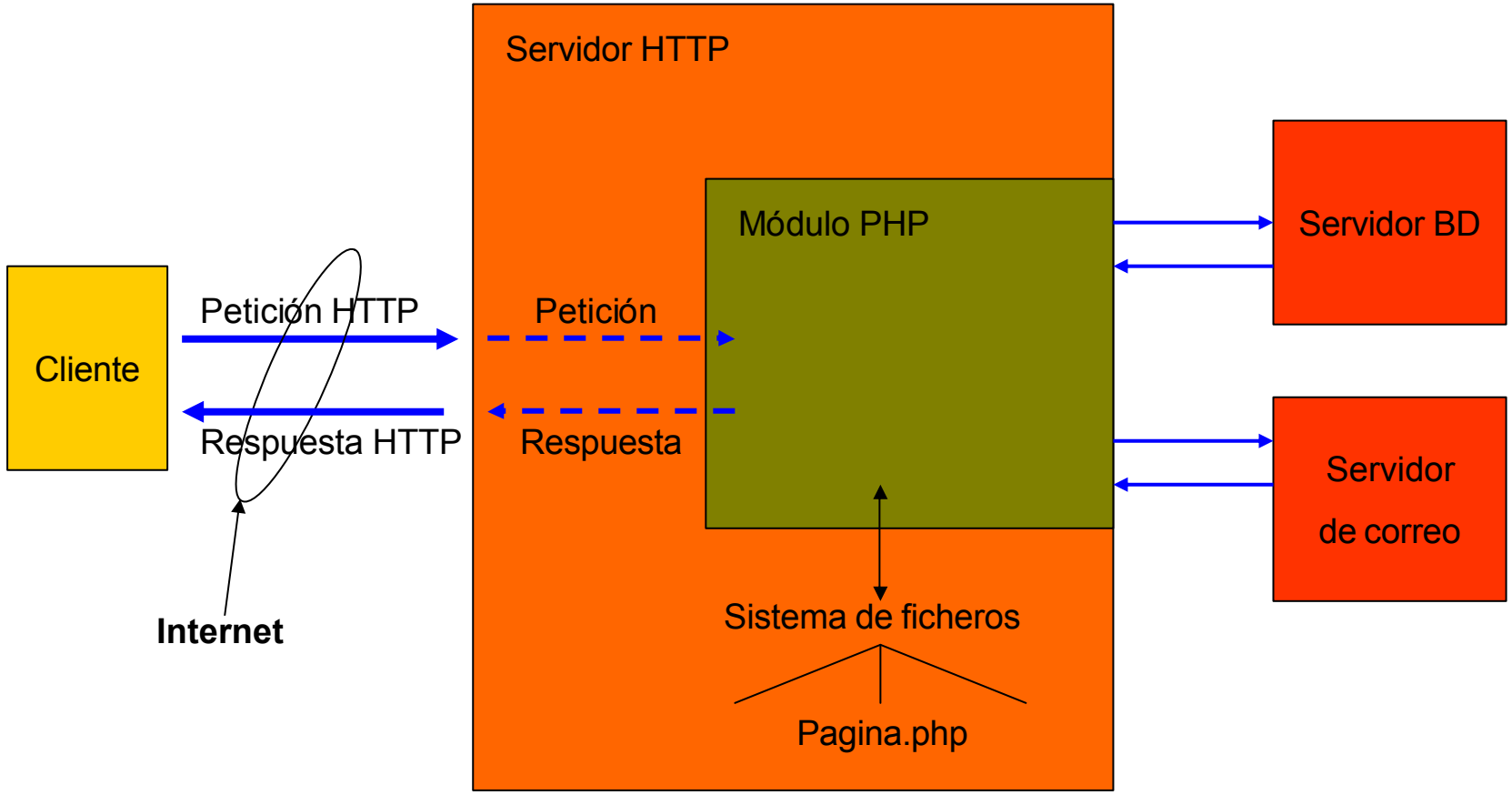
Tema 4: PHP

1. Introducción
2. Sintaxis básica del lenguaje
3. Cadenas
4. Arrays
5. Objetos
6. Formularios
7. Archivos y directorios
8. Bases de Datos
9. Cookies
10. Sesiones
11. Cabeceras HTTP

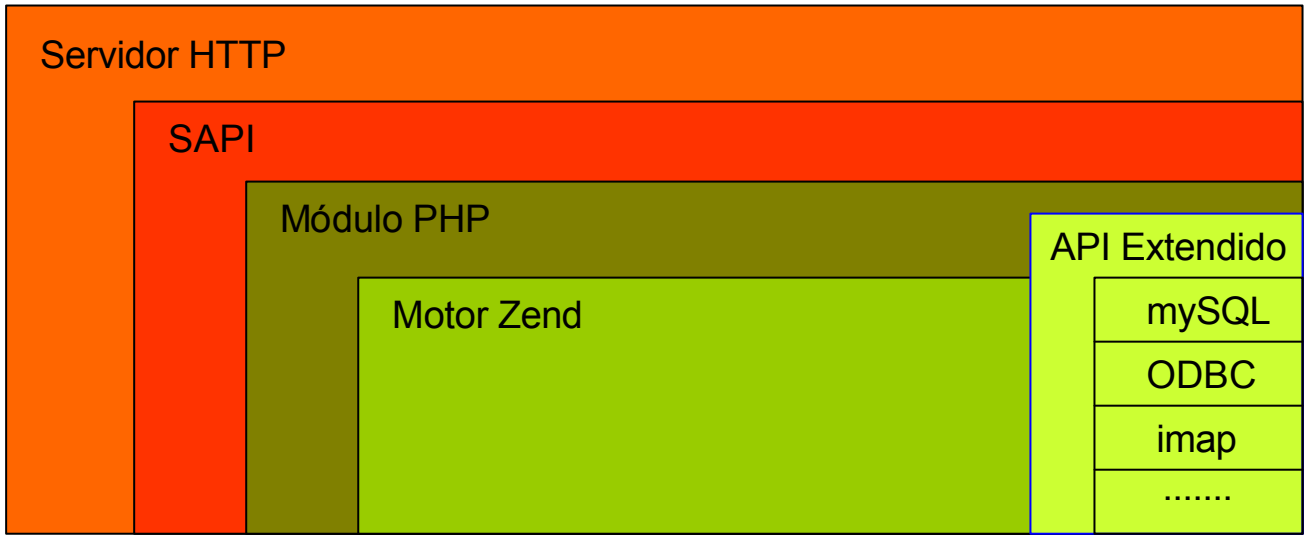
Marco General

- PHP (Hypertext Preprocessor) fue diseñado para la generación dinámica de contenidos en el lado del servidor.
- A diferencia de los CGIs, con PHP:
  - El contenido dinámico es generado utilizando un lenguaje de script propio (PHP), embebido dentro de código HTML.
  - Las peticiones son procesadas por el mismo servidor
    - El script que genera el código dinámicamente es ejecutado por el propio servidor HTTP.
    - De ello se encarga un módulo incluido en el mismo proceso.
- Permite conectarse a BD y otras aplicaciones externas.
- Primera versión: aparece en 1995.
- Ultima versión: PHP 5.1.2 (enero 2006)

Arquitectura PHP

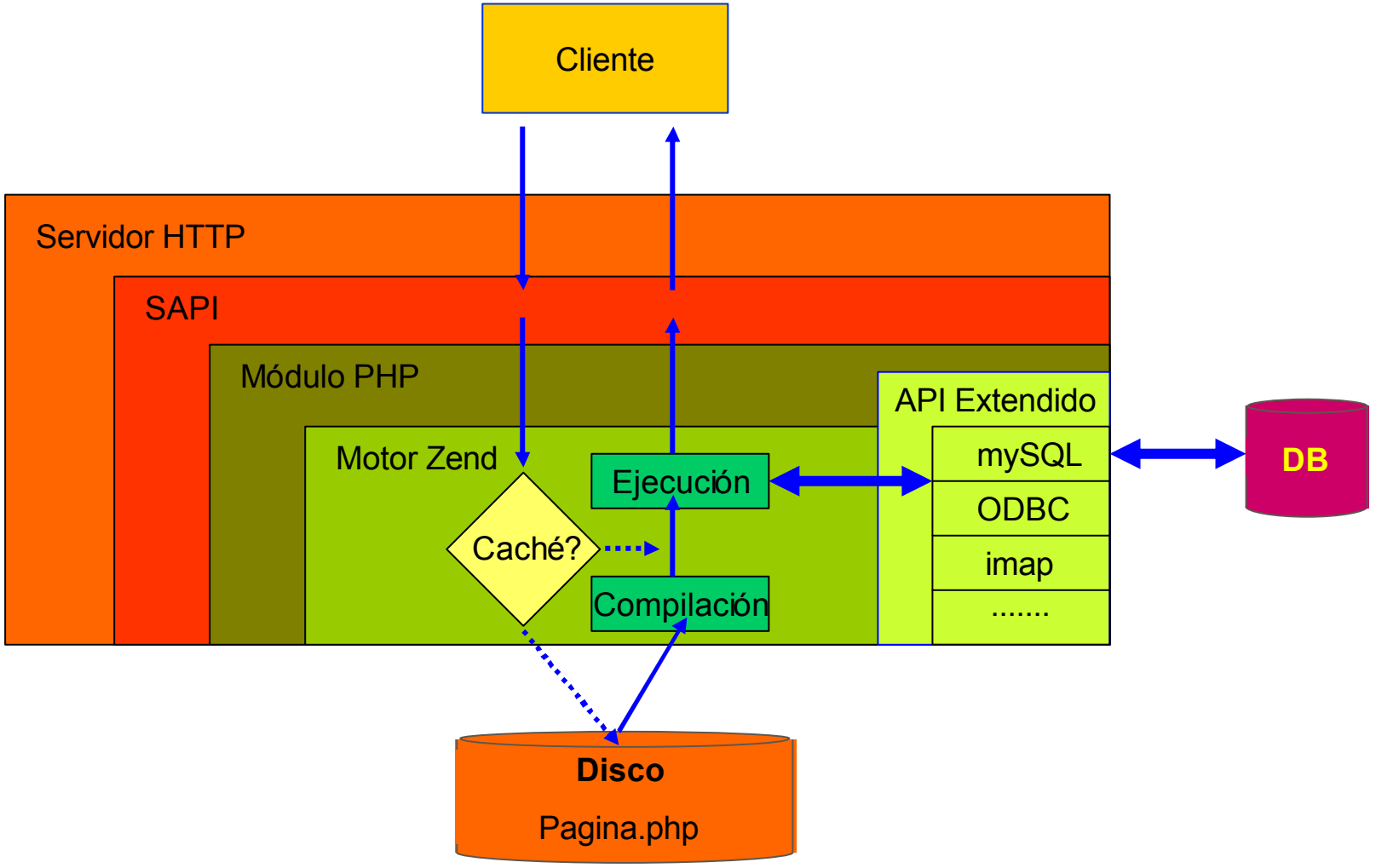


Arquitectura PHP 4.x (I)



- SAPI (Server API) facilita la tarea de añadir soporte para nuevos servidores Web.
  - Existe implementaciones del SAPI actualmente para Apache, IIS (ISAPI), Zeus (idem), AOLServer, etc.
- Las extensiones (como mySQL) forman a su vez un módulo opcional incluido (compilado) junto al de PHP.

Arquitectura PHP 4.x (II)



Ventajas (I)

- PHP es gratis.
  - Licencia de GNU.
- Es de código abierto.
  - El código fuente está disponible, y puede ser modificado libremente (hasta PHP 3.0).
- Es sencillo.
  - Lenguaje de script con sintaxis similar a Javascript, Perl y C.
  - Diseño modular de fácil ampliación.
- Es embebido.
  - Son páginas HTML que “cambian” al modo PHP cuando es necesario.
- No es un lenguaje de etiquetas
  - A diferencia de ColdFusion.
- Permite no sólo generar HTML sino también imágenes, PDFs, Flsah, XML

Ventajas (II)

- Es estable.
  - Apache/PHP tiene una buena estadística de horas de funcionamiento sin fallos.
  - Las versiones de PHP no cambian radicalmente y de forma incompatible entre versión y versión.
- Es rápido.
  - Es mucho más rápido que la mayoría de scripts para CGIs.
  - El intérprete es un módulo dentro del servidor.
- Es multiplataforma.
  - Se puede utilizar con prácticamente cualquier S.O.
  - Integración perfecta con múltiples servidores HTTP.
  - Acceso a múltiples Bases de Datos

Plataformas soportadas

- S.O:
  - Windows, Linux, SunOS, AIX, IRIX, FreeBSD, HP-UX, Solaris, Digital Unix, NetBSD, openBSD, etc.
- Servidores:
  - Apache (UNIX,Win32), ISAPI (IIS, Zeus), NSAPI (iPlanet), SunONE, WebSphere, AOLServer, etc.
- Base de datos:
  - Adabas D, dBase, Empress, IBM DB2, Informix, Ingres, Interbase, Frontbase, mSQL, Direct MS-SQL, MySQL, ODBC, Oracle, PostgreSQL, Velocis, etc.

Ejemplo

```
<html>
<head>
   <title>Ejemplo</title>
</head>
<body>
   <h1> Ejemplo simple.</h1>
   Primer ejemplo de código PHP embebido dentro de
   código HTML.<br>
<?php
echo "Hola Mundo<br>";
?>
</body>
</html>
```

Extensión y localización de los ficheros

- El servidor HTTP distingue que el recurso solicitado se trata de una página php, por la extensión.
  - Pasa la petición al módulo PHP para su procesado.
- Extensiones de los ficheros:
  - **`.php`**
    - Indica código PHP. Extensión genérica.
- Localización de los ficheros:
  - Normalmente, en los mismos directorios donde se ubican el resto de ficheros html.

Delimitadores

- El módulo de PHP busca uno de los tags que emplea para reconocer el comienzo de código PHP.
- Ejecuta el código hasta encontrar una marca de final de código
- Continúa por el documento hasta encontrar otra marca de comienzo
- Todo lo que esté fuera de esas marcas queda inalterado

```
<? echo 'delimitador muy común'; ?>

<?php print("delimitador más recomendable"); ?>

<script language="php">
echo 'Algunos editores (como FrontPage) sólo entienden este
delimitador';
</script>

<% echo 'delimitadores al estilo ASP'; %>
```

✍ Sólo el segundo y el tercer delimitador están siempre disponibles (los otros dos pueden estar desactivados).

PHP embebido

- El código PHP se puede insertar en cualquier lado del fichero, combinándose con el código HTML de cualquier manera.

```
<?php
   if (date('a' == 'pm')) {
        $saludo = 'Buenas tardes/noches!';
    } else {
        $saludo = 'Buenos días!';
    }
?>
<html>
<head><title>Ejemplo</title></head>
<body>
<h1><?php echo $saludo; ?></h1>
</body></html>
```

Primeros elementos del lenguaje

- El final de la sentencia está marcada por un punto y coma. La última no lo necesita.

```
<?php
print( date("M d, Y H:i:s", time()) );
print(
       date("M d, Y H:i:s",
                                  time()
            )
      )
;
?>
```

- Comentarios:

```
<?php
/* Comentarios estilo C.
   Pueden extenderse durante varias líneas.
 */
 // Comentarios estilo C++. Hasta fin de línea.
 # Comentarios estilo Perl. Hasta fin de línea…
```

Variables

- Tipos: boolean, integer, float, string, array, object, resource
- Van precedidas del símbolo **$**
- No necesitan ser declaradas.
- Se pueden declarar empleando la palabra reservada var (independiente del tipo): *var $mivariable;*
- PHP define en tiempo de ejecución el tipo que deben tener según el contexto en el que se empleen.
- Su nombre distingue mayúsculas de minúsculas. Nombre: '[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*'
- Sus valores son asignados con el operador **=**
- No tienen un tipo intrínseco. El tipo de la variable depende de su valor.
- A partir de PHP4 se puede asignar una variable a otra por valor o por referencia.

  Por valor: $mivar="Jose"; $mivar=$miotravar;

  Por referencia: $tercera=&$mivar;

  Más info: http://es2.php.net/manual/en/language.references.php

- Las variable utilizadas, antes de ser inicializadas, tienen valores por defectos.

```
<?php
$var_1 = 123;
$var_2 = 'hola';
$nom = "Juan";
$var_4 = $var_1 * 2 + $var_5;
$var_6 = $nom . $var5;
?>
```

Ambito de las variables

- En una función, son LOCALES a esa función.
- Existen unas variables llamadas *Superglobales* que existen en cualquier punto del script (incluso dentro de funciones sin declararlas como globales).

- Se pueden declarar variables estáticas (como en C) con static que dentro de funciones mantendrán el valor de una llamada a la función a otra. Particuliaridades: http://es.php.net/static

- Dentro de una clase, sólo pueden ser accedidas a través del objeto utilizando el operador "->".
- Dentro de una variable no se ven las variables externas, sino solo las que se declaran ahí.
- Para que una variable externa se pueda emplear dentro de una función hay que declararla en ella como global. Ejemplo:

```
$a="Hola";
function mifunc(){
global $a;
print $a;
}
```

- En el cuerpo de un fichero, las variables son GLOBALES al fichero.
  - A parte, existe un conjunto de variables globales registradas por defecto (versión PHP 4.1 o anteriores), que contienen las variables enviadas por un formulario, incluidas en las cookies, variables asociadas al servidor, a la cabecera HTTP, etc.
    - También se encuentra disponibles estas variables como elementos de arrays asociativos especiales globales.
    - A partir de la versión PHP 4.2, con el fin de generar código más claro, el registro de estas variables está desactivado por defecto (en php.ini).

Funciones para las variables

- **isset()**
  - Comprueba si ha sido creada la variable.

```
<?php if (isset($mivar)) print("$mivar"); ?>
```

- **unset()**
  - Destruye una variable.

```
<?php
 $v = 'Juan'; unset($v);
 if (!isset($v)) print('la variable ya no existe');
?>
```

- **empty()**
  - Comprueba si ha sido inicializada la variable, si tiene un valor igual a cero, o una cadena vacía.

```
<?php print(empty($v)); // devuelve true
$v = ""; print(empty($v)); // devuelve tb. true
?>
```

Constantes

- Pueden contener un boolean, integer, float o string
- Su nombre distingue mayúsculas de minúsculas (por convenio solo mayúsculas)
- Se emplean poniendo su nombre (sin $ delante)
- Se puede acceder al contenido de una constante empleando *constant(constante)*
- Una vez creadas no se puede cambiar su contenido ni eliminarlas
- Se utiliza la función **define** para crear constantes:
  - *boolean define (string nombre, tipo valor);*
  - El nombre se pasa entre comillas.
  - Se distinguen de las variables porque no van precedidas de $
  - Su ámbito es global (con PHP3, necesita ser declarada antes de usarse).
  - No pueden contener vectores.

```
define ('pi', 3.14159 ); $a = pi;
define ('NOMBRE', 'Juan'); print (NOMBRE);
```

- **defined** comprueba que la constante ha sido definida:
  - boolean defined(string nombre)

```
if (defined ('NOMBRE'))  ...
```

Salida

- **echo**, escribe la cadena que se le pasa como argumento.
  - La cadena puede estar o no incluida entre paréntesis.

```
echo " Mensaje 1 "; echo (" Mensaje 2 ");
```

  - Sin paréntesis, admite varios argumentos.

```
echo "Este mensaje", 'aparecerá', "en el browser";
```

- **print**, similar a echo.
  - Sólo acepta un argumento.

```
print ('3.145159'); print (3.14159);
```

  - El comportamiento de **print** y **echo** es distinto si la cadena va incluida entre comillas simple o dobles:

```
print(" El animal $a tiene $p patas\n");
print(' El animal $a tiene $p patas\n');
```

- **printf**: Salida con formato.

Mayúsculas y minúsculas

- Comportamiento mixto en variables y funciones:
  - En las variables, las mayúsculas y minúsculas IMPORTAN.

```
$var = 3;
$Var = 5.6;
print ("$var $Var");
```

  - En los nombres de funciones y palabras reservadas, NO IMPORTAN.

```
if ( $a == 4 )
 PRINT ('a vale 4');
IF ( $a == 3 )
 print ("a vale 3");
```

Tipos en PHP (I)

- **integer**: Enteros, en decimal, octal o hexadecimal.

```
$MiVarDec = -923; $MiVarOct = 0127; $MiVarHex = 0xf1;
print("v1: $MiVarDec  v2: $MiVarOct v3: $MiVarHex");
```

- **double**: Reales.

```
$MiVar = 1.3e4; $var1 = 2.28888; $var2 = 2.21112;
$var3 = $var1 + $var2; print ("$var3");
```

- **boolean:** true o false (PHP 4.x)
- **array**: vectores

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

- **string**: Cadenas.

```
$var1 = "Cadena\n"; $var2 = 'Mas cadenas';
```

- **object**: Objetos.

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

Tipos en PHP (II)

- No hay declaración de tipos. Son asignados por el contexto.

```
$var1 = 55.5;
$var2 = 3;
$var3 = substr(12345, 2, 2);
```

- Variables débilmente tipadas.

```
$mi_variable = 'Inicializamos como texto';
$mi_variable = 3;                        // Entero.
$mi_variable = 3.14 * $mi_variable;      // Real.
$mi_variable = new MiClase();            // Objeto.
```

- PHP realiza conversiones automáticas de tipo:

```
$pi = 3 + 0.14159;
$mivar = '3';
$mivar = 2 + $mivar;          // Se convierte a entero
```

- Conversiones forzadas:

```
$mivar = (string)(123);       // Casting.
$mivar2 = settype($mivar,"integer");
```

Tipos en PHP (III)

- **is_integer(), is_double(), is_string(), is_bool(), is_array(), is_object()**
  - Comprueban si es entero, real, cadena, booleano, array u objeto, respectivamente

```
<html>
<head><title>Ejemplo de variables</title></head>
<body>
  <h1>Ejemplo de funciones para las variables</h1>
<?php
    $ed = 42;
    $nom = 'Juan';
    if ( is_integer($ed) && is_string($nom) )
       print("$ed tiene $nom años");
?>
</body></html>
```

Operadores (I)

- Lógicos:
  - **and**, **or**, **xor**, **&&**, **||**, **!**

```
if( $a == 4 and $b && !$c ) ...
```

- Relacionales:
  - **==**, **!=**, **<**, **<=**, **>**, **>=**, **===** (PHP4, identidad), **!==** (no idéntico)

```
$v1 = 5;                    // Asignación
$v2 = 05;
$v3 = '5A6';
if($v1 == $v2) …      // Cierto, son iguales
if($v1 == $v3) …      // Cierto, son iguales
if($v1 === $v2) …  // Cierto, son idénticas
if($v1 === $v3) …  // FALSO, el tipo no coincide
```

Operadores (II)

- Aritméticos:
  - **+ +**, **--**, **+**, **-**, *****, **/**, **%**

```
$a = 3 + 6; ++$a; $a++;
```

- Bit:
  - **&**, **|**, **^** (OR exclusivo), **~** (complemento), **< <**, **> >**
- Asignación:
  - **=**, **+ =**, **- =**, *** =**, **/ =**, **% =**, **& =**, **| =**, **^ =**, **< < =**, **> > =**, **. =**

```
$v1 <<= 2;     // Equivalente a: $v1 = $v1<<2
$v2 .= 'nuevo'; // Equivalente a: $v2 = $v2 . 'nuevo'
```

Estructuras de control (I)

- Selectivas:
  - **if** (condición) {..} **else** {..}
  - **if** (..) {..} **elseif** (..) {..} **else** {..}

```
if ( $ciudad == 'Madrid') {
          ...
} elseif ($ciudad =='Valencia') {
  ...
} else { ... }
```

  - **switch** (..) { **case**: .... **break**; .... **default**: .... }
  - **FALSE**, 0 , “” o un array vacío, equivalen a un valor de falso. Cualquier otro valor equivale a cierto (**TRUE**).

Estructuras de control (II)

- Bucles:
  - **while** (condición) { .. }
  - **do** { .. } **while** (condición);
  - **for** (.. ; condición ; ..) { .. }
  - **break** y **continue**
  - **foreach** (array **as** variable) { .. } // (PHP4)

```
$v = array(3, 5, 9, 1);
foreach ( $v as $elem)
    print("otro valor: $elem<br>");
```

Funciones (I)

- Ejemplo:

```
function mifuncion($arg1, $arg2){
echo “Esta es mi funcion y me has pasado $arg1 y $arg2”;
return 33; // devolver valor
}
```

- Dentro de la función puede aparecer cualquier código válido PHP.Eso incluye otras funciones, clases, etc.
- No hace falta que la función esté definida antes de la línea donde se emplee (PHP 4)
- En el nombre de la función no distingue mayúsculas
- Se pueden definir dentro de un bloque de condición y entonces no existen hasta que la ejecución pase por él
- Los argumentos pueden tener valores por defecto
  - Ejemplos:

```
function mifunc2($arg1, $arg2=“no me han pasado nada”)
```

```
function Cuadrado($numero = 1) {
          return($numero * $numero);
}
$a = cuadrado(3) + cuadrado();
```

Funciones (II)

- Las variables definidas dentro de la función tiene **ámbito** local. Cualquier variable definida fuera de una función tiene ámbito global.

```
$numero = 5;
function Prueba() {
        $numero = 4;
}
Prueba(); print($numero);  //Se escribe 5
```

- Para acceder a las variables globales:
  - Se declaran dentro de la función precedidas de **global**

```
$numero = 5;
function Prueba() {
        global $numero; $numero = 4;
}
Prueba(); print($numero);  //Se escribe 4
```

Funciones (III)

- También se puede acceder a una variable global a través del array **GLOBALS**

```
$numero = 5;
function Prueba() {
        $GLOBALS['numero'] = 8;
}
Prueba(); print($numero);  //Se escribe 8
```

- Los parámetros de la función se pueden pasar tanto por valor como por referencia (se marcan con **&**)

```
function Copia (&$destino, $origen) {
        $destino = $origen;
        $origen = 0; //  No tiene efecto en $b
}
$a = 2; $b = 3;
Copia($a,$b);
```

Funciones (IV)

- Por defecto, las variables locales se destruyen cuando acaba la ejecución de la función, a no ser que se declaren como estáticas, precedidas de la palabra **static**.

```
function SumaNumeros() {
  static $cont = 0; static $suma = 0;
       $i = $cont; $cont += 10;
       for ( ; $i < $cont; $i ++)
              $suma += $i;
       return suma;
}
for ( $cont = 1; $cont < 6; $cont ++ ) {
       $s = SumaNumeros();
       printf("0..%d suma %d", $cont*10-1, $s);
}
```

Operaciones básicas sobre cadenas

- Las cadenas se crean como una secuencia de caracteres incluidas entre comillas simples o dobles.

```
$var = "Juan"; $var = 'Pepe';
$var = 'Las comillas " forman parte de esta cadena';
```

- Se concatenan con el operador "**.**"

```
$nombre = "Francisco"; $apellidos = 'Tomas Vert';
$nombre_completo = $nombre . ' ' . $apellidos;
```

- Para acceder a los caracteres de la cadena, se trata ésta como si fuera un array de caracteres.

```
function Dobla($cad) {
        for ( $i = 0; $i < strlen($cad); $i ++ )
                $res .= $cad[$i] . $cad[$i];
        return $res;
}
print ( Dobla('Hola') . "<br>");
```

Funciones para el manejo de cadenas (I)

- Comparar dos cadenas.
  - integer **strcmp** (cad1, cad2). Devuelve 0 si son iguales.
  - integer **strcasecmp**(cad1, cad2). Idem que strcmp, pero no distingue entre mayúsculas y minúsculas.
  - Utilizando los operadores: **==, ===, !=, !==, <, >, <=, >=**
- Hallar la longitud de una cadena.
  - integer **strlen** (cadena);
- Extraer subcadenas de una cadena.
  - string **substr**(cadena origen, posición inicial, [longitud])
    - El tercer parámetro es opcional. Si se omite, la subcadena termina en el mismo punto que la cadena.

```
print (substr('Valencia',0,2)."-".
  substr('Valencia',2,3).'-'.substr('Valencia',-3));
```

Funciones para el manejo de cadenas (II)

- Encontrar subcadenas dentro de la cadena.
  - string **strstr** (cadena_origen, cadena_buscada);
    - Devuelve la subcadena de la cadena origen que comienza a partir de la subcadena buscada.

```
$a = strstr ('Valencia', 'en'); // $a = "encia";
$a = strstr ('Valencia', 'val'); // $a = false;
```

  - integer **strpos** (cadena, cadena_buscada [, pos_ini]);
    - Esta vez devuelve la posición de la subcadena dentro de la cadena.

```
$a = strpos ("Valencia", "en"); // $a = 3;
$a = strpos ("Valencia", "val"); // $a = false;
```

- Limpiar una cadena de espacios en blanco.
  - string **trim**(cadena);
    - Devuelve la cadena sin espacio en blanco a su izquierda o derecha (“ ”, \n, \r y \t).

```
$a = trim('  Valencia   '); // $a = "Valencia";
```

Funciones para el manejo de cadenas (III)

- Carácter correspondiente a un código ASCII.
  - string **chr**(numero_ascii);

```
$a = chr(65); // $a = "A";
```

- Código ASCII de un caracter.
  - integer **ord**(caracter);

```
$a = ord('A'); // $a = 65;
```

- Convierte letras mayúsculas en minúsculas.
  - string **strtolower**(cadena);

```
$a = strtolower('Valencia'); // $a = 'valencia';
```

- Convierte letras minúsculas en mayúsculas.
  - string **strtoupper**(cadena);

```
$a = strtoupper('Valencia'); // $a = 'VALENCIA';
```

Arrays en PHP

- Como el resto de variables, los array no se declaran, ni siquiera para indicar su tamaño.
- Hay arrays escalares (índices numericos) y asociativos (indices son cadenas)
- Pueden ser dispersos (se implementan como tablas hash).
  - Los índices de sus elementos no tienen porque ser consecutivos.

```
$vec[1] = '1° elemento';
$vec[8] = '2° elemento';
```

- En realidad contienen un mapeo entre *claves* y *valores* (arrays asociativos)

  Array array([index]=>[valor], [index2]=>[valor], ...);
  - Los índices no tienen porque ser números

```
$vec['tercero'] = '3° elemento';
```

- Los arrays no son homogéneos.
  - Sus elementos pueden ser de cualquier tipo (incluso otros arrays) y ser de tipos diferentes.

```
$vec[5] = '4° elemento';
$vec[1000] = 5.0;
```

Creando y eliminando elementos

- Hay dos formas de crear un array:
  - Asignación directa.
    - Se añaden sus elementos uno a uno, indicando el índice (mediante []).
    - Si el array no existía se crea.

```
$vec[5] = '1° elemento'; $vec[1] = "2° elemento";
$vec[] = '3° elemento'; //$vec[6]= "3° elem..
```

  - Utilizando el constructor **array**().
    - Se añaden entre paréntesis los primeros elementos del array. El primero de todos tiene asignado el índice cero.

```
$vec = array ( 3, 9, 2.4, 'Juan');
// $vec[0] = 3; $vec[1] = 9; $vec[2] = 2.4;...
```

    - Se pueden fijar el índice con el operador **= >**

```
$vec = array ( 2=>3 ,9, 2.4, 'nombre'=>'Juan');
// $vec[2] = 3;$vec[3]=9;.. $vec['nombre']="Juan
```

```
Ejemplo2: $unarray = array("dia" => 15, 1 => "uno");
Ejemplo3: $otro = array("unarray" => array(0=>14, 4=>15),
"nombre" => "Una tabla");
```

    - Para eliminar un elemento del array hay que emplear unset()

    - unset($miarray['nombre']);
    - unset($miarray);

Arrays de varias dimensiones

- Un array puede ser al mismo tiempo un elemento de otro array.

```
$ciudades = array ('España' => array('Madrid',
  "Valencia", 'Barcelona'), 'Italia' => array('Roma',
  'Milan', "Venecia"), Portugal => array ('Lisboa',
  'Oporto');
print($ciudades['Italia'][2]. ' y '.
  $ciudades['Portugal'][0]);
```

- El número de dimensiones del array es fijado por el número de índices de sus elementos.

```
$m[0] = '1 dim.';
$m[1][4] = 'ahora 2 dim.';
$m[2][3][5][1][8] = 'y ahora 5 dim.';
```

Recorriendo arrays secuenciales

- La función **`count()`** determina el número de elementos del array.
- Se recorren utilizando un bucle cualquiera, comenzando por el primer elemento (índice 0).

```
$v = array ( 3 ,9, 2.4, 'Juan');
for ($i = 0 ; $i < count($v); $i ++ )
        print("<br>v[$i]=$valor");
```

Recorriendo arrays no secuenciales (I)

- Para recorrer todos sus elementos en orden se puede utilizar: **foreach** (array **as** índice=>valor)

```
$v = array ( 8=>3 ,9, 1=>2.4, 'nombre'=> 'Juan');
foreach ($v as $i=>$valor)
        print("<br>v[$i]=$valor");
```

```
Otros ejemplos:
    $arr = array("uno" => "one", "dos" => "two", "tres" => "three");
    foreach ($arr as $valor) {
    echo "Valor: $valor<br>\n";
    }

    $arr = array("uno" => "one", "dos" => "two", "tres" => "three");
    foreach ($arr as $clave => $valor) {
    echo "Clave: $clave Valor: $valor<br>\n";
```

- En los array existe un puntero apuntando al índice actual. Si no se modifica ese puntero, apunta al 1º elemento.
  - **current** (array)
    Devuelve el contenido del elemento apuntado por el puntero.
  - **next**(array)
    Avanza el puntero hasta el siguiente elemento, y devuelve el contenido de este elemento.

Recorriendo arrays no secuenciales (II)

```
function RecorreTodos(&$vec) {
        $actual = current($vec);
        if ( $actual ) print("$actual<br>");
        while ($actual = next($vec))
                print("$actual<br>");
}
RecorreTodos($v);
...
RecorreTodos($v); // No escribe nada
```

- **reset**(array)

  Posiciona el puntero al inicio del vector, y devuelve su valor.

```
function RecorreTodos(&$vec) {
        $actual = reset($vec); // Ahora si funciona
        if ( $actual ) print("$actual<br>");
        ..... // idem
```

Recorriendo arrays no secuenciales (III)

- Se puede recorrer el array en orden inverso.
  - **prev** (array)

    Retrocede el puntero hasta el anterior elemento, y devuelve el contenido éste.
  - **end** (array)

    Posiciona el puntero en el último elemento, y devuelve su valor.

```
function RecorreTodosInv($v) {
        $actual = end($v);
        if ( $actual ) print("$actual<br>");
        while ($actual = prev($v))
                print("$actual<br>");
}
```

Recorriendo arrays no secuenciales (IV)

- El índice del elemento al que apunta el puntero puede ser extraído con la función **key**().

```
function RecorreTodos($v) {
   if ($actual = reset($v))
      printf("ind:%d, valor:%d<br>",key($v),$actual);
    while ($actual = next($v))
      printf("ind:%d, valor:%d<br>",key($v),$actual);
}
```

- La función **each**() devuelve un vector con el índice y el valor del elemento al que señala el puntero. A continuación avanza el puntero.

```
function RecorreTodos($v) {
   reset($v);
   while ($celda = each($v))
      print("indice:$celda[0],valor:$celda[1]<br>");
}
```

Arrays predefinidos (I)

- Array predefinidos dentro de cualquier aplicación PHP:
  - $**HTTP_GET_VARS** o $**_GET** (PHP 4.1): contiene los valores enviados por el método GET.

```
print('Variables enviadas por el método GET');
foreach($HTTP_GET_VARS as $nom_variable=>$valor)
{
    echo "$nombre_variable = $valor<br>\n";
}
```

  - $**HTTP_POST_VARS** o $**_POST** (PHP 4.1): contiene los valores enviados por el método POST.
  - $**HTTP_COOKIE_VARS** o $**_COOKIE** (PHP 4.1): contiene los valores del cookie.
  - $**_REQUEST** (PHP 4.1): Contienen todas las variables incluidas en los tres anteriores

Arrays predefinidos (II)

(Cont.)

- $**HTTP_SERVER_VARS** o $**_SERVER** (PHP 4.1): variable asociadas a la cabecera HTTP o a variables del servidor.

```
$acepta = $HTTP_SERVER_VARS ['HTTP_ACCEPT'];
$cliente = $_SERVER['HTTP_USER_AGENT'];
$camino_raiz = $_SERVER['DOCUMENT_ROOT'];
$fichero_php = $_SERVER['PHP_SELF']; //SCRIPT_NAME
```

- $**HTTP_SESSION_VARS** o $**_SESSION** (PHP 4.1): variables de la sesión.
- $**HTTP_ENV_VARS** o $**_ENV** (PHP 4.1): variables de entorno.

```
$camino = $_ENV['PATH'];
```

- $**GLOBALS**: contiene todas las anteriores, además del resto de variables globales.

print_r ($_SERVER)

```
Array
(
    [HTTP_ACCEPT] => image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint,
        application/msword, */*
    [HTTP_REFERER] => http://www.ignside.net/man/php/arrays.php
    [HTTP_ACCEPT_LANGUAGE] => es
    [HTTP_ACCEPT_ENCODING] => gzip, deflate
    [HTTP_USER_AGENT] => Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; InfoPath.1)
    [HTTP_HOST] => www.ignside.net
    [HTTP_CONNECTION] => Keep-Alive
    [PATH] => C:\Python23\.;C:\Perl\bin\;C:\Archivos de programa\Windows Resource
        Kits\Tools\;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\Archivos de programa\ATI Technologies\ATI Control Panel;C:\Archivos de
        programa\Archivos comunes\GTK\2.0\bin;;c:\php;C:\Archivos de programa\MySQL\MySQL Server 5.0\bin
    [SystemRoot] => C:\WINDOWS
    [COMSPEC] => C:\WINDOWS\system32\cmd.exe
    [PATHEXT] => .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.pyo;.pyc;.pyw;.py
    [WINDIR] => C:\WINDOWS
    [SERVER_SIGNATURE] => Apache/2.0.54 (Win32) PHP/5.1.2 DAV/2 Server at www.ignside.net Port 80

    [SERVER_SOFTWARE] => Apache/2.0.54 (Win32) PHP/5.1.2 DAV/2
    [SERVER_NAME] => www.ignside.net
    [SERVER_ADDR] => 192.168.1.6
    [SERVER_PORT] => 80
    [REMOTE_ADDR] => 147.158.228.75
    [DOCUMENT_ROOT] => E:/realbeta
    [SERVER_ADMIN] => irvmail@teleline.es
    [SCRIPT_FILENAME] => E:/realbeta/man/php/ejemplos/print_r.php
    [REMOTE_PORT] => 2445
    [GATEWAY_INTERFACE] => CGI/1.1
    [SERVER_PROTOCOL] => HTTP/1.1
    [REQUEST_METHOD] => GET
    [QUERY_STRING] =>
    [REQUEST_URI] => /man/php/ejemplos/print_r.php
    [SCRIPT_NAME] => /man/php/ejemplos/print_r.php
    [PHP_SELF] => /man/php/ejemplos/print_r.php
    [REQUEST_TIME] => 1143119835
)
```

foreach($_SERVER as $value)

```
<?php
foreach( $_SERVER as $value ) {
   echo "Valor: $value<br>\n";
}
?>
```

RESULTADO:

Valor: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */*
Valor: http://www.ignside.net/man/php/arrays.php
Valor: es
Valor: gzip, deflate
Valor: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; InfoPath.1)
Valor: www.ignside.net
Valor: Keep-Alive
Valor: C:\Python23\;C:\Perl\bin\;C:\Archivos de programa\Windows Resource Kits\Tools\;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\Archivos de programa\ATI Technologies\ATI Control Panel;C:\Archivos de programa\Archivos comunes\GTK\2.0\bin;;c:\php;C:\Archivos de programa\MySQL\MySQL Server 5.0\bin
Valor: C:\WINDOWS
Valor: C:\WINDOWS\system32\cmd.exe
Valor: .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.pyo;.pyc;.pyw;.py
Valor: C:\WINDOWS
Valor: *Apache/2.0.54 (Win32) PHP/5.1.2 DAV/2 Server at www.ignside.net Port 80*
Valor: Apache/2.0.54 (Win32) PHP/5.1.2 DAV/2
Valor: www.ignside.net
Valor: 192.168.1.6
Valor: 80
Valor: 147.158.228.75

Valor: E:/realbeta
Valor: irvmail@teleline.es
Valor: E:/realbeta/man/php/ejemplos/foreach.php
Valor: 2450
Valor: CGI/1.1
Valor: HTTP/1.1
Valor: GET
Valor:
Valor: /man/php/ejemplos/foreach.php
Valor: /man/php/ejemplos/foreach.php
Valor: /man/php/ejemplos/foreach.php
Valor: 1143120347

foreach($_SERVER as $key => $value)

```
<?php

foreach($_SERVER as $key => $value)
{
echo "<b>".$key ."</b> tiene el valor de ". $value ."<br>";
}

?>
```

RESULTADO

HTTP_ACCEPT tiene el valor de image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */*
HTTP_REFERER tiene el valor de http://www.ignside.net/man/php/arrays.php
HTTP_ACCEPT_LANGUAGE tiene el valor de es
HTTP_ACCEPT_ENCODING tiene el valor de gzip, deflate
HTTP_USER_AGENT tiene el valor de Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; InfoPath.1)
HTTP_HOST tiene el valor de www.ignside.net
HTTP_CONNECTION tiene el valor de Keep-Alive
PATH tiene el valor de C:\Python23\.;C:\Perl\bin\;C:\Archivos de programa\Windows Resource Kits\Tools\;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\Archivos de programa\ATI Technologies\ATI Control Panel;C:\Archivos de programa\Archivos comunes\GTK\2.0\bin;;c:\php;C:\Archivos de programa\MySQL\MySQL Server 5.0\bin
SystemRoot tiene el valor de C:\WINDOWS
COMSPEC tiene el valor de C:\WINDOWS\system32\cmd.exe
PATHEXT tiene el valor de .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.pyo;.pyc;.pyw;.py
WINDIR tiene el valor de C:\WINDOWS
SERVER_SIGNATURE tiene el valor de *Apache/2.0.54 (Win32) PHP/5.1.2 DAV/2 Server at www.ignside.net Port 80*
SERVER_SOFTWARE tiene el valor de Apache/2.0.54 (Win32) PHP/5.1.2 DAV/2
SERVER_NAME tiene el valor de www.ignside.net
SERVER_ADDR tiene el valor de 192.168.1.6
SERVER_PORT tiene el valor de 80
REMOTE_ADDR tiene el valor de 147.156.222.72
DOCUMENT_ROOT tiene el valor de E:/realbeta
SERVER_ADMIN tiene el valor de irvmail@teleline.es
SCRIPT_FILENAME tiene el valor de E:/realbeta/man/php/ejemplos/foreach.php
REMOTE_PORT tiene el valor de 2450
GATEWAY_INTERFACE tiene el valor de CGI/1.1
SERVER_PROTOCOL tiene el valor de HTTP/1.1
REQUEST_METHOD tiene el valor de GET
QUERY_STRING tiene el valor de
REQUEST_URI tiene el valor de /man/php/ejemplos/foreach.php
SCRIPT_NAME tiene el valor de /man/php/ejemplos/foreach.php
PHP_SELF tiene el valor de /man/php/ejemplos/foreach.php
REQUEST_TIME tiene el valor de 1143120347

Arrays superglobales

- No precisan ser definidos como globales dentro de una función.
- En las versiones anteriores a PHP 4.1
  - **$GLOBALS**
- En las versiones PHP 4.1 o posteriores, también:
  - **$_GET, $_POST, $_COOKIE, $_REQUEST, $_ENV, $_SERVER, y $_SESSION**

```php
<?php
    function f0() {
        global $HTTP_GET_VARS;
        $nom = $HTTP_GET_VARS['nombre'];
        $nom = $_GET['nombre']; // Mismo resultado.
        return $nom;
    }
?>
```

POO en PHP

- En **PHP 4** existen algunos elementos de la programación orientada a objetos:
  - Las clases.
  - Los objetos.
  - Las variables miembro de la clase.
  - Las funciones miembro.
  - La herencia.
- **PHP 5** incluye además (de forma similar a Java):
  - Métodos y propiedades public/private/protected.
  - Métodos y atributos static
  - Constructores y destructores (_construct y _destruct)
  - Clases abstractas.
  - Interfaces
  - Clonado explícito de objetos (_clone)
  - Etc.

Definición de clases

- En la definición de una clase se suele seguir siempre la misma estructura:

  class NOMBRE_CLASE [**extends** NOMBRE_CLASE_PADRE]
  {
  [atributos]
  [metodos]
  }

- Ejemplo:

```
class mi_clase {
  var $atrib = 'mi_atributo';
  function escribe_atributo() {
    print ("$this->atrib<br>");
  }
}
```

Elementos de la POO en PHP 4 (I)

- Atributos.
  - Se definen precedidas de la palabra **`var`**.
  - Se pueden inicializar con un valor (de tipo entero, real, string, o array, pero nunca un objeto).
- Métodos.
  - Se definen dentro de la clase.
  - Las referencias a los atributos se realizan a través de this.
  - Los constructores son funciones con el mismo nombre que la clase. Sólo puede existir un constructor por clase.
- Herencia.
  - Se utiliza la palabra **`extends`** seguida del nombre de la clase padre.

Elementos de la POO en PHP 4 (II)

```
class Persona {
    var $nombre;

    function setNombre($nombre) {
        $this->nombre = $nombre;
    }
    function getNombre() {
        return $this->nombre;
    }
}
$julia = new Persona();
$julia->setNombre("Julia");
$juan = new Persona();
$juan->nombre = "Juan";
print $julia->nombre . "\n";
print $juan->getName(). "\n";
```

Elementos de la POO en PHP 4 (III)

- Ámbito.
  - Las funciones y atributos definidos en una clase derivada prevalecen sobre los atributos o funciones definidos con el mismo nombre en la clase padre.
  - El operador de ámbito "**::**" permite hacer referencia a funciones miembro de la clase padre.

```
class clasepadre {
        var $A;
        function escribe() { print ("$this->A"); }
}
class subclase extends clasepadre {
        var $B;
        function escribe(){ print ("$this->B <br>");
                print ($this->A . '<br>');
                clasepadre::escribe();  // PHP 4
                // parent::escribe();  // en PHP 5
        }
}
```

Nuevos elementos en PHP 5 (I)

- PHP 5 aporta un nuevo modelo de objetos más potente y eficiente.
  - Reescrito por completo de nuevo.
  - Más potente
    - Similar a Java.
    - Aporta muchos elementos que no existían en PHP 4
  - Más eficiente
    - PHP 4 copia el objeto cuando se asigna una variable o se le pasa un parámetro a una función.
    - En PHP 5 los objetos son referenciados por las variables
      - Para realizar una copia se emplea clone.

Nuevos elementos en PHP 5 (II)

```
class Persona {
    var $nombre;
    function setNombre($nombre) {
        $this->nombre = $nombre;
    }
    function getNombre() {
        return $this->nombre;
    }
}
function cambiaNombre($persona, $nombre) {
    $persona->setNombre($nombre);
}

$person = new Persona();
$person->setNombre("Pepe");
cambiaNombre($person, "Jose");
print $person->getNombre();
```

Nuevos elementos en PHP 5 (III)

- Atributos y métodos public/private/protected
  - Por defecto son públicos

```
class MiClase {
    private $saludo1 = "Hola, Mundo\n";
    protected $saludo2 = "Buenos días\n";
    public $saludo3 = "Hasta luego\n";
    public function printSaludo1() {
        print $this->saludo1;
    }
    function printSaludos() {
        print $this->saludo1;
        print $this->saludo2;
        print $this->saludo3;
    }
}
```

Nuevos elementos en PHP 5 (IV)

- Atributos y métodos static
  - Dentro de la clase se accede a ellos con **self**

```
class MiClase {
    static $varEstatica1;
    static $varestatica2 = 0;
    static function Imprime() {
        self::$varestatica1 = self::$varestatica2;
        print self::$varestatica1;
        self::NuevaLinea();
    }
    private static function NuevaLinea() {
        print "\n";
    }
}
Miclase::varestatica2++;
Miclase::Imprime();
```

Nuevos elementos en PHP 5 (V)

- Constructores y destructores (__construct y __destruct)
  - PHP 4 ya tenía constructores: mismo nombre que la clase
  - No disponía de destructores.
    - Se invoca cuando se destruye el objeto o cuando finaliza el script

```
class Persona {
    function __construct($nombre){
        $this->nombre = $nombre;
    }
    ....
    function __destruct() {
         print "Objeto destruyendose";
     }
    private $nombre;
}
$person = new Persona("Juan");
$person = NULL;
```

Nuevos elementos en PHP 5 (VI)

- Métodos y clases abstractas (abstract).
  - Clases abstractas: al menos uno de sus métodos es abstracto.

```
abstract class Shape {
    function setCenter($x, $y) {
        $this->x = $x; $this->y = $y;
    }
    abstract function draw();
    protected $x, $y;
}
class Square extends Shape {
    function draw() {
            ...
    }
}
class Circle extends Shape {
    function draw() {
            ...
    }
}
```

Nuevos elementos en PHP 5 (VII)

- Interfaces (interface)
  - Sintaxis:

```
class A implements B, C, ... {
    ...
};
```

```
interface Imprimible {
    function Imprime();
}
class Persona implements Imprimible {
 private $nom;
 ....
    function Imprime() {
       print "nombre= $this->nom";
    }
}
class Producto implements Imprimible {
    private $id;
    ....
    function Imprime() {
       print "id= $this->id";
    }
}
```

```
function Pr($obj) {
    if ($obj instanceof Imprimible)
    {
        print $obj->Imprime();
    } else {
        print "Error: El Objeto no
soporta el interfaz Imprimible\n";
    }
}

$person = new Persona();
$product = new Producto();
...
Pr($person);
Pr($product);
```

Entradas simples

- Las variables del formulario son transformadas por el servidor en variables globales, conservando el mismo nombre que las variables del formulario (independientemente del método de envío).

```
<!-- formulario.html -->
<form action="accion.php" method="GET">
Su nombre: <input type=text name=nombre><br>
Su edad: <input type=text name=edad><br>
<input type=submit>
</form>
-----------------------------------------
<!-- accion.php -->
<?php
        print("Hola $nombre<br>");
        print("Tienes $HTTP_GET_VARS['edad'] años<br>");
?>
```

Entradas múltiples

- Los valores de las listas de selección se guardan en un array.

```
<form action="accion.php" method="POST">
<select multiple name= "menu[]">
<option>Tortilla </options>
<option>Paella </options>
<option>Fabada </options>
<option>Lentejas </options>
</select><input type=submit></form>
---------------------------------
<?php
    echo 'Platos seleccionados:<br>';
    foreach( $_POST['menu'] as $plato )
    {
        echo "$plato<br>\n";
    }
?>
```

Insertando contenidos

- Existen dos funciones para incluir y ejecutar código de un fichero diferente:
  - **require**(string *fichero*).

    Reemplaza la llamada por el contenido del fichero (no puede ser utilizado dentro de un bucle).
  - **include**(string *fichero*).

    Permite "incluir" otro script php en ese lugar (similar a C)

    Accede al fichero y procesa su contenido.

```
require('fichero0.php');
if ($incluir == TRUE)
{
        for ($i=1; $i<4; $i++)
        {
                include('fichero' . $i . '.php');
        }
}
```

Apertura/cierre de ficheros

PHP dispone de un conjunto amplio de funciones para el manejo de ficheros.

- integer **fopen** (string *nombre_fichero*, string *modo_apertura*)
  - El modo de apertura puede ser: "**r**", "**r+**", "**w**", "**w+**", "**a**","**a+**".
  - Si el fichero es abierto con éxito devuelve un valor entero que hace referencia al fichero, si no devuelve false.
- integer **fclose** (integer *ref_fichero*)

```
$fp = fopen ('datos.txt', 'r' );
if( !$fp ) die('Imposible abrir datos');
//die escribe el mensaje y finaliza el script.
......
fclose($fp);
```

Lectura/escritura de ficheros

- string **fread** (integer *ref_fichero*, int *num_carac*)

```
$val = fread( $fp, 10 );
```

- integer **fwrite** o **fputs** (integer *ref_fichero*, string *cadena*, [integer *num_carac*])

```
fwrite( $fp, "Mas informacion" );
```

- string **fgets** (integer *ref_fichero*, int *num_carac*)

```
$texto = fgets( $fp, 20 ); // $texto = "Juan<br>"
```

- string **fgetc** (integer *ref_fichero*)

```
$un_car = fgetc($fp); //identico a fread($fp,1);
```

- boolean **feof** (integer *ref_fichero*)

```
while (!feof($fp))
        $todo .= fgetc($fp);
```

Acceso aleatorio a los datos del fichero

- integer **fseek** (integer *ref_fichero*, integer *desp_bytes* [, integer *pos_partida*])
  - El tercer parámetro puede tomar los valores: SEEK_SET, SEEK_CUR (valor por defecto), SEEK_END.
- integer **ftell** (integer *ref_fichero*)
- integer **rewind** (integer *ref_fichero*)
  - Coloca el puntero al inicio del fichero.

```
$fp = fopen('contador.dat', 'r+');
$cont = fread($fp,20); $cont ++
print("Eres el visitante número $cont.");
rewind($fp); fwrite($fp, $counter);
fclose($fp);
```

Información sobre los ficheros

Las siguientes funciones extraen información del fichero sin necesidad de abrirlo.

- boolean **file_exists** (string *fichero*)
  - Comprueba si existe el fichero.
- integer **filesize** (string *fichero*)
  - Tamaño en bytes del fichero.
- Atributos del fichero:
  - boolean **is_dir** (string *fichero*)
  - boolean **is_file** (string *fichero*)
  - boolean **is_link** (string *fichero*)
  - boolean **is_executable** (string *fichero*)
  - boolean **is_readeable** (string *fichero*)
  - boolean **is_writeable** (string *fichero*)

Manejo de directorios (I)

PHP trabajar con directorios de modo similar a como lo hace con los ficheros.

- integer **opendir** (string *directorio*)
  - Abre el directorio, y devuelve una referencia al mismo. Si el directorio no es accesible devuelve false.
- **closedir** (integer *ref_directorio*)
  - Cierra el directorio.
- string **readdir** (integer *ref_directorio*)
  - Lee el nombre del siguiente fichero (o directorio) dentro del directorio.
- **rewinddir** (integer *ref_directorio*)
  - Vuelve al primer fichero dentro del directorio

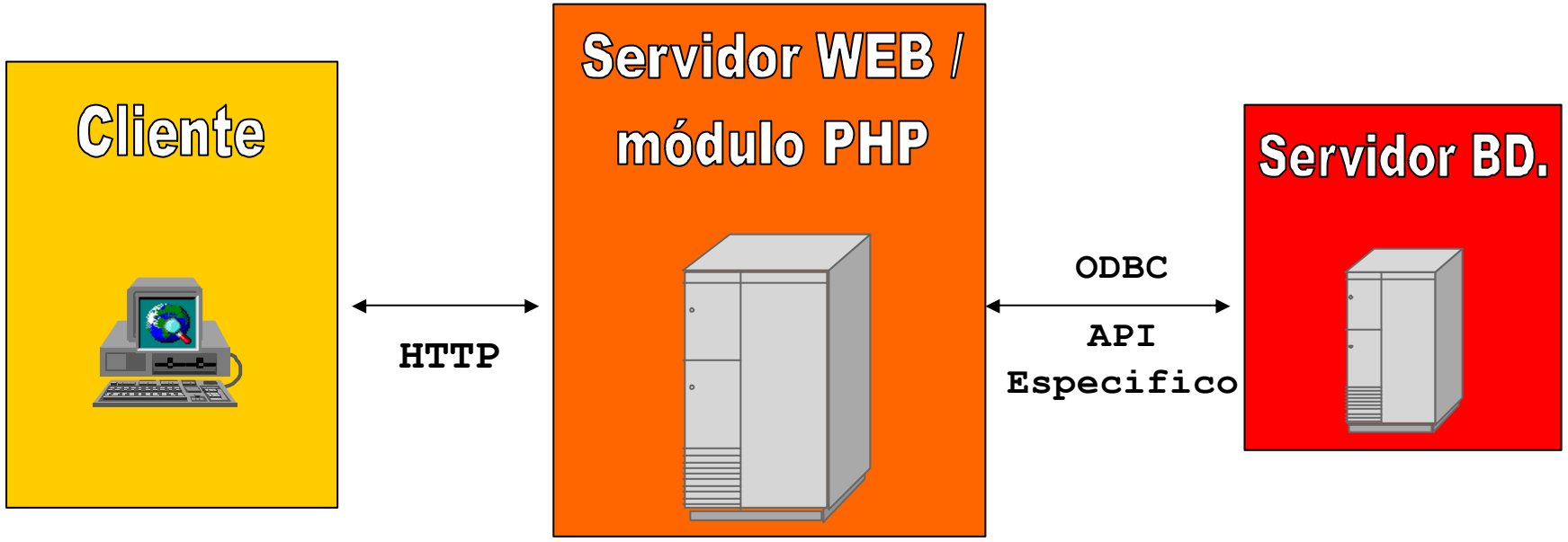
Manejo de directorios (II)

```
<HTML> <HEAD><TITLE> .. </TITLE></HEAD>
<BODY>
<h1> Contenido del directorio /pruebas</h1>
<?php
    $directorio = '/pruebas';
    $ref = opendir ( $directorio )
    rewinddir ( $ref );
    while ( $fichero = readdir($ref) )
    {
       if ($fichero != '.' and $fichero != '..') {
           if ( is_dir($fichero) ) fichero .= '/';
           printf ("$fichero<br>");
       }
    } closedir($ref);
 ?>
 </BODY> </HTML>
```

Introducción

- Una de la taréas más habituales dentro de las aplicaciones PHP (tb. en ASP o JSP) es la generación de contenidos extraídos a partir de una BD.
- La conexión con la BD la realiza el servidor web.
  - A través del driver ODBC (Windows).
  - Directamente, comunicándose con ella utilizando un API especifico para cada BD.
- PHP dispone de APIs para las BD's más utilizadas (son muy similares entre sí):
  - `mysql_connect();` `// función para mySQL.`
  - `ora_logon();` `// función para Oracle.`
  - `odbc_connect();` `// función para ODBC.`
  - `pg_connect();` `// función para PostgreSQL.`
  - `ifx_connect();` `// función para Informix.`

Arquitectura



mySQL

- La BD más utilizada con PHP sobre sistemas Unix.
  - PHP sobre Windows puede conectarse con mySQL, tanto utilizando el API especifico de ésta, como a través del ODBC.
- Es una BD relacional, ideal para aplicaciones de tamaño medio/pequeño:
- Es fácil de usar.
  - Soporte para SQL.
- Es de libre distribución (freeware) para los usuarios Linux/Unix (shareware para los usuarios Windows).

API mySQL. Conexión/desconexión

- integer **mysql_connect** ([string *servidor*
  [:*puerto*][:*/camino/al/socket*]
  [, string *usuario*
  [, string *contraseña*]]])
  - Realiza la conexión con la BD.
  - Devuelve un valor numérico que identifica la conexión.
- integer **mysql_close**([integer *id_conexion*])
  - Cierra la conexión.

```
$id_enlace = mysql_connect ('bd.uv.es:4004', 'user65',
   'pass65');
if (!$id_enlace)
     die ('No se pudo conectar');
print ('Conexión realizada');
.....
mysql_close ($id_enlace);
```

API mySQL. Selección

- boolean **mysql_select_db** (string *nombre_bd* [, integer *id_ conexion*])
  - Selecciona una de las bases de datos. Devuelve cierto si tiene éxito, y falso en caso contrario.

```
if (!mysql_select_db('prueba', $id_enlace))
        die ('No existe la BD');
```

API mySQL. Consulta

- integer **mysql_query** (string *consulta_sql* [, integer *id_conexion*])
  - Devuelve el identificador del resultado de la *consulta_sql* realizada sobre la BD seleccionada. Si no tiene éxito la búsqueda devuelve un valor de falso.

```
$busqueda = 'SELECT codigo, nombre, descripcion,
                    creditos, tipo FROM asignatura';
$id_query = mysql_query($busqueda, $id_enlace);
if (!$id_query)
       die ('Error en la búsqueda');
```

- integer **mysql_affected_rows**(integer *id_query*)
  - Usada después de una operación de INSERT, UPDATE o DELETE (como argumento de mysql_query), devuelve el número de filas alteradas.

API mySQL. Extrayendo Información

- array **mysql_fetch_array**(integer *id_query* [, int *tipo_resultado*])
  - Devuelve el contenido de una consulta SELECT, realizada con mysql_query(), en forma de un array.
  - *tipo_resultado* puede tomar los valores MYSQL_ASSOC (el array es asociativo), MYSQL_NUM (es numerado) y MYSQL_BOTH (valor por defecto).

```
while ($fila = mysql_fetch_array($id_query))
    echo $fila['codigo'].' '.$fila['nombre']. "\n";
```

- object **mysql_fetch_object** (integer *id_query*)
  - Similar, pero esta vez devuelve el resultado como un objeto.

```
while ($fila = mysql_fetch_object ($id_query))
    echo $fila->codigo .' '. $fila->nombre . '\n';
```

- array **mysql_fetch_lengths**(integer *id_query*)
  - Longitud de los campos devueltos por mysql_fetch_*().

API mySQL. Ejemplo

```
<?php
   if( !mysql_connect('db.server.com', 'juan', 'dFGb$3e') )
       die('Error en la conexion');
   if( !mysql_select_db('productos') )
       die('Error al seleccionar la BD');
   $result = mysql_query('SELECT * FROM detalles');
   if ($result) {
?>
   <TABLE>
       <TR><TH>Nombre</TH><TH>Descripción</TH></TR>
<?php while ($a = mysql_fetch_array($result)) { ?>
       <TR><TD><?php echo $a['nom'];?></TD>
           <TD><?php echo $a['descr'];?></TD></TR>
    </TABLE>
<?php  }
   } else print('<P>No hay datos'); ?>
```

Crear/alterar una cookie (I)

- PHP dispone de una única función para crear, borrar, actualizar y limitar las cookies:

  setcookie (string *nombre_cookie*, string *valor_cookie*, [, integer *caducidad*][, string *camino*] [, string *dominio*] [, integer *segura*])

  - Parámetros opcionales:
    - *caducidad* : fecha de expiración de la cookie. Sino se incluyen, la cookie se borra al final de la sesión.
    - *camino* : camino de las páginas PHP con acceso a la cookie.
    - *domino* : dominio de los servidores con acceso a la cookie.
    - *segura* : conexión segura (https).
- Las cookies se sitúan en la cabecera del mensaje
  - setcookie debe invocarse antes de comenzar a escribir el resultado enviado en el cuerpo del mensaje de respuesta.

Crear/alterar una cookie (II)

- Crear un cookie:

```
<?php  SetCookie('usuario', 'Juan');  ?> // volátil.
<?php  // Persistente
  SetCookie('usuario',
  'Juan', mktime(12,0,0,22,11,2003) );
?>
```

- Actualizar el valor de una cookie:

```
SetCookie('usuario', 'Pepe');
```

- Modificar los parámetros de la cookie:

```
SetCookie("usuario", "Juan", time()+86400, "uv.es" );
```

- Borrar una cookie:

```
SetCookie('usuario', '');
SetCookie('usuario', '', mktime(12,0,0,22,11,1970) );
```

Leer los datos de la cookie

- En la página PHP se crea una variable global con el nombre de la cookie (1º parámetro de setcookie), que contiene los datos de la cookie enviada por el cliente.

```
<?php
    print("Bienvenido $usuario<br>");
?>
```

- A partir del array global **$HTTP_COOKIE_VARS** o **$_COOKIE**, también se puede acceder a los valores de las cookies enviadas por el cliente.

```
<?php
    print("Bienvenido $_COOKIE['usuario']<br>");
?>
```

Cookies con varios datos

- Se pueden crear múltiples cookies como un array.
  - Guardar array de cookies:

```
<?php
    SetCookie('datos[0]', 'Juan');
    SetCookie('datos[1]', 'Sánchez Moreno');
    SetCookie('datos[2]', '30');
?>
```

  - Leer array de cookies:

```
<?php
    print ('Bienvenido. Tus datos son:<br>');
    for ($i = 0; $i < count($_COOKIE['datos']); $i++)
       print ("$_COOKIE['datos'][$i]<br>");
?>
```

Introducción

- El término sesión se refiere a un periodo continuo de tiempo en el cual un usuario particular realiza un conjunto de acciones relacionadas en un mismo servidor.
  - Normalmente este conjunto de acciones se reduce a peticiones por parte del cliente de una o varias páginas.
- El módulo de PHP 4.0 (y posteriores) se ocupa (liberando al programador de esa tarea) de:
  - Crear, mantener y enviar los IDs de las sesión.
  - Almacenar y recuperar los datos de la sesión
- >PHP 4.x proporciona un conjunto de funciones para:
  - Controlar el inicio y el final de la sesión.
  - Registrar y borrar variables asociadas a la sesión.

Identificador de sesión

- Cada sesión tiene asignada un identificador único.
  - El identificador es asignado por PHP, cada vez que comienza una nueva sesión.
  - Este valor se guarda (normalmente) dentro del cliente como una cookie, y es enviado por éste en cada nuevo acceso a una de las páginas del servidor. De esta manera el servidor identifica al usuario y a su sesión.
  - Todos los datos generados (registrados) durante la sesión se guardan en el servidor, asociados al identificador.
- integer **session_id** ()
  - Devuelve el identificador de la sesión.
- **session_start** ()
  - Comienza la sesión. En ese momento el servidor crea un identificador único para la sesión.

Datos registrados

- En la mayoría de casos, las sesiones se utilizan para conseguir que los valores de algunas variables “perduren” en la transición entre páginas.
- **session_register** (string nom_variable)
  - Registra una variable dentro de la sesión. El valor de dicha variable podrá ser recuperado en otras páginas, mientras perdure la sesión.
- Para acceder a las variables de la sesión en una página distinta a donde fueron registradas:
  - A través del nombre de la variable (register_global = On)
  - **$HTTP_SESSION_VARS**[‘nombre_variable’]
  - **$_SESSION**[‘nombre_variable’] (PHP 4.1 y posteriores)

Otras funciones

- **session_unregister** (string nom_variable)
  - Elimina el registro de una variable. Esto provoca que los datos de la variable registrados desaparezcan.
- **session_is_registered** (string nom_variable)
  - Devuelve true si la variable está registrada, y false si no.
- **session_destroy**()
  - Destruye la sesión y con ella todos los datos registrados.

Ejemplo (I)

```
<?php
   session_start(); // Comienza la sesion, si es necesario.
   if ( session_is_registered('contador_paginas') )
       $_SESSION['contador_paginas'] ++;
   else {
        $_SESSION['contador_paginas'] = 1;
       session_register('contador_paginas');
   }
   define ('MAX_NUM_PAGINAS', 5);
?>
<html> <head> <title> Prueba </title> </head> <body>
El identificador de tu sesion es
<?php print(session_id()); ?> <br>
<?php
   if ( !session_is_registered('usuario') ) {
       if ( !(isset($_POST['nombre'])) ) {
?>
```

Ejemplo (II)

```
<form action="<?php echo $_SERVER['PHP_SELF']; ?>"
   method="POST">
Introduce tu nombre:<input type="text" name="nombre">
<input type="submit"></form>
<?php
        } // end_if ( !isset($_POST['nombre']) )
        else {
                $_SESSION['usuario'] = $_POST['nombre'];
                session_register('usuario');
        }
   } // end_if ( !session_is_register('usuario') ).
   else{
?>
Bienvenido <?php echo $_SESSION['usuario']; ?> <br>
Has visitado <?php echo $_SESSION['contador_paginas']; ?>
   paginas a lo largo de la sesion.
```

Ejemplo (III)

```
<?php
   } // end_else.
   if ( $_SESSION['contador_paginas'] >= MAX_NUM_PAGINAS ) {
        echo 'Destruyendo la sesion<br>';
        session_destroy();
   }
?>
</body></html>
```

Enviando Cabeceras

- **header** (string cabecera_HTTP)
  - Función PHP para incluir cabeceras HTTP en el mensaje de respuesta.
  - La ejecución de esta función, del mismo modo que para la función setcookie(), debe ocurrir antes de comenzar a escribir el contenido (html) que será devuelto en el cuerpo del mensaje.
  - Ejemplos de cabeceras para controlar las caches intermedias:

```
header('Expires: Mon, 26 Jul 2003 05:00:00 GMT');
header('Last-Modified: '.gmdate("D, d M Y H:i:s").'GMT');
header('Pragma: no-cache');
header('Cache-Control: must-revalidate');
```

Redirección

- La función **header** con el valor “Location:.....”, redirige al cliente hacía una nueva URL.
  - Ejemplo:

```
if ( !$acceso_permitido ) {
  header('Location: http://www.uv.es/error.html');
  exit;
}
```

autenticación

- Para forzar al usuario a identificarse:

```
header( 'WWW-Authenticate: Basic realm= "Acceso restringido"');
```

- Para leer los valores introducidos por el usuario
  - $_**SERVER['HTTP_AUTH_USER']**: Usuario.
  - $_**SERVER['HTTP_AUTH_PW']**: Clave.

```
<?php
if ( !isset($_SERVER['PHP_AUTH_USER']) ||
   !isset($_SERVER['PHP_AUTH_PW']) || $_SERVER['PHP_AUTH_USER'] !=
   'user' || $_SERVER['PHP_AUTH_PW'] != 'open' ) {
            header('HTTP/1.0 401 Unauthorized');
            header('WWW-Authenticate: Basic realm="Privado"');
            die("Se require autorizacion");
} else { ?>
<html><head><title>Información reservada</title></head>
<body>....
```