Lenguaje PHP

Origen

PHP nació en 1994 con el nombre de PHP/FI, creado por Rasmus Lerdorf con el objetivo de rastrear las visitas a su currículum online y lo llamó "Personal Home Page Tools" o "PHP Tools".

En junio de 1995, Rasmus publicó el código fuente después de haberlo reescrito.

PHP tenia una estructura similar a la de C y una sintaxis cercana a Perl, lo que lo convertía en una elección atractiva para los desarrolladores de estos lenguajes.

En 1997, PHP dejó su fase beta y para 1998 ya contaba con alrededor de 60.000 dominios, según una encuesta de Netcraft.

Con el paso de los años, el nombre evolucionó a PHP “Hypertext Preprocessor”.

Características

- PHP es de código abierto → https://github.com/php/php-src
- Está orientado al desarrollo de aplicaciones web dinámicas
- Se puede integrar con HTML
- El código es invisible para el navegador y el cliente, ya que es el servidor el encargado de ejecutar el código y enviar el resultado al cliente
- Tiene conexión con la mayoría de los motores de base de datos, como MySQL y PostgreSQL
- Admite la integración con una amplia variedad de extensiones que pueden ser agregadas al archivo de configuración `php.ini`

  ```
  // Ejemplos para añadir extensiones
  extension=myext.so
  extension=myext.dll
  extension_dir=<directorio de la extensión>
  ```

- Permite controlar excepciones a partir de la version 5 de PHP
- PHP tiene una gran comunidad y una documentación amplia y completa

- Cuenta con diversos frameworks que proporcionan herramientas y estructuras predefinidas que simplifican el desarrollo de aplicaciones web

Sintaxis

- Utiliza etiquetas de inicio y fin para delimitar el código
  - `<?php` código `?>`
  - `<?=` código `?>`
  - `<?` código `?>`
- Convención de nomenclatura `snake_case`
- Variables → `$variable`
- Constantes → `const VARIABLE`
- Todas las declaraciones finalizan con `;`

Operadores:

- Operador de flecha: Accede a elementos de un objeto (no constantes ni estáticos)
  `$objeto->elemento`
- Operador de resolución de ámbito: Accede a elementos constantes o estáticos de un objeto
  `Objeto::$elemento`

Evolución del sistema de tipado en PHP:

PHP 4 → Tipado débil

PHP 5 → Introducción de parámetros

PHP 7 → Modo estricto

PHP 8 → Introducción de tipos `mixed` y `static`

- No es necesario declarar el tipo de variable al ser creada
- El tipo de variable pueda cambiar durante la ejecución

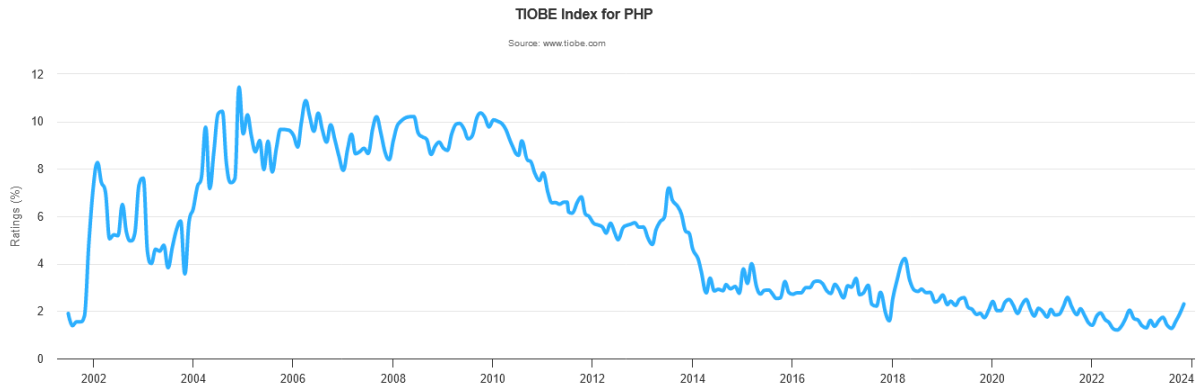
Paradigma

- Es un lenguaje imperativo
  - Se fundamenta en un conjunto de instrucciones secuenciales que indican al computador cómo realizar una tarea
- Desde PHP 3, se permitió la programación orientada a objetos (POO), facilitando la estructuración del código y la reutilización.
- Es un lenguaje interpretado por un módulo del servidor web o como un ejecutable CGI (Common Gateway Interface → interfaz de entrada común)

- El código se ejecuta directamente en el servidor web sin requerir compilarlo previamente

Futuro

En la actualidad, muchos desarrolladores comparten la percepción de que PHP está en declive. Sin embargo, la realidad es que, aunque su popularidad ha disminuido con el tiempo y las empresas buscan migrar hacia otros lenguajes, PHP sigue siendo utilizado en aproximadamente el 77% de todos los sitios web que emplean un lenguaje de programación del lado del servidor.



Parte de la razón por la cual PHP sigue siendo ampliamente utilizado es por el uso de los sistemas de gestión de contenido (CMS) como WordPress, Joomla o Drupal, los cuales impulsan aproximadamente el 40% de todos los sitios web en internet. Estos CMS dependen de PHP, consolidando su presencia en la web.

Cabe destacar que las versiones más recientes de PHP incorporan considerables mejoras en rendimiento, seguridad, compatibilidad y escalabilidad, entre otros aspectos. La combinación de estas mejoras y el creciente uso de los CMS sugiere que PHP no está destinado a quedar obsoleto en un corto plazo.

Métodos mágicos

Son métodos especiales que provee PHP y se ejecutan cuando sucede una determinada condición o evento que los activan, sin necesidad de especificar el nombre del método.

Estos métodos, se identifican usando doble barra baja (`__`) como prefijo → `__metodo()`

__construct()

Es llamado al crear una nueva instancia del objeto.

Permite inyectar parámetros para inicializar el objeto.

```
class Alumno {
  protected $id;
  protected $nombre;
  public function __construct($id, $nombre) {
    $this->id = $id;
    $this->nombre = $nombre;
  }

}

$alumno = new Alumno(54, "Jose");
```

__get()

Es llamado cuando se trata de acceder al valor de una propiedad private o protected.

Permite consultar el valor de una propiedad private o protected o determinar su existencia.

```
class Alumno {
  protected $id;
  protected $nombre;
  public function __construct($id, $nombre) {
    $this->id = $id;
    $this->nombre = $nombre;
  }
    public function __get($propiedad) {
        return $this->$propiedad;
    }
}

$alumno = new Alumno(54, "Jose");
print($alumno->id);
```

__set()

Es llamado cuando se trata de definir o modificar el valor de un atributo private o protected.

Permite definir o modificar el valor de un atributo private o protected.

```
class Alumno {
  protected $id;
  protected $nombre;
  public function __construct($id, $nombre) {
    $this->id = $id;
    $this->nombre = $nombre;
  }
    public function __set($id, $nombre) {
      $this->$propiedad = $valor;
    }
}

$alumno = new Alumno(54, "Jose");
$alumno->nombre = 'Juan';
```

__isset()

Es llamado cuando se invoca la función `isset()` sobre el atributo de un objeto.

Determina la existencia o no de una variable.

```
class Alumno {
    protected $id;
    protected $nombre;
    public function __construct($id, $nombre) {
        $this->id = $id;
        $this->nombre = $nombre;
    }
    public function __isset($propiedad) {
        return isset($this->$propiedad);
    }
}

$alumno = new Alumno(54, "Jose");
echo isset($alumno->id) ? "La id existe" : "La id no existe"
```

__unset()

Es llamado cuando se invoca la función `unset()` sobre un atributo.

Permite destruir una variable.

```
class Alumno {
    protected $id;
```

```
    protected $nombre;
      public function __construct($id, $nombre) {
        $this->id = $id;
        $this->nombre = $nombre;
      }
    public function __unset($propiedad) {
            unset($this->$propiedad);
    }
}

$alumno = new Alumno(54, "Jose");
unset($alumno->nombre);
```

__toString()

Es llamado al invocar una función de impresión `echo()`, `print()` o `printf()`.

Permite devolver el objeto representado en forma de string.

```
class Alumno {
    protected $id;
    protected $nombre;
    public function __construct($id, $nombre) {
        $this->id = $id;
        $this->nombre = $nombre;
    }
    public function __toString() {
        return $nombre . " (" . $id . ")";
    }
}

$alumno = new Alumno(54, "Jose");
echo $alumno;
```

__invoke()

Es llamado cuando se intenta invocar un objeto como si se tratara de una función.

Permite controlar el comportamiento de un objeto cuando este intenta ser llamado como una función.

```
class Operacion {
    public function __invoke($x, $y) {
        return $x + $y;
```

```
    }
}

$operacion = new Operacion();
$resultado = $operacion(5, 3);
echo "El resultado de la suma es: " . $resultado;
```

__call()

Es llamado cuando se intenta llamar a un método que no es accesible públicamente.

```
class Ejemplo {
  public function __call($nombreMetodo, $argumentos) {
    echo "Llamando al método " . $nombreMetodo;
  }
}

$obj = new Ejemplo ();
$obj->noExiste(1, 2, 3);
```

__callStatic()

Es llamado cuando se intenta llamar a un método estático que no es accesible públicamente.

```
class Ejemplo {
  public static function __callStatic($nombreMetodo, $argumentos) {
    echo "Llamando al método estático " . $nombreMetodo;
  }
}

Ejemplo::noExiste(1, 2, 3);
```

__clone()

Se llama en el proceso de clonación de objetos.

Permite realizar acciones específicas antes o después de la clonación.

```
class Ejemplo {
  public $valor;
  public function __construct($valor) {
    $this->valor = $valor;
  }
  public function __clone() {
```

```
    echo "Objeto clonado con el valor " . this->atributo;
  }
}

$obj = new Ejemplo(42);
$obj2 = clone $obj;
```

Funciones anónimas

Son bloques de código que se pueden asignar a variables o invocar directamente desde otras funciones.

Se definen utilizando la palabra clave `function`.

Propósito de las funciones anónimas

Closure: Función anónima que puede capturar y usar variables del ámbito cuando se definen.

- Permiten usar variables mediante la palabra `use`
- Si se altera el valor de la variable, no afectará a la variable original, excepto si se añade un `&` antes de la variable
- Es posible añadir argumentos

```
$color = 'verde';

$mostrarColor = function() use (&$color) {
    echo $color;
  $color = 'azul';
};

$mostrarColor();
echo $color; // Mostrará azul
```

Callbacks: Función anónima llamada por otra función que usa a la primera como parámetro.

```
$saludo = function($nombre) {
    echo "Hola " . $nombre;
};

$saludo('Jose');
```

Referencias

https://www.php.net/manual/es/history.php.php

https://startit.rs/rasmus-lerdorf-kreator-php-a-programski-jezici-su-samo-alat-za-menjanje-sveta/

---

https://www.php.net/manual/es/intro-whatis.php

https://www.php.net/downloads

https://diego.com.es/extensiones-en-php

https://www.php.net/manual/es/install.pecl.windows.php

https://www.php.net/manual/es/lua.installation.php

https://es.wikipedia.org/wiki/PHP

https://www.mindomo.com/es/mindmap/paradigma-de-programacion-a723cf457c474b619e45c877188854b7

https://aulab.es/articulos-guias-avanzadas/31/variables-y-constantes-en-php

https://www.php.net/manual/es/language.oop5.paamayim-nekudotayim.php#example-257

https://www.php.net/manual/es/security.intro.php

---

https://www.tiobe.com/tiobe-index/

https://www.tiobe.com/tiobe-index/php/

https://kinsta.com/es/cuota-mercado-php/

https://kinsta.com/es/base-de-conocimiento/que-es-php/

https://www.mediummultimedia.com/web/la-gente-todavia-usa-php-en-2023/

---

https://www.php.net/manual/es/language.oop5.magic.php

https://diego.com.es/metodos-magicos-en-php

https://styde.net/uso-de-metodos-magicos-en-php/

https://platzi.com/tutoriales/1338-introduccion-php-2018/4753-todo-sobre-metodos-magicos-en-php/

https://platzi.com/tutoriales/1338-introduccion-php-2018/6531-te-enseno-que-son-los-metodos-magicos-en-php-y-sus-diferentes-usos/

---

https://www.php.net/manual/es/functions.anonymous.php

https://diego.com.es/funciones-anonimas-y-clausuras-en-php

https://aulab.es/articulos-guias-avanzadas/41/funciones-anonimas-en-php