1. Lenguajes embebidos en HTML

- Estructura páginas estáticas
  - Sencilla
  - Con tres partes:
    - Cabecera (definición directivas)
    - Zona metadatos
    - Cuerpo del documento (con la información a mostrar).
- Una vez creada la página se inserta el código a compilar e interpretar por el servidor.
- El código de servidor se ejecutará al procesar la petición HTTP del cliente.

1.1 Ámbito de ejecución

- Los lenguajes de servidor sólo pueden tener acceso a los recursos alojados en el servidor o bien otros ofrecidos a través de la red.
- NO pueden acceder a recursos del propio cliente.
- Poseen archivos de configuración para definir el comportamiento de la aplicación (php.ini)

PHP Version 8.2.3

System	Linux 1567db255f68 5.15.0-69-generic #76~20.04.1-Ubuntu SMP Mon Mar 20 15:54:19 UTC 2023 x86_64
Build Date	Mar 1 2023 12:11:00
Build System	Linux 8aafad8917ad 5.10.0-13-cloud-amd64 #1 SMP Debian 5.10.106-1 (2022-03-17) x86_64 GNU/Linux
Configure Command	'./configure' '--build=x86_64-linux-gnu' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/etc/php/conf.d' '--enable-option-checking=fatal' '--with-mhash' '--with-pic' '--enable-ftp' '--enable-mbstring' '--enable-mysqlnd' '--with-password-argon2' '--with-sodium=shared' '--with-pdo-sqlite=/usr' '--with-sqlite3=/usr' '--with-curl' '--with-iconv' '--with-openssl' '--with-readline' '--with-zlib' '--disable-phpdbg' '--with-pear' '--with-libdir=lib/x86_64-linux-gnu' '--disable-cgi' '--with-apxs2' 'build_alias=x86_64-linux-gnu'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	/usr/local/etc/php/php.ini
Scan this dir for additional .ini files	/usr/local/etc/php/conf.d
Additional .ini files parsed	/usr/local/etc/php/conf.d/docker-php-ext-mysqli.ini, /usr/local/etc/php/conf.d/docker-php-ext-pdo_mysql.ini, /usr/local/etc/php/conf.d/docker-php-ext-soap.ini, /usr/local/etc/php/conf.d/docker-php-ext-sodium.ini, /usr/local/etc/php/conf.d/docker-php-ext-zip.ini, /usr/local/etc/php/conf.d/xdebug.ini
PHP API	20220829
PHP Extension	20220829
Zend Extension	420220829
Zend Extension Build	API420220829,NTS
PHP Extension Build	API20220829,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar, zip
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2, tlsv1.3
Registered Stream Filters	zlib.*, convert.iconv.*, string.rot13, string.toupper, string.tolower, convert.*, consumed, dechunk

2. Tecnologías asociadas: PHP

- **PHP**- Hypertext Preprocessor
- Lenguaje de código **abierto**
- Permite su uso junto con documentos HTML mediante la **inserción de fragmentos** de código PHP acotados por etiquetas especiales
- El código PHP es **interpretado** por el servidor web donde se aloja el documento y se genera el correspondiente código HTML para mostrar por el navegador
- Las páginas tienen **extensión .php**
- Tiene una interfaz de desarrollo basada en la **POO**
- Es compatible con diferentes tecnologías
- Admite los principales servidores web y soporte con **diferentes repositorios de BD**

3. Obtención del lenguaje de marcas para mostrar en el cliente

- La incrustación de código PHP en HTML se puede hacer de varias maneras pero la más utilizada es la siguiente ya que es compatible con todas las plataformas:

```
<?php
    ...
?>
```

- Ejemplo:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>
            Desarrollo web
        </title>
    </head>
    <body>
        <h1>Desarrollo en servidor</h1>
        <h2> estas líneas están escritas en HTML</h2>
        <p> esta es una línea incluida en el cuerpo de la página</p>

        <?php
            $expresion="1";
            if ($expresion =="1"){
                echo("1.- Empiezan líneas generadas por PHP <br>");
                echo("2.- El texto está por instrucción print de PHP");
            }
        ?>
    </body>
<html>
```

- Ejercicio : Realiza una página web que presente como título "Mi primer ejemplo" y un mensaje de texto "primer ejemplo", mediante etiquetas HTML y un mensaje "¡Hola mundo!" con un script php

4. Etiquetas para inserción de código

4.1 Combinación de etiquetas html y código PHP

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Hola mundo</title>
    </head>
    <body>
```

```
        <h1>
        <?php
            echo "Hola mundo";
        ?>
        </h1>
    </body>
<html>
```

4.2 Insertar etiquetas dentro de un script PHP

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Hola mundo</title>
    </head>
    <body>
        <?php
            echo "<h1> Hola mundo </h1>";
        ?>
    </body>
<html>
```

4.3 Insertar todas las etiquetas dentro de un script PHP

```
        <?php
            print ("<!DOCTYPE html>");
            print ("<html>");
            print ("<head>");
            print ("<title> hola mundo</title>");
            print ("</head>");
            print ("<body>");
            print ("<h1>Hola mundo</h1>");
            print ("</body>");
            print ("</html>");
        ?>
```

5. Bloques de código

5.1 Sintaxis básica

- PHP es sensible a las mayúsculas
- Los espacios en blanco dentro del código embebido no tienen ningún efecto
- El final de instrucción se indica con ;
- Los scripts embebidos pueden situarse en cualquier parte código HTML.

- El número de scripts es indefinido.
- Cuando se ejecuta un código embebido, el script se sustituye por el resultado de dicha ejecución, incluidas las etiquetas de inicio y fin.

```
<!DOCTYPE html>
<html>
    <?php
        $salida="contenido php";
    ?>
    <head>
        <meta charset="UTF-8">
        <title> <?php  echo $salida;  ?>
        </title>
    </head>
    <body>
        <h1>Otro ejemplo</h1>
        <h2>
        <?php
            echo $salida;
        ?>
        </h2>
    </body>
<html>
```

5.2 Comentarios:

- Comentarios de **una línea** utilizando //
- Comentarios de **una línea** utilizando #
- Comentarios de **varias líneas**. /* ….. */

5.3 Imprimir : echo y print

- echo: muestra **una o más cadenas**

```
echo cadena1 [, cadena2...];

//Ejemplo
echo "Hola Mundo";
echo "Hola", "Mundo";
```

- print: muestra **una cadena**

```
print cadena;

//Ejemplo
print "Hola mundo";
print "Hola "."mundo";
```

6. Variables y tipos de datos

- El tipo de una variable no se suele especificar.

```
$mi_variable = 7;
```

- Se decide en tiempo de ejecución en función del contexto y puede variar.

```
$mi_variable=7;
$mi_variable="cambio de tipo";
```

- Reglas para nombrar una variable en PHP:
    - El nombre debe comenzar con una **letra** o con un **guión bajo** ("_")
    - El nombre únicamente puede contener **caracteres alfanuméricos** y **guiones bajos**.
    - El nombre de una variable **no** debe contener espacios en blanco. Si queremos formar el nombre de una variable con más de una palabra lo que se suele hacer es utilizar un **guión bajo entre ellas** ($mi_variable) o poner **la primera letra de cada palabra en mayúsculas** ($miVariable)
- Tipos de datos:
    - boolean
    - int
    - float
    - string
    - null
- **booleano** (boolean): sus posibles valores son true y false. Además, cualquier número entero se considera como true, salvo el 0 que es false.
- **entero** (int): cualquier número sin decimales. Se pueden representar en formato decimal, octal (comenzando por un 0), o hexadecimal (comenzando por 0x).
- **real** (float): cualquier número con decimales. Se pueden representar también en notación científica.
- **cadena** (string): conjuntos de caracteres delimitados por comillas simples o dobles.
- **null**: es un tipo de datos especial, que se usa para indicar que la variable no tiene valor.

Ejercicios:

1. Comprueba el resultado de las siguientes expresiones visualizando por pantalla su valor:

```
$mi_booleano = false;
$mi_entero= 0x2A;
$mi_real=7.3e-1;
$mi_cadena="texto";
$mi_variable=null;
```

2. ¿De qué tipo es la variable $resultado después de su ejecución?. Compruébalo

```
$mi_entero= 3;
$mi_real=2.3;
$resultado=$mi_entero+$mi_real;
```

6.1 Cadenas de texto

- Se definen tanto con comillas simples como dobles.
- Cuando se pone una variable dentro de unas **comillas dobles**, se procesa y se sustituye por su valor.

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

- PHP al encontrar la variable la sustituye por DWES.
- Para que PHP distinga correctamente el texto que forma la cadena del nombre, a veces es necesario encerrarla **entre llaves**

```
<?php
    echo "<p> Módulo: ${modulo} </p>";
?>
```

Concatenación de cadenas

- Concatenación punto (.)
- 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"
?>
```

> Hoja02_PHP_01

7. Ámbito de las variables

- Se pueden utilizar en cualquier lugar del programa.
- Si la variable no existe se reserva espacio en memoria.
- Si una variable se define dentro de una función es una **variable local** a la función.
- Si aparece fuera de la función se considera distinta a la definida en la función.
- Si dentro de una función se quiere usar una variable definida fuera hay que usar la palabra **global**.
- Si se quiere mantener el valor de una función declara dentro al salir de esta, se debe definir como **static**.

8. Operadores del lenguaje. Tipos

- Operadores **aritméticos**: +, -, *, /, %, ++, --
- Operador de **asignación**: =
- Operadores de **comparación**: ==, !=, <, >, <=, >=
- Operadores **lógicos**: and (&&), or (||), !, xor
- Operadores de **cadena**:
  - concatenación: . (punto)
  - asignación con concatenación: .=

8.1 Precedencia de operadores

- De menor a mayor:
  - ++,-- (operadores unarios)
  - *, /, %
  - +, -
  - <, <=, >, <=
  - ==, !=
  - &&
  - ||
  - and
  - or

9. Funciones relacionadas con los tipos de datos

Funciones para comprobar y establecer el tipo de datos de una variable:

- **gettype( )**: obtiene el tipo de variable

```
gettype($variable);
```

- **settype( )**: convierte una variable al tipo indicado por parámetro.

```
settype($variable,”float”);
```

Funciones para comprobar si es de un tipo concreto:

- **is_array( )**, **is_bool( )**, **is_float( )**, **is_integer( )**...
- **isset( )**: indica si una variable está definida y no es null.

```
isset($variable);
```

- **unset ( )**: destruye una variable.

```
unset($variable);
```

> Hoja02_PHP_02

Funciones para **Fechas**: no hay un tipo especifico. La información fecha y hora se almacena como un número entero y hay una serie de funciones en PHP para trabajar con ellas:

- **date()** es una de las más útiles. Permite obtener una cadena de texto a partir de una fecha y hora con el formato que se elija.

```
//date (string $formato [, int $fechahora]);
$fecha_actual = date('Y-m-d');
echo $fecha_actual; // Imprime: 2023-09-29
```

Si no se indica el segundo parámetro se utiliza la hora actual

Una de las clases más utilizadas para trabajar con fechas y horas es la clase **DateTime**.

```
// Crear un objeto DateTime a partir de una cadena de fecha
$fecha = new DateTime('2023-10-29');
// Añadir un día a la fecha
$fecha->modify('+1 day');
// Establecer la zona horaria a Madrid
$fecha->setTimezone(new DateTimeZone('Europe/Madrid'));
// Imprimir la fecha y hora formateada
echo $fecha->format('Y-m-d H:i:s');
```

- Puedes obtener la diferencia entre dos fechas utilizando la clase DateTime y el método diff(). Por ejemplo:

```
    $fecha1 = new DateTime('2023-06-15');
    $fecha2 = new DateTime('2023-09-22');
    $diferencia = $fecha1->diff($fecha2);
    echo $diferencia->format('%a días'); // Imprime: 99 días
```

La clase **IntlDateFormatter** proporciona una forma más avanzada y flexible de formatear fechas y horas en PHP. Esta clase se basa en la API de internacionalización (Intl) de PHP, lo que significa que puede manejar diferentes formatos de fecha y hora para diferentes idiomas y regiones.

```
$fecha = new DateTime('2023-09-05');
$formateador = new IntlDateFormatter('es_ES');
//https://unicode-
org.github.io/icu/userguide/format_parse/datetime/#dateformat
$formateador->setPattern("MMMM");
echo $formateador->format($fecha);
echo "<br>";

$formateador->setPattern("EEEE d 'de' MMMM");
echo $formateador->format($fecha);
echo "<br>";

$formateador = new IntlDateFormatter('es_ES', IntlDateFormatter::FULL,
IntlDateFormatter::NONE);
echo $formateador->format($fecha);
echo "<br>";

$formateador = new IntlDateFormatter('es_ES', IntlDateFormatter::LONG,
IntlDateFormatter::NONE);
echo $formateador->format($fecha);
echo "<br>";

$formateador = new IntlDateFormatter('es_ES', IntlDateFormatter::MEDIUM,
IntlDateFormatter::NONE);
echo $formateador->format($fecha);
echo "<br>";

$formateador = new IntlDateFormatter('es_ES', IntlDateFormatter::SHORT,
IntlDateFormatter::NONE);
echo $formateador->format($fecha);
echo "<br>";
```

> Hoja02_PHP_03

10. Variables especiales en PHP

PHP incluye variables internas predefinidas que pueden usarse desde cualquier ámbito. Se denominan **superglobales**.

Cada una de estas variables es un array que contiene un conjunto de valores. Son las siguientes:

- $_SERVER: contiene información sobre el entorno del servidor web y de ejecución.
- $_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.
- $_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.