceVariables.php`. The page content displays a PHP notice: `Notice: Undefined variable: variable_entera in /Users/enriqueareyan/Documents/Cursos/Curso - PHP Basico/Ejercicios/AlcanceVariables.php on line 6`. An orange arrow points from the explanatory text above to this error message.

La notificación se genera porque la variable `$variable_entera` no existe dentro del ambiente de la función.

## Variables - Alcance (Scope) (cont.)

```
<?php
$variable_entera= 3;
function imprimir_var(){
    global $variable_entera;
    echo $variable_entera;
}
imprimir_var();
?>
```



Una vez declarada como global la variable \$variable\_entera, esta pasa al ambiente de la función y su valor se puede imprimir.

## Variables - Superglobales

- **`$_GET`**: variables recibidas por url (query\_string)
- **`$_POST`**: variables recibidas por un formulario
- **`$_SERVER`**: variables de ambiente, por ejemplo, `$_SERVER['HTTP_REFERER']`, devuelve el url de la página referenciadora
- **`$_COOKIE`**: información de cookie
- **`$_FILES`**: información sobre archivos subidos por formulario
- **`$_SESSION`**: variables de sesión.
- etc

La lista completa de variables superglobales se encuentra en [php.net](http://php.net)

## Variables - Testeo y Manipulación

Función	Descripción	Ejemplo
isset()	Verifica que una variable exista. Devuelve true or false	isset(\$a)
unset()	Remueve una variable de la memoria	unset(\$a)
empty()	Chequea si una variable contiene un valor no vacío y no falso.	empty(\$a)

Funciones útiles al momento de manipular variables y verificar su existencia.

## Operadores

`$a + $b = ?`

Una vez que se tienen las variables, estas deben poder ser operadas para llevar a cabo tareas fundamentales. Este apartado explora el uso de los operadores en PHP.

## Operadores - Matemáticos

Operador	Nombre	Ejemplo
+	Suma	$\$a + \$b$
-	Resta	$\$a - \$b$
*	Multiplicación	$\$a * \$b$
/	División	$\$a / \$b$
%	Módulo	$\$a \% \$b$

Los operadores matemáticos trabajan sobre datos de tipo entero o real y producen un nuevo dato de tipo entero o real según sea la definición de la operación y los valores de entrada. Los operadores matemáticos son binarios, es decir, actúan sobre dos variables para producir un nuevo valor.

## Operadores - Strings

Operador	Nombre	Ejemplo
.	Concatenación	<code>\$a . \$b</code>

Por ejemplo, si `$a = "Hola"` y `$b = "Mundo"`,  
entonces `$c = $a . $b = "Hola Mundo"`

La concatenación de string es una operación fundamental de PHP y será ampliamente utilizada a lo largo de este curso.

## Operadores - Asignación

Operador	Nombre	Ejemplo
=	Asignación	\$a = 1; \$a = "Cualquier texto";
+=	Combinación de asignaciones	\$a += 1; (\$a = \$a+1;)
-=		\$a -= 1;
*=		\$a *= 2;
/=		\$a /= 2;
%=		\$a %= 2;
.=		\$a .= "texto";
++	Incremento en uno	\$a++; (\$a = \$a+1;)
--	Decremento en uno	\$a--; (\$a = \$a-1;)

Al igual que los operadores matemáticos, los operadores de asignación son operadores binarios. En este caso se tiene, por una parte, una variable a la que se le va a asignar un valor y, por otra parte, una expresión. El resultado de la operación es asignar el resultado de la expresión a la variable en cuestión.

## Operadores - Otros

Operador	Nombre	Ejemplo
+	Ternario	Ver ejemplo (*)
-	Supresión de Error	\$a = @(1/0);

(\*) \$res= (\$edad>= 18) ? 'adulto' : 'niño';

Operadores especiales. No son muy comúnmente utilizados pero pueden resultar útiles en ciertas ocasiones.

## Operadores - Ejemplos

- Ver ejemplos en el archivo Operadores.php

## Strings

“este es un string”

PHP es un lenguaje flexible. Esta flexibilidad se pone de manifiesto en la facilidad con la que se pueden manipular strings o cadenas de caracteres. En este apartado exploramos los strings en el contexto de un script en PHP.

## Strings

- En el manejo de los *strings* reside parte del poder de PHP
- PHP cuenta con gran cantidad de funciones para el manejo de *strings* (ver <http://php.net/manual/en/ref.strings.php>)
- ¡MOSCA!: Un *string* puede ser cualquier cosa lo cual le otorga flexibilidad al sistema pero también puede ser fuente de confusión

Algunas notas sobre strings. Recordar que con mayor flexibilidad también aumenta la posible confusión que se pueda presentar en el código.

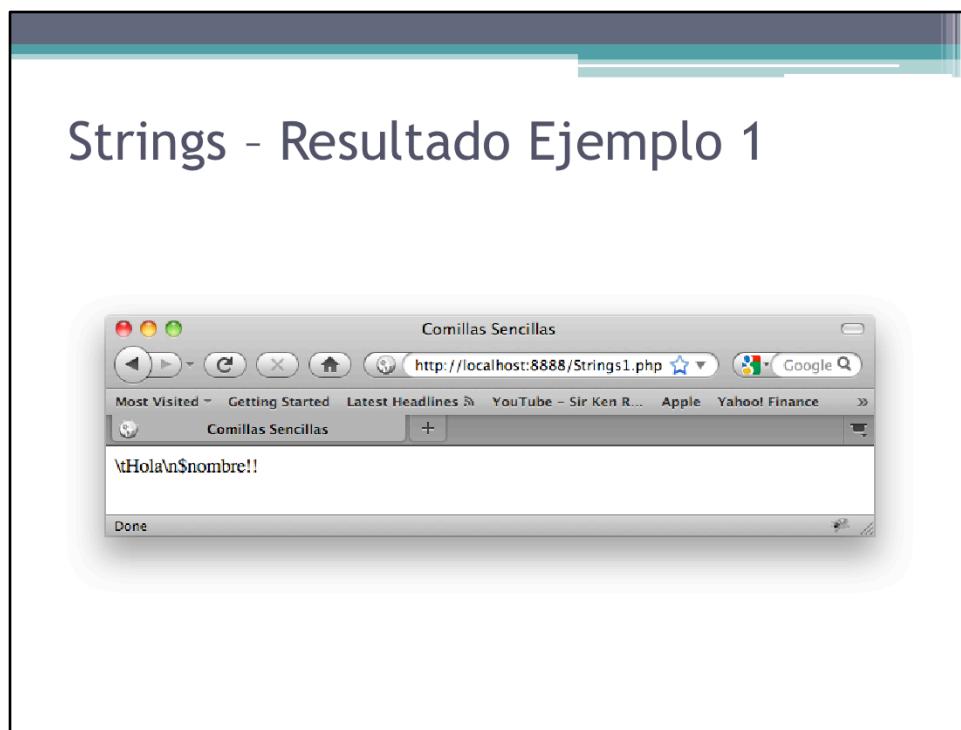
## Strings

- Comillas sencillas vs. Comillas dobles.
  - Para almacenar e imprimir strings sencillos se puede usar comillas dobles (") o sencillas (‘) ▫ Sin embargo, el texto dentro de comillas sencillas (‘) no será interpretado en búsqueda de variables y secuencias de escape (escape sequences)
- Secuencias de escape comunes:
  - \n ,\t
  - \\
  - \"
  - \\$.

La secuencia de escape es utilizada en caracteres para los cuales existe ambigüedad al momento de imprimirlos. La secuencia de escape precede dicho carácter por un backslash (\). Algunas secuencias de escape comúnmente utilizadas son: \n para un salto de línea, \t para tab, \\ para backslash, \" para comillas dobles, y \\$ para signo de dólar.

## Strings - Ejemplo 1

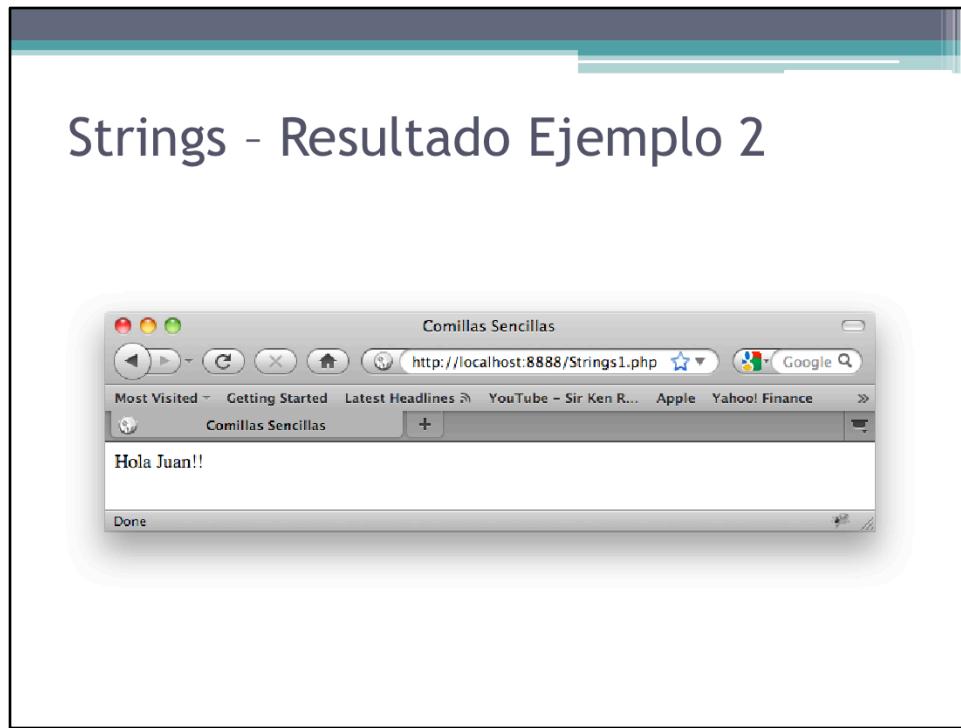
```
<html>
  <head>
    <title>Comillas Sencilla</title>
  </head>
  <body>
    <?php
      $nombre = "Juan";
      echo '\tHola\n$nombre !!';
    ?>
  </body>
</html>
```



Se imprime el string textualmente

## Strings - Ejemplo 2

```
<html>
  <head>
    <title>Comillas Sencilla</title>
  </head>
  <body>
    <?php
      $nombre = "Juan";
      echo "\tHola\n$nombre !!";
    ?>
  </body>
</html>
```



Se interpreta el contenido del string antes de imprimirlo

## Funciones

Reutilizando código

Las funciones nos proveen de herramientas para customizar nuestro código y hacerlo más legible y comprensible. En este apartado trabajamos con funciones definidas por el usuario en PHP. Se enseña como crear una función y se da un ejemplo práctico de su uso.

## Funciones

- Aparte de las funciones incluidas en el lenguaje PHP, el usuario (tu, el programador) puede definir funciones
- Una función definida por el usuario se implementa con tres propósitos principales:
  1. Hace tareas repetitivas más fácilmente
  2. Modularizar el código
  3. Hacer el código más fácil de leer

Algunas de las características de las funciones en general que aplican para PHP.

## Funciones - Definición

```
<?php
function sumarNums($param1, $param2, $param3)
{
    $sum = $param1 + $param2 + $param3;
    return $sum;
?>
```

sumarNums es el nombre de la función

\$param1,\$param2,\$param3, son los parámetros que recibe la función. No hay límite al número de parámetros que puede recibir una función

Una función puede devolver un solo valor, en este caso \$sum. El valor que se devuelve viene inmediatamente después de la palabra clave return

Definición de una función muy sencilla. El concepto de función se inspira en el de una función matemática en cuanto que se pueden tener múltiples variables de entrada pero sólo una salida. Sin embargo, visto desde un mayor nivel de abstracción, esta salida o valor de retorno pudiera ser una variable de tipo estructurado que da cabida a devolver más de un valor.

## Funciones (cont.)

- Al igual que en el caso de las variables, las funciones son débilmente tipadas
  - No se requiere definir tipos de parámetros
  - No se requiere definir tipo de dato del valor de retorno
- A una función se le puede pasar cualquier cosa y devuelve cualquier cosa
  - Puede hacer el código confuso
  - Queda por parte del usuario verificar la validez de los datos de entrada y salida

## Funciones - Ejemplo

```
<?php
function sumarNums($param1, $param2, $param3)
{
    $sum = $param1 + $param2 + $param3;
    return $sum;
}
echo sumarNums(1,2,3);      => 6
echo sumarNums(1,2,"Hola");=>3
echo sumarNums(true,2,'a'); =>3
?>
```

PHP no arroja  
ninguna notificación o  
mensaje de error. ¿Es  
este comportamiento  
lógico?

## Funciones - Ejemplo (cont.)

```
<?php
function sumarNums($param1, $param2, $param3)
{
    if(!is_numeric($param1) || !is_numeric($param2)|| !
        is_numeric($param3)){
        die("Algun parametro no es de tipo numerico");
    }
    $sum = $param1 + $param2 + $param3;
    return $sum;
}
?>
```

## Ejercicio 2 - Funciones

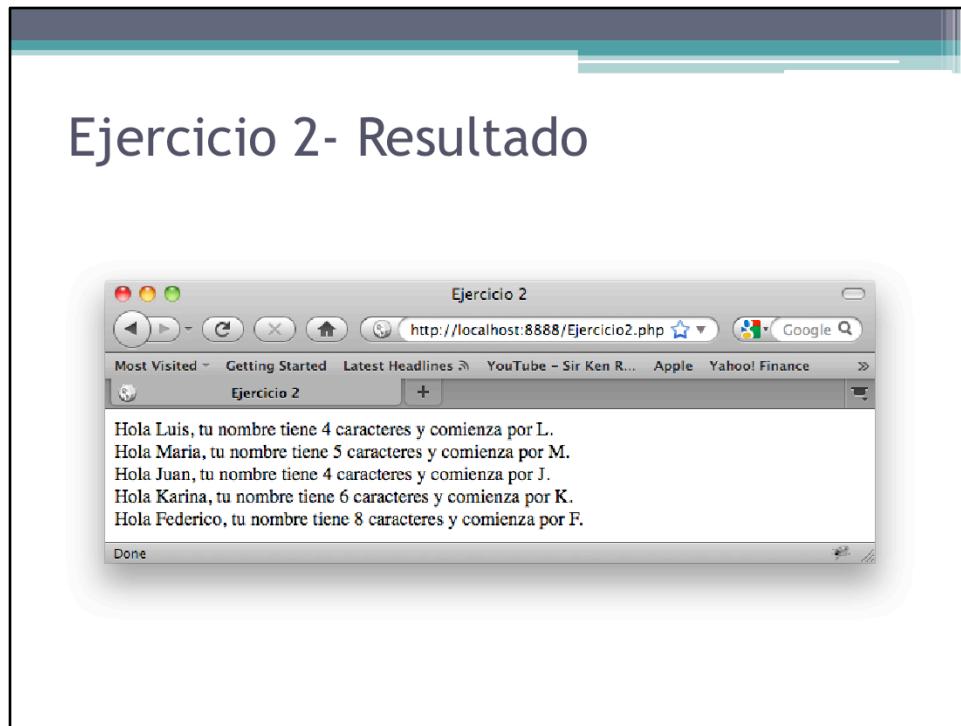
- Duración: de 10 a 15 minutos
- Objetivos:
  - Crear una función desde cero
  - Entender la importancia de las funciones al momento de modularizar código
- Especificaciones:
  - Crear una función que reciba el nombre de una persona y devuelva un mensaje

Este segundo ejercicio añade complejidad a una página web sencilla.

## Ejercicio 2 - Funciones(cont.)

- Instrucciones:
  - Crear un nuevo archivo **Ejercicio2.php**
  - Salvarlo en la carpeta  
*/htdocs/CursoPHPBasico/Ejercicios*
  - Crear una función llamada “saludo” que reciba un string y cree el siguiente mensaje:  
*“Hola \$nombre, tu nombre tiene \$x caracteres y comienza por \$primera letra”*
  - Usar las siguientes funciones PHP: *strlen, substr*
  - Probar con: Luis, María, Juan, Karina y Federico.

Seguir las instrucciones.



El trabajo que se podría tornar tedioso se hace sencillo con el uso de funciones.

## Ejercicio 2- Solución

```
<?php
function saludo($nombre){
    $x = strlen($nombre);
    $primera_letra = substr($nombre,0,1);
    $mensaje = "Hola $nombre, tu nombre tiene $x caracteres y comienza por $primera_letra. <br/>";
    return $mensaje;
}
<?
<html>
    <head>
        <title>Ejercicio 2</title>
    </head>
    <body>
        <?=saludo("Luis")?>
        <?=saludo("Maria")?>
        <?=saludo("Juan")?>
        <?=saludo("Karina")?>
        <?=saludo("Federico")?>
    </body>
</html>
```

El código resulta mucho más sencillo de entender y por ende de actualizar.

## Estructuras de Control

¿Cúal camino tomar?

En los lenguajes imperativos, las estructuras de control, junto con las variables, forman los bloques fundamentales para la construcción de cualquier programa.

## Estructuras de Control

- PHP es un lenguaje con soporte a dos paradigmas de programación:
  - Programación Imperativa
  - Programación Orientada a Objetos (la veremos en detalle luego)
- Por el momento nos ocuparemos de la primera

PHP orientado a objetos es el curso que le sigue a este.

## Estructuras de Control - Programación Imperativa

- Definición:

- “describe la programación en términos del estado del programa y sentencias que cambian dicho estado. Los programas imperativos son un conjunto de instrucciones que le indican al computador cómo realizar una tarea.”

- Notas:

- Es tal vez la forma más común de programación
- Ejemplo: C, C++
- Sentencias: if, if-else, loops: while, do...while,for

Definición tomada de Wikipedia.

## Condicionales

if, if-else, if-elseif-else, switch

En esta sección se estudia los condicionales if, if-else, if-elseif-else, switch

## Estructuras de Control - if

### Declaraciones de if sencillo

```
if (condiciones)
```

```
    Hacer una sola tarea;
```

```
if (condiciones) {
```

```
    Hacer tarea 1;
```

```
    Hacer tarea 2;
```

```
    ...
```

```
    Hacer tarea n;
```

```
}
```

If sencillos. Este es el bloque básico de construcción en un lenguaje imperativa. La sintaxis en PHP es muy sencilla, tal como se muestra arriba.

## Estructuras de Control - if - else

```
if (condiciones) {  
    Hacer esta tarea;  
} else {  
    Hacer esta otra;  
}
```

Tareas  
Mutuamente  
Excluyentes

Las tareas son mutuamente excluyente, lo que significa que la realización de una de ellas impide la realización de la otra.

## Estructuras de Control - if - elseif - else

```
if (condiciones) {  
    Hacer esta tarea;  
} else if (condiciones) {  
    Hacer esta otra;  
}else{  
    Hacer esta;  
}
```

Tareas  
Mutuamente  
Excluyentes

Se puede tener cualquier cantidad de bloques “else if”

## Estructuras de Control

Operadores de Comparación	
==	Igual a
!=	No Igual
>	Mayor que
<	Menor que
>=	Mayor igual que
<=	Menor igual que
====	Exactamente igual (en tipo y valor)
!==	No exactamente igual

Como PHP es un lenguaje débilmente tipado, provee operaciones de comparación sobre valores y tipo de datos como una forma de controlar la ejecución de los scripts.

## Estructuras de Control

Operadores Lógicos		
&&	Y	\$a && \$b
	O	\$a    \$b
!	NO	!\$a

Donde, \$a y \$b son variables booleanas (verdadero, falso)

Operadores lógicos (también llamados booleanos) clásicos.

## Ejercicio 3 - Control

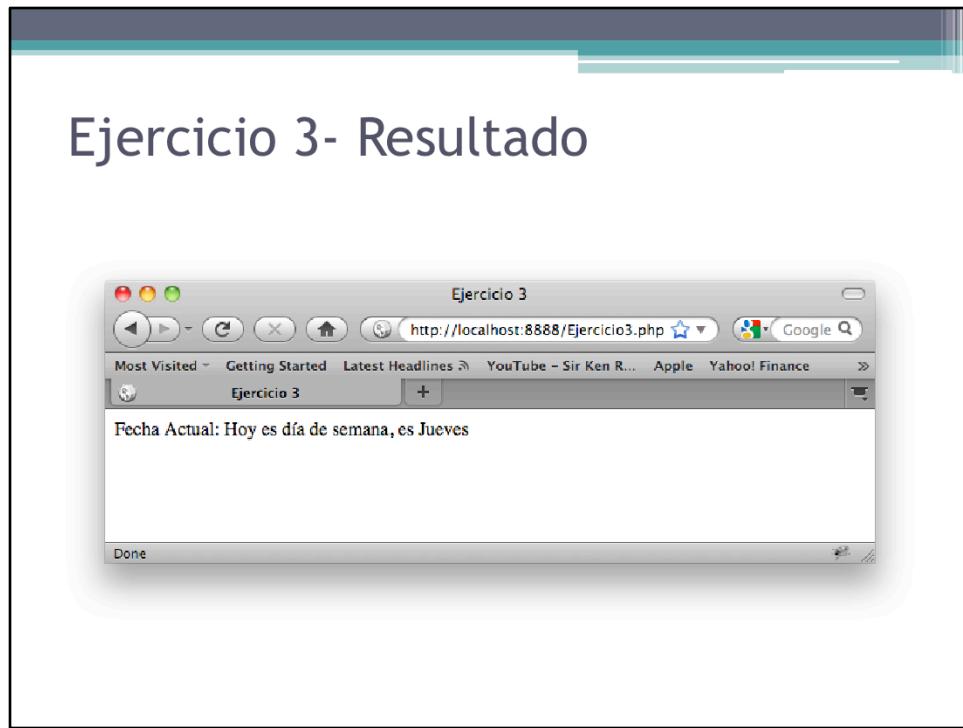
- Duración: de 5 a 10 minutos
- Objetivos:
  - Utilizar las estructuras básicas de control de PHP
  - Cambiar el comportamiento de una página web
- Especificaciones:
  - Realizar una página web que muestre un mensaje distinto dependiendo del día de la semana actual.
  - Si es fin de semana, presentar un mensaje.
  - Si no es fin de semana, presentar otro.
  - Si es jueves, añadir un mensaje adicional.

Un ejercicio para familiarizarse con las estructuras de control básica.

## Ejercicio 3 - Control (cont.)

- Instrucciones:
  - Crear un nuevo archivo **Ejercicio2.php**
  - Salvarlo en la carpeta  
*/htdocs/CursoPHPBasico/Ejercicios*
  - Buscar en la documentación de PHP (php.net) la función **date**
  - Asignar a la variable \$DiaActual el resultado de la función **date** con los parámetros adecuados
  - Estructurar el código de forma que presente distintos mensajes según las condiciones descritas

Seguir las instrucciones.



Se despliega el día de la semana actual, sin necesidad de cambiar el código por parte del usuario.

## Ejercicio 3- (Posible) Solución

```
<?php
    $DiaActual = date("D");
    $Mensaje = "";
    if($DiaActual == "Sat" || $DiaActual == "Sun"){
        //Fin de semana
        $Mensaje = "Feliz fin de semana";
    }else{
        //Dia de semana
        $Mensaje = "Hoy es d&iacute;a de semana";
        if($DiaActual == "Thu"){
            //Particularmente, dia jueves
            $Mensaje .= ", es Jueves";
        }
    }
?>
<html>
    <head>
        <title>Ejercicio 2.a</title>
    </head>
    <body>
        Fecha Actual: <?=$Mensaje?>
    </body>
</html>
```

Código solución Ejercicio 3.

## Estructuras de Control - switch

Combina  
varios if-else

```
switch (expresión) {  
    case 'a':  
        echo 'la expresión es a';  
        break;  
    case 'b':  
        echo 'la expresión es b';  
        break;  
    case 'c':  
        echo 'la expresión es c';  
        break;  
    default :  
        echo 'expresión desconocida';  
        break;  
}
```

Switch clásico. Combina varios if en un solo bloque.

## Loops (bucles)

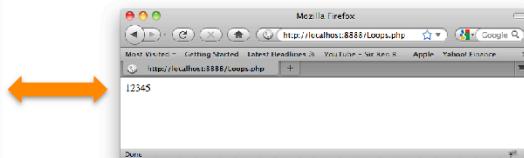
while, do...while, for, foreach

Los loops (en inglés) o bucles (traducción al español) son mecanismos de los lenguajes de programación modernos que le permiten al usuario realizar tareas repetitivas de manera sencilla. En esta sección estudiaremos estos mecanismos en el contexto de PHP y programación para la web.

## Loops - while

- La directiva while se utiliza para ejecutar un bloque de código, repetidamente, mientras una o más condiciones sean verdad

```
$a = 1;  
while($a < 6){  
    echo $a;  
    $a++;  
}
```



## Loops - while (cont.)

### MOSCA: con los loops infinitos

¿Qué pasa si un script se pega en un loop infinito?

- PHP tiene dos mecanismos para manejarlo:
  - Tiempo de ejecución máximo
  - Límite de memoria
- Configuración php.ini

```
;;;;;;
; Resource Limits ;
;;;;;

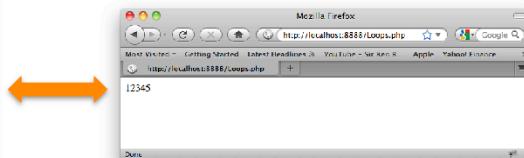
max_execution_time = 30      ; Maximum execution time of each script, in seconds
max_input_time = 60          ; Maximum amount of time each script may spend parsing request data
memory_limit = 32M           ; Maximum amount of memory a script may consume (8MB)
```

Si un script tarda más de 30 segundos en ejecutarse, php lo abortará. Esto también ocurre si el script consume más de 32MB de memoria. En el caso en que un script quede en un loop infinito, en 30 segundos el programa será abortado automáticamente.

## Loops - do...while

- En este caso la condición de parada se chequea luego de ejecutar el bloque de código una primera vez

```
$a=1;  
do {  
    echo $a;  
    $a++;  
} while ($a < 6);
```



## Loops - for

- Un *loop for* toma tres expresiones, separadas por punto y coma (;), y agrupadas en paréntesis, antes del bloque de código sobre el cual se va a iterar. Las reglas del for son:
  1. La primera expresión se ejecuta una sola vez antes de iniciar el loops. Es la condición de inicialización
  2. La segunda expresión se evalúa antes de cada iteración. Si es falsa, entonces termina el loop
  3. La tercera expresión se evalúa al final de cada iteración. Esta se usa como contador o cualquier otra expresión que calcule el final del loop

## Loops - for - sintaxis

The diagram illustrates the syntax of a for loop in PHP. A grey rectangular box contains the code:

```
for ($a=1; $a < 6; $a++)  
{  
    echo $a;  
}
```

Three light blue speech bubbles point to specific parts of the code:

- Inicialización: points to the first part of the condition, `$a=1`.
- Condición de parada: points to the second part of the condition, `$a < 6`.
- Contador: points to the third part of the condition, `$a++`.

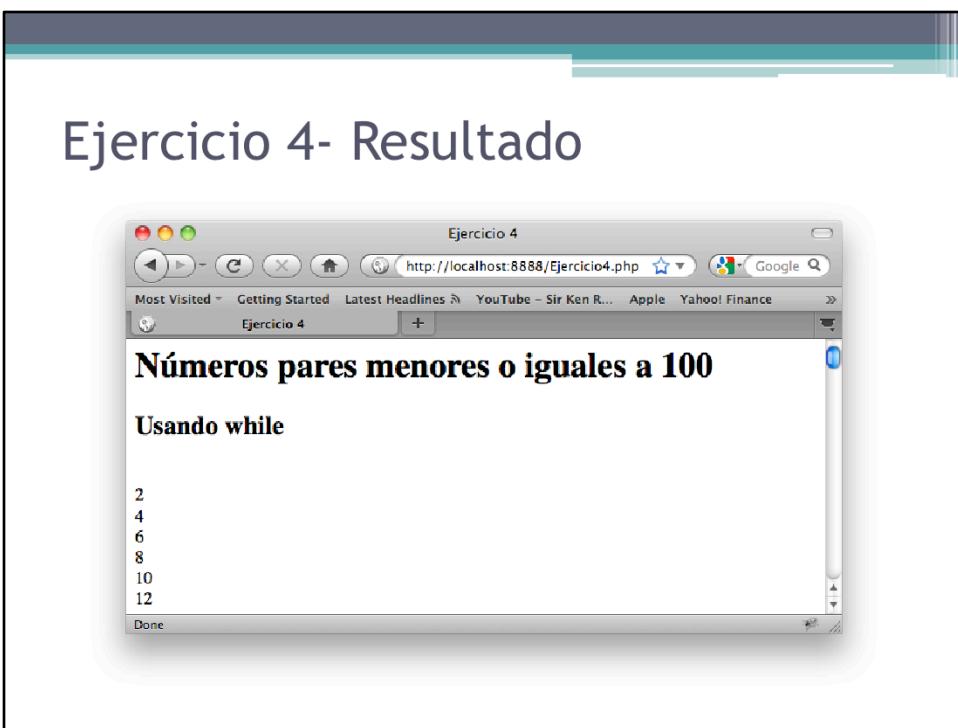
## Ejercicio 4 - Loops

- Duración: de 10 a 15 minutos
- Objetivos:
  - Utilizar las estructuras iterativas de PHP
- Especificaciones:
  - Calcular e imprimir todos los números pares menores o iguales a 100, utilizando un **while**
  - Calcular e imprimir todos los números impares menores o iguales a 100, utilizando un **for**
  - Calcular e imprimir todos los números menores o iguales a 100 que sean divisibles por 2 y 3

## Ejercicio 4 - Loops (cont.)

- Instrucciones:
  - Crear un nuevo archivo **Ejercicio4.php**
  - Salvarlo en la carpeta  
*/htdocs/CursoPHPBasico/Ejercicios*
  - Plantear el for o while correspondiente. Si es while, ¿cuál es la condición de parada? ¿Cómo vas a imprimir los números?
  - Si es for, ¿cuáles son las tres condiciones?
  - Para el último caso, ¿cómo sabemos si un número es divisible por otro?

Seguir las instrucciones.



## Ejercicio 4- (Posible) Solución

```
<html>
    <head>
        <title>Ejercicio 4</title>
    </head>
    <body>
        <h1>Números pares menores o iguales a 100</h1>
        <h2>Usando while</h2>
        <?php
            $numero = 1;
            while($numero<=50){
                echo "<br/>". ($numero * 2);
                $numero++;
            }
        ?>
        <h1>Números impares menores o iguales a 100</h1>
        <h2>Usando for</h2>
        <?php
            for($i=1;$i<100;$i=$i+2){
                echo "<br/>" . $i;
            }
        ?>
        <h1>Números menores o iguales a 100 divisibles por 2 y 3</h1>
        <?php
            for($i=1;$i<100;$i++){
                if($i%2 == 0 && $i%3==0){
                    echo "<br/>" . $i;
                }
            }
        ?>
    </body>
</html>
```

## Recibir parámetros por URL

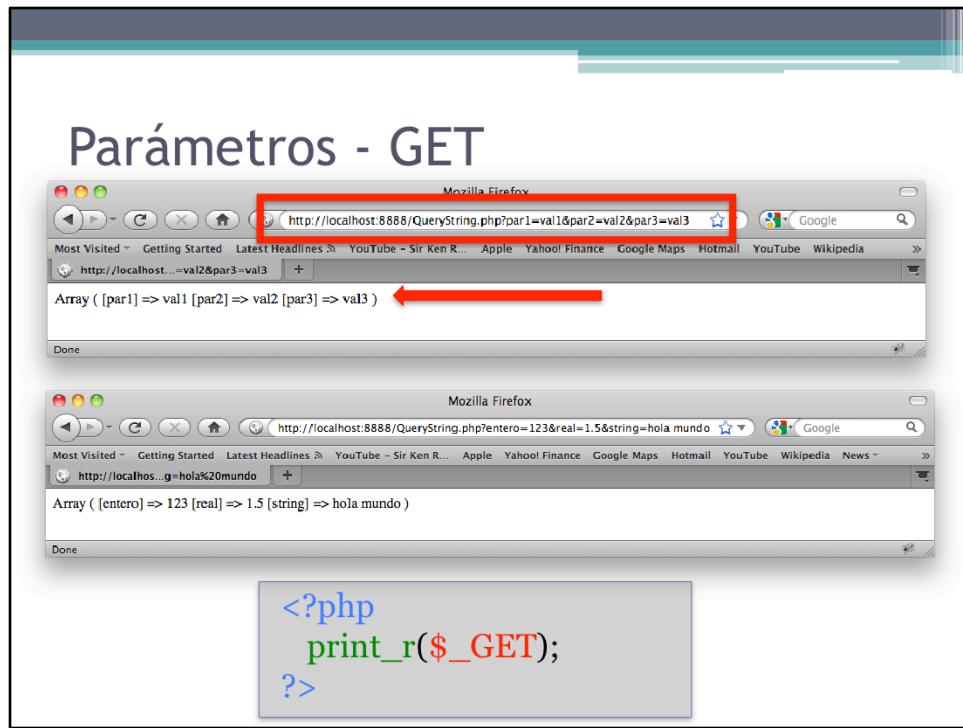
ejercicio.php?par1=val1&par2=val2...

Un programa para la web no puede ser autocontenido o cerrado. Por el contrario, la única forma de que una página web sea dinámica y útil para el cliente es si éste es capaz de proveerle información. En esta sección veremos como es posible esa comunicación entre cliente y código PHP.

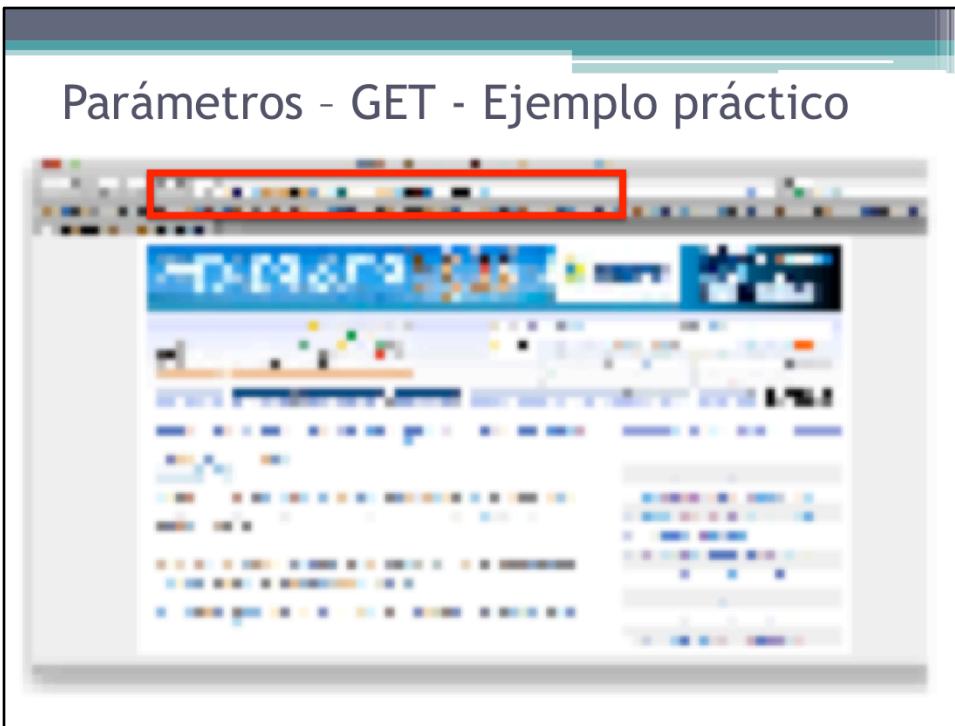
## Parámetros

- Existen dos mecanismos para recibir parámetros por parte del usuario:
  - GET
    - Principalmente se refiere al URL
  - POST
    - Principalmente se refiere al uso de formularios

Existen dos formas básicas de recibir parámetros por parte del cliente, GET y POST. Cada una presenta sus desventajas y ventajas.



Esto sólo funciona para recibir parámetros via web browser. Para recibir parámetros a través del terminal existen otros mecanismos.



Ejemplo práctica del uso de los parámetros tipo GET para comunicar una página web con el cliente.

## Parámetros - GET - Características

- Es visible para todo el mundo
- A través de este mecanismo sólo se debe manipular información poco sensible
- Propenso a ataques SQL Injection
  - Se debe filtrar esta data
  - Veremos mecanismos más adelante (Zend Framework) para evitar que esto ocurra

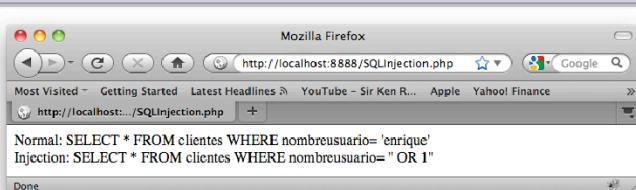
GET son los parámetros que se reciben por el url.

## Parámetros - GET - Características

```
$nombre = "enrique";
$query = "SELECT * FROM clientes WHERE nombreusuario= '$nombre'";
echo "Normal: " . $query . "<br />";
// data que viene de GET y puede inyectar SQL
$nombre_malo= "' OR 1'";

// query Inseguro
$querymalo = "SELECT * FROM clientes WHERE nombreusuario= '$nombre_malo'";

// mostrar el query
echo "Injection: " . $querymalo ;
```



Ejemplo de SQL Injection. Utilizar los valores de los parámetros que se reciben directamente del usuario es fuente común de brechas en la seguridad del sistema. Estos parámetros se deben procesar, es decir, asegurarse que contienen información válida y no riesgosa, antes de utilizarlos.

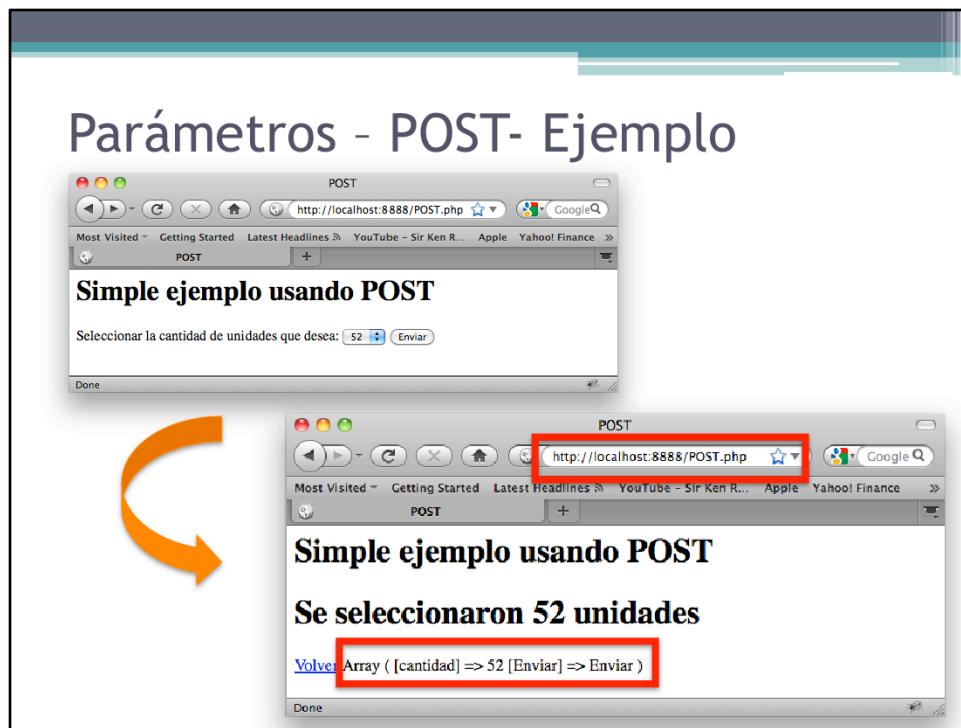
## Parámetros - POST- Características

- Llega a través de formularios HTML
- Oculto al usuario “de a pie”
- Permite enviar mayor cantidad de data que GET
- Es accesible a alguien con conocimientos en web
  - Con firebug y un poquito de tiempo se puede hacer lo mismo que en el caso de GET
- Por lo tanto también se debe manipular con cuidado

La forma más común de pasar parámetros es a través del POST. Permite enviar una cantidad mucho mayor que GET y además esconde el mecanismo como se pasan los parámetros al cliente “de a pie”.

## Parámetros - POST- Ejemplo

```
<html>
    <head>
        <title>POST</title>
    </head>
    <h1>Simple ejemplo usando POST</h1>
    <body>
        <?php if(isset($_POST["cantidad"])):?>
            <h1>Se seleccionaron <?=$_POST["cantidad"]?> unidades</h1>
            <a href="javascript:history.back()">Volver</a>
            <?php print_r($_POST) ?>
        <?php else:?:>
            <form method="post">
                Seleccionar la cantidad de unidades que desea:
                <select name="cantidad">
                    <?php for($i=0;$i<100;$i++):?>
                        <option value=<?=$i?>><?=$i?></option>
                    <?php endfor;?>
                </select>
                <input type="submit" name="Enviar" value="Enviar">
            </form>
        <?php endif;?>
    </body>
</html>
```



## Ejercicio 5 - POST

- Duración: de 15 a 20 minutos
- Objetivos:
  - Entender el funcionamiento de un formulario HTTP
  - Manipular los parámetros recibidos por POST
- Especificaciones:
  - Hacer un formulario con los campos Nombre, Apellido y Edad.
  - Escribir un script que valide si estos campos fueron ingresados o no.

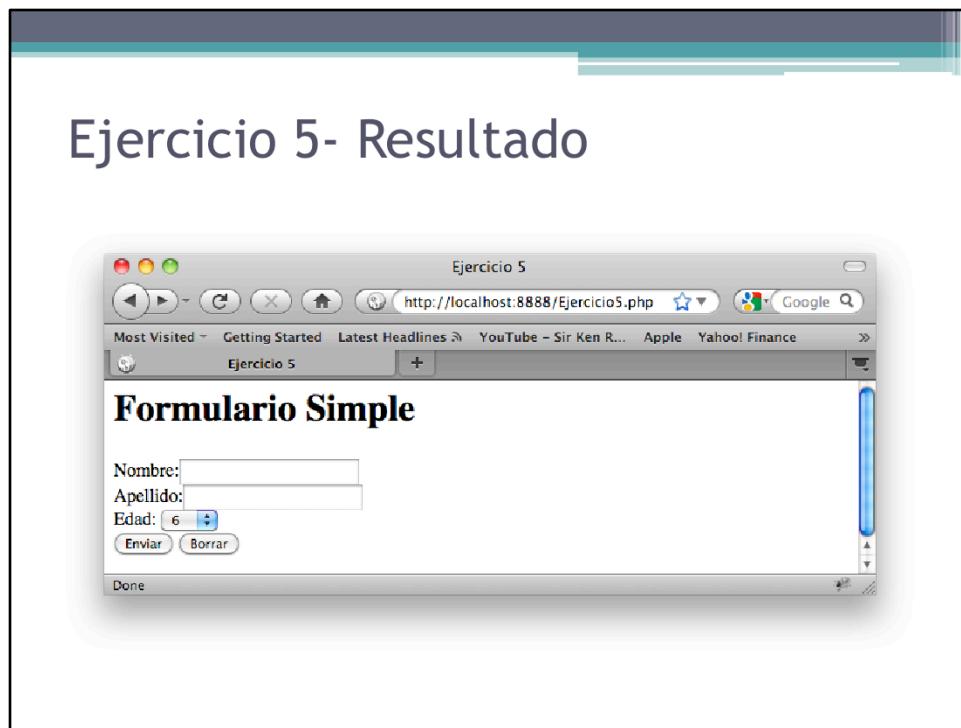
## Ejercicio 5 - POST (cont.)

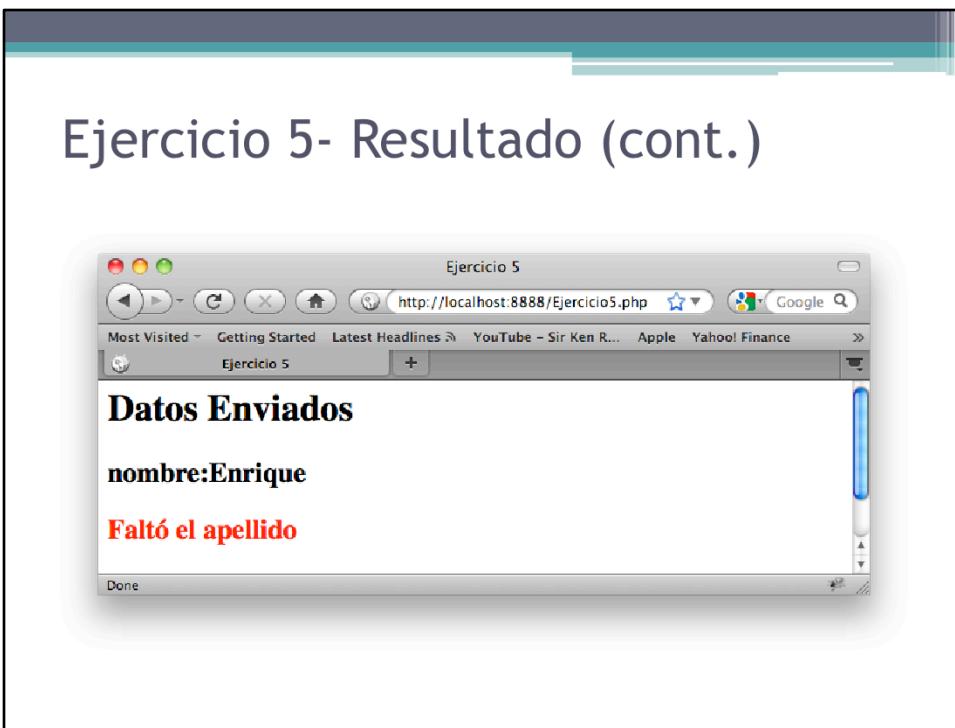
- Instrucciones:
  - Crear un nuevo archivo **Ejercicio5.php**
  - Salvarlo en la carpeta  
*/htdocs/CursoPHPBasico/Ejercicios*
  - Plantear un formulario HTML con los campos  
Nombre, Apellido y Edad
  - Enviar los datos a través de POST al mismo  
archivo
  - Verificar si los datos vienen o no vacío y mostrar  
un mensaje acorde

Seguir las instrucciones.

## Ejercicio 5 - POST (cont.)

- Al momento de hacer el ejercicio, pregúntese:
  - ¿Existirá alguna forma más eficiente de verificar todos los campos, sin repetir código?
  - ¿Existirá algún campo para el cuál se puede generar valores predeterminados?
    - ¿De que forma es más eficiente generar estos valores predeterminados?





## Ejercicio 5- Solución

- Ver archivo Ejercicio5.php

## *Tipos de datos estructurados: arreglos*

```
$Beatles = array();  
$Beatles[0] = 'John';  
$Beatles[1] = 'Paul';  
$Beatles[2] = 'George';  
$Beatles[3] = 'Ringo';
```

Los tipos de datos estructurados elevan el nivel de abstracción del lenguaje y permiten un uso más eficiente de los recursos del sistema, así como una mayor legibilidad y mantenimiento del código. Un nivel aún mayor de abstracción lo posee la programación orientado de objetos. En este curso sólo vamos a tratar con datos estructurados del tipo arreglos. En el curso siguiente trataremos con programación orientada a objetos en PHP.

## Arreglos

- Hasta el momento hemos trabajado con tipos de datos entero, reales, *strings*, y lógicos
- Casi de manera inmediata surge la necesidad de *ordenar* o *estructurar* las cosas
- Los arreglos o *arrays* introducen este orden al permitir al programador ordenar datos relacionados de la forma más conveniente
- Ya vimos dos ejemplos en `$_GET` y `$_POST`

Utilizar un arreglo para ordenar los datos que provienen como parámetros del usuario (`$_GET` y `$_POST`), provee de una forma lógica de agrupar datos relacionados con un solo punto de acceso.

## Arreglos - Inicialización

```
$VecindadDelChavo = array();  
$VecindadDelChavo[0] = "Chavo del ocho";  
$VecindadDelChavo[1] = "Quico";  
$VecindadDelChavo[2] = "La chilindrina";  
$VecindadDelChavo[3] = "Don Ramón";  
$VecindadDelChavo[4] = "Doña Florinda";  
$VecindadDelChavo[5] = "Profesor Jirafales";
```



```
$VecindadDelChavo = array('Chavo del ocho','Quico','La  
chilindrina','Don Ramón','Doña Florinda','Profesor Jirafales');
```

Ejemplo de organización lógica de datos en arreglos.

## Arreglos - Agregar elementos

- Si se conoce la cantidad de elementos

```
$VecindadDelChavo[6] = "Señor Barriga";
```

- Si no se conoce la cantidad de elementos

```
$VecindadDelChavo[] = "Señor Barriga";
```

Agregar datos a un arreglo es muy sencillo, otra característica interesante de PHP.

## Arreglos - Leyendo elementos

- Leer un elemento de un arreglo es tan sencillo como referencia a su posición:

```
<?php  
echo $VecindadDelChavo[3];  
?>
```

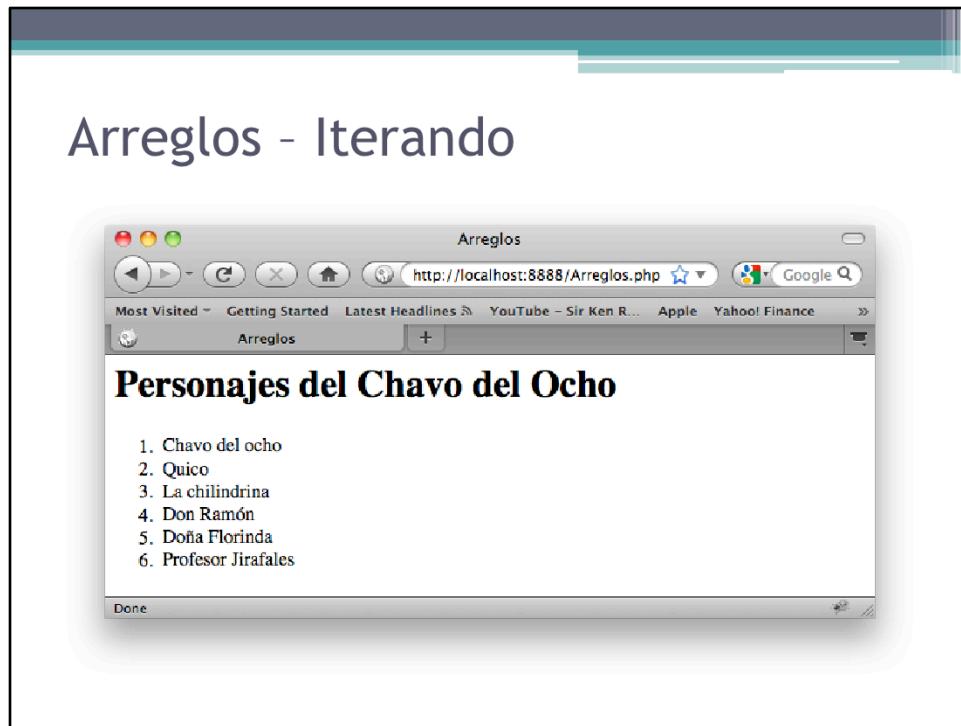
- Este código imprime: Don Ramón

Acceder a un elemento de un arreglo es tan fácil como referenciar su posición.

## Arreglos - Iterando

```
<html>
  <head>
    <title>Arreglos</title>
  </head>
  <body>
    <h1>Personajes del Chavo del Ocho</h1>
    <?php $VecindadDelChavo = array("Chavo del ocho",
        "Quico",
        "La chilindrina",
        "Don Ram&iacute;n",
        "Do&ntilde;a Florinda",
        "Profesor Jirafales");?>
    <ol>
      <?php foreach($VecindadDelChavo as $personaje): ?>
        <li><?=$personaje?></li>
      <?php endforeach;?>
    </ol>
  </body>
</html>
```

Una vez que se tienen los datos en un arreglo, con PHP se puede iterar sobre dichos datos muy cómodamente.



Ejemplo típico de datos provenientes de un arreglo: una lista de elementos.

## Arreglos - Asociativos

- Los arreglos pueden ser indexados con enteros o con *strings*
- Indexación por *strings* le otorga mucha flexibilidad al lenguaje y es una de las características a resaltar de PHP
- Un arreglo asociativo es análogo a un arreglo simple, con la diferencia de que la manera de accesar los datos es a través de *strings*

Los arreglos asociativos se asemejan más al lenguaje natural y pueden resultar más útiles en cantidad de ocasiones.

## Arreglos - Asociativos - Ejemplo

```
$VecindadDelChavo = array();  
$VecindadDelChavo['personaje principal']=“Chavo del ocho”;  
$VecindadDelChavo['personaje secundario'] = “Quico”;
```



```
$VecindadDelChavo =  
array(“personaje principal” => “Chavo del ocho”,  
      “personaje secundario” => “Quico”);
```

Ejemplo de arreglo asociativo.

## Arreglos - Bidimensionales

- Ver archivo ArreglosBidimensionales.php

The screenshot shows a web browser window with the title "Arreglos Bidimensionales". Below the title, a heading reads "Personajes del Chavo del Ocho y sus frases célebres". A numbered list follows, detailing various characters and their iconic catchphrases:

1. Chavo del ocho
  - Fue sin querer queriendo
  - Es que no me tienen paciencia
  - Eso, eso, eso, eso
  - Al cabo que ni quería
2. Quico
  - Te voy a acusar con mi mamá que me...
  - Bueno, así pos si
  - Mamá!
3. La chingadura
  - Fijate, fijate, fijate, fijate
  - Pepero lindo, mi amor
  - Si serás, si serás!
  - Chavo, ¡llegaste tarde a la repartición de cerebro!
4. Don Ramón
  - Tengo usted mucha burriza, Señor Razón
  - ¡Tomá!
  - Me lleva el chanfle!
  - Chingadura!
5. Doña Florinda
  - Vámonos, escroco, no te jentes con esa chusma
  - Tesoro, no le hagas caso a la chusma
  - Tesoro!
6. Profesor Jiribala
  - Ta, ta, ta, taaa... tó!
  - ¿Por qué causa, motivo, razón o circunstancia?
  - Fa primer tigre...
  - Después de usted

## Arreglos - Funciones nativas PHP

- Para manipular arreglos existen un sin fin de funciones nativas en PHP
  - `sort()`
  - `asort()`
  - `shuffle()`
  - `count()`
  - `explode()`
  - Etc
- Revisar documentación  
<http://php.net/manual/en/language.types.array.php>

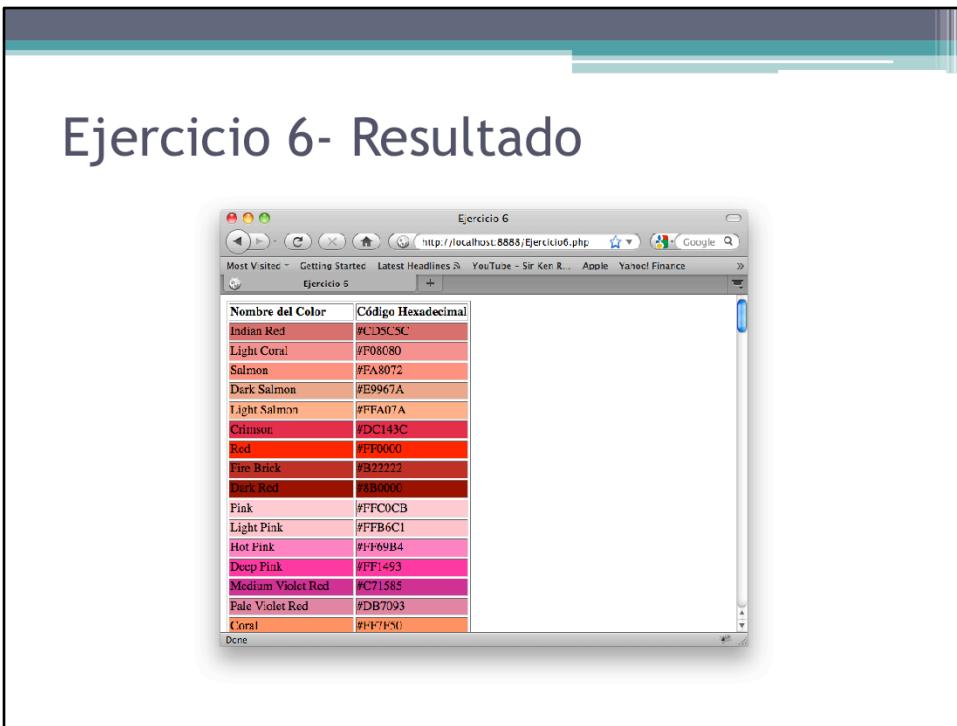
## Ejercicio 6 - Arreglos

- Duración: de 15 a 20 minutos
- Objetivos:
  - Utilizar arreglos en PHP
- Especificaciones:
  - Crear una tabla en la que se listen un conjunto de colores, su nombre y código hexadecimal
  - El color fondo de cada fila debe coincidir con el color en cuestión

## Ejercicio 6 - Arreglos (cont.)

- Instrucciones:
  - Buscar el archivo **Ejercicio6.php** en */htdocs/CursoPHPBasico/Ejercicios*
  - Allí encontrará el arreglo `$colores` con un conjunto de colores predeterminados
  - Se debe crear la estructura HTML necesaria (tablas) e iterar sobre el arreglo para producir la salida necesaria.

Seguir las instrucciones.



## Ejercicio 6- Solución

```
<html>
  <head>
    <title>Ejercicio 6</title>
  </head>
  <body>
    <table border="1">
      <tr>
        <td><b>Nombre del Color</b></td>
        <td><b>Código Hexadecimal</b></td>
      </tr>
      <?php foreach($colores as $nombre=>$codigohex):?>
        <tr style="background-color:<?= $codigohex ?>">
          <td><?= $nombre ?></td>
          <td><?= $codigohex ?></td>
        </tr>
      <?php endforeach;?>
    </table>
  </body>
</html>
```

## *PHP y MySQL básico*

SELECT \* FROM....

Otro pivote sobre el que giran las aplicaciones web es el uso de base de datos. En este curso exploramos como es la interacción entre PHP y las base de datos tipo MySQL para generar aplicaciones web dinámicas.

## PHP y MySQL

- PHP provee librerías nativas para conexión a BD  
(No únicamente MySQL)
- *mysqli* (MySQL **improved**, versión  $\geq 5$ )
  - Interface Orientada a Objetos
  - Soporte para *statements* preparados
  - Soporte de *statements* múltiples
  - Soporte para transacciones
  - Capacidad extendida para *debugging*
  - Soporte para servidor embebido

PHP posee extensiones para varios tipos de base de datos. Sin embargo, en este curso sólo utilizaremos base de datos MySQL.

En esta lámina se muestran algunas de las características de la nueva extensión de PHP para conexión con MySQL. En este curso utilizaremos *mysql* desde un enfoque procedimental. En cursos siguientes utilizaremos la extensión orientada a objetos.

## PHP y MySQL (cont.)

- Crearemos una aplicación Web para manipular (agregar, eliminar y editar) información de películas
- Necesitaremos lo siguiente
  - Una base de datos
  - Una o más tablas dependiendo del modelo de datos
  - Uno o más scripts PHP que interactúen con la base de datos

## PHP y MySQL (cont.)

- Preparando el ambiente. Necesitamos:
  - Crear una base de datos llamada **cursobasicophp**
  - Crear una tabla llamada **peliculas**
- Instrucciones:
  - Acceder al **cPanel**
    - Crear la base de datos
  - Acceder al **phpMyAdmin**
    - Crear la tabla

La implementación de las bases de datos está sujeta a las características del ambiente de desarrollo. Aquí se presenta un esquema común a la fecha, para sistema linux.

## PHP y MySQL (cont.)

Nombre	Tipo
id_pelicula	INT (11), NOT NULL, PRIMARY
nombre_pelicula	VARCHAR (120), NOT NULL
fecha_pelicula	INT (4), NULL

Aquí se almacenará el  
año de estreno de la  
película, por ejemplo,  
2004

The screenshot shows the phpMyAdmin interface with two database structures displayed side-by-side.

**Top Window (localhost:8080/cursophpbasico):**

- Database:** cursophpbasico (0)
- Tables:** No tables found in database.
- Action Bar:** Structure, SQL, Search, Query, Export, Import, Operations, Privileges, Drop.
- Table Structure:** policeus (1 table)
 

Action	Records	Type	Collation	Size	Overhead
Sum	0	InnoDB	latin1_swedish_ci	16.0 kB	-
Sum	0	InnoDB	latin1_swedish_ci	16.0 kB	0 B
- Create New Table:** Create new table on database cursophpbasico
 

Name:	Number of fields:
-------	-------------------

May be approximate. See FAQ 3.11

**Bottom Window (localhost:8080/cursophpbasico/peliculas):**

- Database:** cursophpbasico (1)
- Tables:** peliculas
- Action Bar:** Browse, Structure, SQL, Search, Insert, Export, Import, Operations, Empty, Drop.
- Table Structure:** peliculas
 

Field	Type	Collation	Attributes	Null	Default	Extra	Action
id_pelicula	int(11)	latin1_swedish_ci	No	None			
nombre_pelicula	varchar(120)	latin1_swedish_ci	No	None			
fecha_pelicula	int(4)		Yes	NULL			
- Indexes:**

Action	Keyname	Type	Unique	Packed	Field	Cardinality	Collation	Null	Comment
	PRIMARY	BTREE	Yes	No	id_pelicula	0	A		

Add 1 field(s)  At End of Table  At Beginning of Table  After (id\_pelicula)

Create an Index on 1 columns

+ Details...

## PHP y MySQL (cont.)

- Para interactuar con MySQL desde PHP:
  - Conectarse a la base de datos
  - Enviar un *query* a la base de datos
  - Almacenar resultados obtenidos de la BD
  - Trabajar con los resultados
    - Posiblemente actualizar o eliminar datos de la BD
  - Mostrar resultados al cliente
  - Liberar la base de datos

The screenshot shows a presentation slide with a dark header bar. The main title is "PHP y MySQL - Conexión". Below the title is a code block in PHP:

```
<?php
    $DB = mysqli_connect('localhost', 'root', 'root', 'cursophpbasico');
    if (mysqli_connect_errno()):
        No se pudo conectar con la base de datos: <?=mysqli_connect_error();?>
    <?php
        else:
```

Para chequear si la conexión fue exitosa utilizamos la función `mysqli_connect_errno()`, la cual devuelve un número de error en caso en que ocurra algún error o cero en caso contrario. Si ocurre un error, podemos obtener detalles del mismo a través de la función `mysqli_connect_error()`.

## PHP y MySQL - Buscar datos

```
<?php
else:
$Query = 'SELECT * FROM peliculas';
$result = mysqli_query($DB,$Query);
$NumResults = mysqli_num_rows($result);
?>
<b><?=$NumResults?> Pel&iacute;culas</b>
```

## PHP y MySQL - Mostrar los resultados

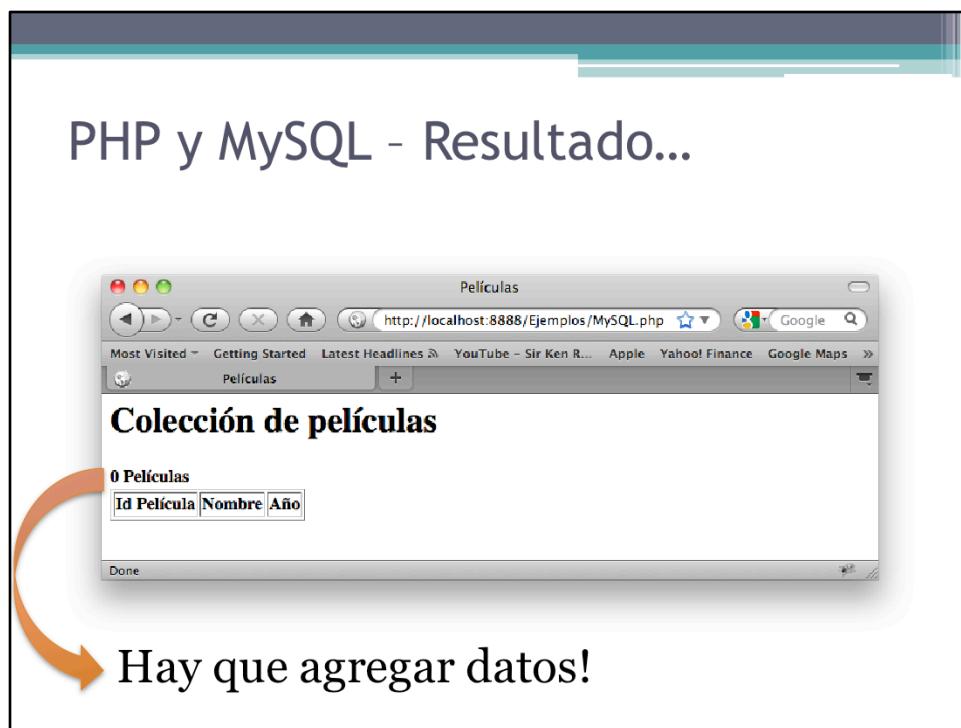
```
<table border="1">
  <tr>
    <th>Id Pel&iacute;cula</th>
    <th>Nombre</th>
    <th>A&ntilde;o</th>
  </tr>
  <?php while ($Row = mysqli_fetch_assoc($Result)) :?>
  <tr>
    <td><?=$Row['id_pelicula']?></td>
    <td><?=$Row['nombre_pelicula']?></td>
    <td><?=$Row['fecha_pelicula']?></td>
  </tr>
  <?php endwhile;?>
</table>
```

## PHP y MySQL - Liberar los recursos

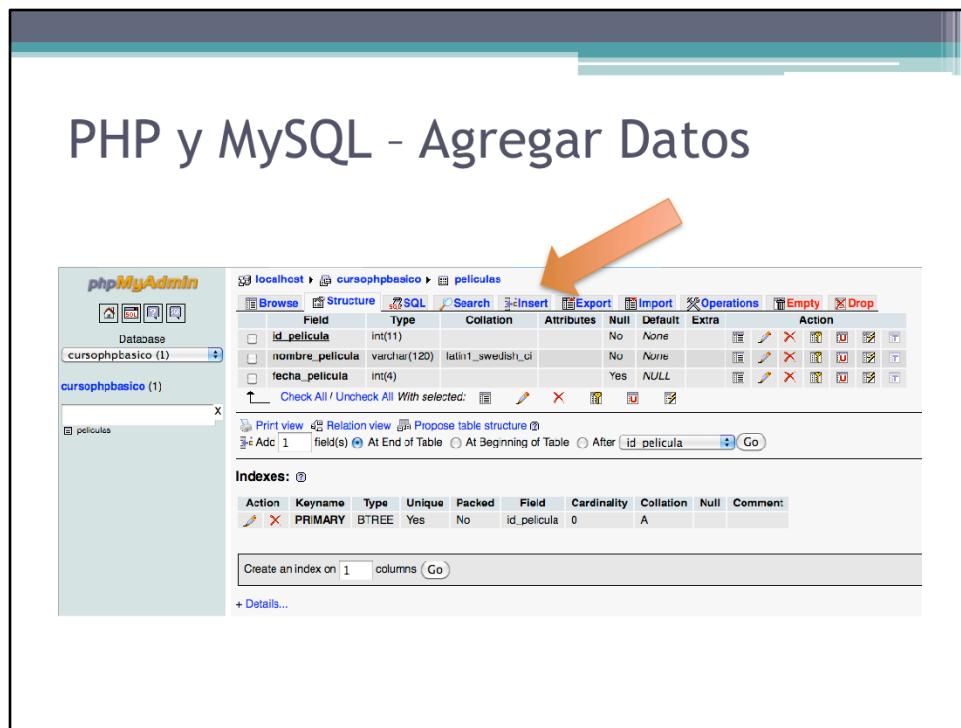
```
<?php  
mysqli_free_result($Result);  
mysqli_close($DB);  
endif;  
?>
```

The screenshot shows a web page titled "PHP y MySQL - Todo Junto". The page contains a table with three columns: "Id Película", "Nombre", and "Fecha". The table has three rows, each representing a movie from the database. The first row is highlighted in yellow. The page is generated by a PHP script that connects to a MySQL database named "cursophbasico". The script retrieves all records from the "peliculas" table and prints them into an HTML table.

```
<head>
<title>Pel&iacute;culas</title>
</head>
<body>
<h1>Colecci&iacute;n de pel&iacute;culas</h1>
<?php
$DB = mysqli_connect('localhost','root','root','cursophbasico');
if (mysqli_connect_errno()):
?>
No se pudo conectar con la base de datos: <?=mysqli_connect_error();?>
<?php
else:
$query = 'SELECT * FROM peliculas';
$result = mysqli_query($DB,$query);
$numResults = mysqli_num_rows($result);
?>
<b><?=$numResults?> Pel&iacute;culas</b>
<table border="1">
<tr>
<th>Id Pel&iacute;cula</th>
<th>Nombre</th>
<th>Fecha</th>
</tr>
<?php while ($Row = mysqli_fetch_assoc($result))?:?>
<tr>
<td><?=$Row['id_pelicula']?></td>
<td><?=$Row['nombre_pelicula']?></td>
<td><?=$Row['fecha_pelicula']?></td>
</tr>
<?php endwhile;?>
</table>
<?php
mysqli_free_result($result);
mysqli_close($DB);
endif;
?>
</body>
...</pre>
```



## PHP y MySQL - Agregar Datos



The screenshot shows the phpMyAdmin interface for a database named 'cursophpbasico'. The current table is 'peliculas'. The top navigation bar includes 'localhost', 'cursophpbasico', 'Structure', 'SQL', 'Search', 'Insert' (which has an orange arrow pointing to it), 'Export', 'Import', 'Operations', 'Empty', and 'Drop'. Below the navigation is a table structure with three columns: 'id\_pelicula' (int(11)), 'nombre\_pelicula' (varchar(120) with collation 'latin1\_swedish\_ci'), and 'fecha\_pelicula' (int(4)). The 'Attributes' column shows 'No', 'None', and 'Yes' respectively. The 'Null' column shows 'None', 'None', and 'NULL'. The 'Default' column shows 'None', 'None', and 'NULL'. The 'Extra' column shows various icons. Below the table is a section for 'Indexes' with one entry: 'PRIMARY' (BTREE, Yes, Field: 'id\_pelicula', Cardinality: 0, Collation: A). At the bottom, there are buttons for 'Add 1 field(s)', 'At End of Table', 'At Beginning of Table', and 'After id\_pelicula'.

## PHP y MySQL - Agregar Datos

localhost > cursophbasico > películas

Browse Structure SQL Search Insert Export Import Operations Empty Drop

Field	Type	Function	Null	Value
id_pelicula	int(11)			
nombre_pelicula	varchar(120)			Back To The Future I
fecha_pelicula	int(4)		<input checked="" type="checkbox"/>	1985

Go

Ignore

Field	Type	Function	Null	Value
id_pelicula	int(11)			
nombre_pelicula	varchar(120)			Back To The Future II
fecha_pelicula	int(4)		<input checked="" type="checkbox"/>	1989

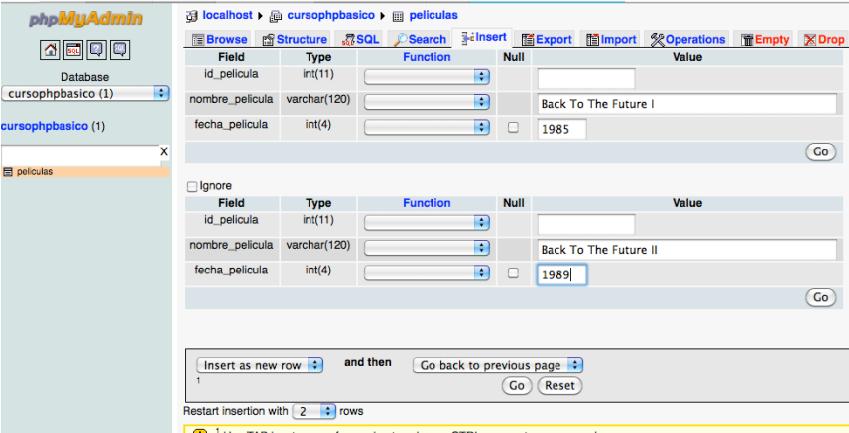
Go

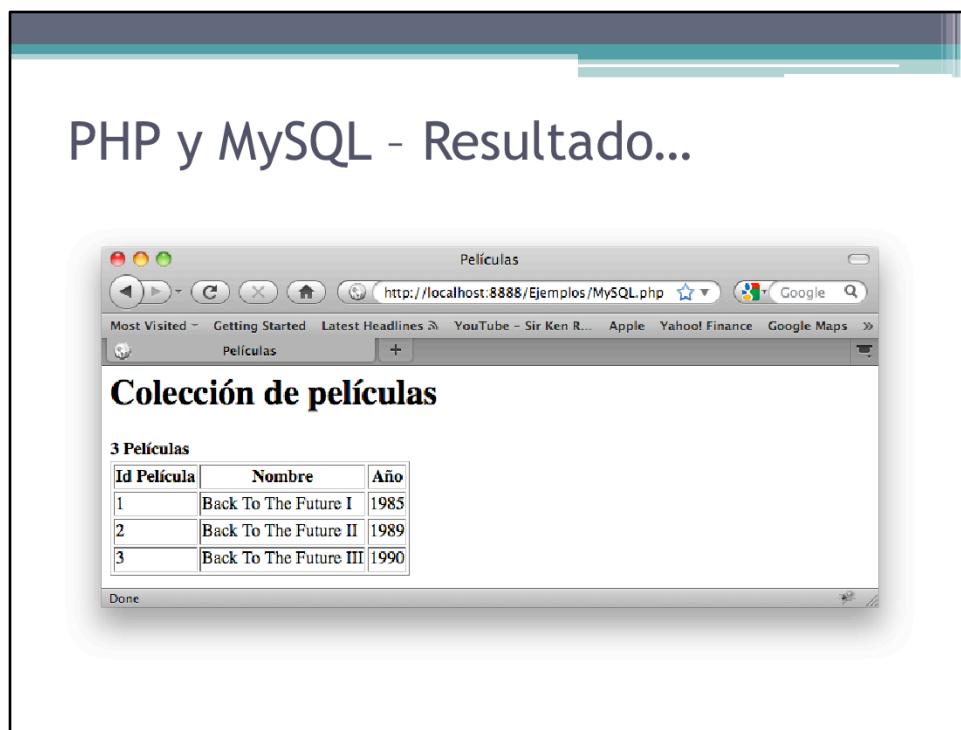
Insert as new row and then Go back to previous page

1 Go Reset

Restart insertion with 2 rows

Use TAB key to move from value to value or CTRL+arrows to move anywhere





## PHP y MySQL - Lo que falta (básico)

- Se debe crear una interfaz desde nuestra aplicación, no desde phpMyAdmin, que nos permita:
  - Agregar registros
  - Modificar registros
  - Eliminar registros

## PHP y MySQL - Lo que falta...

- Ideas:
  - Dividir la aplicación en frontoffice y backoffice
    - Frontoffice: muestra información
    - Backoffice: permite editar información. Hay que autenticar al cliente.
  - Posibilidad de ordenar la lista de películas por campos
  - Directores
  - Actores
  - Premios
  - Trailers
  - Fotos
  - Comentarios y ratings de usuario
  - etc

## *Ejercicio Integral*

Consolidando lo aprendido...

Este ejercicio, de larga duración, busca consolidar conocimientos y acercar al usuario a lo que en realidad sería el desarrollo con PHP

## Ejercicio Integral - Descripción

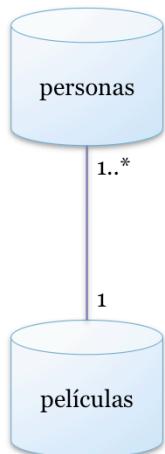
- Duración: de 45 a 60 minutos o más
- Objetivos:
  - Consolidar lo aprendido en este curso
- Especificaciones:
  - Trabajaremos sobre nuestra aplicación de películas y la expandiremos para incorporar nuevas funcionalidades

## Ejercicio Integral - Especificaciones

- Implementaremos las siguientes ideas para la aplicación de películas:
  - Editar películas
  - Eliminar películas
  - Agregar personas
    - Tipos de personas: Actores, Directores, Técnicos
    - Idealmente también borrarlos y editarlos
  - Relacionar toda la información
  - Ordenar películas por fecha ascendente o descendente

De las ideas plasmadas anteriormente vamos a implementar sólo algunas. Se deja como ejercicio al lector implementar el resto.

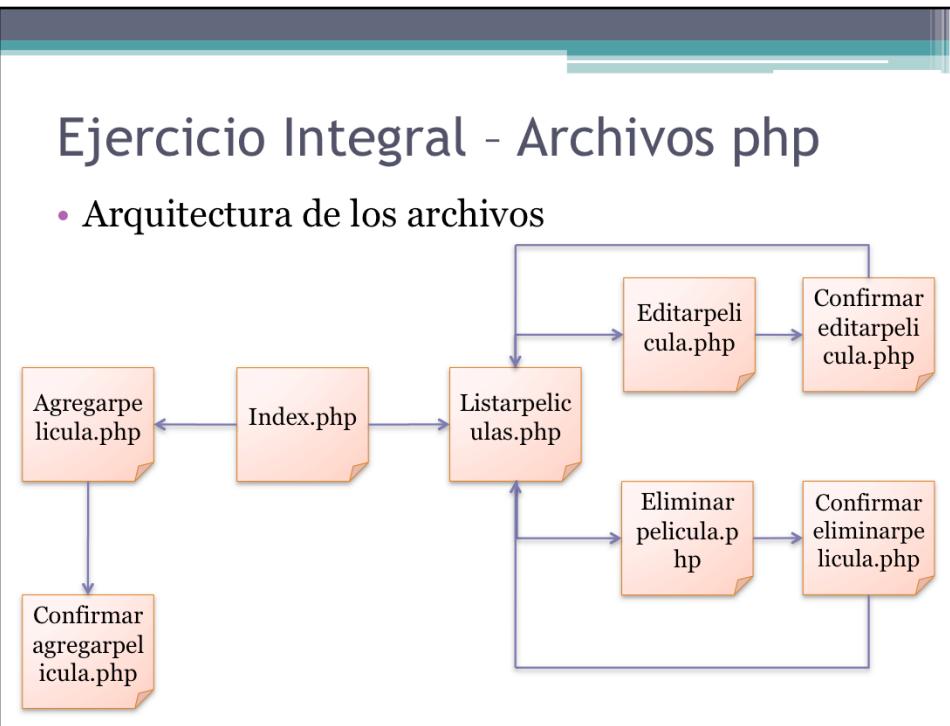
## Ejercicio Integral - Modelo de BD



Modelo sencillo para nuestra aplicación

## Ejercicio Integral - Modelo de BD

- Películas
  - Id
  - Nombre
  - Fecha
- Personas
  - Id
  - Nombre
  - Apellido
  - Fecha de Nacimiento
  - Profesión: Actor, Director, Técnico,...



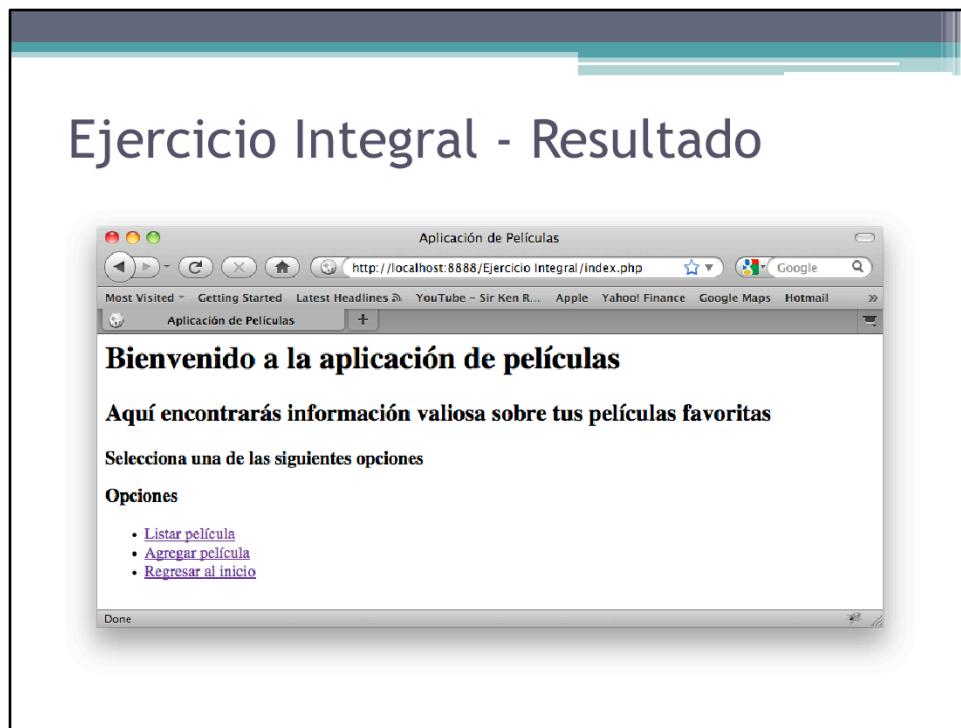
Possible planning of the interaction and structure of the PHP files

## Ejercicio Integral (cont.)

- Instrucciones:

- Se trabajará sobre la carpeta Ejercicio Integral en */htdocs/CursoPHPBasico/Ejercicio Integral*
- Analizar el problema en búsqueda de modularizar el código. Utilice funciones cuando sea posible.
- Evite SQL Injection. Haga su código seguro

Seguir las instrucciones.



## Ejercicio Integral - Solución

- Ver carpeta con la solución

## *Conclusiones*

“La vida es el arte de sacar conclusiones suficientes a partir de datos insuficientes.”

Samuel Butler (novelista) (1835-1902)  
Novelista inglés.

## Conclusiones

- PHP es un lenguaje flexible y versátil
- Aún con una aplicación sencilla es fácil ver el potencial para el desarrollo de aplicaciones complejas
- La flexibilidad introduce desorden estructural
- Necesitamos proveer al lenguaje de mayor estructura para facilitarnos la vida (PHP OO)