Arquitecturas Distribuidas

Tema 4. III – Lenguajes de programación en el servidor: PHP

III. *Lenguajes de programación en el servidor: PHP*

1. Procesado de información en el servidor
2. Lenguajes de programación en el servidor
3. PHP: cambio de paradigma
4. Características
5. Uso básico
6. Descripción del lenguaje

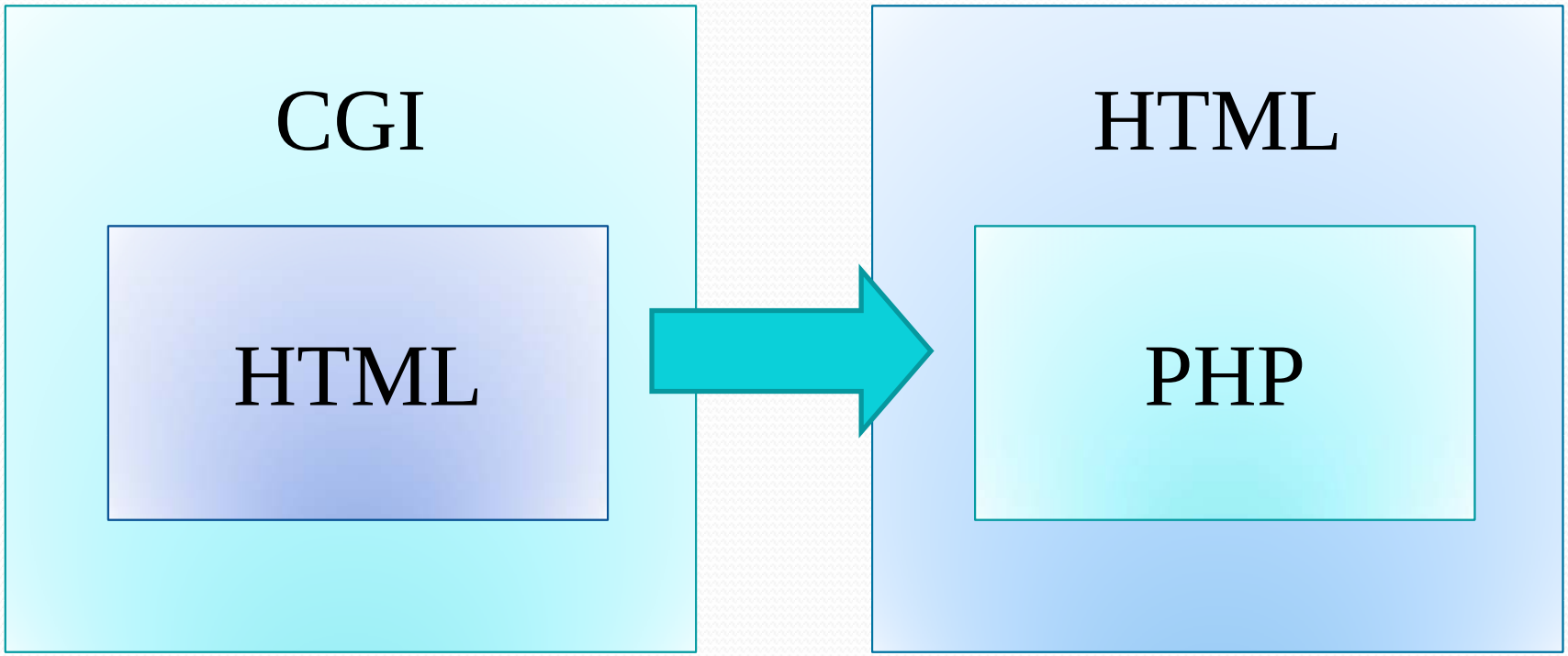
Procesado de información en el servidor

- En la actualidad, las páginas web se generan dinámicamente en el servidor antes de enviarse al cliente, tanto si procesan información del cliente como si no
- Es necesario desarrollar aplicaciones que se ejecuten en el servidor y que generen el código HTML de respuesta.
- El estándar CGI es insuficiente para las necesidades actuales: demasiado costoso computacionalmente ejecutar una aplicación en un contexto diferente para cada petición.
- Se utilizan lenguajes de programación integrados modularmente en el propio servidor web.
  - Cuando el servidor detecta que la URL dirige a una aplicación registrada, el documento solicitado es manejado por el módulo adecuado.
  - Ej. si la URL indica /procesar.php, el archivo procesar.php lo abre y procesa el intérprete de php (el preprocesador php).

Lenguajes de programación en el servidor

- En principio, mediante el estándar CGI cualquier ejecutable puede procesar la petición HTTP.
- En la práctica se suelen utilizar lenguajes interpretados (*script*), ya que las necesidades de procesado no son demasiado exigentes (además de estar optimizados) y se facilita enormemente el desarrollo.
- Algunos ejemplos:
  - Apache y PHP.
  - Apache Tomcat y JSP (*Java Server Pages*)
  - IIS y ASP .NET

Cambio de paradigma



Características

- Sencillo. Lenguaje interpretado.
- Aprovecha lo mejor de varios lenguajes: Perl, C, etc.
- Muchas librerías con funciones: PDF, XML, ODBC, etc.
- Uso **muy** extendido.
- Normalmente integrado directamente en el servidor HTTP

Uso básico

- Un fichero con código PHP incrustado se identifica con la extensión .php
- El servidor reconoce la extensión e invoca al intérprete PHP (preprocesador). El preprocesador interpreta como código PHP todo lo que aparece entre las etiquetas <? y ?>, el resto simplemente lo devuelve por la salida estándar (lo que equivale a enviarlo al cliente).

```
<? echo 'Primer método de delimitar código PHP'; ?>

<?php echo 'Segundo método, el más usado'; ?>
```

Hola Mundo!

```
<html>
<body>

<?php
        echo "Hola Mundo en PHP!";
?>

</html>
```

- Función *echo* envía por la salida estándar la cadena de caracteres siguiente (igual a *print* o *printf*)
- Como la salida estándar es el servidor HTTP, se envía al cliente

Variables

- NO hace falta declararlas
- Llevan delante el signo ‘$’.

```
$var_1 = 123;
$var_2 = ’hola’;
$var_3 = $var_1 * 2;
```

Variables

- PHP realiza conversiones de tipo automáticamente!

```
$mivar = 123;
echo $mivar;  // Se convierte a string
```

```
$mivar = '3';          // Se convierte a entero
$mivar = 2 + $mivar; // para realizar la suma
```

Variables

- Declaradas en el cuerpo de un archivo, las variables son **GLOBALES** a dicho archivo y a los archivos incluidos.
- Declaradas en una función, son **LOCALES** a esa función.

Tipos de datos.

- Enteros, en decimal, octal o hexadecimal.

```
$MiVar = 123;
```

- Punto flotante.

```
$MiVar = 1.3e4;
```

- Arrays.

```
$MiVar[2] = 123;
```

- Strings.

```
$MiVar = “Cadena de texto\n”;
```

- Objetos:

```
$MiVar = new MiClase();
```

Arrays

- PHP admite arrays “especiales”:

```
$MiArray[0] = 1;
$MiArray[1] = “hola!!”;
$MiArray[] = 3;
echo $MiArray[2];  // Imprime 3
$MiArray[“nombre”] = “Homer”;
echo $MiArray[0];          // 1
echo $MiArray[“nombre”]; // “Homer”
```

Strings

- Si se delimitan entre comillas dobles (“), las variables se expanden:

```
$a = hola;
echo “$a”  // Imprime “hola”
```

- Si se delimitan entre comillas simples (’), las variables no se expanden:

```
$a = hola;
echo ‘$a’  // Imprime “$a”
```

Operadores aritméticos.

Operación	Nombre	Resultado
$a + $b	Suma	Suma de $a y $b.
$a - $b	Resta	Diferencia entre $a y $b.
$a * $b	Multiplicación	Producto de $a y $b.
$a / $b	División	Cociente de $a y $b.
$a % $b	Módulo	Resto de la operación $a/$b.

Auto-incremento y decremento.

Operación	Nombre	Resultado
++$a	Pre-incremento	Incrementa $a en 1, y devuelve $a (incrementado).
$a++	Post-incremento	Devuelve $a, y después lo incrementa en 1.
--$a	Pre-decremento	Decrementa $a en 1, y después lo devuelve.
$a--	Post-decremento	Devuelve $a, y después lo incrementa en 1.

Operadores de bits.

Operación	Nombre	Resultado
$a & $b	Y	Se ponen a 1 los bits que están a 1 en $a y $b.
$a | $b	O	Se ponen a 1 los bits que están a 1 en $a o $b.
$a ^ $b	O Exclusivo	Se ponen a 1 los bits que están a 1 en $a o $b, pero no en ambos.
~ $a	No	Se invierten los bits (se cambian 1 por 0 y viceversa.)
$a << $b	Desp. Izq.	Desplaza $b posiciones a la izquierda todos los bits de $a.
$a >> $b	Desp. Drch.	Desplaza $b posiciones a la derecha todos los bits de $a.

Operadores lógicos.

Operación	Nombre	Resultado
$a and $b	Y	Cierto si $a y $b son ciertos.
$a or $b	O	Cierto si $a o $b es cierto.
$a xor $b	O Exclusivo.	Cierto si $a o $b es cierto, pero no ambos.
! $a	No	Cierto si $a es falso.
$a && $b	Y	Cierto si $a y $b son ciertos.
$a || $b	O	Cierto si $a o $b es cierto.

Asignación, igualdad e identidad.

Operación	Nombre	Resultado
$a = $b	Asignación	Asigna el valor de una variable o expresión del segundo término a la variable del primer término.
$a == $b	Igualdad	Compara si el valor de los dos operandos es el mismo.
$a === $b	Identidad	Compara si el valor es el mismo y, además, el tipo coincide.

Comparaciones.

Operación	Nombre	Resultado
$a != $b	No igual	Cierto si el valor de $a no es igual al de $b.
$a !== $b	No idéntico	Cierto si $a no es igual a $b, o si no tienen el mismo tipo.
$a < $b	Menor que	Cierto si $a es estrictamente menor que $b.
$a > $b	Mayor que	Cierto si $a es estrictamente mayor que $b.
$a <= $b	Menor o igual que	Cierto si $a es menor o igual que $b.
$a >= $b	Mayor o igual que	Cierto si $a es mayor o igual que $b.

Estructuras de control

if … elseif … else

```
if (expresion1) {
     comandos1
}
elseif (expresion2) {
     comandos2
}
else {
     comandosElse
}
```

Estructuras de control

while y *do … while*

```
while (expresión)
{
    comandos
}
```

```
do
{
    comandos
}
while (expresión);
```

Estructuras de control

for

```
for (expresión1; expresión2; expresión3)
{
   comandos
}
```

```
$factorial5 = 1;
for ($i = 2; $i <= 5; $i++ )
{
    $factorial5 *= $i;
}
```

Estructuras de control

foreach

```
foreach (array as variable)
{
    comandos
}
```

```
$a = array (1, 2, 3, 17);
foreach ($a as $v)
{
    print "Valor actual de \$a: $v.\n";
}

// Valor actual de $a: 1
// Valor actual de $a: 2
// Valor actual de $a: 3
// Valor actual de $a: 17
```

Estructuras de control

switch

```
switch (variable)
{
    case valor1:
        comandos1
    case valor2:
        comandos2
    ...
    case valorN:
        comandosN
    default:
        comandosDefault
}
```

```
switch ($i)
{
    case 1:
        echo “Código del 1”;

    case 2:
        echo “Código del 2”;

    case 3:
        echo “Código del 3”;
        break;

    case 4:
        echo “Código del 4”;
}
```

Cierto o falso

Valores Numéricos

```
$x = 1;  // $x
if( $x ) // se evalúa a cierto

$x = 0;  // $x definida como el entero 0
if( $x ) // se evalúa a falso
```

Cierto o falso

Strings

```
$x = "hello"; // asignamos una cadena a $x
if( $x )      // se evalúa a cierto

$x = "";      // cadena vacía
if( $x )      // evalúa a falso

              // Excepción:
$x = "0";     // cero en una cadena
if( $x )      // evalúa a falso
              // (se convierte a entero)
```

Cierto o falso

Arrays

```
$x = array();  // $x es un array vacío
if( $x )       // se evalúa como falso

$x = array( "a", "b", "c" );
if( $x )       // se evalúa a cierto
```

Funciones

Declaración

```
function nombre ($arg_1, $arg_2, ..., $arg_n)
{
    comandos
    return $salida;
}
```

Funciones

Ejemplo

```
function factorial ($valor) {
    if ($valor < 0) {
        return -1; // Error
    }

    if ($valor == 0 ) {
        return 1;
    }

    if ($valor == 1 || $valor == 2) {
        return $valor;
    }

    $ret = 1;
    for ($i = 2; $i <= $valor; $i++)
{
        $ret = $ret * $i;
    }
    return $ret;
}

$factorial5 = factorial(5);
```

Formularios

Interacción con PHP

```
<form action="accion.php" method="POST">
Su nombre: <input type=text name=nombre><br>
Su edad: <input type=text name=edad><br>
<input type=submit>
</form>
```

Posible código de accion.php:

```
Hola <? echo $_POST[`nombre`]?>.
Tienes <? echo $_POST[‘edad`]?> años.
```

Formularios

Opciones múltiples

```
<form action="accion1.php" method="POST">
<select multiple name=menu>
<option>Tortilla <option>Paella
<option>Fabada <option>Lentejas
</select><input type=submit></form>
```

```
<?php
    echo "Su elección:<br>";
    foreach($_POST[`menu`] as $plato)
    {
        echo "$plato<br>\n";
    }
?>
```

Cookies

Para enviar una cookie:

```
int setcookie (string nombre [, string valor
               [, int fin [, string camino
               [, string dominio
               [, int seguro]]]]])
```

```
setcookie(“PruebaCookie”,
          “expiraré dentro de una hora”,
          time() + 3600);
```

```
setcookie(“PruebaCookie”, “”, time());
```

Para procesarlas:

```
Se ha recibido la cookie <? echo
$_COOKIE[‘PruebaCookie’] ?>
```

Cadenas

Comparación

```
int strcmp (string str1, string str2)
int strcasecmp (string str1, string str2)

// Ejemplo:
if (strcmp($a, $b) == 0)
{
    echo ‘iguales’;
}
```

Cadenas

Concatenación

```
$a = “Hola”;
$b = “y Adios”;

$c = “$a.$b”;
print “$c”;     // Escribe “Hola y Adios”
```

Cadenas

Impresión y formato

```
int printf (string formato [, mixed args...])
string sprintf (string formato [, mixed args...])
```

1. Relleno
2. Alineación
3. Número de caracteres
4. Precisión
5. Tipo
   - **%** El carácter de tanto por ciento.
   - **b** Entero en binario.
   - **c** Entero como carácter ASCII.
   - **d** Entero en decimal.
   - **f** Double en punto flotante.
   - **o** Entero en octal.
   - **s** Cadena.
   - **x** Entero en hexadecimal (minúsculas).
   - **X** Entero en hexadecimal (mayúsculas).

Cadenas

Impresión y formato

```
printf(“%02d/%02d/%04d”, $dia, $mes, $año);

$pago1 = 68.75;
$pago2 = 54.35;
$pago = $pago1 + $pago2;

// echo $pago mostraría "123.1"
// Mostrar al menos un dígito entero y exactamente
// dos decimales, rellenando con ceros

printf ("%01.2f", $pago);
```

Referencias y bibliografía

- PHP
  - En WWW:
    - www.php.net → Portal web del PHP
    - http://php.resourceindex.com/ → Web con muchos *scripts* de PHP
  - Documentos:
    - http://ait.upct.es/asignaturas/ad/manuales/PHP/PHP-mini-es.html → Resumen lenguaje PHP en castellano
    - http://ait.upct.es/asignaturas/ad/manuales/PHP/php_manual_es.html -→ Manual PHP completo en castellano
    - http://ait.upct.es/asignaturas/ad/manuales/PHP/php_manual_en.pdf → Manual PHP completo en inglés