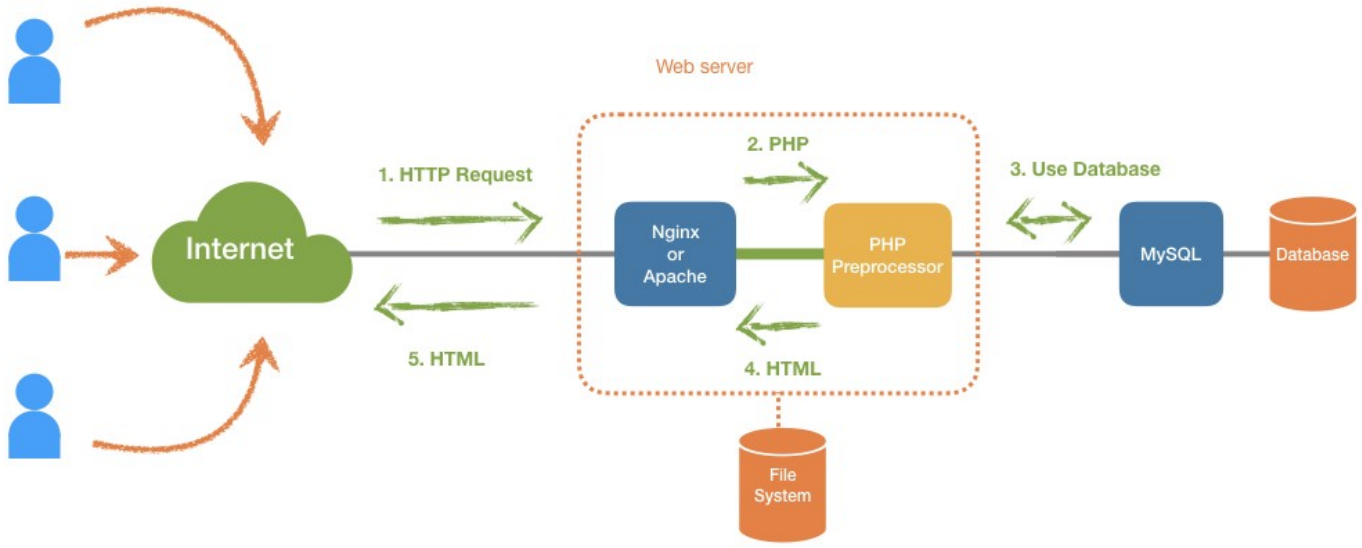
Conceptos básicos

¿Qué es PHP?

PHP es un lenguaje de programación de uso general, especialmente adecuado para el desarrollo web. El código PHP puede ser interpretado y ejecutado desde la interfaz de línea de comandos (CLI) o desde un servidor web que tenga implementado un intérprete PHP.

¿Cómo funciona PHP?



¿Qué sitios web utilizan PHP?

En la actualidad, PHP está siendo utilizado en gran cantidad de sitios web. Entre los sitios web más destacados podemos encontrar:

- Wordpress
- PrestaShop
- Drupal
- Joomla

Sitios como **Facebook** o **Wikipedia**, hacen uso del lenguaje de programación Hack que es una extensión de PHP y se ejecuta en la máquina virtual HHVM (HipHop Virtual Machine).

Frameworks PHP

Un *framework* es una estructura conceptual y tecnológica de asistencia definida, normalmente, con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software. (Fuente: Wikipedia)

A la hora de desarrollar software es muy común hacer uso de *frameworks* ya que nos ayudarán a reducir a cantidad de código que tenemos que escribir y nos proporcionarán una gran cantidad de bibliotecas para realizar funciones de uso común.

Algunos de los *frameworks* PHP más utilizados actualmente son:

- Laravel.
- Symfony.
- CodeIgniter.
- Yii.
- Slim.
- CakePHP.
- Zend Framework.

Sintaxis básica

Etiquetas de apertura y cierre de PHP

El código PHP se encierra entre las etiquetas de apertura y cierre: `<?php` y `?>`. Todo el código que se encuentre estas dos etiquetas será interpretado como código PHP.

Ejemplo:

```
<?php

echo "¡Hola mundo!";

?>
```

Es posible omitir la etiqueta de cierre `?>` cuando el contenido del archivo sólo sea código PHP.

Ejemplo:

```
<?php

echo "¡Hola mundo!";
```

Carácter separador de instrucciones

Todas las instrucciones en PHP terminan con el carácter punto y coma (`;`).

Ejemplo:

```
<?php
    echo "¡Hola ";
    echo "mundo!";
?>
```

El único caso donde se puede omitir el carácter punto y coma (`;`) en la última instrucción de un bloque PHP, ya que la etiqueta de cierre de un bloque PHP (`?>`) implica un punto y coma.

Ejemplo:

```
<?php
    echo "¡Hola ";
    echo "mundo!"
?>
```

Comentarios

Podemos escribir comentarios de una sóla línea con: `//` y #, y comentarios multilínea con `/* ... */`.

Ejemplo:

```
<?php
    // Esto es un comentario de una línea
    echo "Frase 1";

    # Esto es otro comentario de una línea
    echo "Frase 2";

    /* Este comentario
    es un comentario de múltiples líneas */
    echo "Frase 3";
?>
```

Código PHP embebido en documentos HTML

El uso de estas etiquetas de apertura y cierre, nos permite embeber código PHP en documentos HTML. De modo que sólo el código que aparezca entre las etiquetas `<?php` y `?>` será interpretado por el intérprete de PHP y el resto de etiquetas serán ignoradas.

Ejemplo de código PHP embebido en un documento HTML:

```
<!DOCTYPE html>
<html>
<head>
    <title>Ejemplo</title>
</head>
<body>
    <p>Contenido escrito en HTML</p>
    <?php echo "Contenido escrito desde PHP"; ?>
</body>
</html>
```

Tipos de datos

Documentación oficial de PHP.

Variables

Las variables en PHP se representan con el símbolo del dólar ($) seguido por el nombre de la variable. El nombre de la variable es sensible a mayúsculas y minúsculas.

Ejemplo:

```
<?php
    $nombre = "Pepe";
    $edad = 30;
?>
```

Documentación oficial de PHP.

Ámbito de las variables

Documentación oficial de PHP.

Constantes

Las constantes no pueden modificar su valor durante la ejecución del script. El nombre de la constante **no tiene** que ir precedido por el símbolo del dólar y es sensible a mayúsculas y minúsculas. Por convención, las constantes siempre se declaran en mayúsculas.

Ejemplo:

```
<?php
    // Ejemplo de una constante numérica de tipo real
    define("PI", 3.141592);

    // Ejemplo de una constante de tipo string
    define("CONSTANTE", "Hola mundo");
?>
```

Documentación oficial de PHP.

`echo`

`echo` es una construcción del lenguaje (no es una función) que nos permite mostrar cadenas de texto y el contenido de las variables.

Ejemplo:

```
<?php

echo "Esto es una cadena de texto.";

// Con echo también podemos mostrar el contenido de una variable
$numero = 10;

echo "El contenido de la variable es: $numero";

?>
```

Documentación oficial de `echo`.

`var_dump`

`var_dump` es una función nos permite mostrar el contenido de una variable. Esta función muestra el tipo de dato y el valor de la variable.

Ejemplo:

```
<?php

$nombre = "Pepe";
$edad = 30;
$nota = 7.5;

var_dump($nombre);
// string(4) "Pepe"

var_dump($edad);
// int(30)

var_dump($nota);
// float(7.5)

?>
```

Documentación oficial de la función `var_dump`.

`print_r`

`print_r` es una función que nos permite mostrar el contenido de una variable de una forma legible.

Ejemplo:

```
<?php

$lista = array("Pepe", "María", "Juan");

print_r($lista);

//Array
//(
//    [0] => Pepe
//    [1] => María
//    [2] => Juan
//)

?>
```

Documentación oficial de la función `print_r`.

Ejercicios de introducción

1. Escribe un script PHP que muestre información sobre la configuración de PHP que hay en el servidor.

Notas:

- Documentación oficial de la función `phpinfo`.

2. Revise la documentación oficial de PHP para ver qué información podemos obtener de la variable *superglobal* `$_SERVER`**. Escribe un script haciendo uso de la variable** *superglobal* `$_SERVER` **que muestre lo siguiente:**

- La dirección IP del servidor donde se está ejecutando el script.
- El nombre del host del servidor donde se está ejecutando el script.
- El software que está utilizando el servidor para servir el script.
- Información sobre el agente de usuario (*User Agent*) desde el que se está solicitando el script.
- La dirección IP del cliente que está solicitando el script.

Notas:

- Documentación oficial de la variable *superglobal* `$_SERVER`.

3. Revise la documentación oficial para conocer todas las variables *superglobals* **que existen. Con ayuda de la función** `print_r` **muestra el contenido de cada una de las variables** *superglobals***.**

Notas:

- Documentación oficial de las variables *superglobals*.
- Documentación oficial de la función `print_r`.

4: ¿Qué es el tipado dinámico en PHP?. ¿Qué me permite?.

5: Concatenación de Cadenas y Enteros

Escribe un programa que concatene una cadena de texto y un número entero y muestre el resultado. Luego, verifica el tipo de la variable resultante.

6: Suma de Enteros y Cadenas

Escribe un programa que sume un número entero y una cadena de texto y muestre el resultado. Luego, verifica el tipo de la variable resultante.

7: Cambio Dinámico de Tipo

Escribe un programa que muestre el tipo de una variable antes y después de asignarle un valor de un tipo diferente. Observa cómo PHP cambia dinámicamente el tipo de la variable.

8: Operaciones con Variables de Diferentes Tipos

Crea un programa que realice operaciones matemáticas con variables de diferentes tipos y muestre los resultados y tipos intermedios.