TEMA 2

Contenido

Características del lenguaje PHP.

Caso práctico
*El nuevo proyecto va a ponerse en marcha. En **BK Programación** las tres personas asignadas comienzan los preparativos. **Juan**, **el jefe de proyecto**, está elaborando un calendario para intentar definir las distintas fases del desarrollo. **María**, en el tiempo que le puede dedicar, se ha puesto a refrescar sus conocimientos de **programación en PHP**. **Carlos** es el que más trabajo tiene por delante, sabe que si quiere aportar algo, debe aprender a programar aplicaciones web, y pronto.*
***Carlos le pide ayuda a Juan**, que le orienta sobre los conceptos fundamentales del lenguaje y le ofrece su ayuda en todo lo que le sea posible. Sabe que es importante adquirir unos conocimientos sólidos antes de comenzar el desarrollo, para no cometer errores al principio que después sea complicado solucionar.*
***Con la ayuda de María**, ponen en funcionamiento un **servidor de aplicaciones** dentro de la empresa. De momento lo utilizarán para ir haciendo pruebas, pero dentro de poco será la plataforma sobre la que programarán la nueva aplicación.*

1.- Elementos del lenguaje PHP.

Caso práctico
***Carlos comienza su aprendizaje del nuevo lenguaje**. Conforme va avanzando, comprueba que muchos de los conceptos que aprende son similares a lo que ya conocía. Otros, sin embargo, son muy distintos y los tiene que practicar para entender bien su manejo.*
Aunque es consciente que al principio va a cometer fallos, se pone como meta utilizar lo que va aprendiendo para hacer pequeños programas que pueda volver a usar en el futuro. Y si consigue hacer algo que pueda tener alguna utilidad para la nueva aplicación, ¡mejor!

En la unidad anterior, aprendiste a preparar un entorno para programar en PHP. Además también viste algunos de los elementos que se usan en el lenguaje, como las variables y tipos de datos, comentarios, operadores y expresiones.

También sabes ya cómo se integran las etiquetas HTML con el código del lenguaje, utilizando los delimitadores `<?php` y `?>`.

En esta unidad aprenderás a utilizar otros elementos del lenguaje que te permitan crear programas completos en PHP. Los programas escritos en PHP, además encontrarse estructurados normalmente en varias páginas (ya veremos más adelante cómo se pueden comunicar datos de unas páginas a otras), suelen incluir en una misma página varios bloques de código. Cada bloque de código debe ir entre delimitadores, y en caso de que genere alguna salida, ésta se introduce en el código HTML en el mismo punto en el que figuran las instrucciones en PHP.

Por ejemplo, en las siguientes líneas tenemos dos bloques de código en PHP:

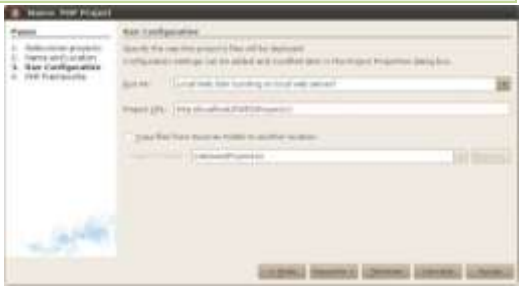
```
<body>
   <?php $a=1; ?>
   <p>Página de prueba</p>
   <?php $b=$a; ?>
…
```

Aunque no se utilice el valor de las variables, en el segundo bloque de código la variable `$a` mantiene el valor 1 que se le ha asignado anteriormente.

En esta unidad empezarás a crear tus propios programas en PHP. Para ello vas a usar el IDE NetBeans, que instalaste anteriormente. Deberías organizar tus programas en proyectos, y almacenarlos en una estructura en árbol, colgando todos por ejemplo de

/home/usuario/NetBeansProjectos/DWES. Para crear un proyecto nuevo vete a **Archivo – Proyecto Nuevo** y selecciona **PHP Application**.
En la siguiente pantalla de configuración del proyecto, debes indicar la ruta que se usará para acceder al mismo desde un navegador. Es decir, tienes que hacer que el servidor web Apache pueda acceder a la ruta anterior en la que vas a almacenar tus proyectos. Esto puedes hacerlo, por ejemplo, creando un enlace simbólico en la raíz del servidor web (`/var/www`):

```
sudo ln -s /home/usuario/NetBeansProjects/DWES/ DWES
```



De esta forma, si creas un proyecto nuevo en la ruta **/home/usuario/NetBeansProjectos/DWES/Proyecto1**, la URL que tendrás que poner en la siguiente pantalla de configuración será **http://localhost/DWES/Proyecto1.**

1.1.- Generación de código HTML.

Existen varias formas incluir contenido en la página web a partir del resultado de la ejecución de código PHP. La forma más sencilla es usando `echo`, que no devuelve nada (`void`), y genera como salida el texto de los parámetros que recibe.

```
void echo (string $arg1, ...);
```

Otra posibilidad es `print`, que funciona de forma similar. La diferencia más importante entre `print` y `echo`, es que print sólo puede recibir un parámetro y devuelve siempre 1.

```
int print (string $arg);
```

Tanto `print` como `echo` no son realmente funciones, por lo que no es obligatorio que pongas paréntesis cuando las utilices. Por ejemplo, el código del siguiente documento puede hacerse igualmente utilizando `echo`.

Utilización de la función print en PHP

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 2 : Características del Lenguaje PHP -->
<!-- Ejemplo: Utilización de print -->
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Desarrollo Web</title>
   </head>
   <body>
      <?php
         $modulo="DWES";
         print "<p>Módulo: ";
         print $modulo;
         print"</p>"
      ?>
   </body>
</html>
```

`printf` es otra opción para generar una salida desde PHP. Puede recibir varios parámetros, el primero de los cuales es siempre una cadena de texto que indica el formato que se ha de aplicar. Esa cadena debe contener un especificador de conversión por cada uno de los demás parámetros que se le pasen a la función, y en el mismo orden en que figuran en la lista de parámetros. Por ejemplo:

```
<?php
   $ciclo="DAW";
   $modulo="DWES";
   print "<p>";
   printf("%s es un módulo de %d curso de %s", $modulo, 2, $ciclo);
   print "</p>";
?>
```

1.2.- Cadenas de texto.

En PHP las cadenas de texto pueden usar tanto comillas simples como comillas dobles. Sin embargo hay una diferencia importante entre usar unas u otras. Cuando se pone una variable dentro de unas comillas dobles, se procesa y se sustituye por su valor. Así, el ejemplo anterior sobre el uso de `print` también podía haberse puesto de la siguiente forma:

```
<?php
   $modulo="DWES";
   print "<p>Módulo: $modulo</p>"
?>
```

La variable `$modulo` se reconoce dentro de las comillas dobles, y se sustituye por el valor "DWES" antes de generar la salida. Si esto mismo lo hubieras hecho utilizando comillas simples, no se realizaría sustitución alguna.

Para que PHP distinga correctamente el texto que forma la cadena del nombre de la variable, a veces es necesario rodearla entre llaves.

```
print "<p>Módulo: ${modulo}</p>"
```

Cuando se usan comillas simples, sólo se realizan dos sustituciones dentro de la cadena: cuando se encuentra la secuencia de caracteres `\'`, se muestra en la salida una comilla simple; y cuando se encuentra la secuencia `\\`, se muestra en la salida una barra invertida.

Estas secuencias se conocen como **secuencias de escape**. En las cadenas que usan comillas dobles, además de la secuencia `\\`, se pueden usar algunas más, pero no la secuencia `\'`. En esta tabla puedes ver las secuencias de escape que se pueden utilizar, y cuál es su resultado.

En PHP tienes dos operadores exclusivos para trabajar con cadenas de texto. Con el operador de concatenación punto (.) puedes unir las dos cadenas de texto que le pases como operandos. El operador de asignación y concatenación (.=) concatena al argumento del lado izquierdo la cadena del lado derecho.

```
<?php
   $a = "Módulo ";
   $b = $a . "DWES"; // ahora $b contiene "Módulo DWES"
   $a .= "DWES"; // ahora $a también contiene "Módulo DWES"
?>
```

En PHP tienes otra alternativa para crear cadenas: la sintaxis **heredoc**. Consiste en poner el operador <<< seguido de un identificador de tu elección, y a continuación y empezando en la línea siguiente la cadena de texto, sin utilizar comillas. La cadena finaliza cuando escribes ese mismo identificador en una nueva línea. Esta línea de cierre no debe llevar más caracteres, ni siquiera espacios o sangría, salvo quizás un punto y coma después del identificador.

```
<?php
   $a = <<<MICADENA
   Desarrollo de Aplicaciones Web<br />
   Desarrollo Web en Entorno Servidor
   MICADENA;
   print $a;
?>
```

El texto se procesa de igual forma que si fuera una cadena entre comillas dobles, sustituyendo variables y secuencias de escape. Si no quisieras que se realizara ninguna sustitución, debes poner el identificador de apertura entre comillas simples.

```
$a = <<<'MICADENA'
...
MICADENA;
```

1.3.- Funciones relacionadas con los tipos de datos(I).

En PHP existen funciones específicas para comprobar y establecer el tipo de datos de una variable, **`gettype`** obtiene el tipo de la variable que se le pasa como parámetro y devuelve una cadena de texto, que puede ser **`array`**, **`boolean`**, **`double`**, **`integer`**, **`object`**, **`string`**, **`null`**, **`resource`** o **`unknown type`**.

También podemos comprobar si la variable es de un tipo concreto utilizando una de las siguientes funciones: **`is_array()`**, **`is_bool()`**, **`is_float()`**,**`is_integer()`**, **`is_null()`**, **`is_numeric()`**, **`is_object()`**, **`is_resource()`**, **`is_scalar()`** e **`is_string()`**. Devuelven `true` si la variable es del tipo indicado.

Análogamente, para establecer el tipo de una variable utilizamos la función **`settype`** pasándole como parámetros la variable a convertir, y una de las siguientes cadenas: **`boolean`**, **`integer`**, **`float`**, **`string`**, **`array`**, **`object`** o **`null`**. La función **`settype`** devuelve true si la conversión se realizó correctamente, o false en caso contrario.

```
<?php
   $a = $b = "3.1416"; // asignamos a las dos variables la misma cadena de texto
   settype($b, "float"); // y cambiamos $b a tipo float
   print "\$a vale $a y es de tipo ".gettype($a);
   print "<br />";
   print "\$b vale $b y es de tipo ".gettype($b);
?>
```

El resultado del código anterior es:

```
$a vale 3.1416 y es de tipo string
$b vale 3.1416 y es de tipo double
```

Si lo único que te interesa es saber si una variable está definida y no es **`null`**, puedes usar la función **`isset`**. La función **`unset`** destruye la variable o variables que se le pasa como parámetro.

```
<?php
   $a = "3.1416";
   if (isset($a)) // la variable $a está definida
   unset($a); //ahora ya no está definida
?>
```

Es importante no confundir el que una variable esté definida o valga `null`, con que se considere como vacía debido al valor que contenga. Esto último es lo que nos indica la función `empty`.

empty

Determina si una variable está vacía. Tiene la sintaxis: `bool empty(*$var*)`

Una variable se considera vacía si no existe o si su valor es igual `FALSE`. No genera una advertencia si a la variable no existe.

`empty()` sólo comprueba variables ya que cualquier otra cosa producirá un error de análisis. En otras palabras, lo siguiente no funcionará: `empty(trim($name))`. Use en su lugar `trim($name) == false`.

No se genera una advertencia si la variable no existe. Esto significa que `empty()` es esencialmente el equivalente conciso de `!isset($var) || $var == false`.

Devuelve `FALSE` si `var` existe y tiene un valor no vacío, distinto de cero. De otro modo devuelve `TRUE`.

Las siguientes expresiones son consideradas como vacías:

- ✓ ***"" (una cadena vacía)***
- ✓ ***0 (0 como un integer)***
- ✓ ***(0 como un float)***
- ✓ ***"0" (0 como un string)***
- ✓ **NULL**
- ✓ **FALSE**
- ✓ ***array() (un array vacío)***
- ✓ ***$var;*** (una variable declarada, pero sin un valor)

Existe también en PHP una función, `define`, con la que puedes definir constantes, esto es, identificadores a los que se les asigna un valor que no cambia durante la ejecución del programa.

```
bool define ( string $identificador , mixed $valor [, bool $case_insensitive = false ] );
```

Los identificadores no van precedidos por el signo "$" y suelen escribirse en mayúsculas, aunque existe un tercer parámetro opcional, que si vale true hace que se reconozca el identificador independientemente de si está escrito en mayúsculas o en minúsculas.

```
<?php
   define ("PI", 3.1416, true);
   print "El valor de PI es ".pi; //El identificador se reconoce tanto por PI como por pi
?>
```

Sólo se permiten los siguientes tipos de valores para las constantes: `integer`, `float`, `string`, `boolean` y `null`.

A partir de la versión 7 de php, se permite también definir arrays como constantes.

1.3.1.- Funciones relacionadas con los tipos de datos (II).

En PHP no existe un tipo de datos específico para trabajar con fechas y horas. La información de fecha y hora se almacena internamente como un número entero. Sin embargo, dentro de las funciones de PHP tienes a tu disposición unas cuantas para trabajar con ese tipo de datos.

Una de las más útiles es quizás la función `date`, que te permite obtener una cadena de texto a partir de una fecha y hora, con el formato que tú elijas. La función recibe dos parámetros, la descripción del formato y el número entero que identifica la fecha, y devuelve una cadena de texto formateada.

```
string date (string $formato [, int $fechahora]);
```

El formato lo debes componer utilizando como base una serie de caracteres de los que figuran en la siguiente tabla.

Función date: caracteres de formato para fechas y horas.

Carácter	Resultado
d	día del mes con dos dígitos.
j	día del mes con uno o dos dígitos (sin ceros iniciales).
z	día del año, comenzando por el cero (0 = 1 de enero).
N	día de la semana (1 = lunes, ..., 7 = domingo).
w	día de la semana (0 = domingo, ..., 6 = sábado).
l	texto del día de la semana, en inglés (Monday, ..., Sunday).
D	texto del día de la semana, solo tres letras, en inglés (Mon, ..., Sun).
W	número de la semana del año.
m	número del mes con dos dígitos.
n	número del mes con uno o dos dígitos (sin ceros iniciales).
t	número de días que tiene el mes.
F	texto del día del mes, en inglés (January, ..., December).
M	texto del día del mes, solo tres letras, en inglés (Jan, ..., Dec).
Y	número del año.
y	dos últimos dígitos del número del año.
L	1 si el año es bisiesto, 0 si no lo es.
h	formato de 12 horas, siempre con dos dígitos.
H	formato de 24 horas, siempre con dos dígitos.
g	formato de 12 horas, con uno o dos dígitos (sin ceros iniciales).
G	formato de 24 horas, con uno o dos dígitos (sin ceros iniciales).
i	minutos, siempre con dos dígitos.
s	segundos, siempre con dos dígitos.
u	microsegundos.
a	am o pm, en minúsculas.
A	AM o PM, en mayúsculas.
r	fecha entera con formato RFC 2822.

Además, el segundo parámetro es opcional. Si no se indica, se utilizará la hora actual para crear la cadena de texto.

Para que el sistema pueda darte información sobre tu fecha y hora, debes indicarle tu zona horaria. Puedes hacerlo con la función `date_default_timezone_set`. Para establecer la zona horaria en España peninsular debes indicar:

```
date_default_timezone_set('Europe/Madrid');
```

En la documentación de PHP puedes consultar las distintas zonas horarias que se pueden indicar.

http://es2.php.net/manual/es/timezones.php

Si utilizas alguna función de fecha y hora sin haber establecido previamente tu zona horaria, lo más probable es que recibas un error o mensaje de advertencia de PHP indicándolo.

Otras funciones como `getdate` devuelven un array con información sobre la fecha y hora actual.

1.4.- Variables especiales de PHP.

En la unidad anterior ya aprendiste qué eran y cómo se utilizaban las variables globales. PHP incluye unas cuantas variables internas predefinidas que pueden usarse desde cualquier ámbito, por lo que reciben el nombre de **variables superglobales**. Ni siquiera es necesario que uses `global` para acceder a ellas.

Cada una de estas variables es un array que contiene un conjunto de valores (en esta unidad veremos más adelante cómo se pueden utilizar los arrays). Las variables superglobales disponibles en PHP son las siguientes:

`$_SERVER`. Contiene información sobre el entorno del servidor web y de ejecución. Entre la información que nos ofrece esta variable, tenemos:

Principales valores de la variable $_SERVER

Valor	Contenido
$_SERVER['PHP_SELF']	guión que se está ejecutando actualmente.
$_SERVER['SERVER_ADDR']	dirección IP del servidor web.
$_SERVER['SERVER_NAME']	nombre del servidor web.
$_SERVER['DOCUMENT_ROOT']	directorio raíz bajo el que se ejecuta el guión actual.
$_SERVER['REMOTE_ADDR']	dirección IP desde la que el usuario está viendo la página.
$_SERVER['REQUEST_METHOD']	método utilizado para acceder a la página ('GET', 'HEAD', 'POST' o 'PUT')

En la documentación de PHP puedes consultar toda la información que ofrece **$_SERVER**:
http://es.php.net/manual/es/reserved.variables.server.php

`$_GET`, `$_POST` y `$_COOKIE` contienen las variables que se han pasado al guión actual utilizando respectivamente los métodos GET (parámetros en la URL), HTTP POST y Cookies HTTP.
`$_REQUEST` junta en uno solo el contenido de los tres arrays anteriores, `$_GET`, `$_POST` y `$_COOKIE`.
`$_ENV` contiene las variables que se puedan haber pasado a PHP desde el entorno en que se ejecuta.
`$_FILES` contiene los ficheros que se puedan haber subido al servidor utilizando el método POST.
`$_SESSION` contiene las variables de sesión disponibles para el guión actual.

2.- Estructuras de control.

Caso práctico
¡Bien! Ya están claros los fundamentos del lenguaje.
Pero con lo visto hasta el momento, solo es posible hacer programas muy sencillos. Para poder empezar a programar, Carlos sabe qué debe estudiar a continuación. Una de las partes más importantes de cualquier lenguaje es la que permite tomar decisiones, es decir, las sentencias que se pueden usar para indicar bajo qué condiciones se debe ejecutar una instrucción o un bloque de instrucciones. Y como no, también las sentencias para repetir la ejecución de ciertas líneas de código. Cuando domine esas estructuras, podrá empezar a probar todo lo que lleva aprendido.

En PHP los guiones se construyen en base a sentencias. Utilizando llaves, puedes agrupar las sentencias en conjuntos, que se comportan como si fueran una única sentencia.

Para definir el flujo de un programa en PHP, al igual que en la mayoría de lenguajes de programación, hay sentencias para dos tipos de **estructuras de control**: **sentencias condicionales**, que permiten definir las condiciones bajo las que debe ejecutarse una sentencia o un bloque de sentencias; y **sentencias de bucle**, con las que puedes definir si una sentencia o conjunto de sentencias se repite o no, y bajo qué condiciones.

Además, en PHP puedes usar también (aunque no es recomendable) la sentencia `goto`, que te permite saltar directamente a otro punto del programa que indiques mediante una etiqueta.

```
<?php
   $a = 1;
   goto salto;
   $a++;  //esta sentencia no se ejecuta
   salto:
   echo $a;  // el valor obtenido es 1
?>
```

2.1.- Condicionales.

`if / elseif / else`. La sentencia **`if`** permite definir una expresión para ejecutar o no la sentencia o conjunto de sentencias siguiente. Si la expresión se evalúa a `true` (verdadero), la sentencia se ejecuta. Si se evalúa a `false` (falso), no se ejecutará.

Cuando el resultado de la expresión sea false, puedes utilizar **`else`** para indicar una sentencia o grupo de sentencias a ejecutar en ese caso. Otra alternativa a **`else`** es utilizar **`elseif`** y escribir una nueva expresión que comenzará un nuevo condicional.

```
<?php
   if ($a < $b)
      print "a es menor que b";
   elseif ($a > $b)
      print "a es mayor que b";
   else
      print "a es igual a b";
?>
```

Cuando, como sucede en el ejemplo, la sentencia **`if elseif`** o **`else`** actúe sobre una única sentencia, no será necesario usar llaves. Tendrás que usar llaves para formar un conjunto de sentencias siempre que quieras que el condicional actúe sobre más de una sentencia.

`switch`. La sentencia **`switch`** es similar a enlazar varias sentencias **`if`** comparando una misma variable con diferentes valores. Cada valor va en una sentencia **`case`**. Cuando se encuentra una coincidencia, comienzan a ejecutarse las sentencias siguientes hasta que acaba el bloque **`switch`**, o hasta que se

encuentra una sentencia **break**. Si no existe coincidencia con el valor de ningún **case**, se ejecutan las sentencias del bloque **default**, en caso de que exista.

```
<?php
   switch ($a) {
      case 0:
         print "a vale 0";
         break;
      case 1:
         print "a vale 1";
         break;
      default:
         print "a no vale 0 ni 1";
   }
?>
```

Haz una página web que muestre la fecha actual en castellano, incluyendo el día de la semana, con un formato similar al siguiente: "Miércoles, 13 de Abril de 2011".

2.2.- Bucles.

- ✓ **`while`:** Usando `while` puedes definir un bucle que se ejecuta mientras se cumpla una expresión. La expresión se evalúa antes de comenzar cada ejecución del bucle.

```
<?php
  $a = 1;
  while ($a < 8)
     $a += 3;
  print $a; // el valor obtenido es 10
?>
```

- ✓ **`do / while`:** Es un bucle similar al anterior, pero la expresión se evalúa al final, con lo cual se asegura que la sentencia o conjunto de sentencias del bucle se ejecutan al menos una vez.

```
<?php
  $a = 5;
  do
     $a -= 3;
  while ($a > 10);
  print $a; // el bucle se ejecuta una sola vez, con lo que el valor obtenido es 2
?>
```

- ✓ **for:** Son los bucles más complejos de PHP. Al igual que los del lenguaje C, se componen de tres expresiones:

```
for (expr1; expr2; expr3)
  sentencia o conjunto de sentencias;
```

La primera expresión, `expr1`, se ejecuta solo una vez al comienzo del bucle.

La segunda expresión, `expr2`, se evalúa para saber si se debe ejecutar o no la sentencia o conjunto de sentencias. Si el resultado el false, el bucle termina.

Si el resultado es true, se ejecutan las sentencias y al finalizar se ejecuta la tercera expresión, `expr3`, y se vuelve a evaluar `expr2` para decidir si se vuelve a ejecutar o no el bucle.

```
<?php
     for ($a = 5; $a<10; $a+=3) {
          print $a; // Se muestran los valores 5 y 8
          print "<br />";
     }
?>
```

Puedes anidar cualquiera de los bucles anteriores en varios niveles. También puedes usar las sentencias **`break`**, para salir del bucle, y **`continue`**, para omitir la ejecución de las sentencias restantes y volver a la comprobación de la expresión respectivamente.

En el siguiente videotutorial puedes ver con ejemplos la utilización de bucles en PHP.
http://www.youtube.com/watch?feature=player_embedded&v=VjqJwSy_BPQ

Realiza los ejercicios de la relación 1

3.- Funciones.

Caso práctico
Juan *observa con agrado los* ***progresos que va haciendo Carlos*** *en su aprendizaje del lenguaje PHP. Con la ilusión que está poniendo, se integrará sin problemas en el nuevo proyecto. Cuantos más puedan colaborar, mejor.*
Tras lo que ya ha visto, le recomienda que aprenda a ***crear y utilizar funciones****. Sabe que no sólo es muy importante saber usarlas, sino también conocer todas las que hay disponibles en el lenguaje, o al menos, saber cómo buscarlas. En un lenguaje abierto como PHP, si sabes utilizar el código que ya hay programado puedes ahorrarte una gran parte del trabajo.*

Cuando quieres repetir la ejecución de un bloque de código, puedes utilizar un bucle. Las **funciones** tienen una utilidad similar: **nos permiten asociar una etiqueta** (el nombre de la función) **con un bloque de código** a ejecutar. Además, al usar funciones estamos ayudando a estructurar mejor el código. Como ya sabes, las funciones permiten crear variables locales que no serán visibles fuera del cuerpo de las mismas.

Como programador puedes aprovecharte de la gran cantidad de funciones disponibles en PHP. De éstas, muchas están incluidas en el núcleo de PHP y se pueden usar directamente. Otras muchas se encuentran disponibles en forma de extensiones, y se pueden incorporar al lenguaje cuando se necesitan.

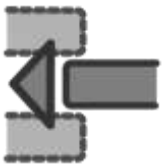
Con la distribución de PHP se incluyen varias extensiones. Para poder usar las funciones de una extensión, tienes que asegurarte de activarla mediante el uso de una directiva **extensión** en el fichero `php.ini`. Muchas otras extensiones no se incluyen con PHP y antes de poder utilizarlas tienes que descargarlas.

Para obtener extensiones para el lenguaje PHP puedes utilizar PECL. PECL es un repositorio de extensiones para PHP. Junto con PHP se incluye un **comando pecl** que puedes utilizar para instalar extensiones de forma sencilla:

```
pecl install nombre_extensión
```

En el manual de PHP tienes más información sobre PECL.
http://es2.php.net/manual/es/install.pecl.php

3.1.- Inclusión de ficheros externos.



Conforme vayan creciendo los programas que hagas, verás que resulta trabajoso encontrar la información que buscas dentro del código. En ocasiones resulta útil agrupar ciertos grupos de funciones o bloques de código, y ponerlos en un fichero aparte. Posteriormente, puedes hacer referencia a esos ficheros para que PHP incluya su contenido como parte del programa actual.

Para incorporar a tu programa contenido de un archivo externo, tienes varias posibilidades:

- ✓ **`include`:** Evalúa el contenido del fichero que se indica y lo incluye como parte del fichero actual, en el mismo punto en que se realiza la llamada. La ubicación del fichero puede especificarse utilizando una ruta absoluta, pero lo más usual es con una ruta relativa. En este caso, se toma como base la ruta que se especifica en la directiva `include path` del fichero `php.ini`. Si no se encuentra en esa ubicación, se buscará también en el directorio del guión actual, y en el directorio de ejecución.

Te presentamos un ejemplo de utilización de `include`.

definiciones.php

```
<?php
   $modulo = 'DWES';
   $ciclo = 'DAW';
?>
```

programa.php

```
<?php
   print "Módulo $modulo del ciclo $ciclo<br />"; //Solo muestra "Modulo del ciclo"
   include 'definiciones.php';
   print " Módulo $modulo del ciclo $ciclo<br />"; // muestra "Modulo DWES del ciclo DAW"
?>
```

Cuando se comienza a evaluar el contenido del fichero externo, se abandona de forma automática el modo PHP y su contenido se trata en principio como etiquetas HTML. Por este motivo, es necesario delimitar el código PHP que contenga nuestro archivo externo utilizando dentro del mismo los delimitadores `<?php` y `?>`.

- ✓ **`include_once`**: Si por equivocación incluyes más de una vez un mismo fichero, lo normal es que obtengas algún tipo de error (por ejemplo, al repetir una definición de una función).`include_once` funciona exactamente igual que `include`, pero solo incluye aquellos ficheros que aún no se hayan incluido.
- ✓ **`require`**: Si el fichero que queremos incluir no se encuentra, `include` da un aviso y continua la ejecución del guión. La diferencia más importante al usar `require` es que en ese caso, cuando no se puede incluir el fichero, se detiene la ejecución del guión.
- ✓ **`require_once`**. Es la combinación de las dos anteriores. Asegura la inclusión del fichero indicado solo una vez, y genera un error si no se puede llevar a cabo.

3.2.- Ejecución y creación de funciones.

Ya sabes que para hacer una llamada a una función, basta con poner su nombre y unos paréntesis.

```
<?php
   phpinfo();
?>
```

Para crear tus propias funciones, deberás usar la palabra **`function`**.

```
<?php
   function precio_con_iva() {
      global $precio;
      $precio_iva = $precio * 1.18;
      print "El precio con IVA es ".$precio iva;
   }
   $precio = 10;
   precio_con_iva();
?>
```

En PHP no es necesario que definas una función antes de utilizarla, excepto cuando está condicionalmente definida como se muestra en el siguiente ejemplo:

```
<?php
   $iva = true;
   $precio = 10;
   precio con iva();  // Da error, pues aquí aún no está definida la función
   if ($iva) {
      function precio_con_iva() {
         global $precio;
         $precio iva = $precio * 1.18;
         print "El precio con IVA es ".$precio_iva;
      }
   }
   precio_con_iva();  // Aquí ya no da error
?>
```

Cuando una función está definida de una forma condicional sus definiciones deben ser procesadas antes de ser llamadas. Por tanto, la definición de la función debe estar antes de cualquier llamada. .

3.3.- Argumentos.

En el ejemplo anterior en la función usabas una variable global, lo cual no es una buena práctica. Siempre es mejor utilizar argumentos o parámetros al hacer la llamada. Además, en lugar de mostrar el resultado en pantalla o guardar el resultado en una variable global, las funciones pueden devolver un valor usando la sentencia `return`. Cuando en una función se encuentra una sentencia `return`, termina su procesamiento y devuelve el valor que se indica.

Puedes reescribir la función anterior de la siguiente forma:

```
<?php
   function precio_con_iva($precio) {
      return $precio * 1.18;
   }
   $precio = 10;
   $precio_iva = precio_con_iva($precio);
   print "El precio con IVA es ".$precio_iva
?>
```

Los argumentos se indican en la definición de la función como una lista de variables separada por comas. No se indica el tipo de cada argumento, al igual que no se indica si la función va a devolver o no un valor (si una función no tiene una sentencia `return`, devuelve `null` al finalizar su procesamiento).

Al definir la función, puedes indicar valores por defecto para los argumentos, de forma que cuando hagas una llamada a la función puedes no indicar el valor de un argumento; en este caso se toma el valor por defecto indicado.

```
<?php
   function precio_con_iva($precio, $iva=0.18) {
      return $precio * (1 + $iva);
   }
   $precio = 10;
   $precio iva = precio con iva($precio);
   print "El precio con IVA es ".$precio_iva
?>
```

Puede haber valores por defecto definidos para varios argumentos, pero en la lista de argumentos de la función todos ellos deben estar a la derecha de cualquier otro argumento sin valor por defecto.

En los ejemplos anteriores los argumentos se pasaban **por valor**. Esto es, cualquier cambio que se haga dentro de la función a los valores de los argumento no se reflejará fuera de la función. Si quieres que esto ocurra debes definir el parámetro para que su valor se pase **por referencia**, añadiendo el símbolo `&` antes de su nombre.

```
<?php
   function precio_con_iva(&$precio, $iva=0.18) {
      $precio *= (1 + $iva);      }
   $precio = 10;
   precio_con_iva($precio);
   print "El precio con IVA es ".$precio
?>
```

Anteriormente hiciste un ejercicio que mostraba la fecha actual en castellano. Con el mismo objetivo (puedes utilizar el código ya hecho), crea una función que devuelva una cadena de texto con la fecha en castellano, e introdúcela en un fichero externo. Después crea una página en PHP que incluya ese fichero y utilice la función para mostrar en pantalla la fecha obtenida.

4.- Tipos de datos compuestos.

Caso práctico

***Es muy raro el programa que utilice solo tipos simples**, y más en PHP. **Carlos ya ha asumido que tendrá que utilizar arrays** para casi cualquier código que haga. La información del servidor, los datos que introduce el usuario, o las cadenas de texto. Gran parte de las variables que se usan en un programa están en forma de array.*

*Por tanto, el siguiente paso está claro: **hay que dominar los arrays**. Crearlos, utilizarlos, recorrerlos... Y como acaba de aprender, antes de ponerse a programar le echará un vistazo a las funciones que ya existen para manejarlos. Si las puede aprovechar en sus programas, ¡mejor!*

Un tipo de datos compuesto es aquel que te permite almacenar más de un valor. En PHP puedes utilizar dos tipos de datos compuestos: el **array** y el **objeto**. Los objetos los veremos más adelante; vamos a empezar con los arrays.

Un **array** es un tipo de datos que nos permite almacenar varios valores. Cada miembro del **array** se almacena en una posición a la que se hace referencia utilizando un valor clave. Las claves pueden ser numéricas o asociativas.

```
// array numérico
$modulos1 = array(0 => "Programación", 1 => "Bases de datos", ..., 9 => "Desarrollo web en entorno servidor");
// array asociativo
$modulos2 = array("PR" => "Programación", "BD" => "Bases de datos", ..., "DWES" => "Desarrollo web en entorno servidor");
```

En PHP existe la función `print_r`, que nos muestra todo el contenido del array que le pasamos. Es muy útil para tareas de depuración.

http://es.php.net/manual/es/function.print-r.php

Para hacer referencia a los elementos almacenados en un array, tienes que utilizar el valor clave entre corchetes:

```
$modulos1 [9]
$modulos2 ["DWES"]
```

Los arrays anteriores son vectores, esto es, arrays unidimensionales. En PHP puedes crear también arrays de varias dimensiones almacenando otro array en cada uno de los elementos de un array.

```
// array bidimensional
$ciclos = array(
   "DAW" => array ("PR" => "Programación", "BD" => "Bases de datos", ..., "DWES" => "Desarrollo web en entorno servidor"),
   "DAM" => array ("PR" => "Programación", "BD" => "Bases de datos", ..., "PMDM" => "Programación multimedia y de dispositivos móviles")
);
```

Para hacer referencia a los elementos almacenados en un **array multidimensional**, debes indicar las claves para cada una de las dimensiones:

```
$ciclos ["DAW"] ["DWES"]
```

En PHP no es necesario que indiques el tamaño del array antes de crearlo. Ni siquiera es necesario indicar que una variable concreta es de tipo array. Simplemente puedes comenzar a asignarle valores:

```
// array numérico
$modulos1 [0] = "Programación";
$modulos1 [1] = "Bases de datos";
...
$modulos1 [9] = "Desarrollo web en entorno servidor";
// array asociativo
$modulos2 ["PR"] = "Programación";
$modulos2 ["BD"] = "Bases de datos";
...
$modulos2 ["DWES"] = "Desarrollo web en entorno servidor";
```

Ni siquiera es necesario que especifiques el valor de la clave. Si la omites, el array se irá llenando a partir de la última clave numérica existente, o de la posición 0 si no existe ninguna:

```
$modulos1 [ ] = "Programación";
$modulos1 [ ] = "Bases de datos";
...
$modulos1 [ ] = "Desarrollo web en entorno servidor";
```

4.1.- Recorrer arrays (I).

Las **cadenas de texto** o **strings** se pueden tratar como arrays en los que se almacena una letra en cada posición, siendo 0 el índice correspondiente a la primera letra, 1 el de la segunda, etc.

```
// cadena de texto
$modulo = "Desarrollo web en entorno servidor";
// $modulo[3] == "a";
```

Para recorrer los elementos de un arrays, en PHP puedes usar un bucle específico: `foreach`. Utiliza una variable temporal para asignarle en cada iteración el valor de cada uno de los elementos del arrays. Puedes usarlo de dos formas. Recorriendo sólo los elementos:

```
$modulos = arrays("PR" => "Programación", "BD" => "Bases de datos", ..., "DWES" => "Desarrollo
web en entorno servidor");
foreach ($modulos as $modulo) {
   print "Módulo: ".$modulo. "<br />"
}
```

O recorriendo los elementos y sus valores clave de forma simultánea:

```
$modulos = array("PR" => "Programación", "BD" => "Bases de datos", ..., "DWES" => "Desarrollo
web en entorno servidor");
foreach ($modulos as $codigo => $modulo) {
   print "El código del módulo ".$modulo." es ".$codigo."<br />"
}
```

Haz una página PHP que utilice foreach para mostrar todos los valores del array $_SERVER en una tabla con dos columnas. La primera columna debe contener el nombre de la variable, y la segunda su valor.

4.2.- Funciones relacionadas con los tipos de datos compuestos.

Además de asignando valores directamente, **la función array permite crear un array con una sola línea de código**, tal y como vimos anteriormente. Esta función recibe un conjunto de parámetros, y crea un array a partir de los valores que se le pasan. Si en los parámetros no se indica el valor de la clave, crea un array numérico (con base 0). Si no se le pasa ningún parámetro, crea un array vacío.

```
$a = array();  // array vacío
$modulos = array("Programación", "Bases de datos", ..., "Desarrollo web en entorno servidor");
// array numérico
```

Una vez definido un array puedes añadir nuevos elementos (no definiendo el índice, o utilizando un índice nuevo) y modificar los ya existentes (utilizando el índice del elemento a modificar). También se pueden eliminar elementos de un array utilizando la función `unset`.

En el caso de los arrays numéricos, eliminar un elemento significa que las claves del mismo ya no estarán consecutivas.

```
unset ($modulos [0]);
// El primer elemento pasa a ser $modulos [1] == "Bases de datos";
```

La función `array_values` recibe un array como parámetro, y devuelve un nuevo array con los mismos elementos y con índices numéricos consecutivos con base 0.

Para comprobar si una variable es de tipo array, utiliza la función `is_array`. Para obtener el número de elementos que contiene un array, tienes la función `count`.

Si quieres buscar un elemento concreto dentro de un array, puedes utilizar la función `in_array`. Recibe como parámetros el elemento a buscar y la variable de tipo array en la que buscar, y devuelve true si encontró el elemento o false en caso contrario.

```
$modulos = array("Programación", "Bases de datos", "Desarrollo web en entorno servidor");
$modulo = "Bases de datos";
if (in_array($modulo, $modulos)) printf "Existe el módulo de nombre ".$modulo;
```

Otra posibilidad es la función `array_search`, que recibe los mismos parámetros pero devuelve la clave correspondiente al elemento, o false si no lo encuentra.

Y si lo que quieres buscar es un clave en un array, tienes la función `array_key_exists`, que devuelve `true` o `false`.

En realidad en PHP hay muchas funciones para gestionar arrays. Puedes consultar una lista completa en el manual online de PHP.

http://es.php.net/manual/es/ref.array.php

Realiza la práctica de arrays

5.- Formularios web.

Caso práctico
***Carlos** está viendo que el esfuerzo que le dedica al **aprendizaje del nuevo lenguaje** empieza a dar sus frutos. Hace unos días casi no sabía ni que existía PHP, y ahora ya es capaz de realizar programas sencillos por sí mismo.*
*Para poder avanzar aún más, sabe cuál ha de ser su siguiente paso: **obtener y utilizar información de un usuario**. De esta forma, los programas que haga no serán lineales, sino que tendrán un comportamiento u otro en función de los datos que aporte el usuario.*
*Como le ha comentado **Juan**, para obtener información de un usuario, en PHP se utilizan los **formularios HTML**. ¡A por ellos!*

La forma natural para hacer llegar a la aplicación web los datos del usuario desde un navegador, es utilizar **formularios HTML**.

Los formularios HTML van encerrados siempre entre las etiquetas `<FORM> </FORM>`. Dentro de un formulario se incluyen los elementos sobre los que puede actuar el usuario, principalmente usando las etiquetas `<INPUT>`, `<SELECT>`, `<TEXTAREA>` y `<BUTTON>`.

El atributo `action` del elemento `FORM` indica la página a la que se le enviarán los datos del formulario. En nuestro caso se tratará de un guión PHP.

Por su parte, el atributo `method` especifica el método usado para enviar la información. Este atributo puede tener dos valores:

- ✓ `get`: con este método los datos del formulario se agregan al URI utilizando un signo de interrogación "?" como separador.
- ✓ `post`: con este método los datos se incluyen en el cuerpo del formulario y se envían utilizando el protocolo HTTP.

Como vamos a ver, los datos se recogerán de distinta forma dependiendo de cómo se envíen.

Crea un formulario HTML para introducir el nombre del alumno y el ciclo que cursa, a escoger entre "Desarrollo Web en Entorno Servidor" y "Desarrollo Web en Entorno Cliente". Envía el resultado a la página "procesa.php", que será la encargada de procesar los datos.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 2 : Características del Lenguaje PHP -->
<!-- Ejemplo: Formulario web -->
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Formulario web</title>
   </head>
   <body>
      <form name="input" action="procesa.php" method="post">
         Nombre del alumno: <input type="text" name="nombre" /><br />
         <p>Ciclos que cursa:</p>
         <input type="checkbox" name="ciclos[]" value="DWES" /> Desarrollo web en entorno
            servidor<br />
         <input type="checkbox" name="ciclos[]" value="DWEC" /> Desarrollo web en entorno
            cliente<br />
         <br />
         <input type="submit" value="Enviar" />
      </form>
   </body>
</html>
```

Fíjate que si en un formulario web tienes que enviar alguna variable en la que sea posible almacenar más de un valor, como es el caso de las casillas de verificación en el ejemplo anterior (se pueden marcar varias a la vez), tendrás que ponerle corchetes al nombre de la variable para indicar que se trata de un array.

5.1.- Procesamiento de la información devuelta por un formulario web.

En el ejemplo anterior creaste un **formulario en una página HTML** que recogía datos del usuario y los enviaba a una página PHP para que los procesara. Como usaste el método **POST**, los datos se pueden recoger utilizando la variable `$_POST`. Si simplemente los quisieras mostrar por pantalla, éste podría ser el código de "procesa.php":

Código de "procesa.php"

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 2 : Características del Lenguaje PHP -->
<!-- Ejemplo: Procesar datos post -->
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Desarrollo Web</title>
   </head>
   <body>
      <?php
         $nombre = $ POST['nombre'];
         $modulos = $ POST['modulos'];
         print "Nombre: ".$nombre."<br />";
         foreach ($modulos as $modulo) {
            print "Modulo: ".$modulo."<br />";
         }
      ?>
   </body>
</html>
```

Si por el contrario hubieras usado el método **GET**, el código necesario para procesar los datos sería similar; simplemente haría falta cambiar la variable `$_POST` por `$_GET`.

Código necesario para procesar los datos

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 2 : Características del Lenguaje PHP -->
<!-- Ejemplo: Procesar datos get -->
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Desarrollo Web</title>
   </head>
   <body>
      <?php
         $nombre = $ GET['nombre'];
         $modulos = $ GET['modulos'];
         print "Nombre: ".$nombre."<br />";
         foreach ($modulos as $modulo) {
            print "Modulo: ".$modulo."<br />";
         }
      ?>
   </body>
</html>
```

En cualquiera de los dos casos podrías haber usado `$_REQUEST` sustituyendo respectivamente a `$_POST` y a `$_GET`.

Ejemplo formulario web utilizando request

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 2 : Características del Lenguaje PHP -->
<!-- Ejemplo: Procesar datos request -->
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Desarrollo Web</title>
   </head>
   <body>
      <?php
         $nombre = $_REQUEST['nombre'];
         $modulos = $_REQUEST['modulos'];
         print "Nombre: ".$nombre."<br />";
         foreach ($modulos as $modulo) {
            print "Modulo: ".$modulo."<br />";
         }
      ?>
   </body>
</html>
```

Siempre que sea posible, es preferible **validar los datos que se introducen en el navegador antes de enviarlos**. Para ello deberás usar código en **lenguaje Javascript**.

Si por algún motivo hay datos que se tengan que validar en el servidor, por ejemplo, porque necesites comprobar que los datos de un usuario no existan ya en la base de datos antes de introducirlos, será necesario hacerlo con código PHP en la página que figura en el atributo `action` del formulario.

En este caso, una posibilidad que deberás tener en cuenta es usar la misma página que muestra el formulario como destino de los datos. Si tras comprobar los datos éstos son correctos, se reenvía a otra página. Si son incorrectos, se rellenan los datos correctos en el formulario y se indican cuáles son incorrectos y por qué.

Para hacerlo de este modo, tienes que comprobar si la página recibe datos (hay que mostrarlos y no generar el formulario), o si no recibe datos (hay que mostrar el formulario). Esto se puede hacer utilizando la función `isset` con una variable de las que se deben recibir (por ejemplo, poniéndole un nombre al botón de enviar y comprobando sobre él). En el siguiente código de ejemplo se muestra cómo hacerlo.

Procesar datos en la misma página que el formulario

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 2 : Características del Lenguaje PHP -->
<!-- Ejemplo: Procesar datos en la misma página que el formulario -->
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Desarrollo Web</title>
   </head>
   <body>
      <?php
         if (isset($_POST['enviar'])) {
            $nombre = $_POST['nombre'];
            $modulos = $ POST['modulos'];
            print "Nombre: ".$nombre."<br />";
            foreach ($modulos as $modulo) {
```

```
                print "Modulo: ".$modulo."<br />";
             }
          } else {
      ?>
      <form name="input" action="<?php echo $_SERVER['PHP_SELF'];?>" method="post">
         Nombre del alumno: <input type="text" name="nombre" /><br />
         <p>Módulos que cursa:</p>
         <input type="checkbox" name="modulos[]" value="DWES" />
         Desarrollo web en entorno servidor<br />
         <input type="checkbox" name="modulos[]" value="DWEC" />
         Desarrollo web en entorno cliente<br />
         <br />
         <input type="submit" value="Enviar" name="enviar"/>
      </form>
      <?php
         }
      ?>
   </body>
</html>
```

Fíjate en la forma de englobar el formulario dentro de una sentencia `else` para que sólo se genere si no se reciben datos en la página. Además, para enviar los datos a la misma página que contiene el formulario puedes usar `$_SERVER['PHP_SELF']` para obtener su nombre; esto hace que no se produzca un error aunque la página se cambie de nombre.

5.2.- Generación de formularios web en PHP.

Vamos a volver sobre el ejemplo anterior, revisando los datos que se obtienen antes de mostrarlos. Concretamente, tienes que comprobar que el nombre no esté vacío, y que se haya seleccionado como mínimo uno de los módulos.

Además, en el caso de que falte algún dato, deberás generar el formulario rellenando aquellos datos que el usuario haya introducido correctamente.

Lo primero que tienes que hacer es la validación de los datos. En el ejemplo propuesto será algo así como:

```
if (!empty($_POST['modulos']) && !empty($_POST['nombre'])) {
   // Aquí se incluye el código a ejecutar cuando los datos son correctos
}
else {
   // Aquí generamos el formulario, indicando los datos incorrectos
   //   y rellenando los valores correctamente introducidos
}
```

Para que el usuario no pierda, después de enviar el formulario, los datos correctamente introducidos, utiliza el atributo `value` en las entradas de texto:

```
   Nombre del alumno:
   <input type="text" name="nombre" value="<?php echo $_POST['nombre'];?>" />
   Y el atributo checked en las casillas de verificación:
   <input type="checkbox" name="modulos[]" value="DWES"
      <?php
         if(in_array("DWES",$_POST['modulos']))
            echo 'checked="checked"';
      ?>
   />
```

Fíjate en el uso de la función `in_array` para buscar un elemento en un array.
Para indicar al usuario los datos que no ha rellenado (o que ha rellenado de forma incorrecta), deberás comprobar si es la primera vez que se visualiza el formulario, o si ya se ha enviado. Se puede hacer por ejemplo de la siguiente forma:

```
Nombre del alumno:
<input type="text" name="nombre" value="<?php echo $_POST['nombre'];?>" />
<?php
   if (isset($_POST['enviar']) && empty($_POST['nombre']))
      echo "<span style='{color:red}'> <-- Debe introducir un nombre!!</span>"
?><br />
```

Completa el ejemplo anterior para que se validen todos los datos

Una forma de enviar información de una página PHP a otra, es incluyéndola en campos ocultos dentro de un formulario.

Modifica el ejercicio que mostraba la fecha en castellano, para que obtenga lo mismo a partir de un día, mes y año introducido por el usuario. Antes de mostrar la fecha, se debe comprobar que es correcta. Utilizar la misma página PHP para el formulario de introducción de datos y para mostrar la fecha obtenida en castellano.
Consultar las funciones `checkdate` y `mktime` en el manual de PHP.