PHP

PHP es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor.

Hay que entender primero como funciona la solicitud de páginas en un navegador para comenzar a programar en PHP.

Comunicación entre el cliente y el servidor sin PHP:

1 - Tipeamos en la barra del navegador la dirección y el archivo a solicitar.

2 - El web browser (navegador) envía el mensaje a través de Internet a la computadora, por ejemplo www.lanacion.com/pagina1.htm solicitando la página (archivo) pagina1.htm

3 - El web server (servidor web, que puede ser el Apache, IIS, etc.) que es un programa que se ejecuta en la máquina www.lanacion.com, recibe el mensaje y lee el archivo solicitado desde el disco duro.

4 - El servidor web envía el archivo solicitado por el navegador tal cual está en el disco duro.

5 - El navegador muestra en pantalla el archivo que envió el servidor web.

Este proceso siempre es el mismo cuando hablamos de páginas estáticas (páginas que no cambian), cualquiera sea el cliente que solicita la página el contenido siempre será el mismo.

La única forma que el contenido del archivo cambie es que el administrador de ese sitio web edite el contenido del archivo pagina1.htm y haga modificaciones.

Comunicación entre el cliente y el servidor con PHP:

1 - Tipeamos en la barra del navegador la dirección y el archivo a solicitar.

2 - El web browser (navegador) envía el mensaje a través de Internet a la computadora llamada www.lanacion.com solicitando la página (archivo) pagina1.php

3 - El web server (servidor web, que puede ser el Apache, IIS, etc.), recibe el mensaje y al ver que la extension es "php" solicita al interprete de PHP (que es otro programa que se ejecuta en el servidor web) que le envíe el archivo.

4 - El intérprete PHP lee desde el disco el archivo pagina1.php

5 - El intérprete PHP ejecuta los comandos contenidos en el archivo y eventualmente se comunica con un gestor de base de datos (ejemplos de ellos pueden ser MySql, MariaDB, Oracle, Informix, SQL Server, etc.)

6 - Luego de ejecutar el programa contenido en el archivo envía éste al servidor web.

7 - El servidor web envía la página al cliente que la había solicitado.

8 - El navegador muestra en pantalla el archivo que envió el servidor web.

La salida de los resultados en PHP se realiza normalmente en una página HTML, al contrario de otros lenguajes de programación “clásicos” cuya salida pueda ser directamente la pantalla.

Primer programa en PHP

Para agregar un programa PHP dentro de una página HTML debemos por un lado al crear el archivo definirlo con extensión php (a diferencia de las páginas estáticas que tienen extensión htm o html) y dentro del contenido de la página, encerrar el programa entre los símbolos

```
<?php [aquí el programa PHP] ?>
```

El comando de PHP para imprimir dentro de la página se llama echo. Nuestro programa "Hola Mundo" será entonces:

pagina1.php

```
<html>

<head></head>

<body>
  <?php
  echo "Hola Mundo";
  ?>
</body>

</html>
```

Es decir que la página que se generará al ejecutarse el programa será:

```
<html>

<head></head>

<body>
  Hola Mundo</body>

</html>
```

Podemos utilizar como editor de texto para codificar el programa PHP el VS Code.

Debemos almacenar el archivo php en la subcarpeta 'htdocs' que depende de la carpeta 'xampp'.

Este directorio es el que el servidor web Apache tiene asignado para recuperar páginas cuando se las solicitamos desde un navegador.

Ya tenemos almacenado el archivo en la carpeta 'htdocs' del 'xampp', ahora procedemos a solicitar dicha página desde un navegador web, para esto en la barra del navegador escribimos:

```
http://localhost/pagina1.php
```

Ejercicio 1.

Realiza un programa que muestre una serie de mensajes en la página empleando el comando echo. Ten en cuenta que cuando utiliza el comando echo el mensaje se debe encerrar entre comillas dobles o simples.
Toda instrucción finaliza con punto y coma.

Ejercicio 2.

Sabiendo que la función rand nos retorna un valor aleatorio entre un rango de dos enteros:

```
$num=rand(1,100);
```

En la variable $num se almacena un valor entero que la computadora genera en forma aleatoria entre 1 y 100.

Hacer un programa que lo muestre por pantalla al valor generado. Mostrar además si es menor o igual a 50 o si es mayor.

Para imprimir el contenido de una variable también utilizamos el comando echo:

```
echo $num;
```

Variables

Los nombres de variables comienzan con el caracter $ y son sensibles a mayúsculas y minúsculas (no así las palabras claves del lenguaje)

En PHP no es necesario definir el tipo de dato que almacena antes de utilizarla, las mismas se crean en el momento de emplearlas. Las variables se declaran cuando se le asigna un valor, por ejemplo:

```
$dia = 24;  //Se declara una variable de tipo integer.
$sueldo = 758.43; //Se declara una variable de tipo
double.
$nombre = "juan"; //Se declara una variable de tipo
string.
$exite = true; //Se declara una variable boolean.
```

También podemos hacer notar que para disponer comentarios de línea debemos utilizar dos caracteres //

Ejercicio 3

Definir una variable de cada tipo: integer, double, string y boolean. Luego imprimirlas en la página, una por línea.

Variables string

Una variable de este tipo puede almacenar una serie de caracteres.

```
$cadena1="Hola";
$cadena2="Mundo";
echo $cadena1." ".$cadena2;
```

Para concatenar string empleamos el operador . (punto)

Tengamos en cuenta que lo anterior sería equivalente a:

```
echo $cadena1;
echo " ";
echo $cadena2;
```

Cuando una cadena encerrada entre comillas dobles contiene una variable en su interior, ésta se trata como tal, por lo tanto se utilizará su contenido para el almacenamiento:

```
$dia=10;
$fecha="Hoy es $dia";
```

```
echo $fecha;
```

En pantalla se muestra: Hoy es 10

Es decir, en la cadena, se sustituye el nombre de la variable $dia, con el contenido de la misma.

Una cadena se puede definir con las comillas simples (pero es importante tener en cuenta que no se sustituyen las variables si empleamos comillas simples):

```
$nombre='juan carlos';
```

Veremos que en muchos casos se utiliza el concepto de sustitución de variables dentro de un string en PHP por lo que nos acostumbraremos en un principio a utilizar las comillas dobles para definir los string en nuestros programas.

Ejercicio 4

Crea un programa en el que pruebes algunas funciones de cadenas, por
ejemplo: substr(), strtolower(), strtoupper(), trim(), strchr(), strcmp(), str_word_count()

Ejercicio 5

Revise y ejecute el código PHP del fichero **cosas_de_variables.php**

Dentro del código del fichero anterior deberá completar los 10 comentarios de manera que **explique por qué y cómo funciona el código** correspondiente a cada ejercicio.

Ejercicio 6(uso del if).

Generar un valor aleatorio entre 1 y 3. Luego imprimir en castellano el número (Ej. si se genera el 3 luego mostrar en la página el string "tres").

Para ver si una variable es igual a cierto valor debemos plantear una condición similar a:

```
if ($valor==3)
{
  //algoritmo
}
```

Ejercicio 7.

Mostrar la tabla de multiplicar del 2. Emplear el for y luego el while o el do/while.
La estructura for permite incrementar una variable de 2 en 2:

```
  for ($f = 2; $f <= 20; $f = $f + 2)
```

Envío de datos de un FORMULARIO (controles text y submit)

Una actividad fundamental en PHP es la recolección de datos de un formulario HTML.

El proceso para el manejo de FORMULARIOS requiere generalmente dos páginas, una que implementa el formulario y otra que procesa los datos cargados en el formulario.

La estructura mínima de un formulario es la siguiente: para la entrada de un nombre de persona, un objeto text y un botón para el envío del dato al servidor:

pagina1.html

```
<html>

<head>
  <title>Formulario de entrada del dato</title>
</head>

<body>
  <form method="post" action="pagina2.php">
    Ingrese su nombre:
    <input type="text" name="nombre">
    <br>
    <input type="submit" value="confirmar">
  </form>
</body>

</html>
```

Ahora necesitamos una página con un pequeño programa en PHP que procese los datos ingresados en el formulario:

pagina2.php

```
<html>

<head>
  <title>Captura de datos del form</title>
</head>

<body>
  <?php
  echo "El nombre ingresado es:";
  echo $_REQUEST['nombre'];
  ?>
```

```
</body>

</html>
```

Para acceder al dato en PHP se cuenta con un vector llamado $_REQUEST indicando como subíndice el nombre del cuadro de texto que definimos en el formulario (dicho nombre es sensible a mayúsculas y minúsculas)
Es común indicar entre comillas simples el subíndice en lugar de comillas dobles (con comillas dobles también funciona)

En nuestro problema sólo mostramos por pantalla el valor ingresado en el formulario HTML:

```
  echo $_REQUEST['nombre'];
```

Ejercicio 8.

(Para la realización de los siguientes ejercicios relacionados con la recogida de datos en formularios se recomienda consultar la siguiente web: Recogida de datos.)

Confeccionar un formulario que solicite la carga de un nombre de persona y su edad, luego mostrar en otra página si es mayor de edad (si la edad es mayor o igual a 18)

Ejercicio 9.

Solicitar que se ingrese por teclado el nombre de una persona y disponer tres controles de tipo radio que nos permitan seleccionar si la persona: 1-no tiene estudios, 2-estudios primarios y 3-estudios secundarios.
En la página que procesa el formulario mostrar el nombre de la persona y un mensaje indicando el tipo de estudios que posee.

Ejercicio 10.

Confeccionar un formulario que solicite la carga del nombre de una persona y que permita seleccionar una serie de deportes que practica (futbol, basket, tennis, voley)
Mostrar en la página que procesa el formulario la cantidad de deportes que practica.

(Si el checkbox no está seleccionado en el formulario no se crea una entrada en el vector asociativo $_REQUEST, para saber si existe una determinada componente en un vector se emplea la función isset, si retorna true significa que existe y por lo tanto el checkbox está seleccionado).

Ejercicio 11.

Confeccionar un formulario que solicite el ingreso del nombre de una persona y un control select (en este último permitir la selección de los ingresos mensuales de la persona: 1-1000,1001-3000,>3000)
En la página que procesa el formulario mostrar un mensaje si debe pagar impuestos a las ganancias (si supera 3000)

Arrays

Un Array es una colección de valores. Los array pueden ser unidimensionales (vectores), bidimensionales (matrices) y multidimensionales (más de dos dimensiones)

Los arrays se utilizan ampliamente en el lenguaje PHP.

Se utiliza el delimitador [] para acceder a los diferentes elementos del vector.

Se lo puede crear al vuelo, sin tener que declararlo:

```
$dias[0]=31;
$dias[1]=28;
```

Luego de estas dos líneas, tenemos creado un vector de dos elementos, a los cuales accedemos por un subíndice.

```
echo $dias[0];    //31
echo $dias[1];    //28
```

También podemos obviar el subíndice cuando asignamos los valores:

```
$dias[]=31;
$dias[]=28;
$dias[]=31;
```

Automáticamente comienza a numerarse desde cero.

Acotaciones

Cuando tenemos que recorrer en forma completa un vector en PHP es muy común utilizar la estructura 'foreach'. Veamos el mismo ejemplo anterior para recorrer el vector $nombres:

pagina1.php

```
<html>
```

```
<head>
  <title>Problema</title>
</head>

<body>

  <?php
  $nombres[] = "juan";
  $nombres[] = "pedro";
  $nombres[] = "ana";
  foreach ($nombres as $nombre) {
    echo $nombre;
    echo "<br>";
  }
  ?>

</body>

</html>
```

Ejercicio 12.

Definir un vector o array con los nombres de los días de la semana. Luego imprimir el primero y el último elemento del vector.

Arrays o vectores asociativos

Este tipo de vectores no es común a otros lenguajes, pero en PHP son de uso indispensable en distintas situaciones (ya lo empleamos cuando recuperamos información de un formulario accediendo al vector $_REQUEST que crea PHP en forma automática, cuando accedamos a datos de una base de datos también lo emplearemos etc.)

Un vector asociativo permite acceder a un elemento del vector por medio de un subíndice de tipo string.

Inicialmente uno piensa que esto nos complica las cosas, como veremos más adelante la realidad nos demuestra lo contrario.

Como ejemplo, consideremos que deseamos guardar en un vector el DNI, nombre y dirección de una persona. Empleando un vector con subíndice entero la solución sería:

```
<?php
```

```
$registro[] = "20456322";
$registro[] = "Martinez Pablo";
$registro[] = "Colon 134";
?>
```

De esta forma debemos recordar que cuando deseamos mostrar el nombre de la persona debemos acceder al subíndice 1. Esto es sencillo si tenemos un vector con tres elementos, pero que sucede si debemos almacenar 20 datos relacionados en un vector.

Un vector asociativo se define de la siguiente forma:

```
<?php
$registro['dni'] = "20456322";
$registro['nombre'] = "Martinez Pablo";
$registro['direccion'] = "Colon 134";
echo $registro['nombre'];
?>
```

Ahora vemos que para imprimir el nombre de la persona no debemos recordar una posición dentro de un vector sino un nombre de clave. Esto se hace indispensable cuando administramos un conjunto de datos grandes.

En un vector asociativo toda componente está asociada a una clave.

Otras formas de crear un vector asociativo:

```
<?php
$registro = array(
  'dni' => '20456322',
  'nombre' => 'Martinez Pablo',
  'direccion' => 'Colon 134'
);
echo $registro['dni'];
?>
```

Ejercicio 13

A partir de la página:

pagina1.html

```
<head>
  <title>Problema</title>
</head>

<body>
  <form action="pagina2.php" method="post">
```

```
    Ingrese primer valor:
    <input type="text" name="valor1">
    <br>
    Ingrese segundo valor:
    <input type="text" name="valor2">
    <br>
    Ingrese tercer valor:
    <input type="text" name="valor3">
    <br>
    Ingrese cuarto valor:
    <input type="text" name="valor4">
    <br>
    Ingrese quinto valor:
    <input type="text" name="valor5">
    <br>
    <input type="submit">
  </form>
</body>

</html>
```

Crea un fichero php que sume todos los valores de los campos mediante el uso de un vector asociativo (investiga como usar en este caso el foreach) (Es importante notar que al input de tipo 'submit' no le definimos la propiedad 'name', con el objetivo que no se cargue en el vector asociativo '$_REQUEST').