Respuestas a las prácticas PHP

E

Respuestas a las prácticas de la parte 4

Respuestas a las prácticas del capítulo 3

1. Hay varias maneras para indicar un bloque de código PHP:

```
<?php  ... ?>
<script languaje=PHP ... /script>
<% ... %>
<? ... ?>
```

2. Código resultante:

```
<HTML>
<HEAD>
<Title>Bienvenidos al curso de PHP 6</Title>
<BODY>
Estas líneas están escritas directamente en código HTML (código
estático).
<BR>
Ésta es una línea incluida dentro de la etiqueta BODY.
<BR>
<BR>
1) Empiezan líneas generadas por PHP 6<BR>
2) texto está por instrucción print de PHP (código dinámico)
</BODY>
</HEAD>
</HTML>
```

3. Nomenclatura de variables.

a. Válida.

b. Inválida.

c. Inválida.

d. Válida.

e. Válida.

f. Válida.

g. Inválida.

h. Inválida.

4. PHP no generará ningún error por usar una variable no declarada previamente.

5. Sí, el tipo de dato de una variable queda determinado por el dato que se le asigne.

6. Una constante sólo se inicializa una vez. La segunda inicialización no se ejecutará aunque la página no producirá un error.

7. Debemos comprobar que está disponible la extensión de PHP que la define.

8. Básicamente es algo que produce un valor.

9. Resultados

   a. 1

   b. -2

   c. 2

10. El operador ternario es (?:) y es un operador de la categoría de comparación. Su funcionalidad es:

```
(exp1) ? (exp2) : (exp3);
```

Si exp1 es TRUE la expresión se evalúa como exp2, en caso contrario, se evalúa como exp3.

11. Evaluaciones:

    a. 4

    b. 3

12. El valor true se considera 1, por lo que $d será igual a 2. Pero $e será 1 porque el operador ++ no tiene efecto en variables booleanas.

13. $var1 será igual a 3 porque la asociatividad del operador = es derecha.

14. Debido a que la precedencia de los operadores lógicos or y | es diferente, los resultados también lo son:

    a. true

    b. false

15. Cadena con comillas, estas dos alternativas son válidas:

```
$var = 'El libro se titulaba "John O\'Reilly\'s car" publicado en
1990.';
$var = "El libro se titulaba \"John O'Reilly's car\" publicado en
1990.";
```

16. Comparación de matrices

```
$a = array("azul", "blanco", "rojo");
$b = array(2 => "rojo", "blanco");
$c = array(2 => "rojo",0 => "azul", 1 => "blanco");
$d = array(0 => "azul", 1=> "blanco", 2 => "rojo");

comparar($a, $b); // desigualdad
comparar($a, $c); // igualdad
comparar($a, $d); // igualdad e identidad
```

Respuestas a las prácticas del capítulo 4

1. Sentencia if recodificada con sintaxis alternativa:

```
// En este caso la evaluación es Falsa
if ($var1 < 7):
    echo "1. La evaluación fue verdadera<BR>";
elseif ($var1 < 8):
    echo "2. La evaluación es verdadera en el 1er. elseif<BR>";
elseif ($var1 < 9):
   // esta sentencia elseif se evalúa como True
    echo "3. La evaluación es verdadera en el 2do. elseif<BR>";
   // esta sentencia else if ya no se evaluará
elseif ($var1 < 10):
    echo "4. La evaluación es verdadera en el 3er. elseif<BR>";
else:
    echo "5. La evaluación fue siempre falsa<BR>";
endif;
// esta sentencia echo se ejecutará siempre, ya que
```

```
// está fuera del bloque if/elseif/else
echo "Salió del if";
```

2. Sí, es válido anidar un número indefinido de estructuras if.
3. No, se puede definir una sentencia if sin cláusula else.
4. Sí, es válido if/elseif.
5. Siempre son sólo dos valores posibles: TRUE o FALSE.
6. No, no son equivalentes. Para serlo la estructura switch debería incluir tres sentencias break:

```
switch ($var):
    case 0:
        echo "var es 0";
  break;
    case 1:
        echo "var es 1";
  break;
    case 2:
        echo "var es 2";
  break;
    default:
        echo "var inesperado";
endswitch;
```

7. Bucle do/while
8. Sí, es válido.
9. El bucle while se evalúa antes de entrar al bloque del bucle, mientras que en el bucle do/while esa evaluación se realizará después.
10.
    a. Ninguna de las expresiones es obligatoria.
    b. Falso.
    c. Verdadero.
    d. Falso.
11. Mediante una sentencia break.
12. Sí que es posible que se produzca un bucle sin salida, por un mal diseño de las expresiones de control. Este fallo puede suceder en todos los tipos de bucles.

13. Se puede utilizar con objetos y como resultado es el acceso en cada bucle a cada una de las propiedades públicas del objeto.

14. break 2;

15. Sí, se podría utilizar goto XX:, con el mismo efecto que break 2.

16. En ambos casos se abandona la ejecución del bucle; en el caso de la sentencia break es para salir del bucle y en el caso de la sentencia continue es para iniciar la siguiente iteración.

17. No está limitada la inclusión de archivos.

18. Sí, es posible.

Respuestas a las prácticas del capítulo 5

1. El uso del constructor array() es sólo un modo para crear una matriz. Existen muchas funciones que devuelven como resultado una matriz e incluso podemos crear una matriz elemento por elemento:

```
$País[0] = "Alemania";
$País[1] = "Andorra";
```

2. ¿Qué diferencia hay entre estas dos matrices?

   La diferencia es la definición de la clave. Una matriz escalar tiene un índice numérico lo que nos permite navegar por la matriz utilizando un índice con la sentencia for. Las matrices asociativas se navegan con foreach.

```
$a[0] = "París"; // matriz escalar
$b["0"]= "París"; // matriz asociativa
$c["Francia"] = "París"; // matriz asociativa
```

3. A una matriz escalar se le puede asignar una clave asociativa pero entonces ya deja de ser escalar.

4. La clave de una matriz sólo puede ser entera o cadena. Si se utiliza un valor float, este valor se trunca a entero. No puede ser ni array ni otra clase de objeto.

5. Cuando la obtención de un elemento de una matriz utiliza dos o más claves. Por ejemplo, una tabla es una matriz de dos dimensiones:

```
$tab[0, 1] = "imagen.jpg";
$tab[20, 15] = "imagen2.jpg";
```

6. Es una matriz escalar y para acceder a sus elementos se utiliza un índice entre 0 y 2. Por ejemplo, $Estad[0].
7. Sí, es válido. No es necesario que una matriz escalar tenga todos los índices consecutivos.

```
8. $mat1 = array(array(valores…), array(valores…), array(valores...);
```

9. Las funciones sizeof() y count() son iguales.
10. Varias funciones dan como resultado una matriz más pequeña, por ejemplo: array_slice(), array_shift() y array_pop().
11. Varias funciones dan como resultado una matriz más grande, por ejemplo: array_push(), array_merge(), array_merge_recursive(), array_unshift() y array_pad().
12. La unión de dos matrices se obtiene con el operador +.
13. Como máximo tendrá 11 elementos, ya que si hay claves duplicadas sólo se incluirá una vez.
14. En ambos casos las matrices tienen la misma lista de clave/valor pero en la identidad también es igual el orden.
15. En la unión se eliminan las claves duplicadas y se mantiene el valor de la primera matriz. Con la función array_merge() se se eliminan las claves duplicadas y se mantiene el valor de la última matriz. Esto es así con claves de tipo cadena.

```
<?php

// Matriz bidimensional
$a = array("A" => "Alemania", "B" => "Perú");
$b = array("B" => "Cataluña", "C" => "Perú");

$c = $a + $b;

// $c es Array ( [A] => Alemania [B] => Perú [C] => Perú )

$c = array_merge($a, $b);
// $c es Array ( [A] => Alemania [B] => Cataluña [C] => Perú )

?>
```

16. Mediante la función array_intersect().

17. Mediante la función array_intersect_assoc().

18.

```
reset($a);
$val = key($a);
```

19.

```
<?php
$a = array("A" => "Alemania", "B" => "Perú", "C" => "Francia");

print "foreach<BR>";
foreach($a as $x)
    print $x . "<BR>";

print "navegación<BR>";
reset($a);
while ($temp = each($a))
// en el elemento 0 está la clave
// en el elemento 1 está el valor
    print $temp[1]. "<BR>";

?>
```

20. Con la función in_array():

```
if (in_array ("Alemania", $a))
    print "Valor buscado existente";
```

21. Con la función array_search():

```
if (array_search ("A", $a))
    print "Clave buscada existente";
```

22.

```
asort($a); // ascendente
arsort($a); // descendente
```

Respuestas a las prácticas del capítulo 6

1. print no utiliza formato sobre la cadena que se imprime, mientras que printf permite definir un formato para una mejor edición de la salida.
2. La función printf() imprime una cadena con formato mientras que vprintf() hace lo mismo con matrices.

3.

```
%s %02d/%02d/%04d
```

4.

```
$a = "QWERTY4Tdd";


$res = stristr($a, "4T");
```

5.

```
$texto = "http://www.microsoft.com/pagina.php";
$nombre_pagina = strrchr($texto, "/" );


print substr($nombre_pagina,1);
```

6.

```
$a = "QWERTY4Tdd";
print strripos($a, "T"); // imprime 7
```

7.

```
$a = similar_text("Carlos Alberto Rodríguez", "Alberto López"); //
resultado 10
```

8.

```
$a = "www.microsoft.com";
$b = str_replace("microsoft","google", $a);
print $b; // imprime "www.google.com"
```

9.

```
<?php
$a = "Este río cambió de color";
$b = strtr($a, "áéíóúÁÉÍÓÚ", "aeiouAEIOU");
ucwords($b);
print $b; // imprime "Este Rio Cambio De Color"
?>
```

10.

```
$cantidad=substr_count(pcadena, pbusca);
```

11.

```
$a = "Este río cambió de color";
```

```
$c = str_split($a, 2); // $c será una matriz
print_r ($c);
```

12.

```
<?php
// parámetro por referencia
function recortar(&$valor)
{
    $valor = trim($valor);
}

// elementos con espacios iniciales y/o finales
$colores = array(' rojo   ','verde   ', ' azul ', '     blanco');

array_walk($colores, 'recortar');
var_dump($colores);
?>
```

13.

```
rtrim($a);
```

14.

```
str_pad($a, 20, "*");
```

15.

```
$var1 = "I'm O'Reilly";
$var2 = addslashes($var1);
print $var2;
```

16.

```
$var2 = "I\'m O\'Reilly";
$var3 = stripslashes($var2);
print $var3;
```

17.

```
<?php
$var1 = "¿Dónde está O'Reilly?";
var_dump($var1);;
$var2 = htmlentities($var1); // convierte la cadena
var_dump($var2); // Observar el cambio en la longitud
?>
```

18.

```
<?php
$var1 = "<HTML><?php print $var . '<BR>';?>Hola. esto quedará</
HTML>";
var_dump($var1);;
$var2 = strip_tags($var1);
var_dump($var2);
?>
```

Respuestas a las prácticas del capítulo 7

1. No, no es obligatorio que se devuelva un valor. En caso de devolver un valor, éste puede ser una variable de cualquier tipo, incluso puede ser un objeto o una matriz.
2. No siempre, si se trata de de una función de una extensión opcional, esta extensión debe estar instalada.
3. Sí, no se hacen diferencias entre minúsculas y mayúsculas.
4. El nombre de una función no puede comenzar con $.
5. Existen algunas excepciones. Si la función está dentro de una condición, sólo se puede llamar si el proceso ya pasó por dentro del código de la condición. Otra excepción es cuando una función está anidada dentro de otra función. No podremos utilizar esa función si previamente no se ha llamado a la función que la contienen.
6. Se debe pasar la variable por referencia. Esto obliga a que en la definición de la función, ese parámetro se defina con &, valor por referencia.
7. Deben estar al final de la lista de argumentos de una función.
8. Sí, esto exige el uso de las funciones:
   - func_num_args()
   - func_get_arg(pnum)
   - func_get_args()

   Con estas funciones se averigüa la cantidad de argumentos y se pueden obtener uno a uno o todos juntos en una matriz.
9. No, puede devolver 0 o 1 valor. Pero como el valor puede ser una matriz o un objeto, por lo que la limitación es relativa.
10. La función se define de la siguiente manera:

```
function &F1()
{
    ...
    return $a;
}
```

Y se usa de la siguiente manera:

```
$val =& F1();
```

11. Las funciones variables no se puede utilizar con distintas construcciones del lenguaje que se suelen confundir con funciones cuando no lo son, por ejemplo: echo(), print(), unset(), isset(), empty(), include(), require(), etc.
12. Es una función que se llama a sí misma.
13. Los parámetros en exceso no se toman en cuenta.
14. Se puede utilizar el operador @ para evitar el mensaje de advertencia.

Respuestas a las prácticas del capítulo 8

1. No, no es obligatorio.
2. Desde fuera de la clase:

```
$obj->miPropiedad;
```

Desde dentro de la clase:

```
$this->miPropiedad;
```

3. Debe usarse la palabra clave public.
4. La propiedad private sólo puede ser accedida desde dentro de la propiedad clase, mientras que una propiedad protected además puede ser accedida por las clases derivadas.
5. Puede utilizarse sin necesidad de crear una instancia de la clase, haciendo referencia al nombre de la clase y no al nombre de un objeto de la clase:

```
print miClase::$var1;
```

Lo mismo sucede con los métodos estáticos.

Para definir una variable estática de utiliza la palabra clave static:

```
static $var1;
```

6. -> es la sintaxis de objeto: $obj->miProp;

   :: es la sintaxis de clase: miClase::miProp;

7. Es una clase que hereda automáticamente las propiedades y los métodos de la clase base y puede redefinir y añadir las propiedades y métodos que precise.

8. Con la palabra clave extends:

```
class MiSubClase extends MiClase {

    ...
}
```

9.

```
parent::__construct();
```

10. Una clase abstracta sirve sólo para ser clase base de otras clases (no se pueden generar ejemplares de la clase). También se la conoce como clase virtual pura.

11. Método abstracto: Un método de una clase abstracta se puede declarar como abstracto, obligando de esa manera que su implementación se realice en la clase derivada.

12. Clase final: Una clase que no puede actuar como clase base (no se puede derivar).

13. No se permite herencia múltiples, sólo se puede derivar de una clase base.

14. Una interfaz es una especie de contrato en el que una clase se compromete a implementar una serie de métodos (los métodos que define una interfaz, sin suministrar ninguna implementación). una clase puede derivar de una única clase base (la cual puede ser abstracta o no) pero puede implementar múltiples interfaces.

```
class Empleado extends Persona implements Imprimir  {
...
```

15. No hay un límite.

16. Se utiliza para determinar si una variable es un ejemplar de una determinada clase o interface.

```
if ($obj instanceof miClase){
...
```

17. Son constantes que se definen dentro de una clase y que se pueden usar sin necesidad de crear un objeto de la clase, como si fuese una variable static.
18. El método _call es un método que se puede incluir opcionalmente en la definición de una clase. Este método recibe el control cuando se llama a un método no existente de la clase. Nos permite imitar lo que se denomina sobrecarga de métodos en la POO clásica.
19. Estos dos métodos intervienen cuando se quiere acceder a una variable miembro no definida o no accesible.
20. Se utiliza un mecanismo predeterminado de copia de propiedades.
21.

```
parent::_destruct();
```

22. Dos objetos son iguales cuando tienen los mismos valores en sus propiedades y son ejemplares de la misma clase. Por lo tanto, si realizamos una comparación con el operador = = (comparación igualdad) y se cumplen esas condiciones obtendremos True.
23. La función interceptora __autoload() se llama automáticamente cuando se intenta crear un ejemplar de una clase que aún no se declaró.
24. Permite utilizar foreach para obtener las propiedades de un objeto con mayor control sobre el resultado dado que éste se puede manipular mediante código en la función getiterator().
25. namespace nos permite evitar la colisión de nombres entre el código creado por el desarrollador con los nombres de funciones, clases y constantes propias del entorno PHP o de extensiones de terceros.

Respuestas a las prácticas del capítulo 9

1. Está compuesta por el comando HTTP, las cabeceras de la petición, el separador entre cabecera y cuerpo (una línea en blanco) y opcionalmente con el cuerpo del mensaje.
2. GET y POST.
3. La cabecera de la petición se genera en el navegador del cliente y la cabecera de la respuesta se genera en el servidor.

4. Las cabeceras incluyen información general sobre el entorno, el software y el contenido del mensaje. La información común en ambos tipos de cabeceras son las siguientes:
    - Directivas para el control del almacenamiento en caché.
    - Información relativa al estado de la conexión después del envío del mensaje.
    - Forma de codificación de los datos incluidos en el mensaje.
    - Longitud en bytes del mensaje.
    - Tipo de información contenida en el mensaje.
    - Fecha y hora local del envío del mensaje e información de zona horaria.
    - Opciones de funcionamiento del protocolo HTTP.
    - Forma de codificación utilizada en el cuerpo del mensaje.
    - Información sobre servidores proxy y gateways.
    - Advertencias adicionales sobre el estado del mensaje.
5. Una cabecera de petición además incluye, entre otras cosas:
    - Lista de tipos MIME que el cliente acepta.
    - Juego de caracteres que acepta el equipo cliente.
    - Codificación aceptada por el cliente.
    - Lista de lenguajes que acepta el cliente.
    - Información relativa a la autenticación.
    - Dirección de correo electrónico del cliente.
    - Nombre y puerto del host direccionado.
    - Dirección URL del recurso desde donde se realizó la petición.
    - Datos del navegador de donde se realizó la petición.
6. Error 404.
7. `getallheaders()` o `apache_request_headers()`.
8. `header()` y se debe usar antes de enviar cualquier otro dato que no sea de cabecera.

9. El servidor Web.
10. En $_SERVER.
11. <FORM ACTION="pagina.php" ...
12. La principal diferencia es que en el método GET los datos del formulario se envían como parte de la URL mientras que en el método POST van dentro del cuerpo del mensaje.
13.
    - Cuadro de texto
    - Cuadro de verificación
    - Botón de opción
    - Lista de selección múltiple
    - Campos oculto
14. Mediante un símbolo ? y después entre cada clave=valor se incluye el símbolo &.
15. En el primer caso, con la matriz $_GET, y en el segundo con la matriz $_POST.

Respuestas a las prácticas del capítulo 10

1. Un protocolo sin estado no almacena ninguna información sobre una petición o una respuesta.
2. El estado de una sesión es el conjunto de datos que permiten mantener un proceso relacionado de una secuencia de peticiones dentro de una aplicación.
3. El uso de datos ocultos en el formulario (hidden), el uso de cookies en el cliente y el almacenamiento de datos en una base de datos en el servidor.
4. Un cookie no es otra cosa que un archivo de texto y que resulta inofensivo para el equipo dado que no es un ejecutable.
5. El archivo cookie forma parte de la cabecera HTTP y PHP tiene funciones específicas para colocar esos datos en la cabecera del mensaje.

```
setcookie("fechahora",date('d/m/Y h:i:s')); // actualiza cookie
setcookie("contador",$contador + 1); // actualiza cookie
```

6. Mediante la superglobal $_SESSION y un conjunto de funciones auxiliares.

7.

```
session_start();
```

8.

```
$id = session_id();
```

9. Es posible pero no obligatorio asignar un nombre a la sesión, esto se hace con la función siguiente:

```
session_name("Nombre");
```

10.

```
$_SESSION['MiDato1'] = "Datos a guardar";
$_SESSION['MiDato2'] = "Más datos a guardar";
```

11. Desde la superglobal $_SESSION, como se hace desde cualquier matriz:

```
$var = $_SESSION['MiDato1'];
```

12.

```
// eliminación de un elemento de la matriz de $_SESSION
unset($_SESSION['MiDato1']);
```

Respuestas a las prácticas del capítulo 11

1. Esta función sirve para verificar archivos y directorios:

```
file_exists();
```

2.

```
fopen("arch.txt",a);
```

3. Indicar qué función de lectura se usa en cada caso:

   a. Leer un carácter por vez: fgetc()

   b. Leer una cantidad fija de caracteres: fgets()

   c. Leer todo el archivo en una única operación: file()

   d. Leer variables según un formato predefinido: fscanf()

4.

```
fseek();
```

5. Indicar qué función de escritura se usa en cada caso:
   a. Escribir un bloque de caracteres: fwrite() o fputs()
   b. Abrir, escribir un bloque de caracteres y cerrar el archivo: file_put_contents()
6. Mediante la función feof().
7. Mediante la función is_uploaded_file().
8. Mediante la función header().

Respuestas a las prácticas del capítulo 12

1. Requisitos mínimos:
   a. Debe tener un único elemento raíz
   b. Los elementos deben mantener un orden jerárquico y entrecruzamientos de etiquetas
   c. Todos los elementos requieren una etiqueta de cierre
   d. Los elementos pueden contener otros elementos, texto o datos
   e. Los carecteres &, <, > ' y " se pueden utilizar en los textos pero deben ser definidos con escapes (por ejemplo, &amp, &lt, etc.)
2. Falso. También puege gestionar archivos binarios.
3. Comentarios en XML:

```
<!-- y finalizan con la cadena -->
```

4. En el inicio de la sintaxis de la declaración XML se deben cuidar estos detalles:
   - Comienza con los caracteres <?xml, y finaliza con los caracteres ?>.
   - La versión es obligatoria, pero los atributos encoding y standalone son opcionales.
   - Los atributos version, encoding y standalone deben aparecer en ese orden.

5. Las secciones CDATA (Character Data) permiten definir texto sin preocuparnos por los escapes.
6. Permiten validar la estructura de un documento XML, más allá de la forma del documento.
7. Son API que generan objetos diferentes (DomDocument y SimpleXML); en ambos casos el documento XML se carga totalmente en memoria pero SimpleXML requiere menos memoria y es más simple y limitado.
8. SAX es un analizador XML basado en eventos, mientras que DOM XML crea un objeto en memoria.

Respuestas a las prácticas del capítulo 13

1. La clase Exception.
2. Salvo el método _ToString(), todos los métodos de la clase Exception no se pueden reemplazar en la clase derivada dado que se han definido con la palabra clave final.
3. Mediante el uso de la palabra clave throw.
4. Si se ha definido un bloque catch para la excepción generada, dentro del bloque catch se puede incluir el código PHP normal.
5. En primer término, debe quedar claro que no es obligatorio derivar la clase Exception, podemos usar la clase tal como está definida en la mayor parte de los casos.

   En algunos casos nos puede interesar hacer un tratamiento personalizado de algunos tipos de error y en esos casos es donde podemos crear una subclase. En estos casos, en la sentencia throw se puede hacer referencia a la clase derivada y se pude añadir un bloque catch específico para ese tipo de excepción.
6. La clase que se hace referencia en throw debe ser una clase Exception o cualquiera de sus clases derivadas.
7. Por norma general eso no debería suceder dado que siempre es conveniente añadir un bloque catch para la clase Exception que actúa como escoba para todas las excepciones.

   En todo caso, si no se incluye un bloque catch general y se genera una excepción no contemplada, entonces la excepción la detectará el sistema.

8. En absoluto. Muchas veces los conceptos se confunden porque una excepción puede ser provocada por un error. Pero una excepción define una circunstancia excepcional que debe ser gestionada por el programa, no necesariamente se trata de un error.

Respuestas a las prácticas del capítulo 14

1. Sí, utilizando la cabecera adicional From:.
2. Falso.
3. Falso, sólo indica que llegó al servidor SMTP.
4. Falso, sólo sirve para enviar mensajes.
5. Se utilizan las siguientes cabeceras en la función mail():

```
$cabs = array();
$cabs[] =  'MIME-Version: 1.0';
$cabs[] =  'Content-type: text/html; charset=iso-8859-1';
$cabs[] =  'Content-Transfer-Encoding: 7bit';
```

Respuestas a las prácticas del capítulo 15

1. En el caso particular de la inyección de código nos evita la entrada de código HTML convirtiendo los caracteres especiales en entidades HTML.
2. Falso, lo que se visualiza es el código HTML puro.
3. display_errors = off;
4. Se debe prevenir la concatenación de sentencias SQL adicionales.
5. Técnicamente es posible pero hay varias técnicas para reducir ese riesgo a un mínimo.

Respuestas a las prácticas del capítulo 16

1. Probar el código a fondo, documentar con los cambios, modularizar el código.
2. Errores de compilación, errores de lógica y errores de ejecución.
3. Sí, en el servidor de desarrollo pero no en producción.
4. Sí, en todo caso.
5. var_dump().