**PHP**

Aníbal Zavala Díaz.

Josselyn Jorge de la Cruz

Marzo 2018.

Universidad Científica del Sur.

Ingeniería de Sistemas de Información y Gestión.

Lenguaje de Programación

**Tabla de Contenidos**

I. Historia básica ……………………………………………………………………………………………… 3

II. Programación orientada a objetos ………………………………………………………………. 6

III. ¿Qué es PHP? ………………………………………………………………………………………………. 8

IV. ¿Qué se puede hacer con PHP? ……………………………………………………………………. 9

V. ¿Cómo funciona el intérprete de PHP con ZEND? ………………………………………… 11

VI. Variables y tipos de dato ……………………………………………………………………………… 13

VII. Clases …………………………………………………………………………………………………………... 15

VIII. Arreglos ……………………………………………………………………………………………………….. 17

IX. Ejemplos ………………………………………………………………………………………………………. 20

**I. Historia básica**

**PHP/FI**

PHP es el heredero de un producto anterior, llamado PHP/FI. PHP/FI fue creado por Rasmus Lerdorf en 1995, inicialmente como un simple conjunto de scripts de Perl para controlar los accesos a su trabajo online. Llamó a ese conjunto de scripts 'Personal Home Page Tools'. Según se requería más funcionalidad, Rasmus fue escribiendo una implementación C mucho mayor, que era capaz de comunicarse con bases de datos, y permitía a los usuarios desarrollar sencillas aplicaciones Web dinámicas. Rasmus eligió [» liberar](http://groups.google.com/group/comp.infosystems.www.authoring.cgi/msg/cc7d43454d64d133) el código fuente de PHP/FI para que cualquiera pudiese utilizarlo, así como arreglar errores y mejorar el código.

PHP/FI, que se mantuvo para páginas personales y como intérprete de formularios, incluía algunas de las funcionalidads básicas de PHP tal y como lo conocemos hoy. Tenía variables como las de Perl, interpretación automática de variables de formulario y sintaxis embebida HTML. La sintaxis por sí misma era similar a la de Perl, aunque mucho más limitada, simple y algo inconsistente.

Por 1997, PHP/FI 2.0, la segunda escritura de la implementación en C, tuvo un seguimiento estimado de varios miles de usuarios en todo el mundo, con aproximadamente 50.000 dominios informando que lo tenían instalado, sumando alrededor del 1% de los dominios de Internet. Mientras había mucha gente contribuyendo con bits de código a este proyecto, era todavía en su mayor parte el proyecto de una sola persona.

PHP/FI 2.0 no se liberó oficialmente hasta Noviembre de 1997, después de gastar la mayoría de su vida en desarrollos beta. Fue sucedido en breve tiempo por las primeras versiones alfa de PHP 3.0.

### PHP 3

PHP 3.0 era la primera versión que se parecía fielmente al PHP tal y como lo conocemos hoy en día. Fue creado por Andi Gutmans y Zeev Zuraski en 1997 reescribiéndolo completamente, después de que encontraran que PHP/FI 2.0 tenía pocas posibilidades para desarrollar una aplicación comercial que estaban desarrollando para un projecto universitario. En un esfuerzo para cooperar y empezar a construir sobre la base de usuarios de PHP/FI existente, Andi, Rasmus y Zeev decidieron cooperar y anunciar PHP 3.0 como el sucesor oficial de PHP/FI 2.0, interrumpiéndose en su mayor parte el desarrollo de PHP/FI 2.0.

Una de las mejores características de PHP 3.0 era su gran extensibilidad. Además de proveer a los usuarios finales de una sólida infraestructura para muchísimas bases de datos, protocolos y APIs, las características de extensibilidad de PHP 3.0 atrajeron a docenas de desarrolladores a unirse y enviar nuevos módulos de extensión. Sin duda, ésta fue la clave del enorme éxito de PHP 3.0. Otras características clave introducidas en PHP 3.0 fueron el soporte de sintáxis orientado a objetos y una sintáxis de lenguaje mucho más potente y consistente.

Todo el nuevo lenguaje fue liberado bajo un nuevo nombre, que borraba la implicación de uso personal limitado que tenía el nombre PHP/FI 2.0. Se llamó 'PHP' a secas, con el significado de ser un acrónimo recursivo - PHP: Hypertext Preprocessor.

A finales de 1998, PHP creció hasta una base de instalación de decenas de millares de usuarios (estimados) y cientos de miles de sitios Web informando de su instalación. En su apogeo, PHP 3.0 estaba instalado en aproximadamente un 10% de los servidores Web en Internet.

PHP 3.0 se liberó oficialmente en Junio de 1998, después de haber gastado unos 9 meses en pruebas públicas.

### PHP 4

En el invierno de 1998, poco después del lanzamiento oficial de PHP 3.0, Andi Gutmans y Zeev Suraski comenzaron a trabajar en la reescritura del núcleo de PHP. Los objetivos de diseño fueron mejorar la ejecución de aplicaciones complejas, y mejorar la modularidad del código base de PHP. Estas aplicaciones se hicieron posibles por las nuevas características de PHP 3.0 y el apoyo de una gran variedad de bases de datos y APIs de terceros, pero PHP 3.0 no fue diseñado para el mantenimiento tan complejo de aplicaciones eficientemente.

El nuevo motor, apodado 'Motor Zend' (comprimido de sus apellidos, Zeev y Andi), alcanzó estos objetivos de diseño satisfactoriamente, y se introdujo por primera vez a mediados de 1999. PHP 4.0, basado en este motor, y acoplado con un gran rango de nuevas características adicionales, fue oficialmente liberado en Mayo de 2000, casi dos años después que su predecesor, PHP 3.0. Además de la mejora de ejecución de esta versión, PHP 4.0 incluía otras características clave como el soporte para la mayoría de los servidores Web, sesiones HTTP, buffers de salida, formas más seguras de controlar las entradas de usuario y muchas nuevas construcciones de lenguaje.

PHP 4 es actualmente la última versión liberada de PHP. Ya se está trabajando en modificar y mejorar el motor Zend para integrar las características que se diseñarían para PHP 5.0.

Hoy, se estima que PHP es usado por cientos de miles de programadores y muchos millones de sitios informan que lo tienen instalado, sumando más del 20% de los dominios en Internet.

El equipo de desarrollo de PHP incluye docenas de programadores, así como otras docenas de personas trabajando en proyectos relacionados con PHP como PEAR y el proyecto de documentación.

### PHP 5

El 13 de julio de 2004, fue lanzado PHP 5, utilizando el motor Zend Engine 2.0 (o Zend Engine 2). La versión más reciente de PHP es la 5.3.6 (17 de marzo de [2011](http://es.wikipedia.org/wiki/2011)), que incluye todas las ventajas que provee el nuevo Zend Engine 2. Tales como:

* Mejor soporte para la [programación orientada a objetos](http://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos), que en versiones anteriores era extremadamente rudimentario.
* Mejoras de rendimiento.
* Mejor soporte para [MySQL](http://es.wikipedia.org/wiki/MySQL) con extensión completamente reescrita.
* Mejor soporte a [XML](http://es.wikipedia.org/wiki/XML) (XPath, DOM, etc.).
* Soporte nativo para [SQLite](http://es.wikipedia.org/wiki/SQLite).
* Soporte integrado para [SOAP](http://es.wikipedia.org/wiki/SOAP).
* [Iteradores](http://es.wikipedia.org/wiki/Iterador_(patr%C3%B3n_de_dise%C3%B1o)) de datos.
* [Manejo de excepciones](http://es.wikipedia.org/wiki/Manejo_de_excepciones).
* Mejoras con la implementación con [Oracle](http://es.wikipedia.org/wiki/Oracle).

**II. Programación Orientado a Objetos**

**1. Introducción**

La programación orientada a objetos (Poo) es un paradigma de programación más orientado a como vemos las cosas en la vida real que otros tipos de programación (programación lógica, funcional, imperativa,...)

Este tipo de programación permite solventar algunos de los problemas que ha tenido el desarrollo del software desde sus comienzos, como por ejemplo la falta de portabilidad y baja reusabilidad del código, junto con la dificultad en la modificación y desarrollo del mismo. Además es una técnica de codificación bastante mas intuitiva que el resto.

La programación orientada a objetos tiene tres propiedades básicas: Debe estar basado en objetos, estar basado en clases y debe de ser capaz de mantener una herencia entre clases. La mayoría de los lenguajes cumplen una o dos de estas propiedades pero pocos logran cumplir las tres. En particular, la herencia.

Posteriormente veremos los conceptos *objeto*, *clase* y *herencia* junto con el resto de conceptos asociadazos a la Poo.

Aprender esta técnica no es complicado, pero es una manera subjetiva de programar que depende del desarrollador. Aunque podemos dar diferentes soluciones a un mismo problema, no todas ellas son válidas. La dificultad no radica en aprender esta técnica, si no en ejecutarla bien. Solo programando bien podemos aprovechar todas las ventajas que nos ofrece la programación orientada a objetos.

1. **Conceptos básicos de POO.**

* **Objeto**: Conjunto concreto de datos y operaciones (métodos). Se corresponden con los objetos que nos rodean. Un ejemplo de objeto puede ser un libro. Tiene una serie de propiedades (número de hojas, autor, editor, ...) y unas operaciones (se puede abrir, leer, anotar, ...)
* **Clase**: Conjunto de las propiedades y métodos de un tipo de objeto determinado. Podemos definir la clase libro, donde un objeto perteneciente a esta clase sería “El Camino” de Miguel Delibes.
* **Método**: Algoritmo asociado a una clase de objetos o a uno de ellos. Se ejecutan tras recibir un "mensaje". Un método es lo que los objetos pueden hacer. No puedo escribir con un libro, por ejemplo. Un método puede producir un cambio en algunas de las propiedades del objeto, y/o la generación de un "evento" junto con un mensaje que se envía a otro objeto o así mismo. De esta forma, en el ejemplo de los libros, el método leer generaría un mensaje dirigido al propio libro para que fuese abierto.
* **Evento**: Es un suceso en el sistema. El programa envía el mensaje adecuado al objeto pertinente.
* **Mensaje**: Información dirigida a un objeto, el cual le dice que ejecute uno de sus métodos con ciertos parámetros asociados al evento que lo generó.
* **Propiedad o atributo**: Características de un objeto o una clase de objetos. Permiten ver los datos asociados a un objeto desde fuera. Su valor puede ser alterado por el evento provocado por un método
* **Estado interno**: Propiedad invisible de los objetos que solo se ve afectada por un método del objeto. Se utiliza para indicar las distintas situaciones del objeto o clases de objetos.

1. **Características de la Poo.**

* **Abstracción:** Permite no preocuparse por los detalles de implementación de cada tipo de dato. Estos toman el nombre de tipos de datos abstractos (TDA), incluidos en los lenguajes Pascal, Ada, Modula-2 y C++ entre otros. La abstracción está presente en casi todos los lenguajes de programación actuales. Algunos ejemplos de abstracción son los ya mencionados TDA, las estructuras de datos, los procedimientos y las funciones.
* **Encapsulamiento:** Consiste en separar la implementación de los objetos de los objetos en si. De este modo solo se puede cambiar el estado del objeto mediante las operaciones asociadas al mismo, sin tener que modificar su código. Así también se evita que el usuario altere su estado de manera incontrolada.

Esta propiedad también recibe el nombre de “ocultación de la información”. El usuario del objeto desconoce la distribución de la información dentro del objeto y la información disponible en él. Pero gracias a los métodos asociados a cada uno puede usarlos y manejarlos. Un ejemplo de encapsulamiento es el circuito eléctrico de una casa. El usuario solo accede a ciertas partes del circuito (enchufes e interruptores), aunque las partes del circuito internas comparten información entre si.

* **Polimorfismo:** Es la posibilidad que ofrece la Poo de asociar a objetos diferentes distintos métodos con el mismo nombre. Cada objeto actuará en consecuencia según como este implementado el método. Por ejemplo, el mensaje "+" a un objeto de tipo entero significa suma, mientras que para un objeto de tipo cadena significaría concatenación.

Cuando esto sucede en tiempo de ejecución, hablamos de *asignación tardía* o *asignación dinámica*. C++ tiene la posibilidad de polimorfismo en tiempo de compilación mediante operadores.

* **Herencia:** las clases de objetos no están desligadas unas de otras, sino que se relacionan entre sí, formando una jerarquía. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes.

Esto suele hacerse agrupando los objetos en clases y las clases en árboles que reflejan un comportamiento común. Cuando un objeto pertenece a más de una clase se llama herencia múltiple.

**III. ¿Qué es PHP?**

PHP (acrónimo de "PHP: Hypertext Preprocessor") es un lenguaje de "código abierto" interpretado, de alto nivel, embebido en páginas HTML y ejecutado en el servidor.

Una respuesta corta y concisa, pero, ¿qué significa realmente? Un ejemplo nos aclarará las cosas:

**Example #1 Un ejemplo introductorio**

<html>  
    <head>  
        <title>Ejemplo</title>  
    </head>  
    <body>  
  
        <?php   
        echo "Hola, Soy un script PHP!";   
        ?>  
  
    </body>  
</html>

Puede apreciarse que no es lo mismo que un script escrito en otro lenguaje de programación como Perl o C -- En vez de escribir un programa con muchos comandos para crear una salida en HTML, escribimos el código HTML con cierto código PHP embebido (incluido) en el mismo, que producirá cierta salida (en nuestro ejemplo, producirá un texto). El código PHP se incluye entre [etiquetas especiales de comienzo y final](http://www.hackingballz.com/herramientas/manual-oficial-de-php/language.basic-syntax.html#language.basic-syntax.phpmode) que nos permitirán entrar y salir del modo PHP.

Lo que distingue a PHP de la tecnología Javascript, la cual se ejecuta en la máquina cliente, es que el código PHP es ejecutado en el servidor. Si tuviésemos un script similar al de nuestro ejemplo en nuestro servidor, el cliente solamente recibiría el resultado de su ejecución en el servidor, sin ninguna posibilidad de determinar qué código ha producido el resultado recibido. El servidor web puede ser incluso configurado para que procese todos los archivos HTML con PHP.

Lo mejor de usar PHP es que es extremadamente simple para el principiante, pero a su vez, ofrece muchas características avanzadas para los programadores profesionales. No sienta miedo de leer la larga lista de características de PHP, en poco tiempo podrá empezar a escribir sus primeros scripts.

Aunque el desarrollo de PHP está concentrado en la programación de scripts en el lado del servidor, se puede utilizar para muchas otras cosas.

**IV. ¿Qué se puede hacer con PHP?**

PHP puede hacer cualquier cosa que se pueda hacer con un script CGI, como procesar la información de formularios, generar páginas con contenidos dinámicos, o enviar y recibir cookies. Y esto no es todo, se puede hacer mucho más.

Existen tres campos en los que se usan scripts escritos en PHP.

* Scripts del lado del servidor. Este es el campo más tradicional y el principal foco de trabajo. Se necesitan tres cosas para que esto funcione. El intérprete PHP (CGI ó módulo), un servidor web y un navegador. Es necesario correr el servidor web con PHP instalado. El resultado del programa PHP se puede obtener a través del navegador, conectándose con el servidor web. Consultar la sección [Instrucciones de instalación](http://www.hackingballz.com/herramientas/manual-oficial-de-php/missing-stuff.html#installation) para más información.
* Scripts en la línea de comandos. Puede crear un script PHP y correrlo sin ningún servidor web o navegador. Solamente necesita el intérprete PHP para usarlo de esta manera. Este tipo de uso es ideal para scripts ejecutados regularmente desde cron (en \*nix o Linux) o el Planificador de tareas (en Windows). Estos scripts también pueden ser usados para tareas simples de procesamiento de texto. Consultar la sección [Usos de PHP en la línea de comandos](http://www.hackingballz.com/herramientas/manual-oficial-de-php/features.commandline.html) para más información.
* Escribir aplicaciones de interfaz gráfica. Probablemente PHP no sea el lenguaje más apropiado para escribir aplicaciones gráficas, pero si conoce bien PHP, y quisiera utilizar algunas características avanzadas en programas clientes, puede utilizar PHP-GTK para escribir dichos programas. También es posible escribir aplicaciones independientes de una plataforma. PHP-GTK es una extensión de PHP, no disponible en la distribución principal. Si está interesado en PHP-GTK, puedes visitar las [» páginas web del proyecto](http://gtk.php.net/).

PHP puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, incluyendo Linux, muchas variantes Unix (incluyendo HP-UX, Solaris y OpenBSD), Microsoft Windows, Mac OS X, RISC OS y probablemente alguno más. PHP soporta la mayoría de servidores web de hoy en día, incluyendo Apache, Microsoft Internet Information Server, Personal Web Server, Netscape e iPlanet, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd y muchos otros. PHP tiene módulos disponibles para la mayoría de los servidores, para aquellos otros que soporten el estándar CGI, PHP puede usarse como procesador CGI.

De modo que, con PHP tiene la libertad de elegir el sistema operativo y el servidor de su gusto. También tiene la posibilidad de usar programación procedimental o programación orientada a objetos.

Con PHP no se encuentra limitado a resultados en HTML. Entre las habilidades de PHP se incluyen: creación de imágenes, archivos PDF y películas Flash (usando libswf y Ming) sobre la marcha. También puede presentar otros resultados, como XHTM y archivos XML. PHP puede autogenerar estos archivos y almacenarlos en el sistema de archivos en vez de presentarlos en la pantalla.

Quizás la característica más potente y destacable de PHP es su soporte para una gran cantidad de bases de datos. Escribir un interfaz vía web para una base de datos es una tarea simple con PHP. Las siguientes bases de datos están soportadas actualmente:

|  |  |
| --- | --- |
| Adabas D | dBase |
| Empress | FilePro (read-only) |
| Hyperwave | IBM DB2 |
| Informix | Ingres |
| InterBase | FrontBase |
| mSQL | Direct MS-SQL |
| MySQL | ODBC |
| Oracle (OCI7 and OCI8) | Ovrimos |
| PostgreSQL | Solid |
| Sybase | Velocis |
| Unix dbm |  |

También contamos con una extensión DBX de abstracción de base de datos que permite usar de forma transparente cualquier base de datos soportada por la extensión. Adicionalmente, PHP soporta ODBC (el Estándar Abierto de Conexión con Bases de Datos), asi que puede conectarse a cualquier base de datos que soporte tal estándar.

PHP también cuenta con soporte para comunicarse con otros servicios usando protocolos tales como LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (en Windows) y muchos otros. También se pueden crear sockets puros. PHP soporta WDDX para el intercambio de datos entre lenguajes de programación en web. Y hablando de interconexión, PHP puede utilizar objetos Java de forma transparente como objetos PHP Y la extensión de CORBA puede ser utilizada para acceder a objetos remotos.

PHP tiene unas características muy útiles para el procesamiento de texto, desde expresiones regulares POSIX extendidas o tipo Perl hasta procesadores de documentos XML. Para procesar y acceder a documentos XML, soportamos los estándares SAX y DOM. Puede utilizar la extensión XSLT para transformar documentos XML.

Si usa PHP en el campo del comercio electrónico, encontrará muy útiles las funciones Cybercash, CyberMUT, VeriSign Payflow Pro y CCVS para sus programas de pago.

Para terminar, contamos con muchas otras extensiones muy interesantes, las funciones del motor de búsquedas mnoGoSearch, funciones para pasarelas de IRC, utilidades de compresión (gzip, bz2),, conversión de calendarios, traducción.

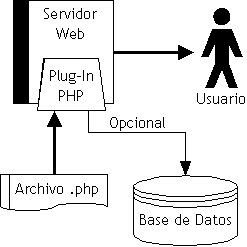
**V. ¿Cómo funciona el intérprete de PHP con ZEND?**



El navegador realiza una llamada al servidor web, que pasa la petición a través de la interfaz PHP del servidor web(1). La interfaz del servidor web llama al intérprete Zend (2), que accede al disco donde están las páginas PHP. El servidor web recupera el código de la página PHP y lo envía al compilador (3). El compilador de Zend crea una versión compilada de la página que es pasada al módulo de ejecución de Zend (4). Este módulo genera el código HTML que será visto por el navegador. Si existieran llamadas a otros módulos, como bases de datos, XML o Java, el intérprete de Zend la envía a través del módulo específico para procesar la petición, y devuelve el resultado a la interfaz del servidor web (5) que enviará el código HTML al navegador (6).

Con Zend Compiler se permitirá a los desarrolladores compilar sus propios script de PHP y distribuirlos. De esta forma se pretende proteger el código fuente y así crear una especie de copyright, permitiendo a las empresas crear aplicaciones y soluciones software basado en PHP. Además también se espera la presentación de productos como Zend Caché, un módulo que almacena en una memoria intermedia del servidor web la aplicación PHP ya interpretada. Este proceso consigue mejorar el tiempo de respuesta de cara al usuario siendo ideal para website con gran cantidad de visitas. De esta forma se trata de evitar las continuas compilaciones de las páginas cada vez que son solicitadas.

¿Cómo interpreta?

PHP toma código dentro de las páginas, lo ejecuta en el servidor y envía el resultado al cliente. El cliente no puede visualizar el código del programa, sólo su resultado. Además, por ser un lenguaje de scripting, los programas no se compilan, sino sólo se interpretan; esto significa que es más lento en ejecutarse que, por ejemplo, un programa en C, pero al mismo tiempo los cambios en el código PHP tienen efecto de inmediato.

No todos los archivos son interpretados por el plug-in de PHP, sino sólo aquellos que hayan sido definidos en la configuración del servidor como tales, puesto que la interpretación de código es un proceso que toma un tiempo mayor al que se emplea en enviar una página tal como está. Es común utilizar una o varias de las siguientes extensiones para informar al servidor que debe interpretarlas: .php, .phtml, .php3 y .php4.

Existen múltiples formas de incluir código PHP:  
  
<?php echo("Hello World") ?>  
  
<? echo("Hello World") ?>  
  
<script language="php"> echo("Hello World"); </script>  
  
<% echo("Hello World") %>

Puede ser que no todas estén disponibles en su sistema, esto depende de la instalación que haya realizado. Ante la duda, utilice la primera forma  
  
Todas las instrucciones se separan de la instrucción siguiente con un ; (punto y coma), y se asume que el final de la inclusión de código limita instrucciones:

<% echo( "hola" ) %> <% echo( "chao" ) %>  
  
<% echo( "hola" ); echo( "chao" ); %>

Comentarios  
  
Los comentarios en PHP se pueden poner en varios formatos, de tipo C, C++ y Shell. Si bien se puede hacer, no es recomendable mezclar distintos tipos de comentario en un archivo, sino elegir una sintaxis y quedarse con ella durante todo el documento.

<?  
  
echo( "Hello World" )  
  
/\* comentario de  
  
varias líneas \*/  
  
// comentario de una línea  
  
?>

Como en la mayoría de los lenguajes, no se pueden poner comentarios dentro de otros comentarios.

**VI. Variables y tipos de dato**  
  
Las variables son enteros, flotantes, strings, arreglos y objetos. Todos comienzan con un signo $ y a continuación un identificador ($a, $linea, etc.) que es sensible a mayúsculas y minúsculas.

Normalmente PHP elegirá un tipo apropiado de acuerdo al contexto para cada variable:

$a = 123; # entero  
  
$a = 123.1; # flotante  
  
$a = "abc"; # string  
  
Los arreglos se denominan $arreglo[indice] en que indice puede ser un entero o un string. Esto permite crear fácilmente arreglos asociativos (tablas de hashing).

$a[0] = "aaa";  
  
$a[1] = 22;  
  
$edad["pedro"] = 30;  
  
$edad["agustin"] = 35;  
  
$direccion["pedro"] = "Las Pataguas 123";

Strings  
  
Sobre strings se define la concatenación $a . $b como operador. Para generar caracteres especiales se usa:

echo " " # (newline)  
  
echo """ # "  
  
echo "$" # $  
  
echo "\" #

**VII. Clases**

Se aplica el concepto de clase que todos conocemos, entidad contenedora de información basada en atributos y en métodos de construcción, modificación y consulta de dichos atributos necesarios y suficientes para representar un objeto con el que trabajar o procesar información relativa al dominio del problema en el que nos estamos moviendo.

Para esto se define la palabra reservada **class**. Vamos a ver un ejemplo de definición de una clase en PHP.

usuario.class.php

1 //--------------------------------------------------------------

2 // Creación de la clase Usuario

3 // Autor: Ildefonso Montero Pérez - monteroperez{arroba}us.es

4 //--------------------------------------------------------------

5 <?php

6 **class** Usuario{

7

8 // ATRIBUTOS

9 **private** $nombreDeUsuario;

10 **private** $palabraClave;

11

12 // CONSTRUCTOR DE LA CLASE

13 **public function** Usuario() { }

14

15 // FUNCIONES CONSULTORAS

16 **public function** getNombreDeUsuario() { return **$this**->nombreDeUsuario; }

17 **public function** getPalabraClave() { return **$this**->palabraClave; }

18

19 // FUNCIONES MODIFICADORAS

20 **public function** setNombreDeUsuario($nu) { **$this**->nombreDeUsuario = nu; }

21 **public function** setPalabraClave($pc) { **$this**->palabraClave = $pc; }

Programación en PHP a través de ejemplos 52

22

23 }

24 ?>

Vamos a comentar detalladamente el ejemplo que se adjunta. En él estamos definiendo una clase que representará a un usuario de una aplicación. Es un ejemplo muy básico en el que se introducirá únicamente el nombre de usuario y su contraseña, típico ejemplo de un sistema de autenticación.

Para ello definiremos dos atributos denominados nombredeUsuario y palabraClave los cuales podremos consultar y modificar mediante sus respectivas funciones consultoras y modificadoras, más comúnmente conocidas como funciones *getters* y *setters.*

Podemos comprobar en las cuatro primeras líneas que está permitido el tipo de comentario propio de C++ ( // ) y vemos que a partir de la sexta línea comenzamos con la definición de la clase mediante la palabra reservada class seguida del nombre de la clase, en este caso Usuario.

Los atributos son definidos como variables con visibilidad privada mediante la palabra clave private, tambien podriamos optar por hacer uso de var pero de esta forma estas variables serian públicas y no nos interesa Esto se define en las lineas nueve y diez.

A continuación definimos el constructor en la linea trece como una función definida como publica mediante public function y donde como vemos no es necesario introducir el tipo de objeto que se devuelve tras la llamada a la función o metodo como ocurre en Java o C++, es decir, nos olvidamos de indicar si la función devuelve un entero, un char o un tipo determinado, o si ni siquiera devuelve nada (void). No es necesario en PHP.

Así pues, ya tenemos definida toda la clase. Merece especial atención la variable $this la cual corresponde como equivalente a this en C++ o Java, es decir, una instancia del propio objeto que estamos modelando.

A continuación, vamos a ver un ejemplo de uso de esta clase en el que crearemos una instancia de un usuario e introduciremos unos datos concretos.

veranoazul.php

1 <?php

2 **require\_once**('usuario.class.php');

3 $usuario = **new** Usuario();

4 $usuario->setNombreDeUsuario('chanquete');

5 $usuario->setPalabraClave('hamuerto');

6

7 **echo** 'Nombre de Usuario: '.$usuario->getNombreDeUsuario().'<br>';

8 **echo** 'Palabra Clave: '.$usuario->getPalabraClave().'<br>';

9

10 $usuario->setPalabraClave('nonosmoveran');

11

12 **echo** 'Nombre de Usuario: '.$usuario->getNombreDeUsuario().'<br>';

13 **echo** 'Palabra Clave: '.$usuario->getPalabraClave().'<br>';

14 ?>

En este ejemplo vemos como hacer uso de la clase que hemos definido en el fichero usuario.class.php con lo que necesitamos que se incluya la definición contenida en este fichero de la clase Usuario. Para ello hacemos uso de require\_once, esta es la opción más óptima para la carga de definiciones ya que forzamos que solo se incluya una única vez dentro de todo el código.

A partir de la línea tres empezamos a trabajar con la clase anteriormente definida. Para ello creamos una variable denominada $usuario y mediante new construimos una nueva instancia de la clase Usuario, a la que vamos a inicializar sus atributos en las líneas cuatro y cinco y posteriormente en la diez. Veamos la salida en el navegador de este código:

Nombre de Usuario: chanquete

Palabra Clave: hamuerto

Nombre de Usuario: chanquete

Palabra Clave: nonosmoveran

**Arreglos**

Un **array** es una matriz/vector/arreglo que almacena valores de 3 maneras posibles dependiendo de su **tipo**.  Existe 3 diferentes tipos tal y como explica la web oficial de PHP:

* Array numérico indexado.
* Array asociativo.
* Array multidimendional (Matrices).

Ejemplo de PHP array con índice numérico (indexado)

Es un tipo de array que tiene índices numéricos y se accede con un número entero a cada valor del mismo.

$variable = array($valor1, $valor2, $valor2,...);

Explico la línea superior:

$variable: Es la variable donde se guardará el array.

$valor1, $valor2 y $valor3: Valores de cada elemento.

En esta demo de ejemplo se muestra como recorre un array con índice numérico:

$nombre = "Aner";

$array = array(1, 2, 3, "casa", $nombre);

//saco el número de elementos

$longitud = count($array);

//Recorro todos los elementos

for($i=0; $i<$longitud; $i++)

{

//saco el valor de cada elemento

echo $array[$i];

echo "<br>";

}

Como puedes notar, se ha introducido diferentes tipos de valores:

Numéricos: 1, 2 y 3

String: “casa”.

Una variable con valor “Aner”.

Luego asigno a la variable $longitud el número de elementos del array sacado con la función count(). Después recorro el array con un bucle for y saco por pantalla cada valor.

Podemos acceder directamente a un elemento del array sin necesidad de recorrerlo, sería así:

//ejemplo para acceder al primer elemento

$array[0];

//ejemplo para acceder al sexto elemento

$array[5];

Ejemplo de PHP array tridimensional (multidimensional)

Es caso de 3 dimensiones. Aquí el ejemplo:

$datos = array(

array(array(0, 0, 0),

array(0, 0, 1),

array(0, 0, 2)

),

array(array(0, 1, 0),

array(0, 1, 1),

array(0, 1, 2)

),

array(array(0, 2, 0),

array(0, 2, 1),

array(0, 2, 2)

)

);

Como puedes ver, en la matriz del ejemplo hay una profundidad de 3 arrays, es decir, el primero contiene un segundo y el segundo de 3 terceros.

Se pone el código necesario en este ejemplo para recorrer el array tridimensional:

foreach($datos as $datos2)

{

foreach($datos2 as $datos3)

{

foreach($datos3 as $dato)

{

echo "$dato ";

}

echo "<br>";

}

echo "<br>";

}

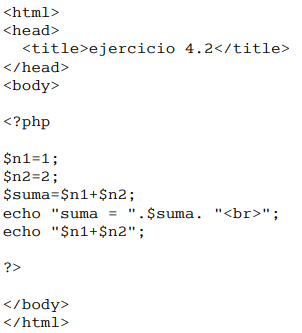
En caso de necesitar acceder a un elemento concreto:

$datos[0][2][1];

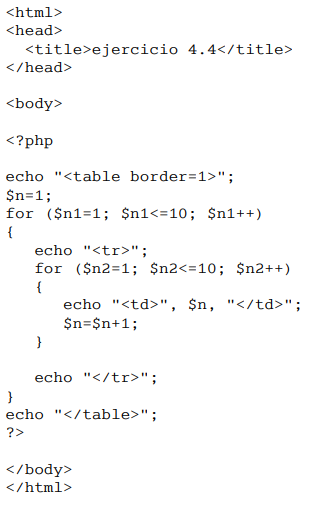
$datos[0][2][1];

**IX. Ejemplos.**

1. Hacer un programa que sume dos variables que almacenan dos números distintos.



2. Mostrar en pantalla una tabla de 10 por 10 con los números del 1 al 100



3. Vectores. Almacene en un vector los 10 primeros número pares. Imprímalos cada uno en una línea

