



Nivel II de PHP

julio, 2016



Objetivos del curso

- Aprender a programar orientado a objetos en PHP
- Manipular variables de sesión
- Generar documentos PDF dinámicamente
- Aprender a usar la tecnología AJAX

Hacia quien está Orientado el curso

Desarrolladores y programadores que deseen profundizar en herramientas basadas en PHP a partir del paradigma POO.

Prerrequisitos del curso

- Conocimientos básicos de programación orientada objetos.
- Haber realizado el Nivel 1 del curso de php

Acerca de este manual

Este manual pertenece al Centro de Asesoramiento y Desarrollo Informático C.A. (CADI F1). Para obtener más información sobre este u otros cursos visite nuestra sitio Web www.cadif1.com, escribanos a la dirección de correo cadi@cadif1.com o visítenos en nuestra sede ubicada en la Av. Pedro León Torres con calle 59, Centro Comercial Sotavento, piso 2 oficina 27, Barquisimeto estado Lara, Venezuela. Tlf. 0251-7179247, 0251-4410268.

Las marcas mencionadas en este manual son propiedad de sus respectivos dueños. Copyright 2016. Todos los derechos reservados.



Contenido del curso

Capítulo 1. Funciones

- 1.1.- Definición de Funciones en php.
- 1.2.- Llamar una Función.
- 1.3.- Paso de Parámetros a las Funciones.

Capítulo 2. Librerías

- 2.1.- Creación de Librerías.
- 2.2.- Incluir Librerías en Páginas.

Capítulo 3. Manejo de Fechas

- 3.1.- Zonas Horarias.
- 3.2.- La Función Getdate().
- 3.3.- La Función Date.
- 3.4.- Time.

Capítulo 4. Programación Orientada a Objetos.

- 4.1.- Concepto.
- 4.2.- Clases y Objetos.
- 4.3.- Atributos y Métodos.

Capítulo 5. Declaración de Clases en Php

- 5.1.- Definición de una Clase en php.
- 5.2.- Creación y uso Objetos.
- 5.3.- Niveles de Acceso.

Capítulo 6. Métodos. Parte 1

- 6.1.- Definición de Métodos.
- 6.2.- Llamado de Los Métodos.
- 6.3.- Acceso a Los Atributos.

Capítulo 7. Métodos. Parte 2

- 7.1.- Getter y Setter.
- 7.2.- Constructores.
- 7.3.- Métodos Estáticos.

Capítulo 8. Manejo de Sesiones

- 8.1.- Concepto de una Sesión.
- 8.2.- Crear una Variable de Sesión.
- 8.3.- Destruir una Variable de Sesión.

Capítulo 9. Herencia

- 9.1.- Herencia.
- 9.2.- Funciones Relacionadas.

Capítulo 10. Generación de Archivos Pdf. Parte 1

- 10.1.- ¿Qué es FPDF?.
- 10.2.- Creación de un archivo PDF.
- 10.3.- Agregar Contenido a un Documento.

Capítulo 11. Generación de Archivos Pdf. Parte 2

- 11.1.- Agregar Imágenes.
- 11.2.- Establecer pie y Cabecera de Pagina.

Capítulo 12. Ajax

- 12.1.- ¿Qué es AJAX?.
- 12.2.- El objeto XMLHttpRequest.
- 12.3.- Usos y Aplicaciones.
- 12.4.- Ventajas y Desventajas.

Capítulo 13. Ajax Con Php. Parte 1

- 13.1.- XAjax.
- 13.2.- La Clase Xajax.
- 13.3.- Llamado Asíncrono de Funciones.

Capítulo 1. FUNCIONES

1.1.- Definición de Funciones en php

Una función se puede definir con la siguiente sintaxis:

```
<?php
    function foo ($arg_1, $arg_2, ..., $arg_n)
    {
        echo "Función de ejemplo.";
        return $retval;
    }
?>
```

El tipo de dato devuelto por la función dependerá de la variable que se coloque en la cláusula return. Cualquier instrucción válida de PHP puede aparecer en el cuerpo de la función, incluso invocar otras funciones y definiciones de clases. La función puede tener cualquier cantidad de parámetros, en caso de no tener parámetros igual se deben colocar los paréntesis.

Dentro del cuerpo de la función se pueden declarar variables, que solo existirán mientras se esté ejecutando la función. Estas variables se declaran igual que cualquier variable en PHP (inicializándola con un valor cualquiera). Por ejemplo:

```
<?php

    function esMayor($edad)
    {
        if ($edad >=18) return true;
        else return false;
    }

?>
```

También se puede hacer referencia a variables globales en la función, pero se debe "declarar" la variable usando la palabra reservada "global", de lo contrario se tomará como una variable local. Por ejemplo:

```
<?php

$dias=array("lunes","martes","miercoles","jueves",
            "viernes","sabado","domingo");

// si no hacemos la definición de la variable $dias con la palabra global,
// ocurrirá un error como si la variable no existiera.

function mostrarDias()
{
    global $dias; // aqui estamos diciendo que vamos a usar la var global

    for ($i=0;$i<count($dias);$i++)
        echo $dias[$i]. " \n";
}
```

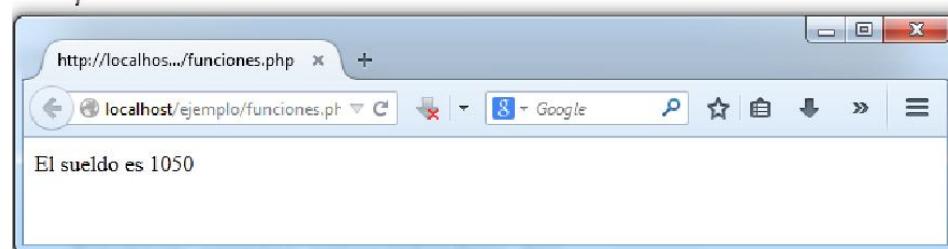
1.2.- Llamar una Función

Las funciones pueden llamarse en cualquier lugar del código PHP, incluso antes de definirse. Se llaman colocando el nombre de la función y entre paréntesis los parámetros que ella espera. Al llamar una función no existe sensibilidad a mayúsculas y minúsculas. Por ejemplo:

```
<?php

echo "El sueldo es " . calcularSueldo(750, 15);

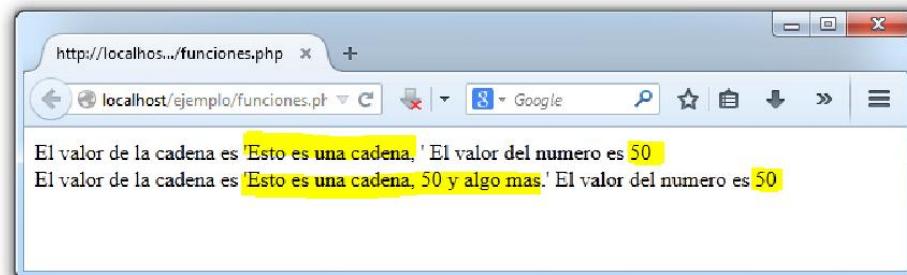
function calcularSueldo($sueldobase, $antiguedad )
{
    return $sueldobase + ($antiguedad * 20 );
}
```



1.3.- Paso de Parámetros a las Funciones

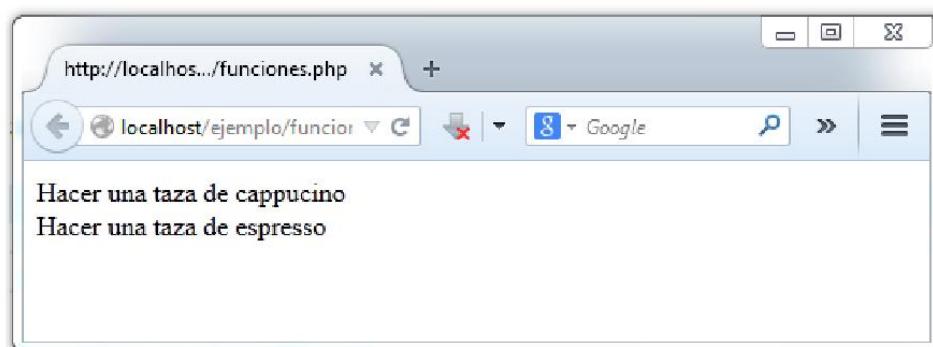
PHP soporta pasar parámetros por valor (el comportamiento por defecto), por referencia y parámetros por defecto. Mientras no se especifique lo contrario, los parámetros de una función se pasan por valor. Si se necesita que un parámetro de una función se pase por referencia, se antepone el ampersand (&) al nombre del parámetro en la definición de la función:

```
<?php
    function imprimir($str,$n)
    {
        echo "El valor de la cadena es '$str' ";
        echo "El valor del numero es $n <br>";
    }
    function agregar_extra(&$cadena,$n)
    {
        $cadena = $cadena . $n ." y algo mas.";
        $n=0;
    }
    $str = "Esto es una cadena, ";
    $n=50;
    imprimir($str,$n);
    agregar_extra($str,$n);
    imprimir($str,$n);
```



Una función puede definir valores por defecto para los parámetros, por ejemplo:

```
<?php  
  
    function makecoffee ($type = "cappuccino")  
    {  
        return "Hacer una taza de $type <br>";  
    }  
  
    echo makecoffee();  
    echo makecoffee("espresso");
```



El valor por defecto tiene que ser una expresión constante y no una variable ó llamada a una función. Cabe destacar que cuando se usan parámetros por defecto, estos tienen que estar a la derecha de cualquier parámetro sin valor por defecto.

Capítulo 2. LIBRERÍAS

2.1.- Creación de Librerías

PHP da la posibilidad de crear un archivo que contenga funciones generales definidas para poder usarlas en cualquier otro archivo, dándole así la posibilidad de reutilizar código. Para esto primero se debe crear un archivo de PHP donde se colocan todas las funciones y declaraciones. Por ejemplo:

```
<?php  
function mensajedeerror($mensaje)  
{  
    echo $mensaje;  
}  
?>
```

Puede definir todas las funciones, variables y/o constantes que necesiten en este archivo, sin importar el orden en el cual se coloquen. Se pueden tener tantos archivos php que sean usados como librerías como hagan falta.

2.2.- Incluir Librerías en Páginas

Para usar las funciones desde otro archivo php, variables o cualquier otra declaración, se debe incluir el archivo donde están declaradas. Esto se hace con la función “require”, “require_once”, “include” o “include_once”. El siguiente es un ejemplo:

```
<?php libreria.php  
  
function calcularSueldo($sueldobase, $antiguedad )  
{  
    return $sueldobase + ($antiguedad * 20 );  
}  
  
?>
```

```
<?php cualquierOtroArchivo.php  
  
// producira un error  
echo "El sueldo es " . calcularSueldo(750, 15);  
include("libreria.php");  
  
// ahora si funcionara  
echo "El sueldo es " . calcularSueldo(750, 15);  
  
?>
```



ACADEMIA DE SOFTWARE

Capítulo 3. MANEJO DE FECHAS

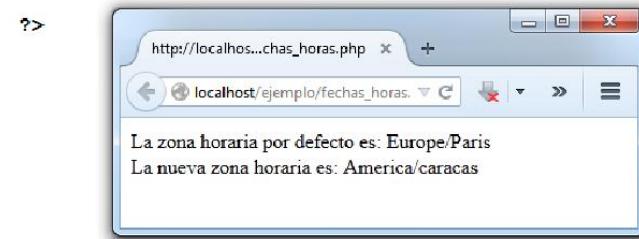
3.1.- Zonas Horarias

Antes de trabajar con las funciones de fecha y hora debemos establecer la zona horaria que necesitamos usar. Si no establecemos la zona horaria correcta, los valores de fecha y hora obtenidos pueden ser diferentes a los que esperamos. La zona horaria la establecemos con la función `date_default_timezone_set ("zona_horaria")`. La zona horaria en nuestro país (Venezuela) es America/Caracas. Para no tener que colocar esta función antes de cualquier función de fecha y hora, podemos establecer en el `php.ini` el valor `date.timezone = mizona`.

La lista completa de los posibles valores de zona horaria la podemos encontrar en la url

<http://php.net/manual/es/timezones.php>. Con la función `date_default_timezone_get ()` podemos determinar la zona horaria que se está usando. Veamos un ejemplo:

```
<?php  
  
echo "La zona horaria por defecto es: ".date_default_timezone_get()."  
date_default_timezone_set("America/caracas");  
echo "La nueva zona horaria es: ".date_default_timezone_get();
```



3.2.- La Función Getdate()

Devuelve un valor array asociativo que contiene información sobre la fecha y la hora local actual. Los valores que devuelve el `getdate` son:

Clave	Descripción	Ejemplo de valores devueltos
"seconds"	Representación numérica de segundos	0 a 59
"minutes"	Representación numérica de minutos	0 a 59
"hours"	Representación numérica de horas	0 a 23
"mday"	Representación numérica del día del mes	1 a 31
"wday"	Representación numérica del día de la semana	0 (para el Domingo) a 6 (para el Sábado)
"mon"	Representación numérica de un mes	1 a 12
"year"	Una representación numérica completa de un año, 4 dígitos	Ejemplos: 1999 o 2003
"yday"	Representación numérica del día del año	0 a 365
"weekday"	Una representación textual completa del día de la semana	Sunday a Saturday
"month"	Una representación textual completa de un mes, como January o March	January a December
0	Segundos desde el Epoch Unix, similar a los valores devueltos por time() y usados por date() .	Depende del sistema, típicamente -2147483648 a 2147483647.

Un ejemplo de como usar el getdate es:



```
<?php
date default timezone set("America/caracas");
$fecha=getdate();
echo "Hoy es ".$fecha['mday'];
if ($fecha['mday']<10) $descuento=0.05;
else
    if ($fecha['mday']<20) $descuento=0.07;
    else $descuento=0.1;
echo ".El descuento es $descuento";
?>
```

La función getdate también devuelve el día de la semana en número o en palabra, con los valores "weekday" y "wday". El día de la semana en palabras es retornado en inglés. Una manera de solventar esto podría ser:

```
<?php
date_default_timezone_set("America/caracas");
$dias=array("domingo","Lunes","Martes","Miercoles","Jueves","Viernes","Sabado");
$fecha=getdate();
echo "Hoy es ".$fecha['weekday']."' (en ingles) <br>";
echo "Hoy es ".$fecha['wday']."' (en numero) <br>";
echo "Hoy es ".$dias[$fecha['wday']]."' (en castellano) ";
?>
```



Es importante recordar que el código php se ejecuta en el servidor y por lo tanto la fecha y la hora que se consigue es la del servidor. Este aspecto se debe tomar en cuenta cuando se está desarrollando una aplicación donde alguna funcionalidad crítica se base en la fecha y la hora.

En muchos casos la zona horaria del servidor no va a ser la misma que la del cliente. Aunque casi nunca interesa la hora del cliente, hay manera de averiguarla. La forma de conocer la fecha y la hora del cliente es usando la función Date de javascript.

3.3.- La Función Date

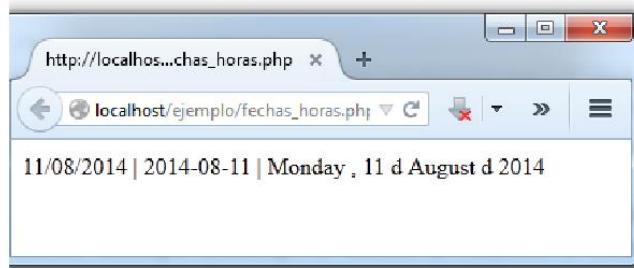
La función Date devuelve un string con la fecha o la hora local con un formato especificado. La forma de usar el date es \$fecha=date("formato"). Algunos ejemplos de como usar la función date son:

<?php

```
$fechahoy=date("d/m/Y"); // fecha de hoy en formato dd/mm/yyyy
$fechaMysql=date("Y-m-d"); // fecha de hoy en el formato de mysql

// fecha larga en el formato Lunes, 2 de Noviembre de 2008
$fechaLarga=date("l \, d \d\le F \d\le Y");

echo "$fechahoy | $fechaMysql | $fechaLarga";
```



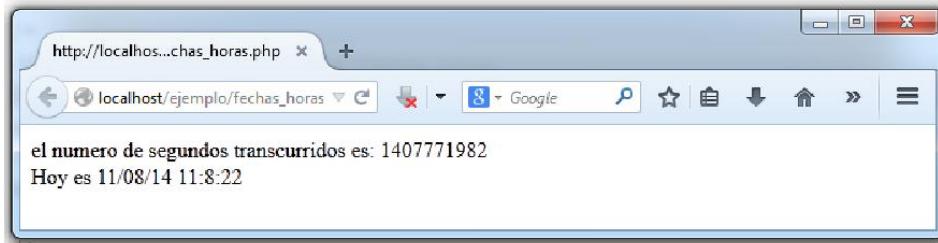
La cadena con el formato puede contener cualquiera de los siguientes caracteres:

<i>d</i>	Día del mes, 2 dígitos con ceros iniciales	01 a 31
<i>D</i>	Una representación textual de un día, tres letras	Mon a Sun
<i>j</i>	Día del mes sin ceros iniciales	1 a 31
<i>\(L\)</i> minúscula	Una representación textual completa del día de la semana	Sunday a Saturday
<i>w</i>	Representación numérica del día de la semana	0 (para el Domingo) a 6 (para el Sábado)
<i>z</i>	El día del año (comenzando en 0)	0 a 365
<i>F</i>	Una representación textual completa de un mes, como January o March	January a December
<i>m</i>	Representación numérica de un mes, con ceros iniciales	01 a 12
<i>M</i>	Una representación textual corta de un mes, tres letras	Jan a Dec
<i>n</i>	Representación numérica de un mes, sin ceros iniciales	1 a 12
<i>t</i>	Número de días en el mes dado	28 a 31
<i>Y</i>	Una representación numérica completa de un año, 4 dígitos	Ejemplos: 1999 o 2003
<i>y</i>	Una representación de dos dígitos de un año	Ejemplos: 99 o 03
<i>a</i>	Ante meridiano y Post meridiano en minúsculas	am o pm
<i>g</i>	formato de 12-horas de una hora sin ceros iniciales	1 a 12
<i>G</i>	formato de 24-horas de una hora sin ceros iniciales	0 a 23
<i>h</i>	formato de 12-horas de una hora con ceros iniciales	01 a 12
<i>H</i>	formato de 24-horas de una hora con ceros iniciales	00 a 23
<i>i</i>	Minutos con ceros iniciales	00 a 59
<i>s</i>	Segundos, con ceros iniciales	00 a 59

3.4.- Time

Otra forma de obtener valores del tiempo es con la función `time()`. Ésta devuelve un número entero que representa el número de segundos que han transcurrido desde la fecha 1 de Enero de 1970 00:00:00 (GMT), conocida también como la época Unix. Este valor es especialmente útil para hacer operaciones con fechas y horas, tales como compararlas, sumarle valores como días, segundos, minutos, calcular la diferencia entre fechas, etc. Veamos un ejemplo sencillo:

```
<?php  
// se establece la zona horaria  
date_default_timezone_set("America/caracas");  
// se determina la/hora en segundos  
$fechahora = time();  
echo "el numero de segundos transcurridos es: $fechahora <br>";  
// se imprime la fecha y hora de hoy con formato entendible  
echo "Hoy es ".date("d/m/y h:n:s")."<br>";
```

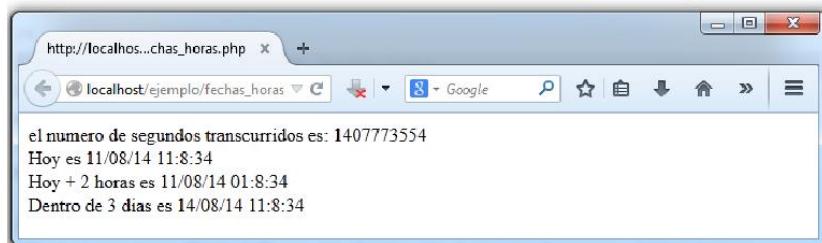


Al valor returnedo por la función `time()` se le puede hacer operaciones aritméticas. Por ejemplo sumarle 2 horas sería algo así como `time() + 2 * 60 * 60`, porque cada hora tiene 60 minutos y cada minuto tiene 60 segundos. El valor en segundos se puede mostrar en su correspondiente formato de fecha normal (d/m/y) u hora (h:m:s) o ambos inclusive, con la función "date", a la cual se le puede pasar el formato y el valor de fecha Unix. Veamos unos ejemplos:

```
<?php
// se establece la zona horaria
date_default_timezone_set("America/caracas");
// se determina la/hora en segundos
$fechahora = time();
echo "el numero de segundos transcurridos es: $fechahora <br>";
// se imprime la fecha y hora de hoy con formato entendible
echo "Hoy es ".date("d/m/y h:n:s")."<br>";

// se imprime la fecha y hora de hoy + 2 horas
echo "Hoy + 2 horas es ".date("d/m/y h:n:s",$fechahora+(2*60*60))."<br>";

// se imprime la fecha y hora de hoy + 3 dias
echo "Dentro de 3 dias es ".date("d/m/y h:n:s",$fechahora+(3*24*60*60))."<br>";
```



Capítulo 4. PROGRAMACIÓN ORIENTADA A OBJETOS.

4.1.- Concepto

La programación orientada a objetos expresa un programa como un conjunto de objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar. De esta forma, un objeto contiene toda la información que permite definirlo e identificarlo frente a otros objetos.

Esto difiere de la programación estructurada tradicional, en la que los datos y los procedimientos están separados y sin relación, ya que lo único que se busca es el procesamiento de unos datos de entrada para obtener otros de salida. La programación estructurada anima al programador a pensar sobre todo en términos de procedimientos o funciones, y en segundo lugar en las estructuras de datos que esos procedimientos manejan.

En la programación estructurada se escriben funciones y después les pasan datos. Los programadores que emplean lenguajes orientados a objetos definen objetos con datos y métodos y después envían mensajes a los objetos diciendo que realicen esos métodos en sí mismos.

Hay un cierto desacuerdo sobre exactamente qué características de un método de programación o lenguaje le definen como "orientado a objetos", pero hay un consenso general en que las características siguientes son las más importantes:

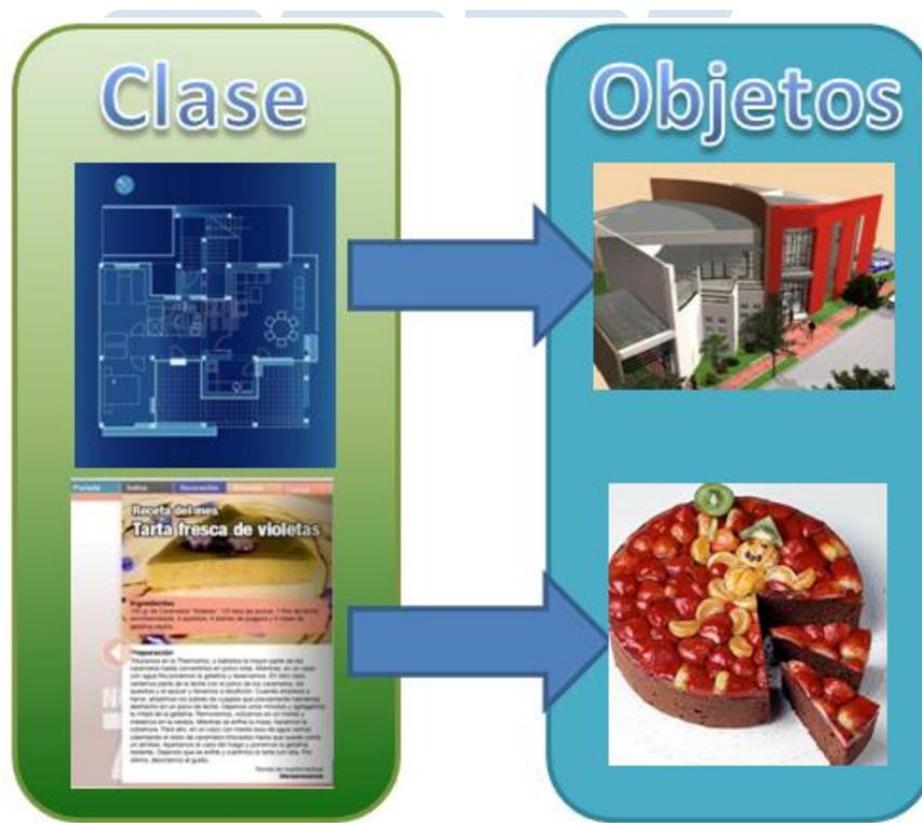
- Encapsulamiento.
- Principio de ocultación.
- Polimorfismo.
- Herencia.

Entre los lenguajes orientados a objetos destacan los siguientes: C++ ,C# ,VB.NET ,Visual FoxPro ,Builder C++ ,Delphi ,Java ,Perl (soporta herencia múltiple) ,PHP (en su versión 5) ,PowerBuilder ,Python ,Ruby , entre otros.

Muchos de estos lenguajes de programación no son puramente orientados a objetos, sino que son híbridos que combinan la OOP con otros paradigmas.

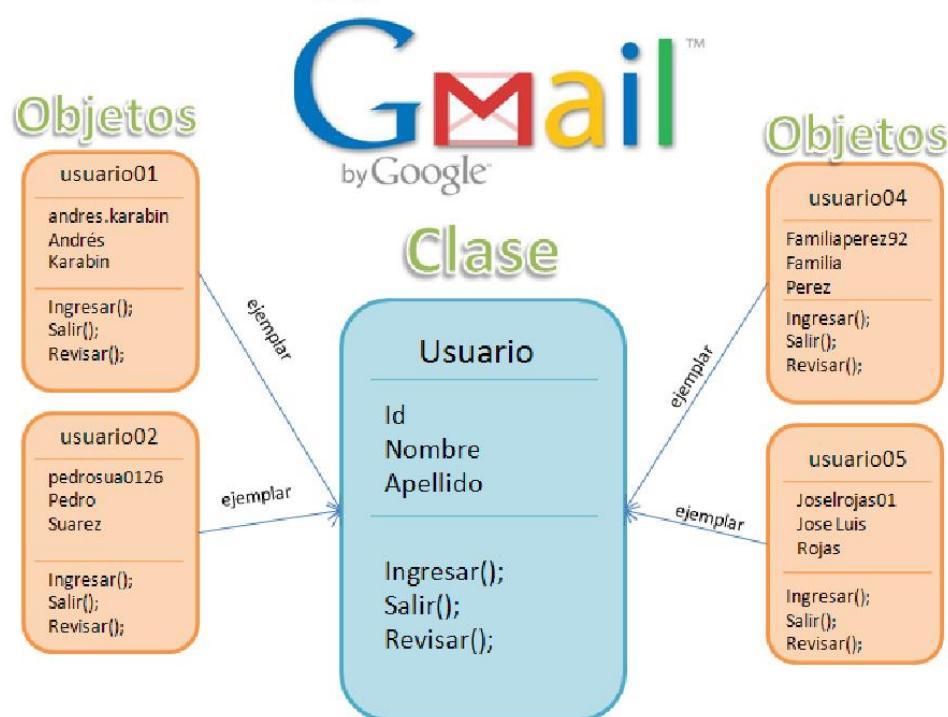
4.2.- Clases y Objetos

En la POO aparecen constantemente dos entes principales: Clases y Objetos. La relación entre éstos, dicho de manera sencilla e intuitiva, es la misma que puede haber entre un plano dibujado por un arquitecto, y la casa o casas construidas a partir de este plano. La receta creada por un cocinero y los platos de comida que han sido cocinados en base a la receta, en programación pudiésemos hablar del código de un programa en particular y el proceso que está ejecutándose en el sistema con el contenido de ese código, podemos crear o ejecutar tantos procesos como queramos a partir del mismo archivo.



4.3.- Atributos y Métodos

Existen variables contenidas dentro de una clase que reciben el nombre de "atributos" y las funciones de estos objetos se denominan "métodos". Los atributos existen porque contienen información de estado de los objetos y los métodos realizan modificaciones sobre ese estado, definiendo así el comportamiento del objeto.



Capítulo 5. DECLARACIÓN DE CLASES EN PHP

5.1.- Definición de una Clase en php

A partir de PHP5 se cubren las carencias de las versiones anteriores, proveyendo un soporte completo para la Programación Orientada a Objetos (POO). Este paradigma se logra gracias al nuevo motor (Zend Engine 2) que proporciona todos los mecanismos necesarios para este propósito, de esta manera PHP puede ser empleado desde dos perspectivas distintas, la Programación Procedural (abordada durante todo el contenido de los niveles anteriores del curso) y Orientada a Objetos (el objetivo de este nivel).



Las clases en php se definen usando la palabra reservada `class`, seguido del nombre de la clase y un par de llaves que indican el ámbito de la clase. Dentro de las llaves, se declaran los atributos y los métodos. Los atributos se declaran usando la palabra reservada `"var"`. Estos atributos no tienen tipo hasta que se les asigna un valor. Un atributo puede ser un entero, un arreglo, un arreglo asociativo o incluso otro objeto. El siguiente es un ejemplo de la declaración de una clase:

```

1  <?php
2      class Usuario
3  {
4      var $nombre;
5      var $contraseña;
6      var $correo;
7
8
9  }
10
11 ?>
```

5.2.- Creación y uso Objetos

Para instanciar una clase (dicho de otra manera, para crear un objeto de una clase), se usa el operador `"new"`. Luego, a través del objeto se hace referencia a cualquier miembro de la clase (propiedad o método), usando el operador `"->"`. El siguiente es un ejemplo:

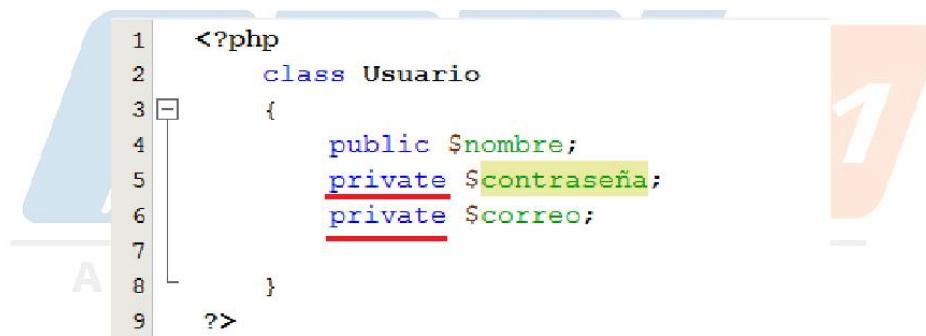
```

1  <?php
2      class Usuario
3  {
4      var $nombre;
5      var $contraseña;
6      var $correo;
7
8  }
9
10 // causa un error porque el objeto no ha sido instanciado
11 $usu_1->nombre="Jose luis";
12 echo "el nombre del usuario 1 es ".$usu_1->nombre;
13
14 // se instancia el objeto
15 $usu_2 = new Usuario();
16 // se accede a sus atributos
17 $usu_2->nombre="Manuel";
18 echo "el nombre del usuario 2 es ".$usu_2->nombre;
19 ?>
```

5.3.- Niveles de Acceso

Hasta ahora, todas las propiedades o atributos que se han usado son "públicos" (en php los atributos son públicos mientras que no se indique lo contrario). Un atributo público es aquel que puede ser accedido desde fuera del objeto, simplemente escribiendo algo como: \$objeto->atributo.

Tener atributos públicos, sin embargo, no es la práctica más recomendable, porque si desde fuera del objeto se pueden modificar sus propiedades, se puede llegar a alterar el propósito o la funcionalidad del mismo. Por esto, para tener el control del uso de las propiedades del objeto, es necesario de indicar que un determinado atributo es interno al objeto, de su uso exclusivo, o dicho de otra forma, privado del objeto. Por tanto, solo accesible y/o modificable desde los métodos internos de la clase.



```
1 <?php
2     class Usuario
3     {
4         public $nombre;
5         private $contraseña;
6         private $correo;
7     }
8
9 ?>
```

Para hacer que un atributo sea privado, se debe declarar usando la palabra reservada "private". Así, cualquier referencia del tipo \$objeto->atributo provocará un error del intérprete de PHP. Es decir, cualquier intento de acceder externamente a un atributo privado no estará permitido.

```
1 <?php
2     class Usuario
3     {
4         public $nombre;
5         private $contraseña;
6         private $correo;
7     }
8
9
10    $usu_1 = new Usuario();
11    $usu_1->nombre="Jose luis";           // es valido
12    $usu_1->correo="joselrojas@cadif1.com"; // es invalido
13    $usu_1->contraseña="45678";          // es invalido
14
15 ?>
```

Como se puede ver en el ejemplo, los intentos de actualización de las propiedades "usuario" y "contraseña" son inválidos. La única forma de modificar y/o acceder a estos atributos es a través de métodos públicos. La práctica más recomendable es que todas las propiedades se declaran privadas y se accede a ellas de manera controlada mediante métodos denominados 'setters' y 'getters', los cuales, respectivamente, modifican y devuelven sus contenidos. Esto se tratará en el próximo capítulo.

ACADEMIA DE SOFTWARE

Capítulo 6. MÉTODOS. PARTE 1

6.1.- Definición de Métodos

Los métodos se definen dentro de las llaves de la clase usando la palabra reservada function. El siguiente es un ejemplo sencillo:

```

1 <?php
2     class Usuario
3     {
4         public $nombre;
5         private $contraseña;
6         private $correo;
7
8         function mostrar_bienvenida()
9         {
10            echo "Bienvenido al sistema";
11        }
12    }
13 ?>
```

Los métodos pueden tener tantos parámetros como haga falta. Las reglas para usar parámetros en los métodos son las mismas que aplican a funciones tradicionales:

```

1 <?php
2     class Usuario
3     {
4         public $nombre;
5         private $contraseña;
6         private $correo;
7
8         function mostrar_bienvenida()
9         {
10            echo "Bienvenido al sistema";
11        }
12        function agregar_lista($n)
13        {
14            echo "<ul>";
15            for ($i=1;$i<$n;$i++)
16            {
17                echo "<li> $i </li>";
18            }
19        }
20 ?>
```

6.2.- Llamado de Los Métodos

El llamado de los métodos se hace de la misma forma que cómo se accede a los atributos públicos. Se instancia el objeto y se accede al nombre del método con el operador "->":

```

1  <?php
2      class Usuario
3      {
4          public $nombre;
5          private $contraseña;
6          private $correo;
7
8          function mostrar_bienvenida()
9          {
10             echo "Bienvenido al sistema";
11         }
12         function agregar_lista($n)
13         {
14             echo "<ul>";
15             for ($i=1;$i<$n;$i++)
16                 echo "<li> $i </li>";
17             echo "</ul>";
18         }
19     }
20     $usu_1 = new Usuario();
21     $usu_1->mostrar_bienvenida();
22     $usu_1->agregar_lista(5);
23 ?>
```

6.3.- Acceso a Los Atributos

Si desde un método se necesita hacer referencia a un atributo (propiedad) de la clase, debemos hacerlo con la variable implícita \$this, que hace referencia al objeto que está ejecutando el método en el momento, y con el operador "->" para acceder a la respectiva propiedad. Si se hace referencia al atributo sin la palabra \$this, se asume que es una variable local al método. El siguiente es un ejemplo:

```
1 <?php
2     class Usuario
3     {
4         public $nombre;
5         private $contraseña;
6         private $correo;
7
8         function mostrar_bienvenida()
9         {
10            echo "Bienvenido al sistema " . $this->nombre;
11        }
12        function agregar_lista($n)
13        {
14            echo "<ul>";
15            for ($i=1;$i<$n;$i++)
16            {
17                echo "<li> Si </li>";
18            }
19        }
20        $usu_1 = new Usuario();
21        $usu_1->mostrar_bienvenida();
22        $usu_1->agregar_lista(5);
23    ?>
```



ACADEMIA DE SOFTWARE

Capítulo 7. MÉTODOS. PARTE 2

7.1.- Getter y Setter

Cuando los atributos de una clase son privados (que es la práctica más recomendable), debe existir una forma de conocer sus valores y de modificarlos. A estos métodos particulares se les denomina getters y setters. Sus nombres provienen de las palabras "get" (obtener) y "set" (establecer). Estos métodos deben ser públicos. El nombre particular generalmente esta compuesto del verbo (get o set) seguido del nombre del atributo. Por ejemplo:

```
1 <?php
2     class Usuario
3     {
4         public $nombre;
5         private $contraseña;
6         private $correo;
7
8         function getCorreo()
9         {
10            return $this->correo;
11        }
12        function setCorreo($correo)
13        {
14            $this->correo=$correo;
15        }
16    }
```

La forma de usarlos es la siguiente:

```
1  <?php
2      class Usuario
3  {
4          public $nombre;
5          private $contraseña;
6          private $correo;
7
8          function getCorreo()
9  {
10             return $this->correo;
11         }
12         function setCorreo($correo)
13  {
14             $this->correo=$correo;
15         }
16     }
17     $usu_1 = new Usuario();
18     $usu_1->setCorreo("joselrojas@cadif1.com");
19     echo "El correo es ".$usu_1->getCorreo();
20 ?>
```

7.2.- Constructores

Al definir los atributos de las clases, es posible que existan algunos de ellos que deban ser inicializados en ciertos valores específicos cuando se instancia un objeto. Un constructor es un método que se ejecuta automáticamente al instanciar una clase (crear un objeto), y su utilidad es la de inicializar los atributos que necesiten estar inicializados antes de usar el objeto. Hay dos formas de indicar que un método es el constructor de la clase: colocándole el nombre "__construct" o colocándole el mismo nombre de la clase.

```
1 <?php
2     class Usuario
3     {
4         public $nombre;
5         private $contraseña;
6         private $correo;
7         private $fecha;
8
9         function Usuario()
10        {
11             $this->fecha=date("d-m-y");
12             $this->nombre="sin nombre";
13         }
14         function imprimir_Datos()
15         {
16             echo "El usuario ".$this->nombre.
17                 " Tiene fecha ".$this->fecha;
18         }
19     }
20     $usu_1 = new Usuario();
21     $usu_1->imprimir_Datos();
22 ?>
```

7.3.- Métodos Estáticos

ACADEMIA DE SOFTWARE

Existen un tipo de métodos que se pueden crear dentro de una clase, con la particularidad de que pueden ser llamados sin necesidad de instanciar un objeto de la clase. A estos métodos se les llama métodos estáticos. El siguiente es un ejemplo:

```
<?php

class Prueba
{
    static function saludar()
    {
        echo "Hola que tal !";
    }
}

// se puede llamar directo con la clas
Prueba::saludar();

// se puede llamar a traves de un objeto
$objeto=new Prueba();
$objeto->saludar();

?>
```

Desde un método estático solo se pueden usar otros métodos estáticos, al tratar de usar métodos de instancia se producirá un error de ejecución:

ACADEMIA DE SOFTWARE

```
<?php

class Prueba
{
    static function saludar()
    {
        echo "Hola que tal !";
        $this->chao(); // prohibido
    }
    function chao()
    {
        echo "chao";
    }
}

// se puede llamar directo con la clas
Prueba::saludar();

// se puede llamar a traves de un objeto
$objeto=new Prueba();
$objeto->saludar();
```

A continuación se verán más ejemplos.

Capítulo 8. MANEJO DE SESIONES

8.1.- Concepto de una Sesión

Básicamente una sesión es la secuencia de páginas que un usuario visita en un sitio web. Desde que entra en nuestro sitio (colocando en la barra de direcciones del navegador el nombre de una página de nuestro sitio), hasta que lo abandona (cerrando la ventana del navegador).

El término sesión en PHP, se aplica a esta secuencia de navegación. Para ello se crea un identificador único que asignamos a cada una de estas sesiones de navegación. A este identificador de sesión se le denomina, comúnmente "la sesión".

El proceso en cualquier lenguaje de programación podría ser algo así: ¿Existe una sesión?, si existe lo retomamos, si no existe la creamos y le damos un identificador único. En resumen, una sesión va a ser una o varias variables, cuyo valor(es) no se perderá(n) cuando pasamos de una página a otra.

8.2.- Crear una Variable de Sesión

Antes de poder hacer referencia o crear una variable de sesión, se debe inicializar la sesión. Para lograrlo se utiliza la función "session_start" antes de intentar utilizar cualquier variable de sesión. Esta función inicializa el manejo interno de las sesiones en PHP, de no invocarlo nada que tenga que ver con las sesiones funcionará.

Existen dos maneras de referirse a las variables de sesión, una con la variable global `$_SESSION` y otra con la variable global `$HTTP_SESSION_VARS`. Ambas son arreglos asociativos. Se recomienda usar `$_SESSION` (o `$HTTP_SESSION_VARS` con PHP 4.0.6 o inferior) por seguridad y para hacer el código más legible.

Las opciones de configuración `track_vars` y `register_globals` influyen notablemente en la forma en que las variables de la sesión se almacenan y restauran.

Por ejemplo, si queremos contar la cantidad de páginas que visita un usuario en nuestro sitio debemos colocar al comienzo de cada una de las páginas lo siguiente:

```
<?php  
    session_start();  
    if (isset( $_SESSION["visitas"])) $_SESSION["visitas"]++;  
    else $_SESSION["visitas"]=0;  
?>
```

Se utiliza la función `isset` para verificar si la variable existe, de no existir se crea inicializándola con un valor arbitrario (como se hace con cualquier otra variable de PHP). La siguiente vez que el usuario visite la página (u otra página), la variable va a existir y se va a incrementar el valor que ella trae. El nombre que se le asigne a la variable (en el ejemplo anterior "visitas") es arbitrario también.

8.3.- Destruir una Variable de Sesión

El valor de una variable de sesión se pierde o desaparece cuando liberamos explícitamente la variable o cuando el usuario cierra la ventana del navegador donde inició la visita a nuestro sitio. Destruir una variable de sesión significa eliminar una variable que se haya creado anteriormente. Esto es útil en sistemas como los de correo en los cuales hay un botón donde el usuario decide salir del sistema sin necesidad de cerrar la ventana del navegador, para así evitar el acceso a las páginas privadas.

La destrucción de las variables de sesión se hace con el procedimiento `unset`, al cual se le pasa por parámetro el nombre de la variable que se quiere destruir. Por ejemplo:

```
<?php  
    session_start();  
    unset($_SESSION["visitas"]);  
?>
```

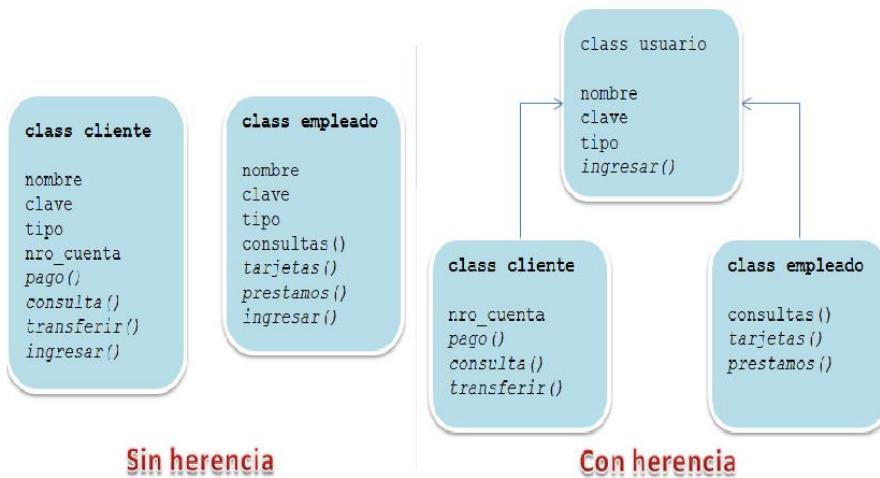
Capítulo 9. HERENCIA

9.1.- Herencia

La herencia en la POO es la relación que se da entre dos clases por la cual una de ellas (a la que llamaremos subclase o clase derivada), además de tener sus propiedades y métodos, tiene a su disposición (hereda) los miembros definidos en la clase padre o superclase.

El propósito final de la herencia es la reutilización de código, si queremos implementar un nuevo tipo de archivo (por ejemplo un PDF) podríamos aprovechar el código de la clase que implementa las características comunes y ahorrarnos trabajo.

Por ejemplo, un sistema web bancario, una sitio web mediante el cual los clientes de esta institución pueden realizar diversas operaciones (pagos, consultas y transferencias). Adicionalmente sabemos que el personal que trabaja en la institución hace uso del sistema para otras operaciones que no son permitidas a los clientes (consultar cuentas, consulta datos personales de los afiliados, habilitar tarjetas de crédito, entre otras). Pero para ambos casos sabemos que existe información común y de la misma naturaleza. En ambos ejemplos el ‘usuario’ tiene un nombre, una contraseña y por supuesto debe existir algún identificador que nos haga diferenciar un usuario de tipo cliente a un usuario de tipo empleado, porque es de esta manera como el sistema decidirá qué funcionalidades debe habilitar.



En PHP 5, para que una clase pueda heredar las características de otra (convirtiéndose en subclase suya) tiene que usar, en su declaración, la palabra reservada extends seguida del identificador de la clase que quiere obtener sus características.

```
<?
class Fichero_PNG extends Fichero {
    private $alto;
    private $ancho;
    private $bits_por_color;

    function bits_por_color() {
        return $this->bits_por_color;
    }

    function alto() {
        return $this->alto;
    }

    function ancho() {
        return $this->ancho;
    }
}

$obj_png = new Fichero_PNG();
echo "La cantidad de bits de la imagen es:" .
    $obj_png->bits_por_color() . "bits<br />";
echo "Las dimensiones del gráfico son: " .
    $obj_png->alto() . "x" . $obj_png->ancho() .
    "<br />";
```

9.2.- Funciones Relacionadas

En este segmento haremos una breve descripción de las funciones que guardan relación con las clases y objetos:

- `class_exists(clase)`: Si clase está definida devuelve TRUE, en caso contrario devuelve FALSE.
- `get_declared_classes()`: Devuelve un arreglo numérico con las clases que están disponibles al momento de ejecutar la función, las nativas de PHP y las definidas por el usuario.
- `Get_class(obj)`: Para el objeto \$obj, devuelve el nombre de la clase de la que es una instancia.
- `Get_parent_class(obj_o_clase)`: Devuelve el nombre de la superclase de la que está derivado el objeto o la clase pasada como parámetro.
- `Get_class_vars(clase)` o `Get_object_vars(obj)`: Devuelve un arreglo asociativo con las propiedades (públicas) de que dispone la clase o el objeto respectivamente.
- `Get_class_methods(clase)`: Devuelve un arreglo asociativo numérico con el nombre de los métodos que dispone la clase.
- Operador `instanceof`: Si la variable \$obj es un ejemplar de la clase nombre_clase, devuelve TRUE, en caso contrario devuelve FALSE. Por ejemplo: `$obj instanceof nombre_clase`.
- Constante `__METHOD__`: Cuando se usa desde dentro de un método devuelve el nombre de la clase y el método desde el que se está ejecutando, si se usa desde una función devuelve su nombre.

Capítulo 10. GENERACIÓN DE ARCHIVOS PDF. PARTE 1

10.1.- ¿Qué es FPDF?

Nos resulta bastante frecuente encontrar documentos en formato PDF (Portable Document Format) como resultado de alguna operación sobre un sitio web en particular. El objetivo de este tipo de recursos es generar información que no pueda ser editada por el usuario, que pueda ser trasladada con facilidad y que a su vez pueda tomar datos dinámicamente de distintas fuentes sin afectar lo primero.

Para php existen varias opciones a la hora de querer generar contenido en formato pdf, que luego pueda ser descargado por el usuario a su computador. Entre ellos están: HTML2PDF, fpdf, tcpdf, entre otros.



En el curso vamos a usar la librería fpdf, que está desarrollada con el paradigma de la programación orientada a objetos y además es bastante simple de utilizar. Para usar esta librería debemos descargar un archivo comprimido de la página oficial y luego descomprimirlo en la carpeta raíz

de nuestro proyecto. Al descomprimir vamos a encontrar un conjuntos de archivos y directorios necesarios para utilizar la librería, entre ellos el archivo fpdf.php, la carpeta font (contiene las fuentes usadas) y la documentación.



10.2.- Creación de un archivo PDF.

Para crear un archivo pdf con la librería fpdf lo primero que debemos hacer es incluir el archivo fpdf.php. Luego debemos instanciar un objeto de la clase FPDF definida dentro de la librería, agregar una página nueva, establecer la fuente y empezar a ejecutar los métodos para agregarle información al documento pdf resultante. Por último debemos ejecutar el método output para cerrar el archivo. Veamos un ejemplo:

```
<?php  
require('fpdf.php');  
  
$pdf=new FPDF();  
$pdf->AddPage();  
$pdf->SetFont('Arial','B',16);  
$pdf->Cell(40,10,'Hola, Mundo!');  
$pdf->Output();  
?>
```

Esto debe realizarse en un documento php exclusivamente hecho para crear el archivo pdf. Este documento php puede ser llamado con un hipervínculo desde cualquier página de nuestro sitio. Vamos a explicar más detalladamente las instrucciones colocadas en este ejemplo:

- La instanciación de la clase: El constructor FPDF() se usa en el ejemplo con sus valores por defecto: las páginas son de tamaño a4 alargado y la unidad de medida es el milímetro. Se podría haber declarado explícitamente con:

```
$pdf=new FPDF("P","mm","A4");
```

Es posible usar el formato apaisado(L), otros formatos de página (como Carta y Legal) y otras unidades de medida (pt, cm, in).

- Agregar una página en blanco: Por el momento no hemos creado ninguna página, así que añadiremos una con AddPage(). El origen de coordenadas está en la esquina superior izquierda y la posición actual está por defecto situada a 1 cm de los bordes; los márgenes pueden cambiarse con SetMargins().
- Establecer la fuente: Antes de que podamos imprimir texto, es obligatorio escoger una fuente con SetFont(), si no, el documento no será válido. Escogemos Arial en negrita de tamaño 16:

```
$pdf->SetFont("Arial","B",16);
```

Podríamos haber especificado itálica con I, subrayado con U o normal con una cadena vacía (o cualquier combinación de las anteriores). Observe que el tamaño de la fuente se determina en puntos, no en milímetros (ni en cualquier otra unidad establecida por el usuario); es la única excepción. Las otras fuentes incorporadas son Times, Courier, Symbol y ZapfDingbats.

Finalmente, el documento se cierra y se envía al navegador con Output(). También podríamos haberlo guardado en un fichero pasando como parámetro el nombre del archivo.

```
$pdf->output("prueba.pdf");
```

Cuidado: en caso de que el PDF se envíe al navegador, nada más debe enviarse, ni antes ni después (el más mínimo espacio en blanco o retorno de carro también cuenta). Si se envía algún dato antes, obtendrá el mensaje de error: "Some data has already been output, can't send PDF file". Si se envía después, su navegador puede que muestre únicamente una página en blanco.

10.3.- Agregar Contenido a un Documento

Para agregar texto al archivo pdf usamos el método cell. Una celda es una superficie rectangular, con borde si se quiere, que contiene texto. Se imprime en la posición actual. Al agregar una página nueva el origen de coordenadas está en la esquina superior izquierda y la posición actual está por defecto situada a 1 cm de los bordes. Al agregar la celda especificamos sus dimensiones, el texto (centrado o alineado), si queremos dibujar o no los bordes, y dónde se ubicará la posición actual después de imprimir la celda (a la derecha, debajo o al principio de la siguiente línea). El formato general del método cell es:

```
Cell(float w [,float h [,string txt[,mixed border[,int ln[, string align[,boolean fill[, mixed link]]]]]]])
```

La explicación de cada uno de los parámetros es:

w

Ancho de Celda. Si es 0, la celda se extiende hasta la márgen derecha.

h

Alto de celda. Valor por defecto: 0.

txt

cadena a ser impresa. Valor por defecto: cadena vacía.

border

Indica si los bordes deben ser dibujados alrededor de la celda. El valor puede ser un número:

- 0: sin borde
- 1: marco

o una cadena conteniendo alguno o todos de los siguientes caracteres (en cualquier orden):

- L: izquierda
- T: superior
- R: derecha
- B: inferior

ln

Indica donde la posición actual debería ir antes de invocar. Los valores posibles son:

- 0: a la derecha
- 1: al comienzo de la siguiente línea
- 2: debajo

Poner 1 es equivalente a poner 0 y llamar justo después Ln(). Valor por defecto: 0.

align

Permite centrar o alinear el texto. Los posibles valores son:

- L o una cadena vacía: alineación izquierda (valor por defecto)
- C: centro
- R: alineación derecha

fill

Indica si el fondo de la celda debe ser dibujada (true) o transparente (false). Valor por defecto: false.

link

URL o identificador retornado por AddLink()

Luego de agregar la celda o en cualquier momento podemos agregar un salto de línea en el documento con el método `ln()`. También podemos ubicar la posición actual en el documento a un lugar arbitrario usando el método `setxy()`. Veamos algunos ejemplos:

Otra forma de agregar texto al documento es utilizando el método `multicell`. Este método permite imprimir texto con saltos de línea. Estos pueden ser automáticos (tan pronto como el texto alcanza el borde derecho de la celda) o explícito (vía el carácter `\n`). Tantas celdas como sean necesarias son creadas, uno debajo de otra. El texto puede ser alineado, centrado o justificado. El bloque de celda puede ser enmarcado y el fondo impreso. Los parámetros de `multicell` son:

```
MultiCell(float w, float h, string txt [, mixed border [, string align [, boolean fill]]])
```



Capítulo 11. GENERACIÓN DE ARCHIVOS PDF. PARTE 2

11.1.- Agregar Imágenes

Para agregar imágenes a un archivo pdf debemos usar el método `image()`. Las dimensiones pueden establecerse de diferentes maneras:

- mediante la especificación explícita de ancho y alto (en unidades definidas por el usuario)
- mediante la indicación de una sola de las dimensiones: la otra se calculará automáticamente para mantener la proporción original
- sin indicar ninguna dimensión explícita. En este caso, la imagen se imprime a 72 puntos por pulgada

Los formatos admitidos son JPEG, PNG y GIF. La extensión GD es necesaria para GIF. La sintaxis general de éste método es:

```
Image(string file [, float x [, float y [, float w [, float h [, string type [, mixed link]]]]]])
```

Los parámetros del método `image` son:

ACADEMIA DE SOFTWARE

file

Nombre del fichero que contiene la imagen.

x

Abscisa de la esquina superior izquierda. Si no se especifica o es igual a null, se utilizará la abscisa actual.

y

Ordenada de la esquina superior izquierda. Si no se especifica o es igual a null, se utilizará la ordenada actual, además, un salto de página es invocado primero si es necesario (en caso de que esté habilitado el salto de página automático) y, después de la llamada, la ordenada actual se mueve a la parte inferior de la imagen.

w

Ancho de la imagen en la página. Si no se especifica o es cero, se calcula automáticamente.

h

Alto de la imagen en la página. Si no se especifica o es cero, se calcula automáticamente.

type

Formato de la imagen. Los posibles valores son (indiferente a mayúsculas): JPG, JPEG, PNG y GIF. Si no se especifica, el tipo se deduce de la extensión del fichero.



11.2.- Establecer pie y Cabecera de Pagina

La clase FPDF tiene los método Header() y Footer() que agregan un encabezado a las páginas y un pie a las páginas automáticamente. Estos métodos en la clase por defecto no hacen nada, es por ello que necesitaremos crear una clase que herede de la clase FPDF y sobreescribir estos 2 métodos para así poder agregar pies de páginas y/o encabezados de página a nuestros documentos pdf. Veamos un ejemplo de cómo crear una clase que herede de la clase fpdf:

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

class PDF extends FPDF
{
//Cabecera de página
function Header()
{
    //Logo
    $this->Image('logo_pb.png',10,8,33);
    //Arial bold 15
    $this->SetFont('arial','B',15);
    //Movernos a la derecha
    $this->Cell(80);
    //Título
    $this->Cell(30,10,'Title',1,0,'C');
    //Salto de línea
    $this->Ln(20);
}

//Pie de página
function Footer()
{
    //Posición: a 1,5 cm del final
    $this->SetY(-15);
    //Arial italic 8
    $this->SetFont('arial','I',8);
    //Número de página
    $this->Cell(0,10,'Page '. $this->PageNo().' / (nb)',0,0,'C');
}
}
```

ACADEMIA DE SOFTWARE

Luego debemos instanciar nuestra clase. Al agregar nuevas páginas automáticamente serán llamados los métodos header() y footer().

```
//Creación del objeto de la clase heredada
$pdf=new PDF();
$pdf->AliasNbPages();
$pdf->AddPage();
$pdf->SetFont('Times','',12);
for($i=1;$i<=40;$i++)
    $pdf->Cell(0,10,'Imprimiendo linea número '.$i,0,1);
$pdf->Output();
```

Capítulo 12. AJAX

12.1.- ¿Qué es AJAX?

Ajax (Asynchronous JavaScript And XML) es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas completamente, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página.

Ajax es una combinación de cuatro tecnologías ya existentes:

- XHTML (o HTML) y hojas de estilos en cascada (CSS) para el diseño que acompaña a la información.
- Document Object Model (DOM) accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones ECMAScript como JavaScript y JScript, para mostrar e interactuar dinámicamente con la información presentada.
- El objeto XMLHttpRequest para intercambiar datos de forma asíncrona con el servidor web. En algunos frameworks y en algunas situaciones concretas, se usa un objeto iframe en lugar del XMLHttpRequest para realizar dichos intercambios.
- XML es el formato usado generalmente para la transferencia de datos solicitados al servidor, aunque cualquier formato puede funcionar, incluyendo HTML preformateado, texto plano, JSON y hasta EBML.

12.2.- El objeto XMLHttpRequest

XMLHTTP (Extensible Markup Language / Hypertext Transfer Protocol), es una interfaz empleada para realizar peticiones HTTP y HTTPS a servidores Web. La interfaz se presenta encapsulada en una clase. Para utilizarlo, la

aplicación cliente debe crear una nueva instancia mediante el constructor adecuado.

La instancia del objeto XMLHttpRequest nos permite la transferencia de datos en formato XML desde los script del lado del cliente (JavaScript, JScript, VBScript) a los del servidor (PHP, Perl, Asp, Java) e inversamente.



El uso de la interfaz XMLHttpRequest no es sencillo ya que requiere de conocimiento en el manejo del protocolo HTTP, manejo correcto de la cache en el servidor, proxy cache intermedios, cache del navegador y además del problema que representa la implementación desigual en cada navegador

Debido a esta dificultad existen frameworks que facilitan el uso de la interfaz XMLHttpRequest permitiendo realizar operaciones asíncronas de manera transparente para el programador.

12.3.- Usos y Aplicaciones

Como se explicó anteriormente, Ajax es una técnica de desarrollo web que permite implementar una comunicación asincrónica entre el cliente y el servidor. Gracias a Ajax, muchas de las tareas que habitualmente se realizan desde el lado servidor (por ejemplo, consultas a bases de datos) pueden realizarse parcialmente desde el lado cliente, evitando recargar la página y brindando una experiencia de navegación más fluida. Los usos más frecuentes que se le da a esta tecnología son los siguientes:

- consulta de nombres de usuarios disponibles durante un registro de nuevo usuario.
- autocompletación con información obtenida de una base de datos.
- resultados automáticos de búsquedas.
- chats.
- traductores.
- actualización de bandejas de entrada sin necesidad de recargar páginas.

Visite el sitio http://www.dynamicajax.com/fr/AJAX_Example_Sites-271_287.html para conocer otras aplicaciones de Ajax.

12.4.- Ventajas y Desventajas

Entre las ventajas de usar Ajax estan:

- Mejor experiencia de usuario. Ajax permite que las páginas se modifiquen sin tener que volver a cargarse, dándole al usuario la sensación de que los cambios se producen instantáneamente. Este comportamiento es propio de los programas de escritorio a los que la mayoría de los usuarios están más acostumbrados. La experiencia se vuelve mucho más interactiva.
- Optimización de recursos. Al no recargarse la página se reduce el tiempo implicado en cada transacción. También se utiliza menos ancho de banda.
- Alta compatibilidad. Ajax es soportado por casi todas las plataformas Web.

Entre las desventajas se pueden describir las siguientes:

- Problemas de acceso. Normalmente, si un usuario refina una consulta a una base de datos a través de muchos criterios (por ejemplo, categoría, precio, forma de pago, etc.), la página se recargará con una URL que reflejará los parámetros ingresados. El usuario puede guardar esa URL para volver a acceder a los resultados ya filtrados fácilmente. Pero con Ajax la URL no se modifica ante la consulta, por lo que deberemos volver a ingresar cada filtro manualmente cuando queramos recuperar los resultados deseados. Existen métodos para modificar este comportamiento, pero agregan dificultad al desarrollo y peso al sitio.
- Problemas de SEO. Los buscadores tienen dificultades al analizar el código escrito en JavaScript. El hecho de que se no se generen nuevas URLs elimina un importante factor de posicionamiento.
- Dificultad. Las aplicaciones con Ajax suelen requerir de un mayor tiempo de desarrollo.

Dadas sus ventajas y desventajas, Ajax sólo debería aplicarse en los casos en que una interacción cliente-servidor tradicional no sea capaz de brindar una buena experiencia de usuario.

ACADEMIA DE SOFTWARE

Capítulo 13. AJAX CON PHP. PARTE 1

13.1.- XAjax

XAJAX es un framework escrito en PHP de código abierto que permite crear fácilmente aplicaciones web que utilizan AJAX sin necesidad siquiera de conocer JavaScript (aunque existe la posibilidad de hacer funciones en este lenguaje). XAJAX utiliza una forma de trabajo de funciones, designando qué funciones de código PHP se convierten en funciones AJAX.

XAJAX permite escribir funciones en PHP que pueden ser accedidas cuando ya la página ha estado enviada al navegador, cuando el usuario ha disparado un evento o la función PHP ha sido llamada desde JavaScript. Éstas funciones PHP modifican el contenido o el aspecto de la página, como lo podría hacer JavaScript, pero hechas desde PHP, lo que abre un abanico de posibilidades ya que puede acceder a bases de datos, al sistema de archivos o realizar cualquier otra cosa que PHP lo permita. XAJAX es un producto Open Source gratuito y compatible con los navegadores más comunes, como Firefox, u otros navegadores basados en Mozilla, Internet Explorer, Opera.

El proyecto puede ser descargado de su sitio web oficial <http://xajaxproject.org/>. Para instalar XAJAX en el proyecto se deben seguir los siguientes pasos:

- Descargar la última versión de XAJAX del sitio web oficial (actualmente la versión estable es la 0.5)
- Incluir las carpetas xajax_core, xajax_js, xajax_controls y xajax_plugins en la proyecto o sitio web.

13.2.- La Clase Xajax

La clase xAjax posee un conjunto de métodos usados para generar el código JavaScript necesario para realizar llamadas asíncronas. Para hacer una página php que haga uso de ajax debemos hacer 3 cosas:

- crear un documento php que hace uso de la clase ajax, donde también se definen las funciones que se llamarán de forma asíncrona.

- crear un documento php con etiquetas html (puede ser una capa, un lista menú, etc) que serán generados dinámicamente de forma asíncrona
- hacer el llamado de las funciones asíncronas en algún lugar, puede ser el click de un botón.

Veamos un ejemplo del archivo donde están las funciones que serán llamadas asíncronamente. Este archivo lo guardaremos con el nombre ajax.php (puede ser cualquier nombre):

```
1 <?php
2 require_once("xajax_core/xajax.inc.php");
3 $xajax = new xajax();
4
5 $xajax->registerFunction("hora");
6
7 function hora()
8 {
9     $objResponse = new xajaxResponse();
10    $objResponse->assign("capa_hora","innerHTML", date("H:m:s"));
11
12    return $objResponse;
13 }
14
15 $xajax->processRequest();
16
17 ?>
```

La explicación de este código es la siguiente:

- en la línea 2 se incluye la librería que contiene la clase xajax.
- en la línea 3 se instancia un objeto de la clase xajax.
- en la línea 5 se registra la función que será llamada de forma asíncrona (debe hacerse con cada función que ejecutará código php asíncrono).
- dentro de la función asíncrona "hora" se instancia un objeto de la clase xajaxResponse.
- se asigna la salida dinámica que generará la llamada de la función, usando el método assign de la clase xajaxResponse. En el ejemplo se le agrega información al div "capa_hora".

- la función debe retornar el objeto xajaxResponse.
- al final del archivo debe hacerse un llamado al método processRequest() de la clase xajax.

13.3.- Llamado Asíncrono de Funciones

Ya en la página php donde se ejecutarán las funciones asíncronamente debemos hacer lo siguiente:

```
1 <?php
2 require 'ajax.php';
3 ?>
4 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
5 <html>
6   <head>
7     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
8     <title></title>
9   </head>
10  <?php $xajax->printJavascript(); ?>
11  <body>
12    <input type="button" value="La hora" name="hora" onclick="xajax_hora()" />
13  <div id="capa_hora">
14  </div>
15  </body>
16 </html>
```

La explicación de este código es la siguiente:

- incluimos el archivo "ajax.php".
- se coloca en algún punto antes del body el llamado al método "printJavaScript".
- se hace el llamado a la función registrada en el archivo ajax.php, anteponiéndole el sufijo "xajax_", como en el ejemplo "xajax_hora".