¿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 un servidor web.

PHP es extremadamente simple para el principiante, pero a su vez, ofrece muchas características avanzadas para los programadores profesionales.

PHP puede ser utilizado en cualquiera de los principales sistemas operativos del mercado; PHP soporta la mayoría de servidores web de hoy en día, incluyendo Apache, Microsoft Internet Information Server y muchos otros.

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. También puede presentar otros resultados, como XHTML y archivos XML, puede crear y manejar archivos de Excel. 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.

Reglas inícials

1. Toda pagina web que se ejecute en el servidor debe tener la extensión *.php* (ninguna debe quedar en *.html* dentro de **htdocs**).

2. Se puede combinar código HTML y PHP dentro de cada archivo (el servidor no coloca problema), solo que se debe distinguir cuando se trabaja con PHP colocando **<?php** al inicio del código PHP y cerrando con ?> Adicionalmente, cada línea debe finalizar en punto y coma (;)

3. Se utiliza el comando **echo** o **print** para escribir en el área de trabajo del navegador. Ejemplo:

```
1  <html>
2      <head>
3          <title>Ejemplo de PHP</title>
4      </head>
5      <body>
6
7  <?php
8      echo "Hola <b>Multimedia</b>";
9  ?>
10
11     </body>
12 </html>
```

4. Todas las variables en PHP se distinguen porque empiezan por el símbolo **$**. Ejemplo:

```
<html>
    <head>
        <title>Ejemplo de PHP</title>
    </head>
    <body>
<?php
    $saludo = "<b>Hola Multimedia</b>";
    echo $saludo;
?>
    </body>
</html>
```

5. En un **echo** que se envíe al navegador se pueden incluir códigos HTML o Javascript(se encierra en la etiqueta **<script>**). Ejemplo:

```
<html>
    <head>
        <title>Ejemplo de PHP</title>
    </head>
    <body>
<?php
    $saludo = "<h1>Hola Multimedia</h1>";
    echo $saludo;

    $saludo = "<script>window.alert('Hola de nuevo');</script>";
    echo $saludo;
?>
    </body>
</html>
```

6. Si se necesita unir cadenas con variables se utiliza el operador punto (**.**) para armar el mensaje. Ejemplo:

```
<html>
    <head>
        <title>Ejemplo de PHP</title>
    </head>
    <body>
<?php
    $msg = "Hola";
    $i = 14;
    echo $msg."  a todos, hoy es el dia ".$i." de Febrero";
?>
    </body>
</html>
```

Variables

En PHP las variables se representan como un signo de pesos (`$`) seguido por el nombre de la variable. El nombre de la variable es sensible a minúsculas y mayúsculas.

```php
<?php
    $foo = 45.3; // asignar el valor 45.3 a $foo
?>
```

Tipos de datos

PHP soporta ocho tipos primitivos.

- Cuatro tipos escalares:
  - boolean
  - integer
  - float (número de punto-flotante, también conocido como 'double')
  - string

- Dos tipos compuestos:
  - array
  - object

- Finalmente dos tipos especiales:
  - resource
  - NULL

El tipo de dato de una variable es definido cuando se asigna un valor o dato a la variable. Ejemplo:

```php
<?php
    $a = 1234;                  // asigna un entero
    $b = 12.34;                 // asigna un flotante o double
    $c = True;                  // asigna un booleano
    $d = "Esto es una cadena";  // asigna una cadena
    $e = 'Tambien es una cadena'; // asigna una cadena
    $f = 'e';                   // asigna un caracter
    $g = array(3,2,4,5,9);      // asigna un vector
    $h = new Clase();           // asigna un objeto
?>
```

Variables dinámicas

Una variable dinámica toma el valor de una variable y lo trata como el nombre de una variable , es decir, se crea una variable nueva a partir del *contenido* de otra variable. Ejemplo:

```php
<?php
    $a = 'Hola';
    $$a = 'Multimedia';  // se crea la variable $Hola

    echo $a.' '.$Hola;

?>
```

En el ejemplo anterior se han declarado dos variables: `$a` que contiene “Hola” y `$Hola` que contiene a “Multimedia ”.

Variables externas a PHP

Formularios HTML (GET y POST)

Cuando se envía un formulario a PHP, los valores del formulario pueden ser enviados usando el método POST o el método GET. Cuando se envía por GET los valores son mostrados en la *url* del navegador, mientras que los valores enviados por POST son ocultos al usuario final y no aparecen en la *url* del navegador. Para que PHP reciba los valores del formulario se utiliza las variables:

- `$_POST['name_del_input']` → si fue enviado por método POST
- `$_GET['name_del_input']` → si fue enviado por método GET

Ejemplo:
eje1.php

```html
<html>
    <head>
        <title>Hello!</title>
    </head>
    <body>
    <form action="" method="POST">
    <table>
        <tr>
            <td align="right">Identificacion</td>
            <td><input type="text" name="identifica" value="" /></td>
        </tr>
        <tr>
            <td align="right">Nombres</td>
            <td><input type="text" name="nombres" value="" /></td>
        </tr>
        <tr>
            <td colspan=""2"><input type="submit" name="" value="Enviar" /></td>
        </tr>
    </table>
    </form>
    </body>
</html>
```

```php
<?php
    error_reporting(E_ALL & ~E_NOTICE);  // No Muestra advertencias

    $identi = $_POST['identifica'];
    $nombre = $_POST['nombres'];

    echo "Los datos enviados son: <br />";
    echo "NOMBRE: ".$nombre."<br />IDENTIFICA: ".$identi;
?>
```

Variables de SESION

Son variables externas muy utilizadas en PHP. Estas variables se almacenan en el equipo remoto hasta que sean destruidas o eliminadas internamente desde PHP. Se utiliza por ejemplo para verificar las *sesiones* de los usuarios que han pasado por una autenticación al ingresar al sistema. Su forma es:

- `$_SESSION['nombre_variable']`

Más adelante se hará el tratamiento de este tipo de variables, utilizadas para manejar y controlar las sesiones de los usuarios.

Operadores

En PHP se utilizan varios tipos de operadores, a continuación presento los más conocidos:

1. Operadores Aritméticos

Operador	Nombre
!	Negación
+	Suma
-	Resta
*	Multiplicación
/	División
%	División módulo

2. Operadores de Incremento/Decremento

Operador	Nombre
++	Incremento
--	Decremento

3. Operadores de Asignación

Operador	Nombre
`=`	Asignar valor
`+=`	Acumulador positivo
`-=`	Acumulador negativo
`*=`	Multiplicador acumulativo
`/=`	Divisor acumulativo
`.=`	Acumulador de cadenas

4. Operadores de Comparación

Operador	Nombre
`==`	Igual
`===`	Idéntico
`!=`	Diferente
`>`	Mayor que
`<`	Menor que
`>=`	Mayor igual
`<=`	Menor igual

5. Operadores Lógicos

Operador	Nombre
`and`	Y
`or`	O
`xor`	XOR
`&&`	Y
`||`	OR
`!`	NOT

Estructuras de Control

Toda la codificación PHP se compone de una serie de sentencias. Las sentencias normalmente acaban con *punto y coma*. Además, las sentencias se pueden agrupar en grupos de sentencias encapsulando un grupo de sentencias con *llaves*. Las estructuras de control son similares al lenguaje JavaScript y C/C++.

1. CONDICIONALES

Simple

```
if(condicion)
{
   Sentencia;
   Sentencia;
   Sentencia;
     ⋮
}
```

Completo

```
if(condicion)
{
   Sentencia;
   Sentencia;
     ⋮
}
else
{
   Sentencia;
   Sentencia;
     ⋮
}
```

Anidado

```
if(condicion1)
{
   Sentencia;
   Sentencia;
}
elseif(condicion2)
{
   Sentencia;
   Sentencia;
}
else
{
   Sentencia;
   Sentencia;
}
```

Caso especial

```
switch(variable)
{
case valor1:
     sentencias;
     break;

case valor2:
     sentencias;
     break;

     ⋮

case valorN:
     sentencias;
     break;
default:
     sentencias;
}
```

2. CICLOS

while

```
variableControl=0;

while(condicion)
{
    Sentencia;
    Sentencia;
    Sentencia;
     ⋮
    Incremento;
}
```

do..while

```
variableControl=0;

do
{
    Sentencia;
    Sentencia;
    Sentencia;
     ⋮
    Incremento;
} while(condicion);
```

for

```
for(variableControl=0; condición; Incremento)
{
    Sentencia;
    Sentencia;
    Sentencia;
     ⋮
}
```

foreach

```
foreach(arreglo as VariableInterna)
{
    Sentencia VariableInterna;
    Sentencia;
    Sentencia;
     ⋮
}
```

Ejemplomultiple:

Eje2.php

```
<html>
    <head>
        <title>Ejemplo 2</title>
    <head>
    <body>
        <form action="eje2.php" method="POST">
            A<input type="text" name="varA" value="" size="4" /><br />
            B<input type="text" name="varB" value="" size="4" /><br />
            C<input type="text" name="varC" value="" size="4" /><br />
            <input type="submit" value="Enviar" /><br />
        </form>
    </body>
</html>
<?php
    error_reporting(E_ALL & ~E_NOTICE);  // Evita los warning y notices

    $a = $_POST['varA'];
    $b = $_POST['varB'];
    $c = $_POST['varC'];
    $cad = '';
    $arreglo = array(5,7,2,9,10,5,3,2,1);

```

```php
if($a > $b)
{
    for($n=0; $n<5; $n++)
    {
        $cad .= " mas".$n." ";
    }
    echo $cad."<br />";
}
elseif($a > $c)
{
    foreach($arreglo as $valor)
    {
        echo $valor.",";
    }
    echo "<br />Longitud arreglo=".sizeof($arreglo)."<br />";
}

$var1 = rand(0,4);  // valores aleatorios entre 0 y 5
switch($var1)
{
    case 0: echo "En la posicion ".$var1." esta almacenado un ".$arreglo[$var1];
            break;
    case 1: echo "En la posicion ".$var1." esta almacenado un ".$arreglo[$var1];
            break;
    case 2: echo "En la posicion ".$var1." esta almacenado un ".$arreglo[$var1];
            break;
    case 3: echo "En la posicion ".$var1." esta almacenado un ".$arreglo[$var1];
            break;
    case 4: echo "En la posicion ".$var1." esta almacenado un ".$arreglo[$var1];
            break;
    default:  echo "valor fuera de rango";
}
```

Algunas funciones especiales PHP

En PHP existen toda una gama de funciones PHP para realizar cualquier acción. A continuación voy a mostrar algunas de ellas.

require_once()

Esta funciónes utilizada para incluir archivos externos *.php* u otro que necesitemos incorporar en nuestro código. La gran ventaja sobre sobre otras funciones que también incorporar archivos es que *Si el archivo externo NO existe se detiene la ejecución del código*.

Ejemplo:

```
require_once('clase.php');
```

isset()

Esta función determina si una variable ya fue declarada. Si la variable ya fue declarada nos devuelve un valor de TRUE y en caso contrario FALSE.

Ejemplo:

```
<?php

$var = '';

// Esto evaluara a TRUE asi que el texto se imprimira.
if (isset($var))
{
    echo "Esta variable esta definida, asi que se imprimira esto.";
}

?>
```

empty()

Esta función considera si una variable declarada está vacía. Si la variable tiene un valor nos devuelve un valor de FALSE y si la variable viene vacía nos devuelve TRUE.

Ejemplo:

```
<?php

$var = 0;

if (empty($var))
{
    echo '$var es 0, una variable vacia, o no esta definida en absoluto';
}

if (isset($var))
{
    echo '$var esta definida aunque este vacia';
}

?>
```

unset()

Esta función destruye una variable, objeto o elemento especifico.

Ejemplo:

```php
<?php
// destruir una variable sencilla
unset($foo);

// destruir un elemento de un vector
unset($bar[5]);

// destruir mas de una variable
unset($foo1, $foo2, $foo3);
?>
```

Si lo que deseamos destruir son un grupo grande de variables, por ejemplo todos los **`$_POST`** ó todos los **`$_GET`** ó todos los **`$_SESSION`**, entonces la recomendación es hacerlo de la siguiente forma:

```php
<?php
// destruye todas las variables POST
$_POST = array();
?>
```

Arrays en PHP

PHP permite hacer una manipulación especial de los *array* a diferencia de otros lenguajes de programación. En PHP los índices o posiciones de los *array* no solo son enteros que empiezan desde cero hasta *n*, también se puede utilizar una *clave* para definir éstos índices y es perfectamente válido. Ejemplo:

```php
<?php
$a = array(3 => 'uno', 4 => 'dos', 5 => 'tres');
$b = array('color' => 'rojo', 'sabor' => 'dulce', 'forma' => 'redonda');
$c = array('fruta' => 'manzana', 'vegetal' => 'zanahoria', 5 => 'dieta');

$d['bar'] = 'enemigo';
$d['valor'] = 500;
$d['bebida'] = 'ginger';
$d[69] = 'mesera';

foreach($a as $i => $valor)
{
   echo "a[".$i. "] = ".$valor. "<br />";
}

foreach($b as $indice => $contenido)
```

```
{
   echo "b[".$indice. "] = ".$contenido. "<br />";
}


foreach($c as $key => $value)
{
   echo "c[".$key. "] = ".$value. "<br />";
}

foreach($d as $clave => $dato)
{
   echo "a[".$clave. "]=".$dato. ",";
}

?>
```

Funciones definidas por el usuario en PHP

Toda función debe tener los siguientes elementos:

- Un nombre, sin espacios o caracteres especiales o comenzando con números.
- Uno o varios parámetros (variables) separadas por comas, actúan como información que se le suministra a la función. *NOTA: A veces la función NO necesita parámetros.*
- Un par de llaves { } que demarcan el inicio y final de las instrucciones que ejecuta la función.
- Dentro de las funciones puede tener cualquiera de las estructuras de control (condicionales, ciclos, etc.)
- La función puede devolver valores usando la instrucción ***return***. Puede devolverse cualquier tipo de valor desde la función.
- La funciones deben ser declaradas secuencialmente antes de ser usadas.

Sintaxis:

```
function NombreFuncion ($Param_1, $Param _2, ..., $Param_n)
{
    instruccion1;
    instruccion2;
    instruccion3;
       . . .
    instruccionN;

    return $valor;
}
```

Ejemplo:

```php
<?php
function suma($a, $b)
{
    return ($a+$b);
}


function imprime($arr)
{
    foreach($arr as $clave => $dato)
    {
       echo 'arreglo[' .$clave. ']= ' .$dato. ",";
    }
}

// llamado de las funciones
$resultado = suma(23,78);
echo $resultado. '<br />';

$a = array(3 => 'uno', 4 => 'dos', 5 => 'tres');

imprime($a);

?>
```

Clases y Objetos en PHP

Una clase es una colección de variables y de funciones que forman un sistema. Una clase se define con la siguiente sintaxis:

```php
<?php
class NombreClase
{
    public $variable1 = valor1;
    private $variable2 = valor2;
    protected $variable3 = valor3;

    function __construct($para1,$para2,...,$paraN)
    {
       $this->$variableN = $paraN;
       instrucciones;
    }
    function __destruct()
    {
       instrucciones;
    }
```

```
    public function NombreFuncion1($para1,$para2,...,$paraN)
    {
       instrucciones;
    }

    private function NombreFuncion2($para1,$para2,...,$paraN)
    {
       instrucciones;
    }

    protected function NombreFuncion3($para1,$para2,...,$paraN)
    {
       instrucciones;
    }
}
```

Dentro de la clase las variables y funciones (miembros de la clase) que son utilizadas se les antepone el operador ***$this->*** al nombre de la variable o función a usar.

Las variables y funciones de la clase se les debe delimitar el alcance o ámbito de uso, es decir, hay que establecer los permisos de uso. Existen (3) ámbitos para las variables y funciones: *public*, *private* y *protected*. El alcance *public* como su nombre lo indica es de carácter público, cualquiera lo puede usar o llamar. El ámbito *private* es de carácter privado, solo se puede usar o llamar dentro de la clase. Y el ámbito *protected* es una combinación de público y privado: es publico solo clases que se deriven de la clase principal (herencia), y es privado para todo lo demás.

Las clases como tal, son plantillas (esqueletos) para ser usadas como variables. Se tiene que crear una variable del tipo deseado con el operador ***new***. A esto es a lo que denominamos crear un objeto (instanciar). Así se crea un objeto de una clase:

```
<?php

// Reserva dinámica de la clase (creación de un objeto)
$objeto1 = new nombreClase($para1,$para2,...,$paraN);

// Llamado de las variables y funciones de la clase
$objeto1->variable1 = valor;
$objeto1->NombreFuncion1(dato1,dato2,...,datoN);

// Destrucción del objeto
unset($objeto1);


?>
```

Las clases pueden ser extensiones de otras clases. Las clases extedidas o derivadas constituyen lo que se denomina Herencia, y acceden a todas las variables y funciones de la clase base o padre que hayan sido declaradas en el ámbito *protected*; las variables y funciones de la clase base declaradas en el ámbito *private* no pueden ser accedidas por la clase derivada.

```php
<?php
class NombreClase extends clasePadre
{
       miembrosDeLaClase;
}
?>
```

El constructor `__construct()` es una función de una clase que se usa o invoca automáticamente al ***crear*** una nueva instancia (objeto) de una clase. Adicionalmente, el constructor es el que permite recibir parámetros (variables) para la clase; es la forma de darle información a la clase y de esa manera se personaliza cada objeto creado. Esta característica de cargue automático es muy utilizada para inicializar variables o usar funciones que se necesiten ejecutar de primero.

El destructor `__destruct()` es una función de una clase que se invoca automáticamente al ***destruir*** el objeto de una clase.

MySQL desde PHP

PHP cuenta con toda una gama de funciones exclusivas para interactuar con diferentes bases de datos. Existen varias formas de hacer operaciones con bases de datos, sin embargo se establecen dos formas: El método básico y El método con protección. En el método básico no se hace verificación ni depuración de los datos recibidos desde formulario. Este método debe ser utilizado en situaciones donde el PHP es una versión anterior a la 5.0. El método protegido funciona a partir de PHP 5.0 y tiene la ventaja de realizar filtrado para minimizar el riesgo de inyecciones de SQL para hackear la base de datos.

A continuación se revisaran ambos métodos para realizar operaciones desde PHP hacia el manejador de bases de datos MYSQL.

Método Básico

1. Conexión.

```php
$conexion = mysql_connect('localhost','root','12345') or die('error en conexion');
```

2. Selección de la DB.

```php
mysql_select_db('ejemplo',$conexion) or die('error en base de datos');
```

3. Operación SQL.

```
$sql = "SELECT * FROM usuarios WHERE id_usuario='41780302';";
$buscar = mysql_query($sql, $conexion) or die('error en consulta');
$registro = mysql_fetch_array($buscar,MYSQL_BOTH);

echo "El nombre es ".$registro[2];
```

4. Cerrar conexión.

```
mysql_close($conexion);
```

Para cualquier operación a realizar con la base de datos se conserva el mismo orden anterior, el único ítem que varía es el 3 con la variable `$sql` (la sintaxis SQL) y puede es mas elaborado en el caso de listados. Para sacar un listado el ítem 3 sera:

```
$sql = "SELECT id_usuario,apellido,nombre,tipo_usuario FROM usuarios";
$buscar = mysql_query($sql, $conexion) or die('error en consulta');
echo "<table>";
while($registro = mysql_fetch_array($buscar,MYSQL_BOTH))
{
    echo "<tr>";
    echo "<td>".$registro[0]. "</td>";
    echo "<td>".$registro[1]. "</td>";
    echo "<td>".$registro['nombre']. "</td>";
    echo "<td>".$registro[3]. "</td>";
    echo "</tr>";
}
echo "</table>";
```

Método con Protección

1. Conexión (*incluye selección de la base de datos*):

```
<?php
//-------------------------------------------------------------------------
function $conectar()
{
    $dbase = 'lagranja';
    $user = 'root';
    $pass = '12345';
    $dsn = "mysql:dbname=".$dbase.";host=localhost";
```

```
    try
    {
            $link = new PDO( $dsn, $user, $pass );
            return $link;
    }
    catch (PDOException $e)
    {
            echo 'Error en conexion: ' . $e->getMessage();
    }
}

//--------------------------------------------------------------------
$conn = $conectar();

?>
```

2. Operaciones SQL

```
<?php

$conn = conectar();

$var1 = $_POST["id_mascota"];
$var2 = $_POST["nombre"];
$var3 = $_POST["raza"];
$var4 = $_POST["edad"];
$var5 = $_POST["propietario"];
$var6 = $_POST["telefono"];

try
{
        $sSQL = "INSERT INTO mascotas VALUES(?, ?, ?, ?, ?, ?);";
        $pst = $conn->prepare( $sSQL );

        $pst->bindValue(1, $var1);
        $pst->bindValue(2, $var2);
        $pst->bindValue(3, $var3);
        $pst->bindValue(4, $var4);
        $pst->bindValue(5, $var5);
        $pst->bindValue(6, $var6);

        $n = $pst->execute();
        if( $n > 0 )
        {
                echo "Registro insertado";
        }
}
catch (PDOException $e)
{
        echo "Error al Insertar: " . $e->getMessage();
}
$conn = null;

?>
```

Otro ejemplo:

```php
<?php

$conn = conectar();

$var1 = $_POST["id"];

try
{
        $sSQL = "SELECT * FROM  mascotas WHERE id_mascota=?;";
        $pst = $conn->prepare( $sSQL );

        $pst->bindValue(1, $var1);
        $pst->execute();

        $rs = $pst->fetch(PDO::FETCH_ASSOC);
        echo $rs["id_mascota"]."<br />";
        echo $rs["nombre"]."<br />";
        echo $rs["raza"]."<br />";
        echo $rs["propietario"]."<br />";
        echo $rs["telefono"]."<br />";
}
catch (PDOException $e)
{
        echo "Error en Busqueda: " . $e->getMessage();
}

$conn = null;
?>
```

3. Listado de Registros

```php
<?php

$conn = conectar();

$var1 = $_POST["filtro"];

try
{
        $sSQL = "SELECT * FROM  mascotas WHERE CONCAT(nombre,' ',propietario) LIKE ?;";
        $pst = $conn->prepare( $sSQL );

        $pst->bindValue(1, "%$var1%");

        $pst->execute();
        echo "<table border='1'>";
        while( $rs = $pst->fetch(PDO::FETCH_ASSOC) )
        {
                echo "<tr>";
                echo "<td>".$rs["id_mascota"]."</td>";
```

```
                echo "<td>".$rs["nombre"]."</td>";
                echo "<td>".$rs["raza"]."</td>";
                echo "<td>".$rs["propietario"]."</td>";
                echo "<td>".$rs["telefono"]."</td>";
                echo "</tr>";
        }
        echo "</table>";
}
catch (PDOException $e)
{
        echo "Error en Listado: " . $e->getMessage();
}

$conn = null;
?>
```