U.T. 2: Características del lenguaje PHP
Parte 1: Elementos del lenguaje PHP y Estructuras de control

Contenidos:

- Elementos del lenguaje PHP
  - Generación de código HTML
  - Cadenas de texto
  - Funciones relacionadas con los tipos de datos
  - Variables especiales de PHP
- Estructuras de control
  - Condicionales
  - Bucles

1.- Elementos del lenguaje PHP

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 de 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.

1.1.- Generación de código HTML

Existen varias formas de 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.

Consulta el Anexo I al final de esta presentación: Utilización de la función print en PHP

1.1.- Generación de código 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>";
?>
```

Cada especificador de conversión va precedido del carácter % y se compone de las siguientes partes:

- **signo** (opcional). Indica si se pone signo a los número negativos (por defecto) o también a los positivos (se indica con un signo +).
- **relleno** (opcional). Indica que carácter se usará para ajustar el tamaño de una cadena. Las opciones son el carácter 0 o el carácter espacio (por defecto se usa el espacio).
- **alineación** (opcional). Indica que tipo de alineación se usará para generar la salida: justificación derecha (por defecto) o izquierda (se indica con el carácter -).
- **ancho** (opcional). Indica el mínimo número de caracteres de salida para un parámetro dado.
- **precisión** (opcional). Indica el número de dígitos decimales que se mostrarán para un número real. Se escribe como un dígito precedido por un punto.
- **tipo** (obligatorio). Indica cómo se debe tratar el valor del parámetro correspondiente. En la siguiente tabla puedes ver una lista con todos los especificadores de tipo.

1.1.- Generación de código HTML

Especificadores de tipo para las funciones printf y sprintf.

Especificador	Significado
b	el argumento es tratado como un entero y presentado como un número binario.
c	el argumento es tratado como un entero, y presentado como el carácter con dicho valor ASCII.
d	el argumento es tratado como un entero y presentado como un número decimal.
u	el argumento es tratado como un entero y presentado como un número decimal sin signo.
o	el argumento es tratado como un entero y presentado como un número octal.
x	el argumento es tratado como un entero y presentado como un número hexadecimal (con minúsculas).
X	el argumento es tratado como un entero y presentado como un número hexadecimal (con mayúsculas).
f	el argumento es tratado como un doble y presentado como un número de coma flotante (teniendo en cuenta la localidad).
F	el argumento es tratado como un doble y presentado como un número de coma flotante (sin tener en cuenta la localidad).
e	el argumento es presentado en notación científica, utilizando la e minúscula (por ejemplo, 1.2e+3).
E	el argumento es presentado en notación científica, utilizando la e mayúscula (por ejemplo, 1.2E+3).
g	se usa la forma más corta entre %f y %e.
G	se usa la forma más corta entre %f y %E.
s	el argumento es tratado como una cadena y es presentado como tal.
%	se muestra el carácter %. No necesita argumento..

Por ejemplo, al ejecutar la línea siguiente, en la salida el número PI se obtiene con signo y sólo con dos decimales.

printf("El número PI vale %+.2f", 3.1416); // +3.14

Existe una función similar a printf pero en vez de generar una salida con la cadena obtenida, permite guardarla en una variable: sprintf.

$txt_pi = sprintf("El número PI vale %+.2f", 3.1416);

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.

1.2.- Cadenas de texto

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

print " Módulo: ${modulo} "

También podemos utilizar las llaves para insertar elementos complejos como elementos de un array o propiedades de un objeto.

Print “ Módulo: {$modulos[0]}”

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.

1.2.- Cadenas de texto

Secuencias de escape.

Secuencia	Resultado
\\	se muestra una barra invertida.
\'	se muestra una comilla simple.
\"	se muestra una comilla doble.
\r	se muestra un retorno de carro (CR o 0x0D (13) en ASCII).
\v	se muestra un tabulador vertical (VT o 0x0B (11) en ASCII).
\f	se muestra un avance de página (FF o 0x0C (12) en ASCII).
\$	se muestra un signo del dólar.

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"
?>
```

1.2.- Cadenas en PHP: Búsqueda

- strstr(cadena, cadBusq)
  - Devuelve desde la primera aparición de la cadena (**incluyéndola**) hasta el final. Case-sensitive

```
strstr(“yo@midominio.es","@");
// @midominio.es
```

- strrchr(cadena, carBusq)
  - Devuelve desde la **última** aparición del 1o carácter (**incluído**) hasta el final. Case-sensitive

```
strrchr ("Esto es muy bonito","so"); // s muy bonito
```

- stristr(cadena, carBusq)
  - Igual que strstr pero no es case-sensitive.

1.2.- Cadenas en PHP: Búsqueda

- **strpos (cad1, cad2 [, desplz])**
  - **Busca la primera posición de aparición de una cadena a partir de desplz (por defecto 0).**
  - **Si no la encuentra devuelve FALSE**

    strpos ("Este espacio es muy bonito","es",7); // 13
- **strrpos (cadena, cadena [, desplz])**
  - **Devuelve la posición de la última aparición o FALSE**

    strrpos ("Este espacio es muy bonito","es"); // 13

1.2.- Cadenas en PHP: Búsqueda

- strspn (cadena, máscara,comienzo,longitud)
  - Devuelve la longitud del segmento inicial de un string que consiste únicamente en caracteres contenidos dentro de una máscara dada .

    Case-sensitive

    strspn ("Este espacio es muy bonito","Estela"); // 4

    strspn ("Este espacio es muy bonito","Estado"); // 3

    strspn ("Este espacio es muy bonito","el");// 0
- strcspn (cadena, máscara, comienzo, longitud)
  - Devuelve la longitud de la subcadena más larga que está formada sólo por caracteres no contenidos en la máscara. Case-sensitive
- preg_match (patron, cadena)
  - Devuelve 1 si la cadena cumple el patrón (expresión regular) y 0 si no.
  - Hay que prestar atención porque los valores devueltos NO SON BOOLEANOS sino que son numéricos.

1.2.- Cadenas en PHP: Comparación

- strcmp (cad1, cad2)
  - Compara dos cadenas (case-sensitive) y devuelve:
    - > 0, si cad1>cad2
    - < 0, si cad1<cad2
    - 0, si ambas cadenas son iguales.

strcmp("Hola","hola") → no iguales
strcasecmp("Hola","hola") → iguales

- strcasecmp(cad1, cad2)
  - Igual que strcmp() pero insensible a mayúsculas y minúsculas.
- strncmp (cad1, cad2,longitud)
  - Sólo compara los longitud primeros caracteres
- strnatcmp (cad1, cad2)
  - Comparación "natural“
    - con strcmp(10,2) 10 es menor que 2
    - con strnatcmp(10,2) 10 es mayor que 2

strncmp("Paco","Paca",4) → distintos
strncmp("Paco","Paca",2) → iguales

1.2.- Cadenas en PHP: Operación

- int strlen (string $cadena)
  - Devuelve el número de caracteres que contiene una cadena.

```
$cadena="Hola que tal";
echo strlen($cadena);        // 12
```

- substr ($cadena, $inicio, $lon)
  - Devuelve una subcadena de longitud $lon a partir de la posición $inicio de la cadena $cadena.

```
$cad="PATITOS";
echo substr($cad,2);         // TITOS
echo substr($cad,2,3);       // TIT
echo substr($cad,-2);        // OS
echo substr($cad,2,-3);      // TI 2o no negativo = caracteres del
                                  final que se quitan
```

1.2.- Cadenas en PHP: Operación

- **substr_replace($cadena, $reemplazo, $ini, $charsABorrar)**
  - **Devuelve una cadena, resultado de reemplazar con el string dado en reemplazo, una copia de cadena empezando por el parámetro inicio hasta (opcionalmente) charsABorrar o hasta el final de cadena.**
  - **La cadena original no sufre ninguna modificación. Case sensitive**

```
$texto="Hola a todos los chicos";
$texto2=substr_replace($texto,"todas", 0)   // todas
$texto2=substr_replace($texto,"todas", 7)   // Hola a todas
$texto2=substr_replace($texto,"todas", 7,5)
    // Hola a todas los chicos
```

1.2.- Cadenas en PHP: Operación

- **str_replace ($cadBusq,$cadReempl,$texto)**
  - **Devuelve una cadena resultado de sustituir $cadBusq en $texto por $cadReempl.**
  - **La cadena original no sufre ninguna modificación.**
- **strtr (cadena, cadBus, cadRee)**
  - **Sustituye carácter a carácter**
  - **La cadena original no sufre ninguna modificación.**

```
echo strtr("Hola a todos los presentes aquí","aeiou", "AEIOU");

HOlA A tOdOs lOs prEsEntEs AqUI
```

  - **En lugar de dos argumentos se puede pasar un array.**

1.2.- Cadena en PHP: Operación

- substr_count ($cadena, $patron, $inicio, $longitud)
  - Devuelve el número de apariciones de $patron dentro de la cadena $cadena, (opcionalmente) empezando desde la posición $inicio hasta $longitud.

```
$cad="Hola a todos los presentes";

echo substr_count($cad,"a");                    // 2
```

- $cad[$pos]
  - Obtenemos el carácter que ocupa la posición $pos de la cadena $cad. La primera posición es la 0.

```
echo $cad{3};        // a
```

1.2.- Cadenas en PHP: Modificar

- rtrim($cadena, $listaChar)
  - Elimina los espacios en blanco y caracteres de fin de línea {“ “, ”\n”, ”\t”, ”\r”, ”\0”, ”\x0B” } del final de la cadena. Permite incluir una lista $listaChar con los caracteres a eliminar del final. Devuelve la cadena modificada.
- chop($cadena)
  - Alias de rtrim().
- ltrim(), trim()
  - Eliminan caracteres a la izquierda o por ambos lados de una cadena.
- str_pad($cadena, $long, $cadenaRelleno)
  - Rellena una cadena con un carácter de relleno (por defecto, es el espacio en blanco) hasta obtener $long caracteres en total.
  - Si $long es negativo o menor que la longitud de la cadena no rellena.
  - Parámetros: STR_PAD_RIGHT, STR_PAD_LEFT, STR_PAD_BOTH
    - Indican la opción para rellenar.

1.2.- Cadenas en PHP: Modificar

- strtolower($cad) / strtoupper($cad)
  - Devuelve una string con todos los caracteres alfabéticos convertidos a minúsculas/ mayúsculas.
- ucfirst($texto)
  - Pone en mayúscula solo la 1a letra de $texto
- ucwords($texto)
  - Pone en mayúscula la 1a letra de cada palabra contenida en $texto.

1.2.- Cadenas en PHP: Modificar

- addslashes($cadena)
  - Antepone barras de escape \ a: ("), (') , el NULL (el byte nulo) y (\).
- stripslashes($cadena)
  - Quita barras de escape de: (\') .
- nl2br($cadena)
  - Todos los saltos de línea serán convertidos a etiquetas
.
    - echo nl2br($row['campo_BD']);
- addcslashes($cadena, $listcar)
  - Escapa cualquier carácter de $listcar.
- stripcslashes($cadena)
  - Contraria a addcslashes ().

```
addcslashes("Hola","a,b");       // Hol\a
```

1.2.- Cadenas en PHP: Modificar

- quotemeta($cadena)
  - Coloca una barra \ delante de : . , \, +, *, ? , [, ^,] , (, $,)
- string htmlspecialchars ($string [,$flags [,$codigo]])
  - Traducen a entidades html
  - Las traducciones realizadas son:
    - &' se convierte en '&'
    - ‘ “ ‘ (comillas dobles) se convierte en '"‘ cuando ENT_NOQUOTES no está establecido.
    - “ ‘ " (comilla simple) se convierte en ''' (o ') sólo cuando ENT_QUOTES está establecido.
    - '<' (menor que) se convierte en '<'
    - '>' (mayor que) se convierte en '>‘

ENT_QUOTES → Convertirá tanto las comillas simples como las dobles
ENT_NOQUOTES → Dejará tanto las comillas simples como las dobles sin convertir

1.2.- Cadenas en PHP: Modificar

- string htmlentities ($string [,$flags [,$codigo]])
  - Esta función es idéntica a htmlspecialchars() en todos los aspectos, excepto que con htmlentities(), todos los caracteres que tienen equivalente HTML son convertidos a esas entidades.
- strtok($cadena, $delimitador)
  - Divide una cadena en subcadenas separadas por $delimitador. Devuelve la primera subcadena obtenida. Invocar desde un bucle para dividir toda una cadena.

    strtok("hola que tal"," ") → divide en palabras y devuelve “hola”

    strtok(“ “) → Devuelve las siguientes (una en cada llamada)
- chunk_split($cadena [, $long [, $separador]])
  - Inserta la cadena separador cada $long caracteres

    chunk_split("hola que tal",3,’-’); → hol-a q-ue -tal

1.2.- Cadenas en PHP: Modificar

- array str_split (string $string [, int $split_length = 1])
  - Convierte un string en un array.
  - Si el parámetro opcional split_length se específica, el array devuelto será separado en fragmentos los cuales cada uno tendrá una longitud de split_length, de otra manera cada fragmento tendrá una longitud de un carácter.
- array explode (string $delimiter ,string $string [, int $limit])
  - Parte la cadena en función de $delimiter y crea un array con los elementos.

```
$dir_correo="prueba@yahoo.com.ar";

$parte_array=explode("@",$dir_correo);

echo $parte_array[0]; // prueba

echo $parte_array[1]; // yahoo.com.ar
```

1.2.- Cadenas en PHP: Modificar

- string implode (string $delimitador , array $matriz)
  - Convierte un array en una cadena de caracteres separando sus elementos con la cadena indicada en $delimitador.

```
$matriz=array(7,'julio',2011);
echo implode(' de ',$matriz);
$unidos=implode("@",$parte_array);
```

- filter_var ($variable, $filtro)
  - Valida el contenido de la variable según el filtro indicado en la constante $filtro:
  - Algunos tipos de filtro:
    - FILTER_VALIDATE_EMAIL: Valida emails según RFC 822
    - FILTER_VALIDATE_FLOAT: Valida números reales
    - FILTER_VALIDATE_IP: Valida IP's
    - FILTER_VALIDATE_URL : Valida URL's según RFC 2396

1.3.- Funciones relacionadas con los tipos de datos

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 **unknowntype**.

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.

1.3.- Funciones relacionadas con los tipos de datos

```
<?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

1.3.- Funciones relacionadas con los tipos de datos

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**.

1.3.- Funciones relacionadas con los tipos de datos

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**.

1.3.1.- Funciones relacionadas con los tipos de datos

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.

1.3.1.- Funciones relacionadas con los tipos de datos

Función date: caracteres de formato para fechas y horas.

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');

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.

En el curso de Moodle encontrarás un enlace con información sobre las distintas zonas horarias (PSM1) y otro con las funciones para gestionar fechas y horas (DC1)

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 es el 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 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	Hora en formato de 12 horas, siempre con dos dígitos
H	Hora en forma de 24 horas, siempre con dos dígitos
g	Hora en formato de 12 horas, con uno o dos dígitos (sin ceros iniciales)
G	Hora en formato 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 enterea con formato RFC 2822

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 el curso de Moodle encontrarás un enlace donde consultar toda la información que ofrece $_SERVER (DC2)

1.4.- Variables especiales de 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.

En posteriores unidades iremos trabajando con estas variables.

En el curso de Moodle encontrarás un enlace donde consultar información sobre estas variables superglobales (PSM2)

2.- Estructuras de control

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.

2.1.- Condicionales

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". (Solución en el Anexo II y en el curso de Moodle (ER1))

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
?>
```

2.2.- Bucles

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 es 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.

2.2.- Bucles

- Sentencia **break**:

  Break [n];

- Nos permite salir de una estructura de control o bucle de manera directa.
- Si la acompañamos de un número entero (opcional), indicamos el número de niveles de la estructura anidada que queremos saltar.
- **NO SE RECOMIENDA SU USO (no es programación estructurada)**

2.2.- Bucles

- Sentencia **continue**:

  continue [n];

- Nos permite abandonar la iteración vigente de una estructura de control.
- Si la acompañamos de un número entero (opcional), indicamos el número de niveles de la estructura anidada que queremos saltar.
- **NO SE RECOMIENDA SU USO (no es programación estructurada)**

Anexo I.- 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>
```

Anexo II.- Solución propuesta I

```
<!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: Mostrar fecha en castellano -->
<html>
    <head>
        <meta http-equiv="content-type" content="text/html; charset=UTF-8">
        <title>Fecha en castellano</title>
    </head>
    <body>
<?php

date_default_timezone_set('Europe/Madrid');
    $numero_mes = date("m");
    $numero_dia_semana = date("N");
switch($numero_mes) {
    case 1: $mes = "Enero";
    break;
    case 2: $mes = "Febrero";
    break;
    case 3: $mes = "Marzo";
    break;
    case 4: $mes = "Abril";
    break;
    case 5: $mes = "Mayo";
    break;
    case 6: $mes = "Junio";
    break;
    case 7: $mes = "Julio";
    break;
    case 8: $mes = "Agosto";
    break;
    case 9: $mes = "Septiembre";
    break;
    case 10: $mes = "Octubre";
    break;
    case 11: $mes = "Noviembre";
    break;
    case 12: $mes = "Diciembre";
    break;
}
```

Anexo II.- Solución propuesta I

```
    switch($numero_dia_semana) {
        case 1: $dia_semana = "Lunes";
        break;
        case 2: $dia_semana = "Martes";
        break;
        case 3: $dia_semana = "Miércoles";
        break;
        case 4: $dia_semana = "Jueves";
        break;
        case 5: $dia_semana = "Viernes";
        break;
        case 6: $dia_semana = "Sábado";
        break;
        case 7: $dia_semana = "Domingo";
        break;
    }
    print $dia_semana.", ".date("j")." de ".$mes." de ".date("Y");
?>
    </body>
</html>
```