PREGUNTAS FRECUENTES SOBRE PHP

¿Qué es PHP?
Las siglas PHP significan Hypertext Preprocessor, en sí PHP es un lenguaje destinado en su mayoría a la Web, aunque existen aplicaciones PHP que pueden correr en consola. El PHP no es un lenguaje orientado a objetos.

¿Qué extensión usa?
La extensión que usa PHP es "*.php", ej. pagina.php, siendo la página principal por defecto index.php

¿Qué tipo de aplicaciones puedo realizar en PHP?
PHP no tiene límites, puedes realizar cualquier caso de la vida real y además crear e implementar tus propias bibliotecas o métodos.

¿Qué es una biblioteca?
Una biblioteca es un archivo en el cual guardas variables u objetos que serán usados en muchas páginas, esto con el objeto de la reutilización de códigos y de no tener que escribir de nuevo las sentencias en página.

¿Puedo combinar PHP con HTML?
Sí, de hecho lo normal es diseñar la página PHP en HTML y luego agregar las sentencias de programación PHP.

¿Puede ejecutar una página PHP haciendo doble clic sobre ella igual que las páginas HTML?
No, PHP se ejecuta en el servidor Web, y no en la computadora del usuario, por lo que para ejecutar una página PHP debe estar colgada en un servidor o dentro de un servidor local de prueba.

IMPRIMIR EN PANTALLA Y USO DE VARIABLES

Imprimir en pantalla: Esta página que elaboraremos nos mostrará en pantalla la frase "Hola mundo", la haremos íntegramente en PHP, el código sería el siguiente:

```
<?php
        echo "Hola mundo";
?>
```

Como podemos ver, se utiliza *echo* para mostrar el texto entre comillas en pantalla, todas las sentencias se acaban con punto y coma (;)

Uso de variables: Del ejemplo anterior usaremos variables para notar la diferencia de código:

```
<?php
        $bienvenida="Hola mundo";
        echo $bienvenida;
?>
```

Como podemos ver, se utiliza la variable bienvenida que almacena el texto "Hola mundo", notamos que todas las variables comienzan con signo de dólar ($), para imprimir en pantalla ya no ponemos la variable entre comillas porque es la variable la qué contiene el texto.

También pueden juntarse varias variables y textos a la vez, esto en programación se llama concatenar.

Concatenar texto y variables: Para concatenar textos y variables usaremos el siguiente código:

© 2008 José Luis Yapur Vallejos. Todos los derechos reservados

```
<?php
        $nombre=”José Luis”;
        $saludo=”Hola”;
        $dia=”16/10/2007”;
        echo $saludo.” “.$nombre.” Hoy es: “.$dia.” Gracias por ingresar al sistema”;
?>
```

Este código imprimirá en pantalla el siguiente mensaje:
“Hola José Luis Hoy es: 16/10/1987 Gracias por ingresar al sistema”
Como podemos ver la unión entre varias variables y textos es el punto (.), también se hubiera podido crear la variable mensaje que contenga todo el texto y luego imprimirlo.

Creación de variable de cadena concatenada: En el siguiente código veremos como crear una variable de cadena concatenada:

```
<?php
        $nombre=”José Luis”;
        $saludo=”Hola”;
        $dia=”16/10/2007”;
        $mensaje=$saludo.” “.$nombre.” Hoy es: “.$dia.” Gracias por ingresar al
        sistema”;
        echo $mensaje;
?>
```

Lo más común es usar este ejemplo ya que si debemos modificar el texto que se imprimirá en pantalla modificaremos variable, la cual localizaremos rápidamente.

Observación: Toda variable que contenga un valor dentro de comillas dobles (“”) o comillas simples (‘ ‘) es un texto.
Los números que se almacenan en variables no deben llevar comillas o serán convertidos a texto.
Las variables no deben contener caracteres distintos a los del rango [a-z], eso excluye a (“ñ”, “`”, “.”, “ “, “!”, etc)

Declaración de variables numéricas: Para crear una variable numérica deberemos seguir el siguiente ejemplo:

```
<?php
        $numero=1523;
        echo “El número ingresado es: “.$numero;
?>
```

Concatenar números: Para concatenar números usaremos el mismo método que con los textos:

```
<?php
        $num1=12;
        $num2=13;
        $num=$num1.$num2;
        echo “Sea un número $\overline{abcd}$ y ab=12, cd=13, entonces $\overline{abcd}$ =”.$num;
?>
```

© 2008 José Luis Yapur Vallejos. Todos los derechos reservados

OPERACIONES BÁSICAS Y REDONDEO

Suma: La suma en PHP se hace de la siguiente manera:

```
<?php
        $num1=12;
        $num2=15;
        $suma=$num1+$num2;
        echo “La suma de “.$num1.”+”.$num2.”=”.$suma;
?>
```

Resta: La resta en PHP se hace de la siguiente manera:

```
<?php
        $num1=15;
        $num2=10;
        $resta=$num1-$num2;
        echo “La resta de “.$num1.”+”.$num2.”=”.$resta;
?>
```

Producto: El producto en PHP se hace de la siguiente manera:

```
<?php
        $num1=15;
        $num2=10;
        $producto=$num1*$num2;
        echo “El producto de “.$num1.”*”.$num2.”=”.$producto;
?>
```

Cociente: El cociente en PHP se hace de la siguiente manera:

```
<?php
        $num1=15;
        $num2=10;
        $cociente=$num1/$num2;
        echo “El cociente de “.$num1.”/”.$num2.”=”.$cociente;
?>
```

Resto: El resto es aquella operación que muestra el residuo de la división de 2 números, se usa para determinar múltiplos y divisores. El resto (%) en PHP se hace de la siguiente manera:

```
<?php
        $num1=15;
        $num2=10;
        $resto=$num1%$num2;
        echo “El resto de “.$num1.”%”.$num2.”=”.$resto;
?>
```

Redondeo: Para redondear se encierra el número en round();
Ejemplo:

```
<?php
        $num=324.234234;
        $redondeado=round($num); //Redondeado sin decimales
        $redondeado2=round($num*100)/100; //Redondeado con 2 decimales
?>
```

Operaciones combinadas:

© 2008 José Luis Yapur Vallejos. Todos los derechos reservados

Problema cotidiano 1: Una persona debe determinar el numerador de una fracción basándose en el algoritmo de la división que es el siguiente: $d(x) = R(x) + (V(x).D(x)), \text{en} : \frac{d(x)}{V(x)} = D(x) \wedge \text{Re}\, siduo = R(x)$, elaborar una solución que permita resolver su problema sin mayor inconveniente, se sabe que V(x)=15, D(x)=3, R(x)=0.

Solución:

```
<?php
        $V=15;
        $d=3;
        $R=0;
        $w=$R+($V*$d); //$w es el denominador
        echo "El denominador es: ".$w;
?>
```

Problema cotidiano 2: Una persona recibió su estado de cuenta de su tarjeta de crédito, pero quiere comprobar que la suma con aplicación de tasas de interés e impuestos sean correctas, para ello sabe lo siguiente:

Compró 2 botellas de Blue de Ralph Lauren a 67.66 dolares cada uno.
Compró 4 latas de leche a 2 soles cada una.
Compró 2 cajas de corn flakes a 7.25 soles cada una.
Compró una lata de café a 8.40.
Además a mitad de mes depositó el 50% del total del mes sin incluir tasas de interés ni impuestos.
El impuesto es 19% del total.
El monto por mantenimiento de cuenta es de 0.7% del total con impuestos.
El recibo calcula el total con 2 cifras decimales.

Calcular el total a pagar a fin de mes.

Solución:

```
<?php
        $tasaDeCambio=3.26;
        $precio1=2*67.66*$tasaDeCambio;
        $precio2=4*2;
        $precio3=2*7.25;
        $precio4=8.4;
        $totalSinImpuesto=$precio1+$precio2+$precio3+$precio4;
        $deposito=50*$totalSinImpuesto/100;
        $totalSinImpuesto=$totalSinImpuesto-$deposito;
        $impuesto=0.19;
        $manteCuenta=0.007;
        $total=$totalSinImpuesto*0.19*0.007;
        $total=round($total*100)/100;
        echo “El total a pagar es: “.$total;
?>
```

© 2008 José Luis Yapur Vallejos. Todos los derechos reservados

ESTRUCTURAS LÓGICAS DE DECISIÓN Y VECTORES

Lógica preposicional: Al igual qué la lógica que llevamos en los cursos de filosofía o matemática, la programación se vale de sentencias lógicas como son:

Preposición	Símbolo
Y (conjunción)	&&
O (disyunción)	||
Igualdad	==
Menor o igual	<=
Mayor o igual	>=
Negación	!
Desigualdad	!=

Estructura de decisión SI (if): Esta estructura nos permite crear condiciones de la manera siguiente, por ejemplo sí "a" es igual a "b" entonces, imprimir en pantalla "a igual a b". Declaración de condición SI;

```
<?php
if(condicion){
        sentencia1;
        sentencia2;
        .
        .
        .
        sentencia n;
}
?>
```

Ejemplo1: Determinar si un número es múltiplo de 2;

```
<?php
        $num=23;
        if($num%2){
        echo "El número es múltiplo de 2";
        }
?>
```

Ejemplo2: Determinar sí el número 1 es mayor que el número 2;

```
<?php
        $num1=12;
        $num2=14;
        //Sí numero 1 no es mayor que número 2 no nos mostrará nada
        $mensaje="";
        //Comenzamos la condición
        if($num1>$num2){
                $mensaje="Número 1 es mayor que número 2";
        }
?>
```

© 2008 José Luis Yapur Vallejos. Todos los derechos reservados

Estructura de decisión SI/SINO (if/else): La estructura de decisión SI/SINO es más completa ya que nos permite procesar información tanto si la condición es falsa como si es verdadera. Se declara de la siguiente manera:

```
<?php
	if(condicion){
		sentencia1		.
		.
		sentencia n
	}
	else{
		sentencia1
		.
		sentencia n
	}
?>
```

Ejemplo1: Elaborar una solución que permita determinar si un número es múltiplo de otro.

```
<?php
	$num1=15;
	$num2=3;
	$mensaje="";
	if($num1%$num2==0){
		$mensaje=$num1." es múltiplo de ".$num2;
	}
	else{
		$mensaje=$num1." no es múltiplo de ".$num2;
	}
	echo $mensaje;
?>
```

Ejemplo2: Elaborar una solución que permita calcular el promedio de 3 notas y determinar si aprobó o no. (Aprobado en azul y desaprobado en rojo)

```
<?php
	$n1=15;
	$n2=3;
	$n3=10;
	$prom=($n1+$n2+$n3)/3;
	$mensaje="";
	if($prom>10){
		$mensaje="<font color=blue>Aprobado</font>";
	}
	else{
		$mensaje="<font color=red>Desaprobado</font>";
	}
	echo $mensaje;
?>
```

Observación: Puede usarse código HTML dentro de PHP, pero como vemos los atributos no deben ir entre comillas.

© 2008 José Luis Yapur Vallejos. Todos los derechos reservados

Ejemplo3: Determinar si un año es bisiesto y dependiendo de eso mostrar el número de días del año.

```
<?php
	$ano=2007;
	$mensaje="";
	if($ano%4==0){
		$mensaje="366 días - Bisiesto";
	}
	else{
		$mensaje="365 días - No bisiesto";
	}
	echo $mensaje;
?>
```

SI/SINO Anidados: Las estructuras SI/SINO anidados sirven para crear condiciones dentro de otras condiciones. La estructura es la siguiente:

```
<?php
	if(condicion){
		if(condicion){
		sentencia
		}
		else{
			if(condicion){
			sentencia
			}
			else{
			sentencia
			}
		}
	}
	else{
		if(condicion){
		sentencia
		}
		else{
			if(condicion){
			sentencia
			}
			else{
			sentencia
			}
		}
	}
?>
```

Ejemplo1: Hallar el área de un cuadrado de base 15, si el área es menor que 10, pero mayor 5 entonces indicar que el cuadrado es pequeño, si el área es igual a 7 mostrar

© 2008 José Luis Yapur Vallejos. Todos los derechos reservados

el cuadrado es pequeño de área 7; si el área es mayor o igual que 10 pero menor que 30 mostrar un mensaje que diga el cuadrado es pequeño, de lo contrario, si es mayor o igual que 30 mostrar el mensaje que diga el cuadrado es grande.

Observación: Sabemos que el área del cuadrado es l^2, por lo que usaremos un *método de la clase matemática* importante en PHP, este método se llama "pow", y se declara de la siguiente manera, sea: $x = a^b$, entonces $x=pow(a, b);

Solución:

```
<?php
$base=15;
$area=pow($base,2); //Usamos el método pow
if($area<10&&$area>5){
$mensaje="El cuadrado es pequeño";
        if($area==7){
                $mensaje="El cuadrado es pequeño de área 7";
        }
}
else{
        if($area<30){
                $mensaje="El cuadrado es mediano";
        }
        else{
                $mensaje="El cuadrado es grande";
        }
}
?>
```

Ejemplo2: De 3 números hallar los números en orden ascendente

```
<?php
$a=12;
$b=12;
$c=14;
$mayor=0;
$menor=0;
$intermedio=0;
if($a<$b){
        $mayor=$a;
        if($b>$c){
                $intermedio=$b;
                $menor=$c
        }
        else{
                $intermedio=$c;
                $menor=$b;
        }
}
else{
        if($b>$c){
        $mayor=$b;
                if($a>$c){
                        $intermedio=$a;
                        $menor=$c;
                }
                else{
                        $intermedio=$c;
                        $menor=$c;
```

© 2008 José Luis Yapur Vallejos. Todos los derechos reservados

```
                }
        }
        else{
        $mayor=$c;
                if($a>$b){
                        $intermedio=$a;
                        $menor=$b;
                }
                else{
                        $intermedio=$b;
                        $menor=$a;
                }
        }
}
echo $mayor."\n".$intermedio."\n".$menor;

?>
```

Decisión con casos CAMBIAR (switch): Esto se usa para no elaborar varias condiciones, para ello se usa el siguiente código:

```
<?php
$opcion=num;
switch($opcion){
case 1: sentencias
break;
case 2: sentencias
break;
case n: sentencias
break;
}
?>
```

Problema cotidiano 1: Un estudiante de ingeniería debe elaborar una calculadora que le permita calcular suma, resta, división, producto y potencia de 2 números. (Usar Switch)

```
<?php
$num1=12;
$num2=14;
$opcion=1;
echo "Seleccione la opción a calcular [1]Suma [2]Resta [3]Cociente [4]Producto [5]Potencia";
switch($opcion){
case 1:
$operacion=$num1+$num2;
break;
case 2:
$operacion=$num1-$num;
break;
case 3:
$operacion=$num1/$num;
break;
case 4:
$operacion=$num1*$num2;
break;
case 5:
$operacion=pow($num1,$num2);
```

© 2008 José Luis Yapur Vallejos. Todos los derechos reservados

```
break;
}
echo $operacion;
?>
```

Repetición con decisión MIENTRAS (while): La estructura de repetición MIENTRAS se usa para repetir sentencias muchas veces hasta que la condición lógica deje de cumplirse.

Ejemplo1: Solución que cuenta del 1 al 20.

```
<?php
$i=0;
while($i<=20){ //Mientras $i sea menor o igual que 20, hacer {
$i++; //$i es un acumulador, que aumenta de uno en uno a medida que se va repitiendo la aplicación.
echo $i."</br>";
}
?>
```

Ejemplo2: Desarrollar una solución que permita mostrar una progresión aritmética con razón 2, que comienza desde 0 y termina en 9000.

```
<?php
$i=0;
while($i<=9000){
        if($i%2==0){
        echo $i;
        echo "</br>";
        }
$i++;
}
echo $acumulador;
?>
```

Repetición con DESDE HASTA (for): Es la más útil de todas, declara, condiciona y aumenta el contador en un solo proceso.

```
<?php
for($i=0; $i<num; $i++){
        sentencias;
        }
?>
```

Ejemplo 1: Imprima los números múltiplos de 2 desde 0 hasta 9999.

```
<?php
for($i=0; $i<9999; $i++){
        if($i%2==0){
        echo $i;
        echo "</br>";
        }
}
?>
```

Vectores: Un vector es un arreglo de una sola variable que almacena n datos, además es de naturaleza volátil, lo que significa que al cerrarse la aplicación PHP o entrar a otro enlace todo lo almacenado en dicho vector será borrado.

Un vector se puede declarar de 2 maneras:

$vector[longitud];

$vector={elementos};

Ejemplo1: Modificación del ejemplo anterior mostrando datos almacenados en vector:

```
<?php
```

© 2008 José Luis Yapur Vallejos. Todos los derechos reservados

```
$i=0; $j=0;
$vector[10000];
//INGRESAMOS DATOS AL VECTOR
while($i<=9000){
        if($i%2==0){
                if($i!=0){
                $vector[$i]=$i;
                }
                else{
                $vector[$i-1]=$i;
                }
        }
        $i++;
}
//RECORREMOS EL VECTOR
while($j<$i){
echo $vector[$j];
echo "</br>";
$j++;
}
?>
```

Ejemplo 2: Suma de todos los números múltiplos de 2 mayores que 20 pero menores que 10,000:

```
<?php
//COMENZAMOS DESDE 21 POR SER LOS NUMERO MAYORES QUE 20
$i=21; $num=0; $conta=0; $suma=0; $num2=0 ;$vector[10000-21];
while($i<=10000){
        if($i%2==0){
                $vector[$num2]=$i;
                $conta++;
        }
        $i++;
        $num2++:
}
//RECORREMOS EL VECTOR
while($num<$conta){
//+= es lo mismo que decir variable=variable+valor
        $suma+=$vector[$num];
        $num++;
}
echo "La suma es:"+$suma;
?>
```

© 2008 José Luis Yapur Vallejos. Todos los derechos reservados

INTEGRACIÓN PHP CON HTML

Manejo de formularios: El PHP se vale de los formularios en HTML para su correcta ejecución, es así que PHP obtiene los datos a trabajar valiéndose de las variables de tipo $_POST['variable'] y $_GET['variable'].

$_POST: También conocida como variable de formulario. Este tipo de variables se utilizan sí y solo sí obtenemos información de un formulario HTML que procesa la información mediante el método POST. Imaginemos que en un formulario tenemos un campo de texto llamado nombres, entonces, al hacer clic en enviar el script PHP debe procesar el nombre y mostrarlo:

En el documento prueba.php copiamos lo siguiente, dentro del body.

```
<form id="form1" name="form1" method="post" action="">
  <input name="nombres" type="text" id="nombres" />
  <input type="submit" name="Submit" value="Enviar" />
</form>
<?php
//NULL representa vacio
if($_POST['nombres']!=NULL){
echo "Hola ".$_POST['nombres'];
}
?>
```

$_GET: También conocida como variable URL. Este tipo de variables se utilizan sí y solo sí obtenemos información de un formulario HTML que procesa la información mediante el método GET. Imaginemos que en un formulario tenemos un campo de texto llamado nombres, entonces, al hacer clic en enviar el script PHP debe procesar el nombre y mostrarlo:

En el documento prueba.php copiamos lo siguiente, dentro del body.

```
<form id="form1" name="form1" method="post" action="">
  <input name="nombres" type="text" id="nombres" />
  <input type="submit" name="Submit" value="Enviar" />
</form>
<?php
//NULL representa vacio
if($_GET['nombres']!=NULL){
echo "Hola ".$_ GET['nombres'];
}
?>
```

¿Cuál es la diferencia entre ambos?: La diferencia es que el método POST oculta los datos enviados (el proceso no es visto por el usuario), y el método GET, muestra los datos enviados en la barra de url, ejemplo: http://direccion.com/pagina.php?usuario=jose&password=miclave&nombre=jose%20luis
El método POST solo se llama desde un formulario, mientras que el método GET puede ser llamado desde cualquier parte de la página, incluso un enlace.

© 2008 José Luis Yapur Vallejos. Todos los derechos reservados

¡Pruébate!

1. Dar una solución al siguiente problema de la vida real: una empresa tiene una cantidad n de trabajadores, cada uno de ellos recibe un sueldo correspondiente a su categoría, existen 3 categorías:

Categoría 1	S/.1200
Categoría 2	S/.1598.23
Categoría 3	S/.2393.23

Además se descuenta el 17% del total del sueldo por categoría a cada trabajador, y si el mes ingresado es julio o diciembre entonces aumentar el 100% del total del sueldo contando el descuento. Se debe calcular además el total pagado a la categoría 1, a la 2 y a la 3, determinar el porcentaje de trabajadores de cada categoría y el total pagado a todos los trabajadores de la empresa.
Almacenar datos en vector.

2. Dar una solución al siguiente problema de la vida real: Se debe crear un calendario en PHP para el mes de febrero, considerar si el año es o no bisiesto, dar la solución a ese problema.

3. Dar una solución al siguiente problema de la vida real: Crear un menú desplegable que contenga los años desde el 1910 hasta el 2007, incluyendo las etiquetas y valores correspondientes.

4. Dar una solución al siguiente problema de la vida real: Hacer un programa que calcule el monto a pagar por la compra de "n" productos. Que calcule también el IGV (19%) correspondiente y el monto total a pagar (incluido el IGV).
Considere:
   - Los productos que cuestan más de S/. 50 tienen 10% de descuento.
   - Si la cantidad de productos comprados es mayor a 10, que se haga un descuento de 2% sobre el total a pagar (antes de aplicarle el IGV).
   - Si la cantidad de productos es menor o igual a 5 que muestre un mensaje de envío a Caja Rápida.
   - Si el precio de un producto es 0 que envíe el mensaje "Producto Gratis", y que no permita más de 3 productos gratis.
   - Que al terminar la operación pregunte por la modalidad de pago, con tarjeta o en efectivo; en caso de usar tarjeta que aplique un recargo del 10% sobre el total (incluido el IGV).

© 2008 José Luis Yapur Vallejos. Todos los derechos reservados

MANEJO DE MÉTODOS Y FUNCIONES

Variables globales: Son todas aquellas declaradas fuera de los métodos y que pueden ser leídas por todos los métodos.
Variables locales: Son todas aquellas declaradas dentro de los métodos y solo son válidas en los métodos en los que fueron declarados.
Métodos que no reciben ni retornan valor: Estos métodos son usados frecuentemente para trabajar con variable globales, o simplemente para ser ejecutados en el momento en que son llamados.

```
<?php
function nombreFuncion(){
        sintaxis;
}
?>
```

Métodos que reciben pero retornan valor: Estos métodos son usados frecuentemente para trabajar entre métodos y mostrar un resultado en el instante en que es llamado.

```
<?php
function nombreFuncion($variable1, $variable2, ...., $variableN){
        $suma=$variable1 + $variable2+....+$variableN;
}
?>
```

Métodos que no reciben pero retornan valor: Estos métodos son usados para que una operación creada dentro del método sea guardada o mostrada fuera del método

```
<?php
function nombreFuncion(){
        return $variable;
}
?>
```

Métodos que reciben y retornan valor: Estos métodos son los más usados por los programadores, se usan frecuentemente para reutilización de código.

```
<?php
function nombreFuncion($variable1, $variable2, ...., $variableN){
        return $suma=$variable1 + $variable2+....+$variableN;
}
?>
```

Llamado de métodos: Para que un método se ejecute debe ser llamado indicando el nombre del método: nombreFuncion();

Ejemplo 1: Crear un método que calcule la suma de 2 números (n1 y n2) recibidos de un formulario los cuales son variables globales.

```
<?php
$n1=$_GET['n1'];
$n2=$_GET['n2'];

        function suma(){
                echo $n1+n2;
        }
        suma();
?>
```

© 2008 José Luis Yapur Vallejos. Todos los derechos reservados

Ejemplo 2: Implemente el método anterior pero procurando que el método retorne valor.

```
<?php
$n1=$_GET['n1'];
$n2=$_GET['n2'];

        function suma(){
                $n1+n2;
        }
        echo suma();
?>
```

Ejemplo 3: Implemente el método anterior pero desde 4 métodos, dos que almacenen los números, otro que los sume y otro que muestre la suma.

```
<?php
//variables
function N1(){
                return $_GET['n1'];
}
function N2(){
                return $_GET['n2'];
}
//metodo que recibe datos y los suma
function suma($n1, $n2){
                return $n1+n2;
}
//metodo que recibe la suma y la muestra
function mostrar(){
                echo suma(N1(), N2());
}
//llamada al método
mostrar();
?>
```

© 2008 José Luis Yapur Vallejos. Todos los derechos reservados

COOKIES, SESIONES, Y MANEJO DE HORAS Y FECHAS

Cookies: Fragmento de información que se almacena en el ordenador del usuario que accede a un sitio Web.
Sesión: Información que se almacena en el servidor de manera temporal, hasta que el usuario decida destruir la sesión, se utiliza normalmente para el manejo de login de usuarios.

```
<?php
//Se declara una cookie
setcookie(string nombre, string valor, int fecha, string path, string dominio, int segura);
//Se lee una cookie
$_cookie['nombre'];
$http_cookie_vars['nombre'];
?>
```

Ejemplo 1: Creamos una cookie que expire el 1 de Enero del 2010 a las 0.0.0 horas.

```
<?php
$fecha=mktime(0,0,0,1,1,2010);
setcookie('micookie',$valor,$fecha,' ',' ',0);
 /*Ahora visualizamos los campos*/
echo $micookie;
echo $http_cookie_vars['micookie'];
?>
```

Las Cookies tienen una serie de campos, como mínimo se envía el nombre y el valor de la cookie. Otros campos de las cookies son:

Fecha: las cookies pueden tener una fecha de expiración. Hay que destacar que esta fecha es un número entero, por lo que habrá que calcularla con las funciones mktime() y time() de PHP.
Path: especifica el subconjunto de URLs en el servidor de origen para las cuales se aplica la cookie.
Dominio: establece el dominio en el cual es válida la cookie.
Segura: indica si la cookie necesita una conexión segura, toma valores enteros, si no queremos especificar nada, poner a cero.

Deberás tener en cuenta que muchos de los navegadores no aceptan el uso de cookies por medidas de seguridad por lo que se deberán tomar medidas de acceso y funcionalidad del sistema para que soporte medios alternativos.

Para el caso de las sesiones:
Se inicializa:

```
session_start();
```

//Esto debe ir antes que cualquier otra línea de código o nos mostrará error.
Se declara:

```
$_SESSION[‘nombre’] = valor;
```

Se lee:

```
$_SESSION[‘nombre’];
```

Se destruye la sesión:

```
destroy_session();
```

© 2008 José Luis Yapur Vallejos. Todos los derechos reservados

Ejemplo 2: Desarrolle un código que almacene en una variable de sesión el nombre de un usuario ingresado por un formulario mediante el método POST.

```
<?php
session_start();
if(isset($_POST['nombre']))
{
$_SESSION['nomUsuario'] = $_POST['nombre'];
}
//Leer en cualquier página del sitio
echo $_SESSION['nomUsuario'];
?>
```

Fechas y horas: Son necesarias para el manejo de eventos, y nos dan muchas opciones como timestamp y date**.**

```
<?php
//Declaración y lectura función date:
date("d-m-Y");
//esto nos mostrará día-mes-año(YYYY)
time()
//esto nos mostrará diamesañohoraminseg
/*Para horas hacemos lo mismo, usamos la función date, pero esta vez con la
siguiente variación:*/
date("h:i:s");
//esto mostrará hora:minuto:segundo
?>
```

© 2008 José Luis Yapur Vallejos. Todos los derechos reservados

CLASES

Son declaraciones o abstracciones de objetos, aunque en PHP no son muy usadas ni difundidas, existen variedad de opciones para el manejo de las mismas así como accesibilidad en la declaración y uso.

Para iniciar una clase debemos indicar lo siguiente:

```
class nomClase
{
        variables globales si es que hubieran
        ….
        metodo1(){
        }
}
```

Para declarar variables globales haremos lo siguiente:

```
class nomClase
{
        var $nomVariable;
        public $nomVariable1;
        private $nomVariable2;

        metodo1(){
        }
}
```

Var: define una variable como default.
Public: define una variable como publica, lo que significa que puede ser acezada fuera de la clase.
Private: define una variable como privada, lo que significa que no puede ser acezada fuera de la clase.

Para usar un método constructor:

```
class nomClase
{
        var $nomVariable;
        public $nomVariable1;
        private $nomVariable2;

        public function __construct($param1, $param2){
                contenido
        }
}
```

© 2008 José Luis Yapur Vallejos. Todos los derechos reservados

Para darle valores a las variables globales desde el método constructor:

```
class nomClase
{
	var $a;
	public $b;
	private $c;

	public function __construct($a, $b, $c){
		$this->a = $a;
		$this->b = $b;
		$this->c = $c;
	}
}
```

Usamos $this-> para indicar que esa es la variable global y no la que recibe como parámetro el método constructor.

Ahora añadiremos algunos métodos:

```
<?php
class nomClase
{
   var $a;
   public $b;
   private $c;

   public function __construct($a, $b, $c)
   {
      $this->a = a;
      $this->b = b;
      $this->c = c;
   }

   //como c es privada creamos sus metodos GET y SET para poder trabajarla fuera
de la clase
   public function getC()
   {
        return $this->c;
   }
   public function setC($c)
   {
        $this->c = $c;
   }

   //devuelve la suma de a + b + c
   public function sumaABC()
   {
        return $this->a + $this->b + $this->c;
   }
}

?>
```

© 2008 José Luis Yapur Vallejos. Todos los derechos reservados

Ejemplo 1: Desarrollo de clase conexión

```
class Conexion
{

	var $servidor;
	var $usuario;
	var $password;
	var $bdatos;
	var $conexion;

	public function __construct($Servidor, $Usuario, $Password, $Bdatos)
	{
		$this->servidor = $Servidor;
		$this->usuario = $Usuario;
		$this->password = $Password;
		$this->bdatos = $Bdatos;
		$this->conexion = @mysql_connect($this->servidor, $this->usuario,
$this->password) or die("ERROR 701 -CONSTRUCTOR- Error de conexión,
parametros incorrectos");
	}

	public function mostrar($sentencia)
	{
		@mysql_select_db($this->bdatos, $this->conexion) or die("ERROR 702
-MOSTRAR- No se pudo seleccionar la base de datos");
		$consulta = @mysql_query($sentencia, $this->conexion) or
die("ERROR 703 -MOSTRAR- No se puede realizar la consulta");
		$consultaMAX = mysql_num_fields($consulta);
		$w = 0;
		for($i = 0; $i < $consultaMAX; $i++)
		{
			$campos[$i] = mysql_field_name ($consulta, $i);
		}
		while ($row = mysql_fetch_array($consulta))
		{
			for($j = 0; $j < $consultaMAX; $j++)
			{
				$datos[$w][$campos[$j]] = $row[$campos[$j]];
			}
			$w++;
		}
		mysql_free_result($consulta);
		return $datos;
	}
	public function insertar($tabla, $campos, $valores)
	{
		@mysql_select_db($this->bdatos, $this->conexion) or die("ERROR 702
-INSERTAR- No se pudo seleccionar la base de datos");
		$consulta = @mysql_query("SELECT * FROM ".$tabla, $this->conexion)
or die("ERROR 703 -INSERTAR(1)- No se puede realizar la consulta");
		$sentencia = "INSERT INTO ".$tabla."(".$campos.") VALUES
(".$valores.")";
		$consulta = @mysql_query($sentencia, $this->conexion) or
die("ERROR 703 -INSERTAR(2)- No se puede realizar la consulta");
```

© 2008 José Luis Yapur Vallejos. Todos los derechos reservados

```
                return true;
        }

        public function eliminar($tabla, $condicion)
        {
                @mysql_select_db($this->bdatos, $this->conexion) or die("ERROR 702
-ELIMINAR- No se pudo seleccionar la base de datos");
                $consulta = @mysql_query("DELETE ".$tabla. "WHERE ".$condicion,
                $this->conexion) or die("ERROR 703 -ELIMINAR- No se puede realizar
la consulta");
                return true;
        }

        public function sentencia($sentencia)
        {
                @mysql_select_db($this->bdatos, $this->conexion) or die("ERROR 702
-SENTENCIA- No se pudo seleccionar la base de datos");
                $consulta = @mysql_query($sentencia, $this->conexion) or
die("ERROR 703 -SENTENCIA- No se puede realizar la consulta");
                if($consulta!=NULL && $consulta!=false)
                {
                        return true;
                }
                return false;
        }

        public function mantenimiento()
        {
                @mysql_select_db($this->bdatos, $this->conexion) or die("ERROR 702
-MANTENIMIENTO- No se pudo seleccionar la base de datos");
                $consulta = mysql_list_tables($this->bdatos);
                $num_rows = mysql_num_rows($consulta);
                for ($i = 0; $i < $num_rows; $i++)
                {
                        @mysql_query("OPTIMIZE TABLE
".mysql_tablename($consulta, $i));
                        @mysql_query("REPAIR TABLE ".mysql_tablename($consulta,
$i));
                }
                mysql_free_result($consulta);
                return true;
        }

}
```

© 2008 José Luis Yapur Vallejos. Todos los derechos reservados

OBJETOS

Representación detallada y particular de algo de la realidad, si bien es cierto PHP no está orientado a objetos, sin embargo esto no significa que no se pueda trabajar con ellos, en estas últimas versiones se han ido adaptando y agregando muchas opciones que hacen que PHP sea un potente lenguaje de programación.

Para declarar e inicializar un objeto en PHP se hace lo siguiente:
$nomObjeto = new nomClase();
Con esto habremos inicializado y declarado el objeto $nomObjeto, ahora si la clase nomClase requiere el envio de parámetros a través del método constructor harémos lo siguiente:
$nomObjeto = new nomClase($var1, $var2);
Para poder acceder a los métodos de la clase nomClase a través del objeto haremos lo siguiente:
$nomObjeto->nomMétodo($var1, $var2);
Para poder acceder a las variables permitidas de la clase nomClase el proceso es el mismo:
$nomObjeto->nomVariable;

Ejemplo 1: Crear un objeto de la clase conexión mostrada anteriormente y enviar los parámetros de conexión, a su vez acceder al método de mantenimiento.

```
<?php
        $usuario = “root”;
        $password = “1234”;
        $servidor = “localhost”;
        $bdatos = “pruebas”;
        $ob = new Conexion($servidor, $usuario, $password, $bdatos);
        $mantenimiento = $ob->mantenimiento();
        if($mantenimiento)
                echo "mantenimiento ok";
        else
                echo "error";
?>
```

Ejemplo 2: Acceder al método mostrar de la clase conexión y almacenar los resultados en el vector $datos, sabiendo que la sentencia de consulta es “SELECT * FROM examenes ORDER BY id_examen ASC”

```
<?php
        $usuario = “root”;
        $password = “1234”;
        $servidor = “localhost”;
        $bdatos = “pruebas”;
        $ob = new Conexion($servidor, $usuario, $password, $bdatos);
        $datos = $ob->mostrar("SELECT * FROM examenes ORDER BY id_examen
ASC", "error");

?>
```

© 2008 José Luis Yapur Vallejos. Todos los derechos reservados