UD0. Inserción de código PHP en páginas web

Obtención del lenguaje de marcas para mostrar en el cliente

El código PHP está embebido en documentos HTML, para introducir dinamismo fácilmente a un sitio web.

El intérprete PHP ignora el texto del fichero HTML hasta que encuentra una etiqueta de inicio del bloque de código PHP embebido.

Lenguajes y tecnologías del servidor

- Como PHP se ejecuta del lado del servidor sólo puede tener acceso a los datos del propio servidor.

- No puede acceder a los recursos del cliente
  - No puede saber qué hora es en el cliente
  - No puede acceder a los archivos del cliente
    - Salvo la excepción de las Cookies

Etiquetas para la inserción de código

```
<?php
        echo (“si quiere servir documentos XML,haga esto\n”);
        Instrucciones PHP
?>
```

Obtención del lenguaje de marcas para mostrar en el cliente

```
<HTML>
<HEAD>
<Title>Bienvenido al curso de PHP 6</Title>
</HEAD>
<BODY>
  Estas líneas están escritas directamente en HTML
 <br>
        Esta es una línea incluida directamente en el cuerpo de la
página web
 <br>
<?php
        $expresión="1";
        if ($expresion == "1"){
                print("1.Empiezan líneas generadas por PHP<br>");
        print("2.El texto está por instrucción print de PHP");
        }
?>
</BODY></HTML>
```

Sintaxis básica de PHP

- PHP es sensible a las mayúsculas
- Las instrucciones se separan con un ; como en C. La marca final ?> implica un ;
- Comentarios (como en C):
  - /* … */ varias líneas
  - // una línea
  - # Comentario estilo shell para una línea

- Para imprimir: **echo** y **print**
  - **echo**: muestra una o más cadenas separadas por comas

    echo “Hola mundo”;

    echo “Hola “, “mundo”;

- **print**: muestra una cadena o varias unidas por el operador punto(.)

  print “Hola mundo”;

  print “Hola “ . “mundo”;

Sintaxis básica de PHP

Uso de \n para generar código HTML legible

- Sin el carácter ***\n***

Código PHP

```
print ("<P>Párrafo 1</P>");
print ("<P>Párrafo 2</P>");
```

Código HTML

```
<P>Párrafo 1</P><P>Párrafo 2</P>
```

Salida

```
Párrafo 1

Párrafo 2
```

Sintaxis básica de PHP

Uso de \n para generar código HTML legible

- Con el carácter ***\n***

Código PHP

```
print ("<P>Párrafo 1</P>\n");
print ("<P>Párrafo 2</P>\n");
```

Código HTML

```
<P>Párrafo 1</P>
<P>Párrafo 2</P>
```

Salida

```
Párrafo 1

Párrafo 2
```

Tipos de datos

- PHP soporta los tipos **de datos primitivos**:
  - Números enteros
  - Números en coma flotante
  - Cadenas de caracteres
  - Booleanos
  - Objetos
  - Recursos
  - NULL
- El tipo de una variable no se suele especificar. Se decide en tiempo de ejecución en función del contexto y puede variar.

Tipos de datos

- Números enteros: Enteros positivos y negativos

```
$var = 20;      $var = -20;      // asignación decimal
$var = 024;     $var = -024;     // asignación octal
$var = 0x14;    $var = -0x14;    // asignación hexadecimal
```

- Números en coma flotante: Permiten almacenar una parte fraccionaria.

```
$var  = 260.78;
$var = 26078e-2;
```

- Booleanos: Pueden almacenar los valores **True** (1) y **False** (0).
- Recursos: Son valores especiales que hacen referencia a una información de estado o memoria de origen externo a PHP. P.e. una conexión a una base de datos.

Tipos de datos

- Funciones de interés:
  - La función gettype() devuelve el tipo de una variable
  - Las funciones is_*type* comprueban si una variable es de un tipo dado:
    - is_array(), is_bool(), is_float(), is_integer(), is_null(), is_numeric(), is_object(), is_resource(), is_scalar(),is_string()
  - La función var_dump() muestra el tipo y el valor de una variable. Es especialmente interesante con los arrays.

Tipos de datos

- Tipo string:
  - Las cadenas se encierran entre comillas simples o dobles
  - ‘simples’: admite los caracteres de escape \’ (comilla simple) y \\ (barra). Las variables **NO** se expanden
  - “dobles”: admite más caracteres de escape, como \n, \r, \t, \\, \$, \”. Los nombres de variables **SÍ** se expanden
  - Si no necesitamos la funcionalidad de sustitución de variables dentro de una cadena es preferible usar comillas simples porque se interpretan más rápido.

Tipos de datos

- Tipo string:
  - Otra forma de inicializar cadenas es utilizar la sintaxis heredoc (PHP 4) y nowdoc (PHP 5.3) que utilizan el símbolo de documento incrustado (“<<<“) y un identificador para marcar el final del documento.

    Ejemplo: PlantillaPagina.php

    - Es importante no escribir ningún carácter, salvo \n, ***antes y después*** del identificador de cierre de la cadena.
- Acceso a un carácter de la cadena:
  - La forma es $inicial = $nombre{0};

Tipos de datos

- Ejemplos de inicialización de cadenas:

```
$a = 9;
print ‘a vale $a\n’;                    // muestra a vale $a\n
print “a vale $a\n”;                    // muestra a vale 9 y avanza una línea
print “<IMG SRC=‘logo.gif’>”;           // muestra <IMG SRC=‘logo.gif’>
print “<IMG SRC=\”logo.gif\”>”;         //muestra <IMG SRC=“logo.gif”>
$nombre=“Pepe”;
$var = <<<xxx                           // Sintaxis heredoc
Esta es una cadena que termina al encontrarse xxx. $nombre
xxx;                                    // Muestra:
        // Esta es una cadena que termina al encontrarse xxx. Pepe
```

Variables

- Las variables siempre van precedidas de un signo **$**
- El nombre es sensible a las mayúsculas
- Comienzan por letra o subrayado, seguido de letras, números o subrayado
- Además de las variables definidas por el programador, existen gran cantidad de ***variables predefinidas*** que se pueden usar libremente:
  - ***Variables de entorno***: Variables que el servidor pone a disposición de PHP e indirectamente del programador
  - ***Variables de PHP***: Variables predefinidas que pertenecen al intérprete PHP y que éste pone a disposición del programador.
- A partir de PHP 4.1.0 se incluyen ***matrices superglobales*** que centralizan todas las variables predefinidas.
  - $GLOBALS, $_SERVER, $_GET, $_POST, $_COOKIE, $_FILES,
  - $_ENV, $_REQUEST, $_SESSION

Declaración de Variables

PHP es flexible en lo que se refiere a la declaración de las variables.

- No hace falta declarar una variable antes de utilizarla
- Una variable no se define como perteneciente a un tipo de dato determinado
- El tipo de una variable puede cambiar según los valores que contenga durante la ejecución del programa.
- El último valor asignado es el que define el tipo de la variable.
- PHP aplica una asignación de dato predeterminado que consiste en asignar el dato 0 a las variables que intervienen en operaciones matemáticas, y una cadena vacía a las variables que intervienen en operaciones con cadenas de caracteres.

Declaración de Variables

Ejemplos :

```
<?php
$Cadena = "Tipo de dato de cadena";
$NúmeroEntero = 1;                  // Un valor entero
$NúmeroFlotante = 1.55;             // Un valor numérico con decimales
$Booleano = True;                   // Un valor booleano True (1) o False (0)
$Matriz[0]= "A";                    // Un valor de matriz con subíndice 0
$Matriz[2] = 3;                     // Un valor de matriz con subíndice 2
$NúmeroOctal = 012;                 // Un número octal 12 es decimal 10
$NúmeroHexadecimal = 0x1C; //Un número hexadecimal 1c igual a decimal 28
$NúmeroNegativo = -33; // Los números negativos llevan el signo adelante
$NúmeroFlotanteExp = 1.55e3;
echo $Cadena;
echo $NúmeroEntero;
echo $NúmeroFlotante;
echo $Booleano;
echo $Matriz[0];
echo $Matriz[2];
echo $NúmeroOctal;
echo $NúmeroHexadecimal;
echo $NúmeroNegativo;
echo $NúmeroFlotanteExp;
?>
```

Conversión automática de tipos

- PHP es muy flexible en el manejo de los tipos de datos.
- PHP evalúa la operación a realizar y el tipo de los operandos, y adapta los operandos para poder realizar la operación lo más correctamente posible.
- *Ejemplo:*

```
$varN=1;
$varC='2 flores';
$varC=$varC+$varN;          // el resultado es 3
```

- En una operación aritmética con cadenas intenta obtener el valor numérico de las cadenas.
- En operaciones entre enteros y coma flotante resulta un número en coma flotante.

Conversión automática de tipos

- Una concatenación de cadenas con una variable numérica hace que ésta última sea convertida a cadena.

```
$varN=1;
$varC='4 flores';
$varC=$varC.$varN;          // el resultado es 14 flores
```

Conversión automática de tipos

Reglas automáticas de conversión de tipos:

- En operaciones lógicas, los datos NULL, 0, '0' y ‘ ' se consideran FALSE. Cualquier otro dato se considera TRUE (incluida la cadena 'FALSE').
- En operaciones aritméticas no unitarias las cadenas se intentan leer como números y, si no se puede, se convierten en 0, TRUE se convierte en 1, y FALSE se convierte en 0.
- En operaciones de comparación, si un operando es un número, el otro también se convertirá en un número. Sólo si ambos operandos son cadenas se compararán como cadena.
- En operaciones de cadenas de caracteres, NULL y FALSE se convierten en ‘ ', y TRUE se convierte en '1'.

Conversión forzada de tipos

- La conversión automática que realiza PHP no siempre es lo que queremos.
- PHP permite otras conversiones implícitas de tipos :
  - (int) : Fuerza la conversión a entero
  - (real), (double), (float): Fuerza la conversión a coma flotante.
  - (string): Fuerza la conversión a cadena de caracteres.
  - (array): Fuerza la conversión a matriz
  - (object): Fuerza la conversión a un objeto.

Variable de variables

- Se pueden crear nombres de variables dinámicamente anteponiendo $$ a una variable.
- La variable *variable* toma su nombre del valor de otra variable previamente declarada.

Ejemplo:

```
<?php
 $var = "uno";
 $$var = "dos";
 print ($var);   // produce el texto: "uno"
 print ($uno);   // produce el texto: "dos"
 print ($$var);          // produce el texto: "dos"
 print (${$var});
```

- A diferencia de las variables por referencia, se están creando dos variables distintas que ocupan direcciones de memoria distintas.

Variables variable

- Ejemplo 1

```
<?PHP
    $mensaje_es="Hola";
    $mensaje_en="Hello";
    $idioma = "es";
    $mensaje = "mensaje_" . $idioma;
    print $$mensaje;
?>
```

Variables variable

- Ejemplo 2

```
<?PHP
    $mensaje_es="Hola";
    $mensaje_en="Hello";
    $idioma = "en";
    $mensaje = "mensaje_" . $idioma;
    print $$mensaje;
?>
```

Ámbito de las variables

- Contexto en el que se puede acceder a una variable.
- En PHP existen variables ***locales*** y ***globales***.
- Las variables se definen como globales precediéndolas de la palabra ***global.***
- También las podemos definir como globales asignándolas a la matriz superglobal *$GLOBALS.*
- Si queremos mantener el valor de una variable local en las sucesivas llamadas a la función hay que definirla como ***static***.

Ámbito de las variables

```
<?php
  function PruebaSinGlobal(){
    $var++;
    echo "Prueba sin global. \$var: ".$var."<br>";
}
function PruebaConGlobal(){
    global $var;
    $var++;
    echo "Prueba con global. \$var: ".$var."<br>";
}
function PruebaConGlobals(){
    $GLOBALS["var"]++;
    echo "Prueba con GLOBALS. \$var: ".
$GLOBALS["var"]."<br>";
}
$var=20;                    //variable global
PruebaSinGlobal();
PruebaConGlobal();
PruebaConGlobals();
?>
```

Constantes

- Una constante es un identificador de un dato que no cambia de valor durante toda la ejecución de un programa.
- Las constantes no se asignan con el operador =, sino con la función ***define*** :
  - define(*nombre_constante_entre_comillas*, *dato_constante*);
  - define (“PI”, 3.1416);
  - print PI;
- No llevan $ delante
- La función defined(“PI”) devuelve TRUE si existe la constante.
- Son siempre globales por defecto.
- Sólo se pueden definir constantes de los tipos escalares (boolean, integer, double, string)

Constantes predefinidas

Dependen de las extensiones que se hayan cargado en el servidor, aunque hay constantes predefinidas que siempre están presentes :

- PHP_VERSION: Indica la versión de PHP que se está utilizando.
- PHP_OS: Nombre del sistema operativo que ejecuta PHP.
- TRUE
- FALSE
- E_ERROR: Indica los errores de interpretación que no se pueden recuperar.
- E_PARSE: Indica errores de sintaxis que no se pueden recuperar.
- E_ALL: Representa a todas las constantes que empiezan por E_.

Variables en la URL

- Un mecanismo práctico aunque no muy seguro de intercambio de información entre una página y otra consiste en pasar las variables a través de un sufijo en la URL de la página llamada.

http://www.bookclub.com/compra/vercesta.php?id=24key=7

- El programa PHP recibe estas variables dentro de las matrices superglobales ***$_REQUEST*** o ***$_GET,***

```
If  ($_GET [ 'id' ] == 24)  {
…..
}
```

- Utilizar la función urlencode() cuando los valores de las variables en la URL contienen caracteres especiales.

Expresiones y operadores

- Operadores aritméticos:

  +, -, *, /, %, ++, --

- Operador de asignación:

  =

- Operadores combinados:

  -=, +=, *=, /=, .=, %=

  Ejemplos:

```
$a = 3; $a += 5;          //  a vale 8
$b = “hola ”;
$b .= “mundo”;            // b vale “hola mundo”
```

- Operadores de comparación:

  ==, !=, <, >, <=, >= y otros

- Operador ternario:

  exp1 ? exp2 : exp3

Expresiones y operadores

- Operador de identidad === : Compara también el tipo de las variables.
- Operador de control de error: @
  - Antepuesto a una expresión, evita cualquier mensaje de error que pueda ser generado por la expresión y continua la ejecución

```
<?php
   $var1=3; $var2=0;
   $huboerror="no se produce resultado por error";
   $nohuboerror="variable con valor";
   @$resultado = $var1/$var2;
   echo (empty($resultado))? huboerror : nohuboerror;
```

- Operadores lógicos:

  && (and), || (or), ! , xor

- Operadores de cadena:

  concatenación: . (punto)

  asignación con concatenación: .=

Precedencia de operadores

Asociatividad	Operadores
N/A	++ --
N/A	Instanceof
Derecha	!
Izquierda	* / %
Izquierda	+ - .
N/A	< <= > >=
N/A	== != === !===
Izquierda	&&
Izquierda	||
Izquierda	?:
Derecha	= += -= *= /= .= %=
Izquierda	and
Izquierda	or

Inclusión de ficheros externos en PHP

- La inclusión de ficheros externos se consigue con:
  - **include**()
  - **require**()
- Ambos incluyen y evalúan el fichero especificado
- Diferencia: en caso de error include() produce un warning y require() un error fatal
- Se usará require() si al producirse un error debe interrumpirse la carga de la página

Sintaxis básica de PHP

Ejemplo:

```
<HTML>
<HEAD>
   <TITLE>Título</TITLE>
<?PHP
// Incluir bibliotecas de funciones
   require  ("conecta.php");
   require ("fecha.php");
   require ("cadena.php");
   require ("globals.php");
?>
</HEAD>
<BODY>
<?PHP
   include ("cabecera.html");
?>
// Código HTML + PHP
. . .
<?PHP
   include ("pie.html");
?>
</BODY>
</HTML>
```

Variables por referencia

- La variable no contiene un valor sino la dirección de otra variable.
- En PHP 5 las variables se pasan por referencia al precederlas del símbolo &.

```
<?php
    $cadena="Tipo de dato cadena";
    $ref=&$cadena;
    $cadena="nueva asignación";
    echo $ref;
?>
```

- El signo & indica que se está almacenando la dirección de la variable y no su contenido.