|  |  |
| --- | --- |
|  | **UNIVERSIDAD DON BOSCO**  **FACULTAD DE ESTUDIOS TECNOLÓGICOS**  **COORDINACIÓN DE COMPUTACIÓN Y MÓVILES** |
| **Ciclo II** | **Desarrollo de Avanzado de Software para móviles**  Guía de Laboratorio No. 3  Introducción a PHP |

* **I Objetivos**
* Que el alumno sea capaz de crear scripts en PHP.
* Que el alumno haga uso de la programación orientada a objetos en PHP.
* **II Introducción**

**Definición de PHP**

PHP es, hoy en día, un lenguaje de programación diseñado para desarrollar páginas web dinámicas que son ejecutadas en un servidor web y, luego, devueltas en formato HTML al navegador del usuario que las solicita. No obstante, en sus principios PHP fue más bien un lenguaje de secuencias de comando, guiones o scripts del lado del servidor desarrollado por Rasmus Lerdorf en 1995 con el propósito de crear un conjunto de scripts que le permitieran contabilizar el número de visitantes que accedían a su hoja de vida que había puesto en línea para conseguir empleo o contratos de trabajo.

Debido a que PHP estaba diseñado para trabajar en un ambiente web y que este se podía insertar directamente dentro del código (X)HTML propició que se volviera muy popular para procesar datos ingresados a través de formularios.

El significado actual de PHP es Hypertext PreProcessor y a partir de la versión 4.0 es considerado todo un lenguaje de programación y una plataforma sólida para el desarrollo de aplicaciones web del lado del servidor.

**Evolución de PHP**

Después de alcanzar su objetivo Rasmus Lerdorf libera su código, que fue desarrollado en lenguaje Perl primero, y luego, en lenguaje C. Para ese entonces se hacía alusión al lenguaje como PHP/FI y su significado era Personal Home Page/Form Interpreter, siendo la versión PHP/FI 2.0 liberada en 1997. En este año el proyecto de PHP pasó de ser un proyecto personal de Rasmus Lerdorf a ser un proyecto desarrollado en conjunto. En ese entonces PHP era usado en unos 50,000 dominios en Internet, lo que representaba solamente el 1% del total de dominios en Internet.

La versión PHP 3.0 fue liberada en 1998 y desarrollada principalmente por dos programadores israelís del Instituto Tecnológico de Israel (Technion) de nombres Andi Gutmans y Zeev Zurasky. Una de las mejores características de esta versión era su fácil extensibilidad reflejada en la incorporación de nuevos módulos y soporte para trabajar con una gran cantidad de bases de datos. En su apogeo la versión 3.0 de PHP era utilizada en un 10% de servidores en Internet.

El siguiente paso en la evolución de PHP fue la creación de un nuevo núcleo, escrito desde cero para mejorar su rendimiento y convertirse por primera vez en un lenguaje orientado a objetos. El nuevo motor fue denominado Zend (acrónimo de los nombres de sus nuevos desarrolladores **Ze**ev y A**nd**y). Esta nueva versión conocida como PHP 4.0 fue liberada en mayo del 2000. Las nuevas prestaciones de esta versión fueron el soporte para la mayoría de servidores web, el manejo de sesiones HTTP de forma nativa, facilidades para programar utilizando orientación a objetos (aunque muchos insisten en que esta versión no es totalmente orientado a objetos), compatibilidad con las expresiones regulares de Perl, encriptación, manejo de formas más seguras para el control de las entradas de usuario.

El 13 del julio del 2007 se anunció la suspensión del soporte para la versión 4 de PHP, no obstante, se siguen liberando mejoras para solventar fallos críticos a nivel de seguridad, es así que se lanzaron posterior a esa fecha las versiones 4.4.8 el 13 de enero del 2008 y posteriormente, la versión 4.4.9 el 7 de agosto del mismo año.

La versión 5 de PHP fue lanzada el 13 de julio del 2004, utilizando el motor Zend Engine 2.0 que entre otras cosas incluye:

* Mejor soporte para la programación orientada a objetos.
* Mejoras en el rendimiento.
* Mejor soporte para MySQL.
* Mejor soporte a XML.
* Soporte nativo para SQLite.
* Soporte integrado para SOAP.
* Iteradores de datos.
* Manejo de excepciones.
* Mejoras para la implementación con Oracle.

NOTA: Puede consultar sobre la evolución de PHP en forma detallada en el sitio oficial de PHP en http://www.php.net o en el sitio de wikipedia http://www.wikipedia.org.

En la actualidad se escucha y se lee mucho acerca de la versión 6 de PHP, que todavía no ha sido liberada; pero que ya se pueden leer libros completos de esta nueva versión que estaría por venir en próximos años. Entre las características que se anuncian están:

* Incorporación de soporte para Unicode.
* Eliminación del parámetro REGISTER\_GLOBALS del php.ini.
* Desaparece la característica de usar MAGIC\_QUOTES.

De acuerdo a las últimas noticias el retraso en el lanzamiento y liberación de la versión 6 de PHP se debe principalmente a que el equipo de desarrollo decidió cambiar el enfoque con el que se estaba tratando las cadenas Unicode, lo cual ha llevado a iniciar un nuevo planteamiento para enfrentar el problema.

**Utilización de PHP**

PHP se puede utilizar para:

* Desarrollar aplicaciones web del lado del servidor. Este es el campo de uso más tradicional de PHP y el que le ha significado una infinidad de seguidores y adeptos.
* Realizar scripts que se ejecuten desde la línea de comandos. Estos scripts se pueden ejecutar sin la necesidad de un servidor web ni de un navegador.
* Escribir aplicaciones de interfaz gráfica. Este es el campo más nuevo en el que PHP ha hecho incursión, para utilizarlo es necesario incluir la extensión PHP-GTK que no viene incluida en la distribución principal.

**Tipo de aplicaciones que se pueden realizar con PHP**

PHP se puede utilizar para crear aplicaciones de:

* Comercio electrónico.
* Educación a distancia.
* Foros de discusión.
* Sistemas de Gestión de Contenidos.
* Blogs.
* Exámenes en línea.
* Aplicaciones de correo electrónico.

**Requerimientos para desarrollar aplicaciones con PHP**

Para poder realizar scripts de PHP en el lado del servidor es necesario tener instalado:

* El intérprete de PHP (CGI o módulo).
* Un servidor web (Apache Web Server es ideal para trabajar con PHP; sin embargo, hoy en día se puede instalar fácilmente en Internet Information Server también).
* Un navegador web (Internet Explorer, Chrome, Firefox, Safari y Opera son los más difundidos).
* Un gestor de bases de datos (MySQL es la mejor opción de base de datos para trabajar con PHP).
* Un editor de texto, de preferencia especializado en sintaxis de PHP.

Sintaxis básica.

El lenguaje PHP es bastante sencillo en cuanto a su sintaxis. Alguien con experiencia en programación con lenguaje C o Perl, no debería tener ningún problema de adaptación. Sin embargo, hay que decir que es más complicado que simplemente escribir código HTML.

**Delimitadores de bloque de código PHP.**

Existen cuatro diferentes tipos de delimitadores de código PHP. Estos son:

1. Delimitadores estilo XML:

<?php

*//instrucciones php*

?>

1. Delimitador estilo script:

<script language="php">

*//instrucciones php*

</script>

1. Delimitador corto:

<?

*//instrucciones php*

?>

1. Delimitador estilo ASP:

<%

*//instrucciones php*

%>

Hay que mencionar que solamente los primeros dos tipos de delimitadores están disponibles de forma automática, sin necesidad de realizar configuración alguna en el archivo php.ini. Los últimos dos delimitadores deben ser habilitados en dicho archivo de configuración. Este archivo está en la carpeta de instalación de PHP y deberá modificarlo si necesita utilizar etiquetas cortas o las de estilo ASP y reiniciar los servicios del Wamp para que funcionen.

**Delimitador de sentencias.**

El terminador o delimitador de sentencias en PHP es el punto y coma (;), el mismo que se utiliza en C/C++. Por lo tanto, cuando desee terminar una sentencia para iniciar otra debe utilizar el punto y coma. Existen dos casos en los que se puede omitir el punto y coma. Uno es cuando sólo existe una instrucción PHP en el script y el otro es cuando la instrucción sea la última línea del bloque de código PHP. Se sugiere que siempre utilice el punto y coma al final de cualquier instrucción, incluso en los casos antes mencionados para evitar cometer errores por tratar de recordar estos casos especiales. Es más fácil recordar que siempre debe utilizarse que recordar cuando puede omitirse.

**Comentarios.**

PHP utiliza los estilos de comentarios del lenguaje C/C++ y del *shell* de la interfaz de comandos de Unix y Linux. Estos son el comentario de una sola línea “//” y el comentario de bloque “/\* … \*/”, también utilizados en el lenguaje C. Además, se puede utilizar el comentario de una sola línea utilizado en el *shell* de Unix y Linux, “#”

Veamos algunos ejemplos:

1. Comentario de una sola línea "//" estilo C++:

<?php

$valor = 5.2;

//Este es un comentario de una sola línea estilo C

echo "El valor es: " . $valor;

?>

1. Comentario de una sola línea "#" estilo *shell*:

<?php

$nombre = "Julio";

#Este es un comentario de una sola línea estilo shell

echo "Su nombre es:. " . $nombre;

?>

1. Comentario de bloque (varias líneas) "/\* … \*/":

<?php

$precio = 25;

$total = $precio \* $POST['descuento'];

/\* En este caso estamos utilizando un comentario

de bloque, esto quiere decir que todo este texto

que está leyendo es comentario y que por tanto será

ignorado por el intérprete de PHP

\*/

echo "El total es " . $total;

?>

**Salida a pantalla.**

En PHP existen dos formas principales de mandar a imprimir texto en la ventana de un navegador. La primera es utilizando la instrucción *echo* y la segunda es utilizar la función *printf*().

**echo**

La sentencia *echo* es fácil de utilizar, su sintaxis es la siguiente:

echo cadena\_de\_texto;

Donde, *cadena\_de\_texto* puede ser un literal de cadena delimitado por comillas que pueden ser simples (') o dobles ("). La diferencia más importante es que entre comillas dobles se interpretan las variables y ciertos caracteres especiales, incluyendo etiquetas HTML. En cambio, con comillas simples sólo se interpretan la comilla simple y la barra invertida, por tanto, para evitar que sean interpretados estos caracteres deberá hacer uso de secuencias de escape.

**printf()**

La función *printf()* es mucho más versátil que la instrucción *echo*. Con esta función se pueden mandar a imprimir varios tipos de variables a la vez, utilizando códigos de formato, que indican cómo debe ser formateada la variable que se desea mostrar a la salida. La sintaxis es la siguiente:

printf("cadena\_de\_texto [%s %d %f %c]", $cadena, $entero, $flotante, $caracter);

Donde, *cadena\_de\_texto* es una cadena delimitada por comillas dobles que puede incluir ciertos códigos de formato opcionales. Si se desea imprimir el contenido de variables deben especificarse códigos de formato para formatear la salida adecuadamente. Los códigos de formato más importantes son:

|  |  |
| --- | --- |
| **Elemento** | **Tipo de variable** |
| %s | Cadena de caracteres. |
| %d | Número sin decimales. |
| %f | Número con decimales. |
| %c | Carácter ASCII. |
| Aunque existen otros tipos, estos son los más importantes. | |

**Variables**

Las variables en PHP se definen anteponiendo el símbolo dólar ($) al nombre de la variable. A diferencia de otros lenguajes, PHP posee una gran flexibilidad a la hora de operar con variables. En efecto, cuando definimos una variable asignándole un valor, PHP le atribuye un tipo. Si por ejemplo definimos una variable entre comillas, la variable será considerada de tipo cadena:

$variable = "5"; //esto es una cadena

Ahora bien, si en el script se realiza una operación matemática con esta variable, no se lanzará ningún mensaje de error sino que la variable cadena será convertida automáticamente en numérica al incluirla en una expresión que involucre un operador matemático:

<?php  
 $cadena = "5"; //esto es una cadena

$entero = 3; //esto es un entero

echo $cadena + $entero

?>

Este *script* dará como resultado "8". La variable cadena con valor de "5", ha sido asimilada como entero (aunque su tipo sigue siendo cadena) para poder realizar la operación matemática. Del mismo modo, podemos operar entre variables tipo entero y real. No debemos preocuparnos de nada, PHP se encarga durante la ejecución de interpretar el tipo de variable necesario para el buen funcionamiento del programa.

Sin embargo, si hay que tener cuidado en no cambiar mayúsculas por minúsculas en el identificador de la variable, ya que, en este sentido, PHP es sensible. Conviene por lo tanto, trabajar ya sea siempre en mayúsculas, o siempre en minúsculas para evitar este tipo de malentendidos a veces muy difíciles de localizar. Durante las prácticas de laboratorio se convendrá que los nombres de **variables** se digitarán siempre **en minúsculas** y las **constantes en mayúsculas**.

**Variables predefinidas de PHP**

Estas son variables que están disponibles para cualquier script PHP que se ejecute en un servidor web con el módulo PHP instalado. Algunas de ellas pueden ser muy útiles para obtener información del cliente o del mismo servidor. El comportamiento y disponibilidad de estas variables depende del servidor sobre el que se estén ejecutando, específicamente de su configuración, de la versión de PHP y de otros factores.

A partir de la versión 4.1.0 PHP dispone de un conjunto de matrices predefinidas que contienen variables del servidor web, variables del entorno y variables de entrada del usuario. Estas matrices son automáticamente globales o, también llamadas, superglobales. Entre estas matrices se pueden mencionar:

**$GLOBALS**, es una matriz asociativa que contiene una referencia a cada variable disponible en el ámbito de las variables globales del script. La forma de acceder a las variables es utilizando el nombre de las variables globales entre comillas (dobles o simples) como índice de la matriz.

**$\_SERVER**, es una matriz asociativa que contiene información como cabeceras, rutas y ubicaciones de scripts. Las entradas de esta matriz se crean en el servidor web. No hay garantía alguna de que el servidor vaya a proveer estos valores realmente. Dentro de las entradas que pueden encontrarse en esta matriz se pueden mencionar:

'PHP\_SELF': que proporciona el nombre del archivo de script ejecutándose actualmente, relativo a la raíz del documento.

'SERVER\_ADDR': que proporciona la dirección IP del servidor web en el que se está ejecutando el script actual.

'SERVER\_NAME': que proporciona el nombre del servidor web bajo el que está siendo ejecutado el *script* actual. Si se ejecuta en un host virtual devolverá el nombre definido para tal host.

'SCRIPT\_FILENAME': que proporciona la ruta absoluta del nombre del *script* que está siendo ejecutado actualmente.

Existen muchas más entradas para la matriz $\_SERVER que pueden consultar en el manual oficial de PHP.

**$\_GET**, que es una matriz asociativa que contiene variables proporcionadas al script por medio del método HTTP GET. Esto significa que cuando define que las variables de un formulario serán pasadas por el método GET, es en esta matriz donde se almacenarán sus valores de acuerdo al nombre que asignó al control de formulario HTML.

**$\_POST**, es una matriz asociativa que contiene las variables pasadas al script a través de método HTTP POST. Al igual que $\_GET, cuando se define que el método de paso de valores provenientes de un formulario será POST, la matriz contendrá dichos valores y para tener acceso a ellos deberá usar como llave de la matriz el nombre que le dio al control de formulario.

**$\_COOKIE**, es una matriz asociativa que contiene las variables pasadas al script mediante cookies HTTP.

**$\_SESSION**, es una matriz asociativa que contiene las variables de sesión disponibles en el script actual.

**$\_REQUEST**, es una matriz asociativa que contiene cualquiera de los contenidos de las matrices superglobales $\_GET, $\_POST y $\_COOKIE.

Existen otras matrices superglobales que se dejan como investigación al estudiante.

Ejemplo:

<?php

$cad = "El script que est&aacute;s ejecutando: " . $\_SERVER['PHP\_SELF'] . ". ";

$cad .= "En el servidor: " . $\_SERVER["SERVER\_NAME"] . ".<br>";

echo "<h3>" . $cad . "</h3>";

?>

**Constantes.**

**Sintaxis**

Se puede definir una constante utilizando la función define(). Una vez definida, no se puede modificar ni eliminar.

Sólo se puede definir como constantes valores escalares (boolean, integer, float y string).

Para obtener el valor de una constante únicamente es necesario especificar su nombre. A diferencia de las variables, no se tiene que especificar el prefijo $. También se puede utilizar la función constant(), para obtener el valor de una constante, en el caso de que queramos expresarla de forma dinámica Usa la función get\_defined\_constants() parar obtener una lista de todas las constantes definidas.

**Nota: Las constantes y las variables (globales) se encuentran en un espacio de nombres distinto. Esto implica que por ejemplo TRUE y $TRUE son diferentes.**

Cuando se utiliza una constante que todavía no ha sido definida, PHP asume que se está refiriendo al nombre de la constante en sí. Se lanzará un aviso si esto sucede. Usa la función define() para comprobar la existencia de dicha constante.

Estas son las diferencias entre constantes y variables:

* Las constantes no son precedidas por un símbolo de dólar ($)
* Las constantes solo pueden ser definidas usando la función() define , nunca por simple asignación
* Las constantes pueden ser definidas y accedidas sin tener en cuenta las reglas de alcance del ámbito.
* Las constantes no pueden ser redefinidas o eliminadas después de establecerse; y
* Las constantes solo puede albergar valores escalares

Ejemplo. Definiendo constantes

<?php

define("CONSTANT", "LIS.");

echo CONSTANT; // muestra el mensaje "LIS."

echo "<br>", Constant; // muestra "Constant".

?>

**Tratamiento de cadenas**

Existen tres formas de asignar cadenas a una variable en PHP, que son: delimitándolas entre comillas dobles, entre comillas simples y usando delimitadores tipo Perl (HereDoc):

Para asignar a una variable una cadena **usando comillas dobles** debe hacer una declaración de este tipo:

$cadena = "Esta es la información de mi variable";

Para mostrar el valor de una variable pueden usarse la instrucción *echo* o la función *print()*:

echo $cadena //obtendríamos: Esta es la información de mi variable;

echo "Esta es la información de mi variable"; //daría el mismo resultado

Algo importante con respecto a los delimitadores de comillas dobles es que interpretan variables si son colocadas dentro de comillas dobles. En terminología de programación, se dice que son interpoladas. Por ejemplo:

<?php  
 $cadena1 = "Aplicaciones";  
 $cadena2 = " Prácticas de Software II";  
 $cadena3 = "Materia: $cadena1 $cadena2";  
 echo $cadena3 //El resultado es: Aplicaciones Prácticas de Software II   
?>

También podemos introducir variables dentro de nuestra cadena lo cual nos puede ayudar mucho en el desarrollo de nuestros scripts. Lo que veremos no es el nombre, sino el valor que almacena la variable:

<?php  
 $a=55;  
 $mensaje = "Tengo $a años";  
 echo $mensaje //El resultado es: "Tengo 55 años"  
?>

Puede ser que en lugar de imprimir el valor de la variable, lo que se desee es imprimir el nombre mismo de la variable. Al colocarlo entre comillas dobles, como se hizo en el ejemplo anterior, no sería posible. La única solución sería encerrarlo entre comillas simples o utilizar código o secuencias de escape en la cadena delimitada por comillas dobles. Como se muestra a continuación:

<?php  
 $a=55;

$mensaje = "Tengo \$a años";

echo $mensaje //El resultado es: "Tengo 55 años"

?>

Si se usan **comillas simples** debe realizar una instrucción como la siguiente:

$cadena = 'Coloque acá su cadena';

Para poder mostrar una comilla simple dentro de una cadena delimitada por comillas simples debe utilizarse una secuencia de escape colocando el símbolo de barra invertida antes de ella. Así:

$cadena='Todos lo llamaban \'el mesías\'';

Los únicos caracteres que deben escaparse cuando se encierran entre comillas simples son la comilla simple y la barra invertida.

Otra forma de delimitar cadenas es mediante el uso de la **sintaxis heredoc ("<<<")**. Debe indicarse un identificador después de la secuencia <<<, luego la cadena, y luego el mismo identificador para cerrar la cita.

<?php  
 $frase = <<<ANILLOS

"Hay muchos vivos que merecen la muerte y hay

muchos muertos que merecen la vida,

¿quién eres tú para impartir ese derecho?" – Gandalf.

ANILLOS;

echo $frase;

?>

El identificador de cierre debe comenzar en la primera columna de la línea. Asimismo, el identificador usado debe seguir las mismas reglas que cualquier otra etiqueta en PHP: debe contener solo caracteres alfanuméricos y de subrayado, y debe iniciar con un caracter no-dígito o de subrayado.

Unos aspectos importantes a considerar acerca de esta sintaxis son:

* La línea con el identificador de cierre no puede contener otros caracteres, excepto quizás por un punto-y-coma (;). Esto quiere decir en especial que el identificador no debe usar sangría, y no debe haber espacios o tabuladores antes o después del punto-y-coma.
* Es importante también notar que el primer caracter antes del identificador de cierre debe ser un salto de línea, tal y como lo defina su sistema operativo. Esto quiere decir \r en Macintosh, por ejemplo.

**Códigos o secuencias de escape**

En PHP, al igual que en otros lenguajes existen ciertos caracteres que generan problemas cuando se encuentran dentro de cadenas, debido a que dentro del lenguaje se interpretan de alguna forma especial. Para poder visualizar correctamente dichos caracteres en una operación de salida se utilizan secuencias de escape que involucran dichos caracteres especiales. La siguiente tabla muestra varios de estos caracteres especiales con su correspondiente secuencia de escape:

|  |  |
| --- | --- |
| **Secuencia de escape** | **Significado** |
| \b | Espacio hacia atrás (*backspace*) |
| \f | Cambio de página (*form feed*) |
| \n | Cambio de línea (*line feed*) |
| \r | Retorno de carro (*carriage return*) |
| \t | Tabulación horizontal |
| \\ | Barra inversa (*backslash*) |
| \' | Comilla simple |
| \" | Comilla doble |
| \$ | Carácter dólar ($) |
| \### | Carácter ASCII octal (#: 0-7) |
| \x## | Carácter ASCII hexadecimal (#: 0-F) |

**Operadores**

Los operadores son símbolos especiales que se utilizan en los lenguajes de programación para poder realizar operaciones con las expresiones. Como lo que se obtiene en dichas operaciones es un valor, el resultado también será una expresión.

Los operadores se pueden clasificar como: unarios (que operan sobre un único valor o expresión), binarios (que operan sobre dos valores o expresiones) y ternarios (que consta de tres valores o expresiones)

En PHP existen diversos tipos de operadores y se pueden clasificar de la siguiente manera: aritméticos, lógicos, de cadena, de ejecución, de comparación, de asignación, de incremento/decremento, etc.

**Operadores aritméticos**

Son los operadores que permiten realizar operaciones numéricas sobre las variables y expresiones. Se muestran en la siguiente tabla:

| **Símbolo** | **Nombre** | **Ejemplo** | **Resultado** |
| --- | --- | --- | --- |
| + | Adición | $a + $b | Suma de $a y $b. |
| - | Substracción | $a - $b | Diferencia entre $a y $b. |
| \* | Multiplicación | $a \* $b | Producto de $a y $b. |
| / | División | $a / $b | Cociente de $a y $b. |
| % | Módulo | $a % $b | Resto de $a dividido por $b. |

Debe tener en cuenta que el operador de división “/” devuelve un número con punto flotante en todos los casos, incluso cuando los dos operandos son enteros.

**Operadores lógicos**

Los operadores lógicos se utilizan para realizar comparaciones entre expresiones. Pueden combinarse para formar expresiones de comparación más complejas. Los operadores lógicos de PHP se muestran en la siguiente tabla:

| **Símbolo** | **Nombre** | **Ejemplo** | **Resultado** |
| --- | --- | --- | --- |
| and | Y | $a and $b | **TRUE** si tanto $a como $b son **TRUE**. |
| or | O | $a or $b | **TRUE** si cualquiera de $a o $b es **TRUE**. |
| xor | O exclusivo (Xor) | $a xor $b | **TRUE** si $a o $b es **TRUE**, pero no ambos. |
| ! | No | ! $a | **TRUE** si $a no es **TRUE**. |
| && | Y | $a && $b | **TRUE** si tanto $a como $b son **TRUE**. |
| || | O | $a || $b | **TRUE** si cualquiera de $a o $b es **TRUE**. |

**Operadores de cadena**

Estos operadores se utilizan en combinación con variables o expresiones de cadena. Los dos operadores válidos en PHP para operar con cadenas son el operador de concatenación que se representa con un símbolo de punto (.) y el operador de asignación sobre concatenación, representado por un punto seguido de un símbolo de igual que (.=). Vea la siguiente tabla:

| **Símbolo** | **Nombre** | **Ejemplo** | **Resultado** |
| --- | --- | --- | --- |
| . | Concatenación | $var1=”Hola “;  $var2=”mundo”;  $saludo=$var1 . $var2; | **Se imprimirá en pantalla:**  **Hola mundo** |
| .= | Concatenación y asignación | $var1=”Hola mundo ”;  $var1.="cruel y perverso"; | **Se imprimirá en pantalla:**  **Hola mundo cruel y perverso** |

**NOTA: Observe que no debe olvidar colocar espacios en blanco al terminar o al comenzar la subcadena para que no queden palabras unidas.**

**Operadores de comparación**

Se utilizan para verificación de condiciones en ciertas expresiones, sobre todo en expresiones condicionales. En la siguiente tabla se muestra la lista completa de ellos:

| **Símbolo** | **Nombre** | **Ejemplo** | **Resultado** |
| --- | --- | --- | --- |
| == | Igual | $a == $b | **TRUE** si $a es igual a $b. |
| === | Idéntico | $a === $b | **TRUE** si $a es igual a $b, y son del mismo tipo. (PHP 4 o superior) |
| != | Diferente | $a != $b | **TRUE** si $a no es igual a $b. |
| <> | Diferente | $a <> $b | **TRUE** si $a no es igual a $b. |
| !== | No idénticos | $a !== $b | **TRUE** si $a no es igual a $b, o si no son del mismo tipo. (PHP 4 o superior) |
| < | Menor que | $a < $b | **TRUE** si $a es estrictamente menor que $b. |
| > | Mayor que | $a > $b | **TRUE** si $a es estrictamente mayor que $b. |
| <= | Menor o igual que | $a <= $b | **TRUE** si $a es menor o igual que $b. |
| >= | Mayor o igual que | $a >= $b | **TRUE** si $a es mayor o igual que $b. |

**Operadores de asignación**

Existe un solo operador básico de asignación, que se lee “se asigna a” y no “es igual a”, como podría parecer. Además de este operador de asignación existen operadores combinados con operador de asignación que también se mostrarán en esta parte. Vea la siguiente tabla:

| **Símbolo** | **Nombre** | **Ejemplo** |
| --- | --- | --- |
| = | Se asigna a | $a = “Hola” |
| .= | Concatenación y asignación | $a .= “ mundo” |
| += | Adición y asignación | $a += $b |
| -= | Sustracción y asignación | $a -= $b |
| \*= | Multiplicación y asignación | $a \*= $b |
| /= | División y asignación | $a /= $b |
| %= | Módulo y asignación | $a %= $b |

**Operadores de ejecución**

PHP soporta un operador de ejecución: las comillas invertidas (``). ¡Note que no se trata de comillas sencillas! PHP intentará ejecutar el contenido entre las comillas como si se tratara de un comando del intérprete de comandos; su salida será devuelta (es decir, no será simplemente volcada como salida; puede ser asignada a una variable).

Ejemplo:

<?php  
 $salida = `ls - al`;  
 echo "<pre>$salida</pre>";  
?>

**Operadores de incremento/decremento**

Estos operadores son, en realidad, una mejora a los operadores aritméticos de adición y sustracción, para el caso muy particular en que uno de los operandos sea la unidad. Existen variantes para este operador dependiendo si primero se hace la asignación y luego el incremento/decremento o viceversa. Veamos la siguiente tabla:

| **Símbolo** | **Nombre** | **Finalidad** |
| --- | --- | --- |
| ++$var | Pre-incremento | Incrementa el valor de $var en 1 y luego retorna el nuevo valor de $var |
| $var++ | Post-incremento | Retorna primero el valor actual de $var y después incrementa este valor en 1 |
| ­­--$var | Pre-decremento | Decrementa el valor de $var en 1 y luego retorna el nuevo valor de $var |
| $var-- | Post-decremento | Retorna primero el valor actual de $var y después decrementa este valor en 1 |

* **III Desarrollo**

#### Ejemplo 1: En este ejemplo se evaluará si un año ingresado por un usuario a través de un formulario resulta ser bisiesto o no.

#### Archivo 1: bisiesto.php

<!DOCTYPE html>

<html lang="es">

<head>

<title>A&ntilde;o bisiesto</title>

<link type="text/css" rel="stylesheet" media="screen" href="css/bisiesto.css" />

</head>

<body>

<img id="background" class="source-image" src="bg/butterflybg1280x800.jpg" alt="" />

<header>

<h1 class="Off" onmouseover="this.className='On';" onmouseout="this.className='Off';">

Determinar si un a&ntilde;o es bisiesto

</h1>

</header>

<section>

<?php

if(!isset($\_POST['enviar'])):

?>

<article>

<fieldset>

<legend>Probar a&ntilde;o:</legend>

<form action="bisiesto.php" method="POST">

<label for="year">A&ntilde;o: </label>

<input type="text" name="year" id="year" size="10" maxlength="4" /><br />

<input type="submit" name="enviar" id="enviar" value="Enviar" />

</form>

</fieldset>

<?php

else:

//Script que determina si un año es bisiesto o no

$year = isset($\_POST['year']) ? $\_POST['year'] : 0;

if(($year%4==0 && $year%100!=0) || $year%400==0){

echo "<p class=\"bisiesto\">";

echo "<span style=\"color:Green;font:bold 10pt 'Lucida Sans';\">El año $year es

bisiesto</span><br />\n";

echo "<span style=\"color:Green;font:bold 10pt 'Lucida Sans';\"><a

href=\"{$\_SERVER['PHP\_SELF']}\">Probar otro año</a>";

echo "</p>";

}

else{

echo "<p class=\"bisiesto\">";

echo "<span style=\"color:FireBrick;font:bold 10pt 'Lucida Sans';\">El año $year

no es bisiesto</span><br />\n";

echo "<span style=\"color:Green;font:bold 10pt 'Lucida Sans';\"><a

href=\"{$\_SERVER['PHP\_SELF']}\">Probar otro año</a>";

echo "</p>";

}

endif;

?>

</article>

</section>

</body>

</html>

#### Archivo 2: bisiesto.css

body {

margin-left:50px;

margin-right:50px;

overflow:hidden;

/\* background:#f8f16d url(bg/butterflybg1280x800.jpg); \*/

}

#background.source-image {

width: 100%;

position: absolute;

top: 0;

left: 0;

z-index:-100;

}

hr {

border:2px ridge Purple;

height:6px;

background-color:Orange;

}

fieldset {

width:200px;

margin:auto;

padding:1em;

border:1px solid #ccc;

background-color:#F9E9CC;

}

label, legend {

font:bold 85.5%/1 "Myriad Pro",Frutiger,"Lucida Sans",sans-serif;

color:Purple;

}

label {

float:left;

}

.On {

border:6px outset Purple;

background-color:Orange;

color:Purple;

font:bold 22pt "Arial Black";

}

.Off {

border:6px outset Orange;

background-color:Purple;

color:Orange;

font:bold 22pt "Arial Black";

}

#year {

float:right;

font:bold 78.5%/1 Frutiger,sans-serif;

color:Green;

}

#year:hover, #year:focus {

background-color:#FCCFE9;

}

#enviar {

width:40%;

background-color:#669933;

font:bold 80.5%/1 Frutiger,sans-serif;

color:Yellow;

}

#enviar:hover {

background-color:Orange;

color:Purple;

}

p.bisiesto {

background:#F8F16D;

}

#### Ejemplo 2: El siguiente ejemplo ilustra cómo utilizar ciclos o lazos do-while para acumular valores y obtener datos como el valor menor y mayor de una serie de números así como el total de números pares presentes en la misma.

#### Archivo 1: intervalos.php

<!DOCTYPE html>

<html lang="es">

<head>

<title>Bucle do-while</title>

<meta charset="utf-8" />

<link type="text/css" rel="stylesheet" media="screen" href="css/intervalos.css" />

<script type="text/javascript" src="js/modernizr.custom.lis.js"></script>

</head>

<body>

<header>

<h1>Ciclos, lazos o bucles</h1>

</header>

<section>

<article>

<?php

$min = 1;

echo "<div id=\"main\">";

echo "<h2>Ejemplos con el lazo o bucle <em>do-while</em></h2>";

echo "<hr />";

echo "<p>\n";

echo "Esta iteraci&oacute;n sencilla con ";

echo "<em>do-while</em> imprime una serie de ";

echo "20 n&uacute;meros enteros\n";

echo "</p>\n";

echo "<table>\n";

do {

echo "<td>$min</td>\n";

$min++;

}while ($min<=20);

echo "</table>\n";

$min=1;

$max=95070;

echo "<p>&nbsp;</p>\n<p>&nbsp;</p>\n";

echo "<p>\nLa siguiente iteraci&oacute;n muestra ";

echo "el total de n&uacute;meros pares en un ";

echo "intervalo que va desde $min hasta $max\n</p>\n";

$contador=0;

if($max<$min){

$temp=$max;

$max=$min;

$min=$temp;

}

echo "<table>\n<tr>\n";

echo "<td>El valor m&iacute;nimo de la serie es:</td>\n";

echo "<td>$min</td>\n";

echo "<td>El valor máximo de la serie es:</td>\n";

echo "<td>$max</td>\n";

echo "</tr>\n</table>\n";

do {

if($min%2==0) $contador++;

$min++;

}while ($min <= $max);

echo "&nbsp;<table>\n";

echo "<tr>\n<td>\n";

echo "El total de n&uacute;meros ";

echo "pares en este intervalo es:\n</td>\n";

echo "<td>$contador</td>\n";

echo "</tr>\n</table>\n";

?>

</article>

</section>

</body>

</html>

**Archivo 2: intervalos.css**

body {

background:rgb(15,60,18) url(../img/numbers.jpg) left repeat-y;

padding-left:128px;

}

h1 {

text-align:center;

font:900 20pt "Arial Rounded",Helvetica,sans-serif;

color:rgb(255,255,66);

text-shadow:3px 6px 3px rgba(120,120,30,0.6);

}

h2 {

text-align:center;

font-family:"Arial Rounded",Helvetica,sans-serif;

font-weight:900;

font-size:16pt;

border:outset 8px ForestGreen;

background-color:Khaki;

}

p {

font-family:Garamond,Georgia;

font-weight:400;

font-size:12pt;

color:ForestGreen;

}

table {

border:outset 5px SaddleBrown;

}

th {

background-color:Sienna;

font-family:"Bookman Old Style";

font-weight:800;

font-size:11pt;

}

td {

border:solid 1px Sienna;

background-color:rgb(245,245,199);

font-family:Batang,Garamond,Georgia;

font-weight:300;

font-size:12pt;

color:ForestGreen;

}

#main {

background-color:rgb(255,255,66);

}

**Ejemplo #3: El siguiente ejemplo muestra una lista de estudiantes y su nota registrada en una matriz asociativa, mostrando cómo recorrer los elementos de una matriz haciendo uso de la sentencia *foreach*.**

**Archivo 1: notasalumnos.php**

<?php

$nombres = array(

'Claudia Salazar' => 6.4,

'Lorena L&oacute;pez' => 6.3,

'Carlos &Aacute;valos' => 7.5,

'Jennifer V&aacute;squez' => 6.2,

'Roxana Calder&oacute;n' => 7.5,

'Pedro Hidalgo' => 5.8,

'Morena Barraza' => 6.0,

'Beatriz Hern&aacute;ndez' => 5.0,

'Alberto Figueroa' => 9.5,

'Julia Garc&iacute;a' => 8.5,

'Ileana Dominguez' => 8.2,

'Oscar Reyes' => 8.5,

'Lorena Linares' => 4.8,

'Ramiro Zepeda' => 5.5,

'Luis Pe&ntilde;ate' => 9.6,

'Jos&eacute; Menjivar' => 6.0,

'Sonia Leiva' => 5.1,

'Ernesto Fuentes' => 6.0,

'Claudia Ponce' => 6.8,

'Cristina Morales' => 7.6,

'Javier Hurtado' => 7.0

);

echo "<!DOCTYPE html>";

echo "<html>\n";

echo "<head>\n";

echo "\t<title>Uso del foreach para recorrer una matriz</title>\n";

echo "\t<link type=\"text/css\" rel=\"stylesheet\" media=\"screen\" href=\"css/notasalumnos.css\" />\n";

echo "</head>\n";

echo "<body>\n";

echo "<header>\n";

echo "\t<h1>Notas de los estudiantes</h1><hr>\n";

echo "</header>\n";

echo "<section>\n";

echo "<article>\n";

echo "\t<table>\n";

echo "\t\t<thead>\n";

echo "\t\t\t<tr>\n";

echo "\t\t\t\t<th>\n";

echo "\t\t\t\t\tNombre\n";

echo "\t\t\t\t</th>\n";

echo "\t\t\t\t<th>\n";

echo "\t\t\t\t\tCUM\n";

echo "\t\t\t\t</th>\n";

echo "\t\t\t</tr>\n";

echo "\t\t</thead>\n";

echo "\t\t<tbody>\n";

foreach($nombres as $name => $nota){

echo "\t\t\t<tr>\n";

echo "\t\t\t\t<td>\n";

echo "\t\t\t\t\t$name\n</td>\n";

echo "\t\t\t\t<td class=\"cum\">\n";

echo "\t\t\t\t\t" . number\_format($nota, 1, '.', ',') . "\n";

echo "\t\t\t\t</td>\n";

echo "\t\t\t</tr>\n";

}

echo "\t\t</tbody>\n";

echo "\t</table>\n";

echo "</article>\n";

echo "</section>\n";

echo "</body>\n";

echo "</html>";

?>

**Archivo 2: notasalumnos.css**

body {

background-color:#6cbd70;

filter:progid:DXImageTransform.Microsoft.gradient(GradientType=0,startColorstr=#6cbd70, endColorstr=#b086ab);

background-image:-moz-linear-gradient(top, #6cbd70 0%, #b086ab 100%);

background-image:-webkit-linear-gradient(top, #6cbd70 0%, #b086ab 100%);

background-image:-ms-linear-gradient(top, #6cbd70 0%, #b086ab 100%);

background-image:linear-gradient(top, #6cbd70 0%, #b086ab 100%);

background-image:-o-linear-gradient(top, #6cbd70 0%, #b086ab 100%);

background-image:-webkit-gradient(linear, right top, right bottom, color-

stop(0%,#6cbd70), color-stop(100%,#b086ab));

}

h1 {

font:bold 20pt "Myriad Pro","Bodoni MT Black",Rockwell,Helvetica,sans-serif;

color:#033006;

text-align:center;

}

table {

border-collapse:collapse;

border:8px outset DarkSlateBlue;

left:-10%;

margin-left:50%;

position:relative;

width:20%;

}

tr {

border:1px solid DarkSlateBlue;

}

th {

border:1px solid DarkSlateBlue;

background-color:rgb(0,100,100);

color:Gold;

font:bold 12pt "Lucida Sans","Century Gothic",Helvetica,sans-serif;

line-height:1.4em;

text-align:center;

}

td {

color:rgb(60,0,90);

font:normal 11pt "Century Gothic","Lucida Sans",Helvitica,sans-serif;

}

tr:hover {

background-color:#f5f59d;

filter:progid:DXImageTransform.Microsoft.gradient(GradientType=0,startColorstr=#f5f59d, endColorstr=#e4f222);

background-image:-moz-linear-gradient(top, #f5f59d 0%, #e4f222 100%);

background-image:-webkit-linear-gradient(top, #f5f59d 0%, #e4f222 100%);

background-image:-ms-linear-gradient(top, #f5f59d 0%, #e4f222 100%);

background-image:linear-gradient(top, #f5f59d 0%, #e4f222 100%);

background-image:-o-linear-gradient(top, #f5f59d 0%, #e4f222 100%);

background-image:-webkit-gradient(linear, right top, right bottom, color-

stop(0%,#f5f59d), color-stop(100%,#e4f222));

color: rgb(60,0,90);

font-weight: Bold;

}

.cum {

text-align:center;

}

**Ejemplo #4: El siguiente ejemplo muestra una aplicación orientada a objetos donde se define una clase auto con propiedades como la marca, el modelo, el color y una imagen del auto, un método constructor para inicializar las propiedades del objeto y un método más para mostrar la información de todos los autos existentes.**

**Archivo 1: autos.class.php**

<?php

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Descripción: Ejemplo de definición de una clase auto \*

\* Autor: Ricardo Ernesto Elías Guandique \*

\* Archivo: autopoo.php \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Definición de la clase

class auto {

//Propiedades de la clase auto

private $marca;

private $modelo;

private $color;

private $image;

//Método constructor

function \_\_construct($marca='Peugeot', $modelo='307', $color='Gray',

$image='img/peugeot.jpg'){

$this->marca = $marca;

$this->modelo = $modelo;

$this->color = $color;

$this->image = $image;

}

//Métodos de la clase

function mostrar(){

$tabla = "<table style=\"width:200;border:ridge 5px rgb(200,50,150)\">\n";

$tabla .= "<caption>Compra un " . $this->marca . "</caption>";

$tabla .= "<tr>\n<td style=\"width:35%;\">MARCA</td>\n";

$tabla .= "<td style=\"width:35%\">" . $this->marca . "</td>\n";

$tabla .= "<td rowspan=\"3\" style=\"width:35%\"><img src=\"" . $this->image .

"\"></td></tr>";

$tabla .= "<tr>\n<td>MODELO</td>\n";

$tabla .= "<td>" . $this->modelo . "</td>\n</tr>\n";

$tabla .= "<tr>\n<td>COLOR</td>\n";

$tabla .= "<td>\n" . $this->color . "</td>\n</tr>\n";

$tabla .= "</table>\n";

echo $tabla;

}

}

?>

**Archivo 2: autospoo.php**

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="utf-8" />

<title>Venta de autos</title>

<link type="text/css" rel="stylesheet" media="screen" href="css/autospoo.css" />

<!--[if lt IE 9]>

<script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>

<![endif]-->

</head>

<body>

<header>

<h1>Autos disponibles</h1>

</header>

<section>

<article>

<?php

//Incluyendo el archivo de clase

include\_once("autos.class.php");

//Creando los objetos para cada tipo de auto. Notar que se están

//asignando a elementos de una matriz que tendrá por nombre $movil

$movil[0] = new auto("Peugeot", "307", "Gris", "img/peugeot.jpg");

$movil[1] = new auto("Renault", "Clio", "Marron", "img/renaultclio.jpg");

$movil[2] = new auto("BMW", "Serie6", "Azul", "img/bmwserie6.jpg");

//Esta llamada mostrará los valores por defecto en los argumentos

//del método constructor.

$movil[3] = new auto();

//Mostrando la tabla con los autos disponibles

for($i=0; $i<count($movil); $i++){

$movil[$i]->mostrar();

}

?>

</article>

</section>

</body>

</html>

**Archivo 3: autospoo.css**

body {

font-size: 16px;

}

caption {

background-color:Pink;

font-family:Helvetica,monoscope,sans-serif;

font-size:11pt;

font-weight:bold;

font-style:italic;

color:ForestGreen;

}

td {

border-collapse:collapse;

border-width:2px;

border-style:ridge;

border-color:maroon;

font: normal 0.85em "Lucida Sans",Helvetica,sans-serif;

padding: 0.25em 0.2em;

}

**Ejemplo #5: El siguiente ejemplo crea un script con salida .json que contiene una serie de instancias con la información de las principales exportaciones e importaciones de El Salvador y esta es almacenada temporalmente en un arreglo. Posteriormente la información es convertida al formato de salida.**

Archivo 1: Rubro.php

|  |
| --- |
| <?php  class Rubro{  public $type;  public $description;  public $value;  }  ?> |

Archivo 2: Servicio.php

|  |
| --- |
| <?php  include "Rubro.php";  $reporte["pais"]="El Salvador";  $exportaciones = array();  $importaciones = array();  $rubro = new Rubro;  $rubro->type ="Exportacion";  $rubro->description="Tejer Camisetas";  $rubro->value=783000000;  $exportaciones[]=$rubro;  $rubro = new Rubro;  $rubro->type ="Exportacion";  $rubro->description="Condensadores electricos";  $rubro->value=365000000;  $exportaciones[]=$rubro;  $rubro = new Rubro;  $rubro->type ="Exportacion";  $rubro->description="Jersey de punto";  $rubro->value=309000000;  $exportaciones[]=$rubro;  $rubro = new Rubro;  $rubro->type ="Exportacion";  $rubro->description="Ropa interior de hombre";  $rubro->value=226000000;  $exportaciones[]=$rubro;  $rubro = new Rubro;  $rubro->type ="Importacion";  $rubro->description="Refinado de petroleo";  $rubro->value=889000000;  $importaciones[]=$rubro;  $rubro = new Rubro;  $rubro->type ="Importacion";  $rubro->description="Medicamentos envasados";  $rubro->value=282000000;  $importaciones[]=$rubro;  $rubro = new Rubro;  $rubro->type ="Importacion";  $rubro->description="Coches";  $rubro->value=190000000;  $importaciones[]=$rubro;  $reporte["exportaciones"]=$exportaciones;  $reporte["importaciones"]=$importaciones;  header('Content-type: application/json');  echo json\_encode($reporte);    ?> |

* **IV Ejercicios complementarios**
  1. Crear un script en PHP que pemita generar una tabla de multiplicar a partir de un formulario que solicite el número de la tabla. El resultado debe ser html.
  2. Crear un script en php que sea capaz de calcular una base y potencia. La potencia puede ser positiva, negativa o cero (0).
  3. Investigue qué es mysqli y para qué se utiliza en php.
  4. Investigue cómo agregar boostrap a sus páginas html. Agregue estilo a sus páginas de los ejercicios 1 y 2.
  5. Crear una clase en PHP para registrar los países de Centro América. La información que debe soportar es El nombre del país, la población, la extensión territorial y una url de bandera. Posteriormente debe crear un script en php que permita recibir un parámetro (query string o por medio de post) con el nombre del país y a partir de ese resultado generar un json.
  6. Para los datos generados en el ejercicio anterior, crear una aplicación en Android para mostrar la información retornada.