
# Auxiliar Nº 2- PHP

IN3501 - DII

INGENIERIA INDUSTRIAL
UNIVERSIDAD DE CHILE



# ¿Que es PHP?

- PHP : Hypertext Preprocessor
- Lenguaje Interpretado.
- Se ejecuta en el lado del servidor.
- Independiente del navegador.
- Sintaxis estilo Java, C, Perl, etc.
- Orientado principalmente a la generación de contenidos web de forma dinámica.
- Es el lenguaje mas popular en esta área.



# ¿Cómo se relaciona PHP y HTML?

- PHP no es un reemplazo al HTML
- PHP tiene su potencial en crear contenido HTML de manera dinámica

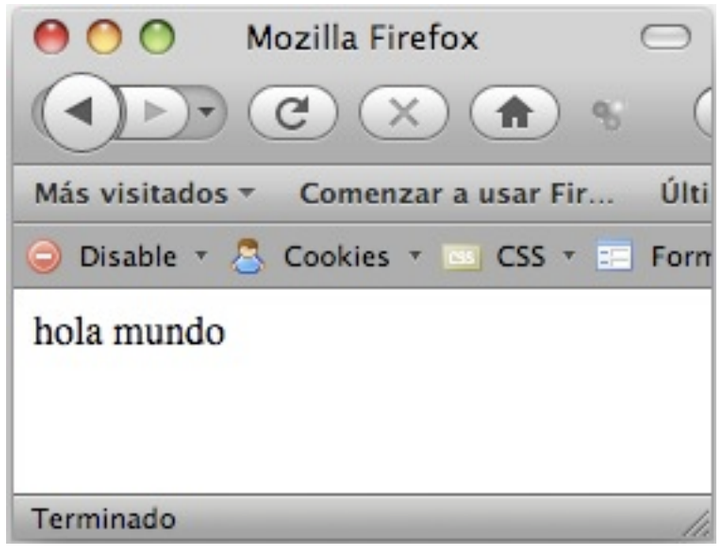
# Codigo PHP

- El codigo PHP debe encontrarse dentro de los tags <?php ?> o <? ?> .
- Se recomienda la primera opción pues ser estandar, la segunda es una abreviación que no siempre es reconocida
- Al finalizar una expresión se debe colocar un “;” .Tal como en Java

# Hola Mundo en PHP

- <?php echo “Hola Mundo”; ?>

- Resultado →



# Variables

- Las variables en PHP se deben declarar anteponiendo un “$” , ej: $mi_variable
- Las variables cambian de tipo según el contexto.
- A diferencia de Java no es necesario decir de que tipo son las variables.
  Ej: int variable = 3 ⟶ Java
  $variable = 3 ⟶ PHP
  variable = “hola”; Error
  $variable = “hola”; Toma valor “hola”



# Comentarios en PHP

- Los comentarios en PHP son igual que en Java.
- // esto es un comentario de una linea
- /*
  Esto es un comentario de varias lineas
  */

# Arreglos

- No se deben declarar como si ocurre en Java
- Pueden guardar cualquier tipo de datos
  $arreglo[0]=”hola”;
  $arreglo[1]= 4;
- Se autoincrementan automáticamente
- Para saber la cantidad de elementos, se usa la función count
  Ej: count($arreglo) ⟶ devolverá 2



# Arreglos Asociativos

- Funcionan como los arreglos normales, pero aceptan cadenas como llave.
- Dentro de los arreglos se pueden guardar otros.
- Se pueden inicializar asignando los valores individuales o a través de la función array().
- Ej: $persona['nombre'] = “Juan”;
  $persona['apellido'] = “Gonzalez”;
- Utilizando array();
- $persona = array(“nombre” => “Juan”,
  “apellido” => “Gonzalez”);

# Strings

- Se declaran como cualquier variable con el valor entre comillas.
  Ej: $var = “hola”;
- Si las variables tienen valores numericos se pueden operar como tal.
  Ej: $var1= 2; $var2= 3;
  $var3 = $var1+ $var2 ⟶ $var3 vale 5

# Strings(2)

- La concatenación es con el operador “.” (punto).
- $final = $rut.”-”.$dv;
- Concatenación recursiva con .=
- $final = $final . $rut <=> $final .= $rut
- Como alternativa se pueden unir variables al declarar variables cadenas:
- $final = “$rut-$dv”;

# Operadores Aritmeticos

| Operador | |
|---|---|
| + | Suma |
| - | Resta |
| * | Multiplicación |
| / | División |
| % | Módulo |
| ++ | Suma 1 |
| -- | Resta 1 |



# Operadores Comparación

| Operador | Nombre |
|---|---|
| "==" | Igual |
| != | Distinto |
| < | Menor |
| > | Mayor |
| <= | Menor igual |
| >= | Mayor igual |



# Operadores Comparación

| Operador | Nombre |
|---|---|
| && | Y Lógico |
| And | Y Lógico |
| \|\| | O Lógico |
| Or | O Lógico |
| ! | Negación |



# Control Flujo

- If(condicion) {...} else if( condicion) {...} else {...}
- while(condicion) {...}
- for(inicializacion ; condicion ; incremento){}



# Salidas PHP

- En PHP hay muchas formas de generar salidas de texto.
- print – echo: Permiten imprimir una cadena o una unión de éstas.
- printf(formato, args...) : Imprime una cadena con formato.
- printf("El rut es %d-%d", $rut, $dv);



# Funciones

- Permiten encapsular grupos de código para ser reutilizado posteriormente.
- Sintaxis:
  -function *nombre*(arg1, ..., argn) {}
- Con la instrucción *return* la función devuelve un valor.

# Funciones(2)

- function fibo($n){
  if($n<=2) return 1;
  return fibo($n-1) + fibo($n-2);}

- <?php
  $res = fibo(3);
  echo $res;
  ?>

- Dará como resultado 2

# Funciones(3)

- Los nombres de las variables dentro de las funciones no tienen relación con las declaradas afuera.

- Para usar variables externas dentro de funciones se utiliza la instrucción *global*.

```
<?
$var = “Variable externa”;
fn(); /* Se llama a la función */
function fn() {
global $var;
echo $var;
} ?>
```

# Inclusion archivos externos

- El código escrito en PHP no debe estar necesariamente en solo un archivo.
- Es posible modularizar código para lograr la reutilización de éste en distintas aplicaciones.
- Existen 4 formas de incluir código externo:
  - include(archivo);
  - require(archivo);
  - include_once(archivo);
  - require_once(archivo);



# Archivos en PHP

- Desde PHP se pueden manipular archivos guardados en el servidor. Se usa la función fopen();
  $fp = fopen(Archivo, Modo);
- *$fp:* puntero del archivo abierto.
- *Archivo*: ruta donde se encuentra el archivo, puede ser relativa o absoluta en el disco local o una URL.
- *Modo*: Especifica como se va a manejar el archivo abierto.

# Modos soportados PHP

| Modo | Descripción |
|---|---|
| r | Apertura para sólo lectura; ubica el apuntador de archivo al comienzo del mismo. |
| r+ | Apertura para lectura y escritura; ubica el apuntador de archivo al comienzo del mismo. |
| w | Apertura para sólo escritura; ubica el apuntador de archivo al comienzo de éste y lo trunca a una longitud de cero. Si el archivo no existe, intenta crearlo. |
| w+ | Apertura para lectura y escritura; ubica el apuntador de archivo al comienzo de éste y lo trunca a una longitud cero. Si el archivo no existe, intenta crearlo. |
| a | Apertura para sólo escritura; ubica el apuntador de archivo al final del mismo. Si el archivo no existe, intenta crearlo. |
| a+ | Apertura para lectura y escritura; ubica el apuntador de archivo al final del mismo. Si el archivo no existe, intenta crearlo. |
| x | Creación y apertura para sólo escritura; ubica el apuntador de archivo al comienzo de éste. Si el archivo ya existe, la llamada a fopen() fallará devolviendo FALSE |
| x+ | Creación y apertura para lectura y escritura; ubica el apuntador de archivo al comienzo de éste. Si el archivo ya existe, la llamada a fopen() fallará devolviendo FALSE |

# Archivos en PHP

- Todos los archivos abiertos con fopen() deben cerrarse con la función *fclose(handler);*
- Para escribir se utiliza *fwrite(handler, string)*.
- Para leer se puede utilizar:
  - *fread(handler, longitud)*. Se lee y retorna hasta la cantidad indicada o hasta que se acaba la información donde apunta el handler.
  - *fgets(handler)*. Se lee hasta alcanzar un fin de línea o de archivo.

INGENIERIA INDUSTRIAL
UNIVERSIDAD DE CHILE
IN3501

# Leer y Escribir Archivos

## Escritura

```
$fp = fopen("archivo.dat", "w");
fputs($fp, "$nombre $apellido\n");
fclose($fp);
```

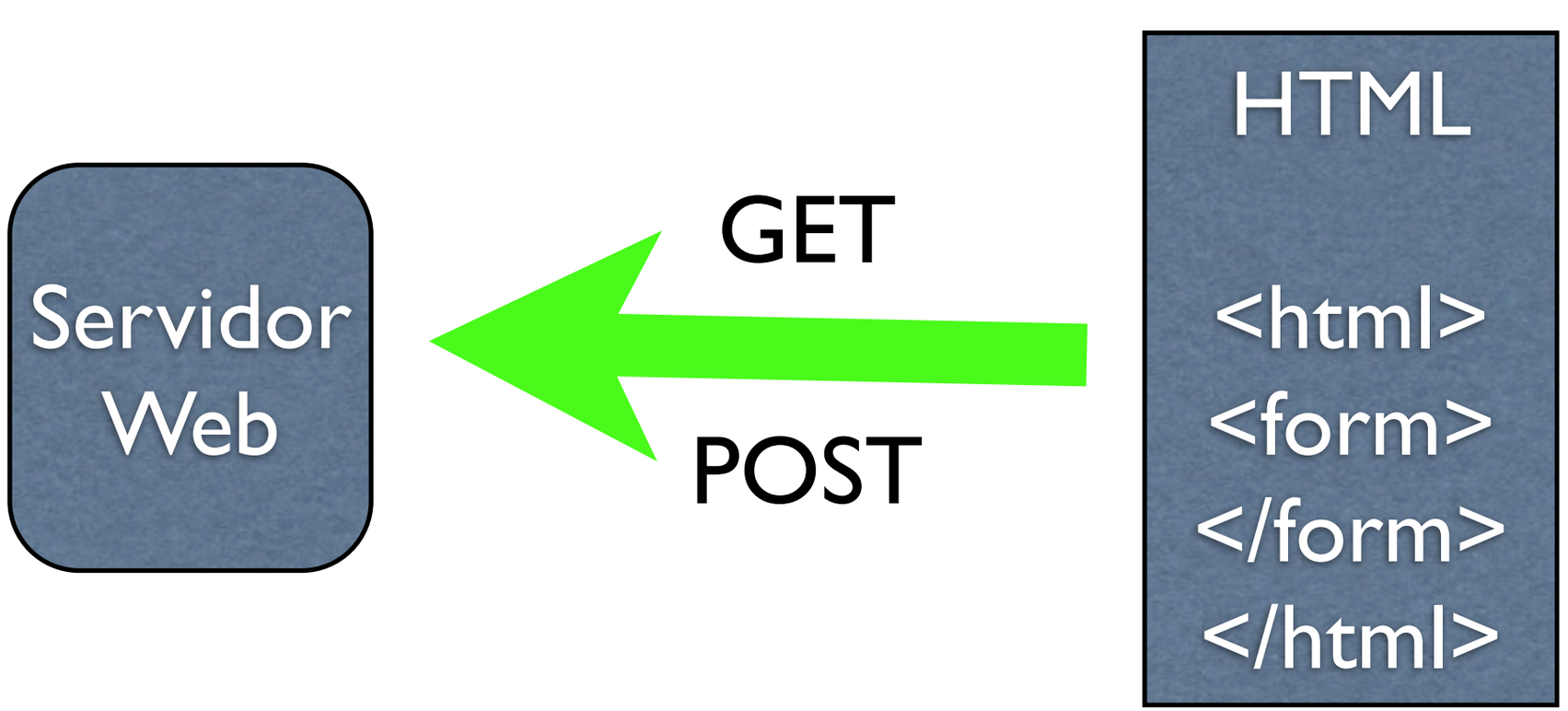
## Lectura

```
$fp = fopen("archivo.dat", "r");
while(!feof($fp)){
$line=fgets($fp, 512);
print($line);
}
fclose($fp);
```

# Formularios

INGENIERIA INDUSTRIAL
UNIVERSIDAD DE CHILE
IN3501

La comunicación entre el usuario y la aplicación web se realiza principalmente a través de formularios Html.



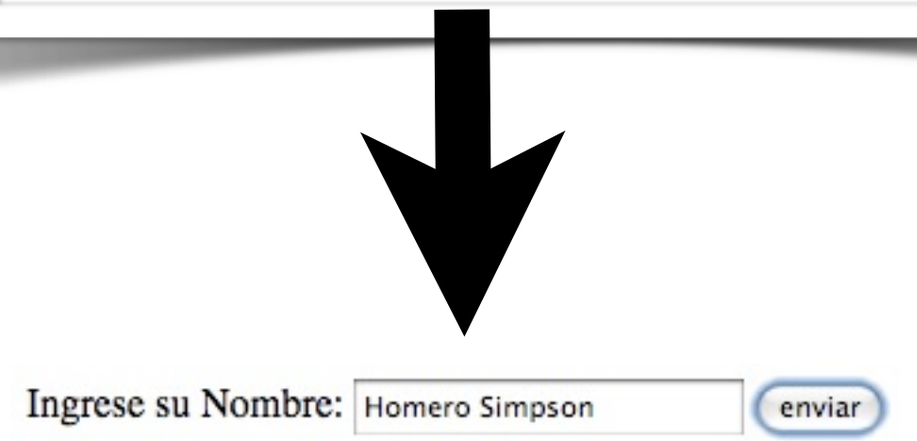


# Formularios(2)

- Los formularios proveen de 2 métodos para enviar información al servidor:
  - GET: Los datos son enviados por la URL.
  - POST: Los datos son enviados a través de la conexión TCP.
- PHP por su parte recibe los datos a través de los arreglos asociativos $_GET y $_POST para cada método.

# Formularios(3)

```
<form action=”recibe_datos.php” method=”GET”>
<input type=”text” name=”nombre”>
<input type=”submit”>
</form>
```




```
<?
echo “Nombre:”.$_GET['nombre'];
?>
```

Nombre: Homero Simpson

INGENIERIA INDUSTRIAL
UNIVERSIDAD DE CHILE

IN3501

# Tips Programación

- Aplicaciones medianas a grandes requieren mantener una organización de sus archivos y el código fuente en ellos.
- Cada organización puede determinar sus propios estándares que les permitan generar software mantenible.
- Con reglas simples es posible tener un orden moderado de las aplicaciones.

# Tips Programación(2)

- Separar archivos en carpetas según su tipo

- –**imagenes**: todos los archivos que correspondan a imágenes debe residir aqui.
–**include**: para los archivos que correspondan a librerias o códigos comunes para la aplicación.
–**template**: las plantillas deben encontrarse aqui.
–Los archivos de aplicación se pueden dejar en la raiz del directorio.

# Tips Programación(3)

- Modularización
  - Separar código común.
  - Crear archivo de configuracion común.
  - Se recomienda utilizar el comando *require_once()* para las llamadas de éstos.



# Tips Programación(4)

- Separación de código

  –Busca separar inteligentemente el código fuente en pequeñas piezas para un mejor rastreo.

  –Funciones de uso común, como llamadas a bases de datos o manejo de propiedades.



# Tarea N°2

- Realizar un programa para manejar nombre, apellidos y tipo de usuario(profesor y alumno dentro de un select)
- Para esto debe tener un archivo llamado curso.txt, el cual se debe poder leer y escribir.
- Se debe crear un archivo llamado tarea2.php, dentro de este se debe leer el archivo y listar los registros existentes, al final del listado debe existir un formulario que permita agregar un nuevo registro al archivo.

Cliente envia requerimiento a Servidor Web

Apache sabe sólo leer HTML, cuando llega un req. PHP apache dice no se como ejecutar eso, pero se quien sabe (motor de PHP) entonces le envia el req. a el.


Apache envia pagina HTML, que armó con la información desde Motor PHP + codigo HTML que pudiese existir

Motor de PHP, ejecuta el codigo PHP que envió Apache, y su salida(del motor php) es redirigida hacia Apache (cuando hacen echo, print, printf, etc)