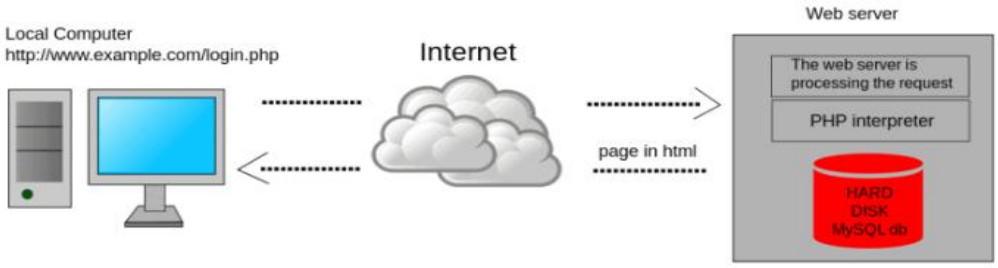
Tema 1
01-Sintaxis PHP

- PHP: acrónimo de Hypertext Preprocessor.
- Lenguaje web de guiones que se ejecuta en el lado del servidor/backend
- Código abierto y gratuito
- Multiplataforma (Compatible con distintos sistemas operativos y servidores)
- Bajo costo: software libre, código abierto
- Es simple de aprender y se ejecuta de forma eficiente en el servidor
- Código embebido dentro del lenguaje de marcas
- Sintaxis basada en la de C/C++, y por lo tanto, similar a Java

Petición de página PHP

- El cliente (navegador) solicita una página php
- El servidor recibe la petición, la procesa y produce salida: HTML
- La salida se envía al cliente, quien no tiene acceso al script PHP



Delimitadores **<?php** y **?>**.

- Un bloque PHP puede ser ubicado en cualquier lugar del documento.

```
<html><body><b><?php
                    echo "Hola Visitante";
              ?></b>
</body></html>
```

- Las páginas que incluyen código PHP deben ser almacenadas con extensión **.php**, si no no se ejecutarán.
- El código se ejecuta por un entorno de ejecución con el que se integra el servidor web (normalmente utilizando Apache con el módulo php_module).

Comentarios

- De línea: **// o #**
- De bloque: **/* */**

No figurarán en la página web enviada al navegador (al contrario de comentarios HTML).

Variables y tipos de datos en PHP

- Las variables en PHP deben comenzar por el signo **$,** seguido de una letra o del el carácter _
- PHP es "case sensitive"
- No es necesario declarar variables. Basta con empezar a usarlas
- El tipo de una variables es el del valor que se le asigne y puede cambiar en tiempo de ejecución

```
$mi_variable = 7;            // variable es de tipo "entero"
$mi_variable = "siete";      // pasa a ser de tipo "cadena"
```

- Los tipos de variables en PHP son:
  - **Bool:** valores true y false. Además, 0 es false y cualquier nº positivo true
  - **Integer:** nºs sin decimales. Pueden ser decimal, octal (0___) o hexadecimal (0x___).
  - **Float:** nºs con decimales.
  - **String**: Conjuntos de caracteres entre comillas simples o dobles.
  - Objects
  - Resources Referencias a funciones o recursos fuera de PHP
  - NULL Variables sin valor

Ejemplos:

```
$mi_booleano = false;
$mi_entero = 0x2A;   (Es un número hexadecimal. 035 sería un número octal, 0b11110011 un binario)
$mi_real = 7.3e-1;
$mi_cadena = "texto";
$mi_variable = NULL;
```

Constantes

- Definidas por el usuario: **define** o **const**
  - **define** (nombre, valor, [es_case_sensitive]); **define** ("PI",3.1415);
  - Los identificadores de constantes no van precedidos por "$" y suelen ir en mayúsculas.
- Constantes PHP predeterminadas: __LINE__, __FILE__, __DIR__, __FUNCTION__, __CLASS__, etc

Expresiones y operadores

- Aritméticos: negación (**-**), suma (**+**), resta (**-**) , multiplicación (*****) , división (**/**) y módulo (**%**)pre y post incremento (**++**) y pre y post decremento (**--**)
- Concatenador: .
- Asignaciones: = += -= *= /= %= .=
- Comparadores: == === != <> !== > >= < <=

  * Los operadores <> y != son equivalentes.
  * El operador === devuelve True si los operandos tienen el mismo tipo y valor
  * El operador !== devuelve T si los operandos tienen distinto tipo o distinto valor

  Por ejemplo: para $x = 0, la expresión **$x == false** es verdad, **$x === false** no es cierta

- Boolean: **!** **and** **&&** **or** **||** **xor**
- De bits

Generación de código HTML

- Sentencias **echo** y **print**

```
<?php
   echo “<td>Texto en celda</td>”;
   echo “Varible x con valor $x”;
   echo “$x al cubo=”. fcion_potencia($x,3) .”</br>”;
?>
```

Estructuras de control: CONDICIONAL

Condicionales

- if
- if /else
- switch/case/default
- if / elseif / elseif / …. / else
- Operador ternario:
  (condición) **?**”valor si verdadero”**:**”valor si falso”

Bucles

- while
- do / while
- for
- foreach
- **break** Salir del bucle
- **continue** Omitir ejecución de las sentencias restantes de la repetición actual del bucle.

Cadenas de texto

- Van entre comillas dobles " " o simples ' ':Dentro de comillas dobles se sustituyen valores de variables y se interpretan caracteres de escape: echo "x vale \"$x\" y z vale \"$z\"";

- Sintaxis **heredoc**: línea con **<<<**un identificador, la cadena en una nueva línea y se cierra en otra línea repitiendo el identificador. Se comporta como un texto entre " " sin necesidad de escaparlas

```
?php
$str = <<<EOD
Ejemplo de una cadena
expandida en varias líneas
empleando la sintaxis heredoc.
EOD;
```

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

```
<?php
        $var = "way";
        echo "Two $vars to define a variable";
?>
```

→SALIDA: Undefined variable: vars

```
<?php
        $var = "way";
        echo "Two {$var}s to define a variable ";
?>
```

➔ SALIDA: Two ways to define a variable

- **Secuencias de escape**: utilizadas para escapar(ignorar) ciertos caracteres especiales de una cadena, antes de que esta se parsee: \n \t \$ \" \' \\

- PHP permite acceder a un carácter de un string como si fuera un array de caracteres (índice entre corchetes)

$cad**="Buenos días" → **$cad[1] contiene "**u**"

Funciones de cadena

- **addcslashes()—** Devuelve una cadena con barras invertidas delante de los caracteres especificados.
- **addslashes()—** Devuelve una cadena con barras invertidas delante de los caracteres que deben escaparse
- **chop()—** Elimina espacios u otros caracteres del extremo derecho de una cadena.
- **chr()—** Devuelve un carácter de un valor ASCII especificado
- **chunk_split()—** Divide una cadena en una serie de trozos más pequeños
- **convert_uudecode()—** Decodifica una cadena codificada sin formato
- **convert_uuencode()—** Codifica una cadena usando uuencode
- **count_chars()—** Devuelve información sobre los caracteres de una cadena.
- **crypt()—** Devuelve una cadena hash
- **echo() or echo ''**— Genera una o varias cadenas
- **explode()—** Divide una cadena en una matriz
- **get_html_translation_table()—** Devuelve la tabla de traducción utilizada por htmlspecialchars()yhtmlentities()
- **hex2bin()—** Traducir valores hexadecimales a caracteres ASCII
- **html_entity_decode()—** Convierte entidades HTML en caracteres
- **htmlentities()—** Convierte caracteres en entidades HTML
- **htmlspecialchars_decode()—** Transforma entidades HTML especiales en caracteres.
- **htmlspecialchars()—** Cambia caracteres predefinidos a entidades HTML
- **implode()—** Recupera una cadena de los elementos de una matriz, igual quejoin()
- **lcfirst()—** Cambia el primer carácter de una cadena a minúsculas.

- **localeconv()—** Devuelve información sobre el formato numérico y monetario de la configuración regional.
- **ltrim()—** Elimina espacios u otros caracteres del lado izquierdo de una cadena.
- **md5()—** Calcula el hash MD5 de una cadena y lo devuelve
- **md5_file()—** Calcula el hash MD5 de un archivo
- **money_format()—** Devuelve una cadena como cadena de moneda
- **nl_langinfo()—** Proporciona información local específica
- **nl2br()—** Inserta saltos de línea HTML para cada nueva línea en una cadena
- **number_format()—** Formatea un número que incluye miles agrupados
- **ord()—** Devuelve el valor ASCII del primer carácter de una cadena
- **parse_str()—** Analiza una cadena en variables
- **rtrim()—** Elimina espacios en blanco u otros caracteres del lado derecho de una cadena
- **setlocale()—** Establece información local
- **sha1()—** Calcula el hash SHA-1 de una cadena
- **sha1_file()—** Hace lo mismo con un archivo.
- **similar_text()—** Determina la similitud entre dos cadenas.
- **str_ireplace()—** Reemplaza caracteres especificados en una cadena con reemplazos especificados (no distingue entre mayúsculas y minúsculas)
- **str_repeat()—** Repite una cadena un número predeterminado de veces
- **str_replace()—** Reemplaza caracteres especificados en una cadena (distingue entre mayúsculas y minúsculas)
- **str_shuffle()—** Mezcla aleatoriamente los caracteres de una cadena
- **str_split()—** Divide cadenas en matrices
- **str_word_count()—** Devuelve el número de palabras en una cadena.
- **strcasecmp()—** Comparación que no distingue entre mayúsculas y minúsculas de dos cadenas
- **strcmp()—** Comparación de cadenas binarias seguras (distingue entre mayúsculas y minúsculas)
- **strcoll()—** Compara dos cadenas según la configuración regional
- **strcspn()—** Devuelve el número de caracteres encontrados en una cadena antes de que aparezcan los caracteres especificados.
- **strip_tags()—** Elimina etiquetas HTML y PHP de una cadena
- **stripcslashes()-** Opuesto de addcslashes()
- **stripslashes()-** Opuesto de addslashes()
- **stripos(**)— Encuentra la posición de la primera aparición de una subcadena dentro de una cadena (no distingue entre mayúsculas y minúsculas)
- **strlen()—** Devuelve la longitud de una cadena
- **strpbrk()—** Busca en una cadena cualquier número de caracteres
- **strpos()—** Devuelve la posición de la primera aparición de una subcadena en una cadena (distingue entre mayúsculas y minúsculas)
- **strrchr()—** Encuentra la última aparición de una cadena dentro de otra cadena
- **strrev()—** Invierte una cadena
- **strripos()—** Encuentra la posición de la última aparición de la subcadena de una cadena (no distingue entre mayúsculas y minúsculas)
- **strstr()—**Búsqueda **de** la primera aparición de una cadena dentro de otra cadena
- **strtok()—** Divide una cuerda en trozos más pequeños
- **strtolower()—** Convierte todos los caracteres de una cadena a minúsculas
- **strtoupper()—** Lo mismo pero para letras mayúsculas.
- **substr()**— Devuelve una parte especificada de una cadena
- **substr_compare()—** Compara dos cadenas desde una posición inicial especificada hasta una longitud determinada, opcionalmente distingue entre mayúsculas y minúsculas
- **substr_count()—** Cuenta el número de veces que aparece una subcadena dentro de una cadena.
- **substr_replace()—** Reemplaza una subcadena con otra cosa
- **trim()—** Elimina espacios u otros caracteres de ambos lados de una cadena.
- **ucfirst()—** Transforma el primer carácter de una cadena a mayúsculas.
- **ucwords()—** Convierte el primer carácter de cada palabra de una cadena a mayúsculas.
- **vfprintf()—** Escribe una cadena formateada en un flujo de salida específico
- **wordwrap()—** Acorta una cadena a un número determinado de caracteres

Funciones personalizadas

- **Permiten asociar una etiqueta** (el nombre de la función) **con un bloque de código** a ejecutar.
- Usar funciones ayuda a estructurar mejor el código.
- Las variables locales declaradas en las funciones no serán visibles fuera del cuerpo de las mismas.
- Para llamar a una función, basta con poner su nombre y unos paréntesis.
- Para crear tus propias funciones,
    - Palabra **function**.
    - Los parámetros de entrada se tratarán como variables locales
    - Podemos devolver un resultado con la instrucción *return valor;*
    - No es obligatorio indicar los tipos de los parámetros
    - Podemos dar valores por defecto a los parámetros de entrada

Ejemplo

```
<?php

  function media(float $a, float $b):float
  {
    $media=($a+$b)/2;
    return $media;
  }
   …..
   echo media(4,6)."</br>";
  ?>
```

Ejemplo

```
<?php

        function verPelicula($titulo = "Avatar"){
                return "Vas a ver la película $titulo</br>";
        }
        ……
        echo verPelicula();
        echo verPelicula("Braveheart");
?>
```

- Desde PHP 7.1, una misma función puede devolver un tipo o null
    - function raiz(float $numero): ?float
- Desde PHP 8, una misma función puede devolver 2 tipos diferentes:
    - function raiz(float $numero): float|string

Funciones relacionadas con los tipos de datos

- Funciones para chequear tipo de una variable: **gettype**, **is_array()**, **is_bool()**, **is_float()**, **is_integer()**, **is_null()**, **is_numeric()**, **is_object()**, **is_resource()**, **is_scalar(), is_string()**

- Funciones para establecer tipo de una variable: **settype**

- Destruir variable: **unset**

- Comprobar si una variable está definida (no es null): **isset**
  Es importante no confundir el que una variable no esté definida o valga null, con que se considere como vacía debido al valor que contenga. Esto lo comprueba la función **empty.**

Arrays

- Existen 2 tipos de arrays: numéricos y asociativos
  - Arrays numéricos: la clave o índice corresponde a un número, desde 0.
  - Arrays asociativos: donde la clave o índice corresponde a un string

- Ejemplo array numérico

```
$modulos1 = array(0 => "Programación", 1 => "Bases de datos", ...,,9 => "Desarrollo web servidor");
$modulos1 = array("Programación","Bases de datos", ...,….."Desarrollo web servidor");
$modulos1 = ["Programación","Bases de datos", ...,….."Desarrollo web servidor"];
```

Sus elementos se referencian: $modulos[0], $modulos[1], $modulos[2]…..

- Ejemplo de array asociativo

```
$modulos2 = array("PR" => "Programación", "BD" => "Bases de datos", ....., "DWES" => "Desarrollo servidor");
```

Sus elementos se referencian: $modulos2 [“PR”], $modulos2 ["BD"],

- Array bidimensional

```
$ciclos = array(
   "DAW" => array ("PR" => "Programación", "BD" => "Bases de datos", ..., "DWES" => "Desarrollo servidor"),
   "DAM" => array ("PR" => "Programación", "BD" => "Bases de datos", ..., "PMDM" => "Prog multimedia")
);
```

Sus elementos se referencian: $ciclos ["DAW"] ["DWES"]

- En PHP no es necesario indicar el tamaño del array al crearlo. Ni siquiera es necesario declararlo. Basta con comenzar a asignarle valores:

```
// array numérico
$modulos1 [0] = "Programación";
$modulos1 [1] = "Bases de datos";
...
$modulos1 [9] = "Desarrollo servidor";
```

```
// array asociativo
$modulos2 ["PR"] = "Programación";
$modulos2 ["BD"] = "Bases de datos";
...
$modulos2 ["DWES"] = "Desarrollo servidor";
```

- En arrays numéricos tampoco es necesario especificar el valor de la clave para agregar elementos: 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 servidor";
```

- Recorrido con foreach. Usa una variable temporal a la que asigna en cada iteración el valor de cada elemento del array. Puedes usarlo de dos formas.

Recorrido de valores:

```
$modulos = array("PR" => "Programación",
"BD" => "Bases de datos", ...,
"DWES" => "Desarrollo servidor");

foreach ($modulos as $modulo) {
        echo "Módulo: ".$modulo. "<br />"
}
```

Recorrido de claves y valores:

```
$modulos = arrays("PR" => "Programación",
"BD" => "Bases de datos", ...,
"DWES" => "Desarrollo servidor");

foreach ($modulos as $codigo => $modulo) {
        echo "El código del módulo ".$modulo." es
".$codigo."<br />"
}
```

- Creación de array vacío: $array=array(); o $array=[];

Funciones de arrays

- **array_change_key_case**— Cambia todas las claves de una matriz a mayúsculas o minúsculas
- **array_chunk**— Divide una matriz en trozos
- **array_column**— Recupera los valores de una sola columna en una matriz
- **array_combine**— Fusiona las claves de una matriz y los valores de otra en una nueva matriz
- **array_count_values**— Cuenta todos los valores en una matriz
- **array_diff**— Compara matrices, devuelve la diferencia (solo valores)
- **array_fill**— Llena una matriz con valores
- **array_filter**— Filtra los elementos de una matriz mediante una función
- **array_flip**— Intercambia todas las claves en una matriz con sus valores asociados
- **array_intersect**— Compara matrices y devuelve sus coincidencias (solo valores)
- **array_intersect_assoc**— Compara matrices y devuelve sus coincidencias (claves y valores)
- **array_intersect_key**— Compara matrices y devuelve sus coincidencias (solo claves)
- **array_key_exists**— Comprueba si existe una clave especificada en una matriz, alternativa:key_exists
- **array_keys**— Devuelve todas las claves o un subconjunto de claves en una matriz
- **array_map**— Aplica una llamada a los elementos de una matriz determinada.
- **array_merge**— Fusionar una o varias matrices
- **array_multisort**- Ordena varios arrays o arrays multidimensionales.
- **array_pad**— Inserta un número específico de elementos (con un valor específico) en una matriz
- **array_pop**— Elimina un elemento del final de una matriz.
- **array_product**— Calcular el producto de todos los valores en una matriz.
- **array_push**— Empujar uno o varios elementos hasta el final de la matriz
- **array_rand**— Elija una o más entradas aleatorias de una matriz
- **array_replace**— Reemplaza elementos en la primera matriz con valores de las siguientes matrices
- **array_reverse**— Devuelve una matriz en orden inverso
- **array_search**— Busca en la matriz un valor determinado y devuelve la primera clave si tiene éxito
- **array_shift**— Cambia un elemento desde el principio de una matriz
- **array_slice**— Extrae una porción de una matriz
- **array_splice**— Elimina una parte de la matriz y la reemplaza
- **array_sum**— Calcular la suma de los valores en una matriz.
- **array_unique**— Elimina valores duplicados de una matriz.
- **array_unshift**— Agrega uno o más elementos al comienzo de una matriz
- **array_values**— Devuelve todos los valores de una matriz
- **array_walk**— Aplica una función de usuario a cada elemento de una matriz.
- **arsort**— Ordena una matriz asociativa en orden descendente según el valor
- **asort**— Ordena una matriz asociativa en orden ascendente según el valor
- **compact**— Crear una matriz que contenga variables y sus valores.
- **count**— Cuente todos los elementos en una matriz, alternativamente usesizeof
- **each**— Devuelve el par clave y valor actual de una matriz
- **in_array**— Comprueba si existe un valor en una matriz
- **key**— Obtiene una clave de una matriz
- **krsort**— Ordena una matriz asociativa por clave en orden inverso
- **ksort**— Ordena una matriz asociativa por clave
- **list**— Asigna variables como si fueran una matriz.
- **natsort**— Ordena una matriz usando un algoritmo de "orden natural"
- **range**— Crea una matriz a partir de una variedad de elementos.
- **rsort**— Ordenar una matriz en orden inverso
- **shuffle**— Mezclar una matriz
- **sort**— Ordena una matriz indexada en orden ascendente

Funciones relacionadas con las fechas

- La fecha y hora se almacenan como un entero (marca de tiempo o timestamp). Sin embargo, hay funciones de PHP para trabajar con ese tipo de datos.

- Función **time** : devuelve la marca de tiempo actual, es decir, la fecha y hora actual expresadas en segundos desde el 1 de enero de 1970.

- Función **strtotime** : convierte un texto en timestamp

- Función **date**: permite obtener una string de hora con el formato elegido a partir de un timestamp.

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

  - El segundo parámetro es opcional. Si no se indica, se utilizará la fecha actual.
  - El primer parámetro (formato) lo debes componer usando caracteres de esta tabla:

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 (Monday, ..., Sunday).
D	texto del día de la semana, tres letras (Mon, ..., Sun).
m	número del mes con dos dígitos.
F	texto del día del mes (January, ..., December).
M	texto del día del mes, tres letras (Jan, ..., Dec).
Y	número del año.
y	dos últimos dígitos del número del año.
h	hora, formato de 12 horas, siempre con dos dígitos.
H	hora, formato de 24 horas, siempre con dos dígitos.
i	minutos, siempre con dos dígitos.
s	segundos, siempre con dos dígitos.
a	am o pm, en minúsculas.
A	AM o PM, en mayúsculas.

Ejemplos:

```
date_default_timezone_set('Europe/Madrid');
echo date("l");                    //Visualiza "Wednesday", o el nombre del día de la semana actual
echo  "Son las" .date("h:i:sa");   //Visualiza "Son las 03:17:08pm"
```

Ejemplo: Visualizar Años completos transcurridos entre el 12/7/2010 y hoy

```
$str_fecha1="2010/07/12";
$fecha1=strtotime($str_fecha1);
$fecha2=time();
//Ya tenemos 2 fechas expresadas como timestamp (en segundos)
        echo (int)(($fecha2-$fecha1)/(60*60*24*365));
```

Otras funciones de fecha

- **checkdate()—** Comprueba la validez de una fecha gregoriana.
- **date_add()—** Agrega una cantidad de días, meses, años, horas, minutos y segundos a un objeto de fecha.
- **date_create_from_format()—** Devuelve un objeto DateTime formateado
- **date_create()—** Crea un nuevo objeto DateTime
- **date_date_set()—** Establece una nueva fecha
- **date_default_timezone_set()—** Establece la zona horaria predeterminada
- **date_diff()—** Calcula la diferencia entre dos fechas.
- **date_format()—** Devuelve una fecha formateada según un formato específico
- **date_modify()—** Modifica la marca de tiempo
- **date_offset_get()—** Devuelve el desplazamiento de la zona horaria.
- **date_parse()—** Devuelve una matriz con información detallada sobre una fecha específica
- **date_sub()—** Resta días, meses, años, horas, minutos y segundos de una fecha
- **date_time_set()—** Establece la hora
- **date_timestamp_get()—** Devuelve la marca de tiempo de Unix.
- **date_timezone_get()—** Devuelve la zona horaria de un objeto DateTime determinado
- **date_timezone_set()—** Establece la zona horaria para un objeto DateTime
- **date()—** Formatea una fecha y hora locales
- **getdate()—** Información de fecha/hora de una marca de tiempo o la fecha/hora local actual
- **gettimeofday()—** La hora actual
- **gmdate()—** Formatea una fecha y hora GMT/UTC
- **gmmktime()—** La marca de tiempo Unix para una fecha GMT
- **gmstrftime()—** Formatea una fecha y hora GMT/UTC según la configuración local
- **localtime()—** La hora local
- **mktime()—** La marca de tiempo de Unix para una fecha.
- **strtotime()—** Transforma un DateTime textual en inglés en una marca de tiempo Unix
- **time()—** La hora actual como marca de tiempo Unix.

Inclusión de ficheros externos.

Para incorporar a tu programa contenido de 1 archivo externo, hay varias opciones:

- 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. Ejemplo

- include_once **:** Si incluyes más de una vez un mismo fichero, obtendrás errores (por ejemplo, al repetir una definición de una función), include_once funciona como include, pero solo incluye los ficheros aún no incluidos.

- Require y require_once **:** Igual que "include", pero dan un error fatal y detiene la ejecución en caso de errores.

Variables especiales de PHP

PHP incluye varias variables predefinidas que pueden usarse desde cualquier ámbito, por lo que reciben el nombre de **variables superglobales**. Cada una de estas variables es un array asociativo. Veremos estas:

- **$_SERVER** 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 del cliente web.
$_SERVER['REQUEST_METHOD']	Método utilizado para acceder a la página ('GET', 'HEAD', 'POST' o 'PUT')

- **$_GET**, **$_POST** y **$_COOKIE** contienen las variables pasadas al guión actual utilizando respectivamente los métodos GET (parámetros en la URL), HTTP POST y Cookies HTTP. **$_REQUEST** junta en uno solo el contenido de los 3 arrays anteriores, $_GET, $_POST y $_COOKIE
- **$_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.

Redirección en PHP

- La forma más sencilla de realizar un redirect en PHP es indicarle al navegador en las cabeceras de respuesta que debe llevar al usuario a otra página. La sintaxis es:

```
<?php
        header("Location: http://www.google.com/");        // Redirecionamos a Google
        exit();                                            //terminamos la ejecución del script
?>
```

Es importante tener en cuenta que la función header se debe utilizar antes de que se imprima cualquier otra información de la página. Es decir, no debe estar precedida de echos ni prints ni ninguna salida html,

```
<html>
<?php
        header("Location: http://www.google.com/");        → Mal
        exit();
?>
```