Universidade Federal do Espírito Santo
Departamento de Informática

Linguagens de Programação

Heitor Costa Oliveira
Gabriel Brozinga Zandonadi
Vivian dos Reis Moreira

Histórico

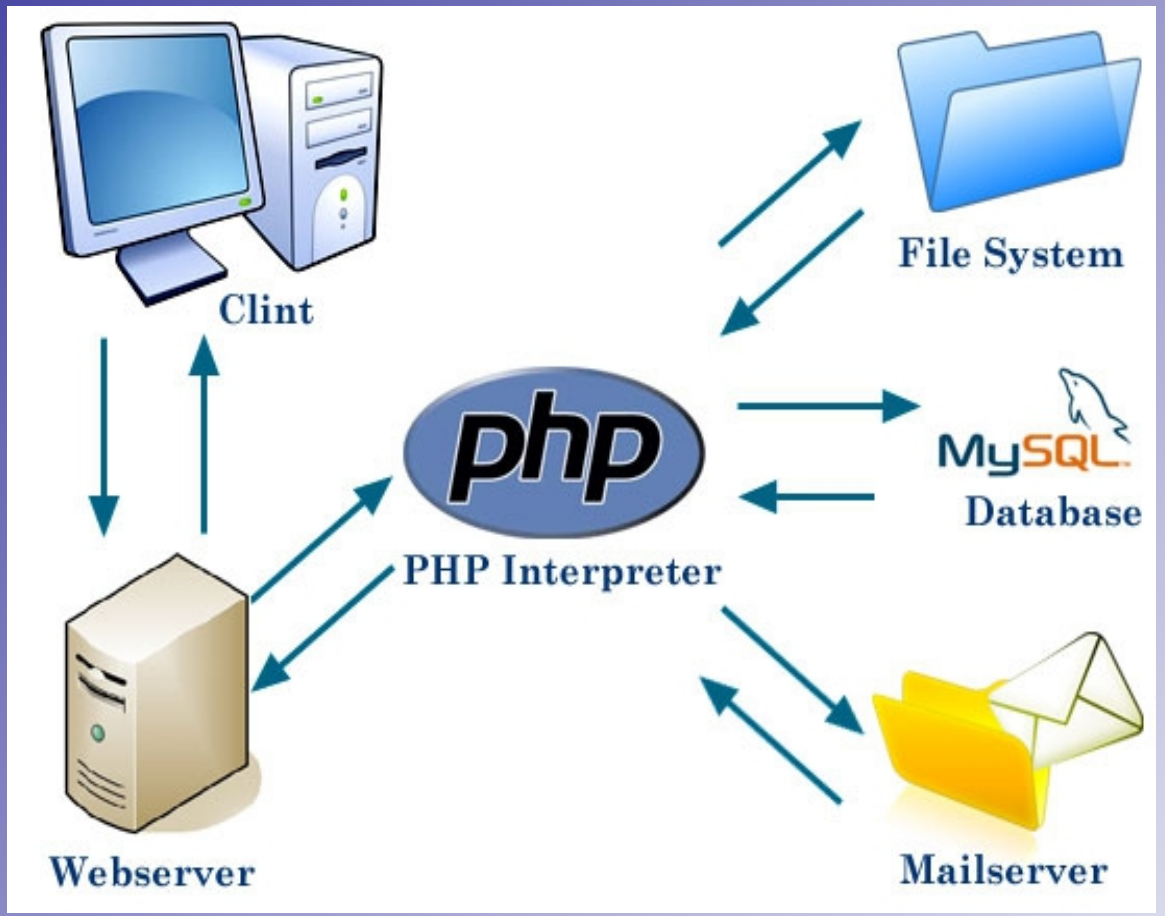
- Criada em 1995 por Rasmus Lerdorf - Personal Home Page.
- 1997: PHP/FI – Com interpretador de comandos SQL.
- 1998: PHP 3 – Orientação a objetos com certas limitações.
- 2000: PHP 4 – Maior suporte a OO.
- 2004: PHP 5 – Versão atual.

Propriedades

- Orientação a objetos e código estruturado.
- Tipagem dinâmica.
- Sintaxe muito semelhante a C/C++.
- Grande portabilidade.
- Feito para programação Web veloz, simples e eficiente.
- Código interpretado.

Como Funciona

- Cliente recebe apenas HTML. O servidor processa o código e retorna apenas saída.



Operadores - Aritméticos

Operador	Símbolo	Exemplo	Resultado
Adição	+	10 + 4	14
Subtração	-	10 - 4	6
Multiplicação	*	10 * 4	40
Divisão	/	10 / 4	2.5
Resto da divisão	%	10 % 4	2

Operadores - Atribuição

Operador	Símbolo
Atribuição com adição	+=
Atribuição com subtração	-=
Atribuição com multiplicação	*=
Atribuição com divisão	/=
Atribuição com resto da divisão	%=
Atribuição com concatenação	.=
Atribuição com operador binário OR	|=
Atribuição com operador binário AND	&=
Atribuição com operador binário XOR	^=
Atribuição com deslocamento à esquerda	<<=
Atribuição com deslocamento à direita	>>=

Operadores - Binário

Operador	Símbolo
Negação (NOT)	~
Ou (OR)	|
E (AND)	&
Ou exclusivo (XOR)	^
Deslocamento à esquerda	<<
Deslocamento à direita	>>

Operadores – Incremento/Decremento

Operador	Descrição	Exemplo
Pré incremento	Incrementa a variável em um e retorna seu valor	++$x
Pós incremento	Retorna o valor da variável e incrementa seu valor em um	$x++
Pré decremento	Decrementa a variável em um e retorna seu valor	--$x
Pós decremento	Retorna o valor da variável e decrementa seu valor em um	$x--

Operadores - Lógico

Operador	Símbolo
Negação (NOT)	!
E (AND)	&&
AND (equivale a &&)	AND
Ou (OR)	||
OR (equivale a ||)	OR
Ou exclusivo (XOR)	XOR

Operadores - Relacionais

Operador	Símbolo
Igualdade	==
Testa igualdade e tipo	===
Maior	>
Menor	<
Maior ou igual	>=
Menor ou igual	<=
Diferente	!=
Testa diferença e tipo	!==

Palavras Chave

Palavras-chave do PHP

abstract (a partir do PHP 5)	and	array()	as	break
case	catch (as of PHP 5)	cfunction (PHP 4 only)	class	clone (as of PHP 5)
const	continue	declare	default	do
else	elseif	enddeclare	endfor	endforeach
endif	endswitch	endwhile	extends	final (as of PHP 5)
for	foreach	function	global	goto (apartir do PHP 5.3)
if	implements (a partir do PHP 5)	interface (a partir do PHP 5)	instanceof (a partir do PHP 5)	
namespace (a partir do PHP 5.3)	new	old_function (PHP 4 somente)	or	private (a partir do PHP 5)
protected (a partir do PHP 5)	public (a partir do PHP 5)	static	switch	throw (a partir do PHP 5)
try (a partir do PHP 5)	use	var	while	xor

Palavras Chave - Constantes

- __CLASS__
- __DIR__
- __FILE__
- __LINE__
- __FUNCTION__
- __METHOD__
- __NAMESPACE__

Palavras Chave - Construtores

- die()
- echo
- empty()
- exit()
- eval()
- include include_once
- isset()
- list()
- require
- require_once
- return
- print
- unset()

Amarrações

- A partir da versão 5.3 do PHP, a vinculação estática tardia, ou dinâmica, passou a ser suportada.

```
class A {
   static $word = "hello";
   static function hello() {print static::$word;}
}

class B extends A {
   static $word = "bye";
}

B::hello();
```

Amarrações

- Desvantagens:
  - Menor confiabilidade.
  - Maior custo.
- A verificação de tipo deve ser feita em tempo de execução.
- Gasto maior de memória: toda variável precisa ter, em tempo de execução, um descritor associado a ela.
- A implementação da LP tem de ser feita através de interpretação pura (10x mais demorado em relação à compilação). Isso porque os computadores não tem instruções capazes de manipular operandos de tipo desconhecido.

Tipos de Dados

- PHP utiliza checagem de tipos dinâmica. Por este motivo não é necessário declarar o tipo de uma variável para usá-la. O interpretador PHP decidirá qual o tipo daquela variável, verificando o conteúdo em tempo de execução.
- Ainda assim, é permitido converter os valores de um tipo para outro desejado, utilizando o typecasting ou a função settype.

Tipos de Dados

- Inteiro (Integer ou Long).

```
<?php
    $tipo = 1234;
    $tipo = -2345;
    $tipo = 0234; //base octal. 156 na base decimal.
    $tipo = 0x34; //base hexadecimal. 52 na base decimal.
 ?>
```

Tipos de Dados

- Ponto flutuante (Float ou Double).

```
<?php
    $tipo = 1.543;
    $tipo = 23e4; // (equivale a 230.000)
?>
```

Tipo de Dados

- Booleano (TRUE ou FALSE).

```
<?php
    $noite = TRUE;
    $chuva = FALSE;
    if($noite == TRUE)
        echo "Está de noite.";
    else
        echo "Está de dia.";
    if($chuva)
        echo "Está chovendo.";
    else
        echo "Não está chovendo.";
    //saída: Está de noite.Não está chovendo.
?>
```

Tipos de Dados

- String – série de caracteres de 1 byte cada.

```
<?php
    $testando = "Olá Mundo Novo";
    $testando1 = '--- $testando ---';
    echo $testando1;
    //--- $testando ---
?>
<?php
    $testando = "Olá Mundo Novo";
    $testando1 = "--- $testando ---";
    echo $testando1;
    //--- Olá Mundo Novo ---
?>
```

Tipos de Dados

- Array.

```
<?php
    $vetor [1] = "Alemanha ";
    $vetor [2] = "Italia ";
    $vetor [3] = "França";
    $vetor ["testando"] = 1;
?>
<?php
    $vetor = array(1 => "Alemanha", 2 => "Italia", 3 => "França", "testando" => 1);
?>
<?php
    $vetor = array("Alemanha", "Italia", "França", "Noruega");
    echo $vetor[1]; //Italia
?>
```

Tipos de Dados

- Objeto – instâncias de classes.

```
<?php
    class CD {
        var $titulo;
        var $banda;
        var $ano_lancamento;
    }
    $disco = new CD();
    $disco->titulo = "The Number of the Beast";
    $disco->banda = "Iron Maiden";
    $disco->ano_lancamento = 1982;
?>
```

Tipos de Dados

- Resource - variável especial que mantém referência de recurso externo, como manipuladores especiais para arquivos abertos ou conexões com bancos de dados.

```
<?php
    $mysql_access = mysql_connect($server, $user, $pw);
    if(is_resource($mysql_access)){
        echo 'A variável $mysql_access é do tipo resource.';
    }
?>
```

Tipos de Dados

- NULL.

```
<?php
    $var = NULL;
    $a = array();
    $a == null; //true
?>
```

Variáveis

- Nomes de variáveis devem começar com letra ou sublinhado seguidos por letras, números ou sublinhados.
- Variáveis são precedidas por cifrão.
- Tipagem dinâmica – o interpretador infere tipos de variáveis na atribuição, sem necessidade de definição.

```
$novavar = 3; // Cria um variável integer com valor 3

$novavar = “Nova”; // Agora $novavar é string com valor “Nova”
```

Variáveis

- Modificador static - uma variável estática é visível num escopo local, mas ela é inicializada apenas uma vez e seu valor não é perdido quando a execução do script deixa esse escopo.

```
function Teste() {
    static $a = 0;
    echo $a;
    $a++;
}
```

Variáveis

- Variáveis de ambiente:

  $GLOBALS - Referencia todas variáveis disponíveis no escopo global.

  $_SERVER - Informação do servidor e ambiente de execução.

  $_GET - HTTP GET variáveis.

  $_POST - HTTP POST variáveis.

  $_FILES - HTTP File Upload variáveis.

  $_REQUEST - Variáveis de requisição HTTP.

  $_SESSION - Variáveis de sessão.

  $argc - O número de argumentos passados para o script.

  $argv - Array de argumentos passados para o script.

Variáveis

- URLencode – variáveis enviadas pelo navegador.
  - Script: http://localhost/teste.php3
  - Chamada: http://localhost/teste.php3?vivas=teste
  - automaticamente o PHP criará uma variável com o nome $vivas contendo a string “teste”. Note que o conteúdo da variável está no formato urlencode.
  - Os formulários html já enviam informações automaticamente nesse formato, e o PHP decodifica sem necessitar de tratamento pelo programador.

Variáveis

- URLencode – variáveis enviadas pelo navegador:
  - o texto “Testando 1 2 3 !!” em urlencode fica “Testando+1+2+3+%21%21”
  - string urlencode(string texto); //codificar
  - string urldecode(string texto); //decodificar

Constantes

- Nomes de constantes devem vir em maiúsculas por convenção.

```
define(“CONSTANTE1”, ”Conteúdo da constante1”);
define(“CRAQUE”, “Neymar”);
define(“PI” , 3.14);
```

Expressões

- Em PHP, quase tudo são expressões. Expressões são tudo aquilo que tem algum valor.

  $varA = 5; // Atribuímos 5 para $a.

  Neste caso, ‘5’ é uma expressão com valor inteiro 5.

- Normalmente, são seguidas por ; assim como em C/C++.

Expressões

- $x = 1; // Atribui 1 a $x
- $x += VALOR; // Incrementa o valor de $x de VALOR
- $x -= VALOR; // Diminui o valor de $x de VALOR
- $x++; // Incrementa o valor de $x de 1
- $x--; // Diminui o valor de $x de 1
- $x != 1 // FALSE se x = 1 TRUE se não for
- $x == 1 // TRUE se x = 1 FALSE se não for
- (expressao1) ? (expressao2) : (expressao3) //operador ternário

Comandos

- Condicionais:

```
if($a > $b)
    $a = $b;
elseif($b > $a)
    $b = $a;
else
    $a = $b = 1;
```

Comandos

- Loops:

```
while($i < 3)
    $i++;

do {$i++} while ($i <3);
```

Comandos

- Loops:

```
for($i=0; $i < 10; $i++)
    $umVetor[$i] = $i;

foreach($umArray as $elemento)
    echo $elemento; // Imprime valores de um vetor
```

Comandos

- Quebra de fluxo – break:

```
while ($x > 0){
    …
    if($x == 20){
        echo “erro! X = 20”;
        break;
        ...
    }
}
```

Comandos

- Quebra de fluxo - continue:

```
for($i = 0;$i < 100;$i++){
    if($i%2)
        continue;
    echo “  $i “;
}
```

Gerenciamento de Memória

- PHP tem um coletor de lixo que atua em diversos momentos. Quando especificado pelo usuário, em fim de função ou no final de um script.
- O usuário pode utilizar funções de liberação de memória como unset() e mysql free result(). No final de uma função, o interpretador limpa os recursos implicitamente.
- Variáveis fora de escopo são liberadas. No fim de um script, naturalmente todos os recursos do mesmo são liberados.

Gerenciamento de Memória

- Existe a opção de habilitar ou desabilitar o coletor de lixo utilizando gc enable() e gc disable(). Pode ainda verificar se está ativo com gc enabled(). A função gc collect cycles() força a ação do coletor e retorna o número de ciclos coletados.

Modularização

- Utiliza-se as funções include() e require(). Assim pode-se modularizar layout de sites ou programas separando o código em arquivos diferentes.

```
include "cliente.php";
include "entreador.php"; //warning
require "item.php";
require "pedid.php"; //erro
```

Funções

- Cria-se funções em PHP com a seguinte sintaxe:

```
function retornaEntrada($entrada) {
    return $entrada;
}
```

- Como em PHP a tipagem é dinâmica, não precisamos denotar o tipo do retorno ou se há algum retorno.

Funções

- Pode-se criar funções dentro de blocos condicionais.

```
if(isset($minhaString)) {
    function imprimeString($input) {
        echo “$input”;
    }
}
imprimeString($minhaString);
```

- No exemplo, se $minhaString contém algum valor, a função imprimeString será criada. Nesse caso podemos utilizar a função imprimeString. Se $minhaString não existir, o interpretador gerará um erro.

Parâmetros

- Por valor:

```
function mais5($numero) {
    $numero += 5;
}
//main
$a = 3;
mais5($a); //$a continua valendo 3
```

Parâmetros

- Por referência:

```
function mais5(&$num1, $num2) {
    $num1 += 5;
    $num2 += 5;
}
$a = $b = 1;
mais5($a, $b); //$num1 terá seu valor alterado
mais5($a, &$b); //Aqui as duas variáveis terão seus valores alterados.
```

Parâmetros

- Default:
  - Quando algum parâmetro é declarado desta maneira, a passagem do mesmo na chamada da função torna-se opcional.

```
function teste($a = “testando”) {
    echo $a;
}

teste(); //imprime “testando”

teste(“outro teste”); //imprime “outro teste”
```

  - É bom lembrar que quando a função tem mais de um parâmetro, o que tem valor default deve ser declarado por último.

Polimorfismo – ad-Hoc

- Sobrecarga – a linguagem não suporta porém é possível sobrecarregar alguns métodos e comandos já existentes.

```
func_get_args();
func_get_arg();
func_num_args();
__set($nome, $valor);
__get($nome);
__isset($nome);
__unset($nome);
public function sobrecarga() {
    var_dump(func_get_args());
}
```

Polimorfismo – ad-Hoc

- Coerção:

  O sistema de tipos converte argumentos automaticamente para o tipo esperado pela função, em situações que gerariam erro.

```
$soma = 1+"1"; // Somando uma string e um integer
//$soma é 2.
```

Polimorfismo - Universal

- Inclusão - um objeto pode ser visto como pertencendo a diferentes classes que não precisam ser disjuntas.
- O PHP adota esse tipo de polimorfismo, pois é uma linguagem orientada a objeto que permite recursos de herança.

Exceções

- throw, try e catch.
  - O código que pode ocasionar uma exceção deve ser inserido num bloco precedido da palavra chave try. Após o try deve ser colocado o catch que será executado quando ocorrer uma exceção.

Exceções

- Pode-se criar uma classe de exceção herdando da classe Exception, utilizando os métodos:
  - getMessage() - retorna a mensagem de erro.
  - getCode() - retorna o código do erro.
  - getFile() - retorna o caminho do arquivo onde a exceção foi gerada.
  - getLine() - retorna o número da linha onde a exceção foi gerada.

Exceções

```
try{
    if (!file_exists($argv[$indice])){
        throw new Exception($argv[$indice].": arquivo nao encontrado\n");
}
$fileEntregador = fopen($argv[$indice],"r");    //abre
}
catch (Exception $e){
    echo $e->getMessage();
    return;
}
```

Orientação a Objetos

- Propriedades e métodos:
  - Public: a propriedade ou método pode ser acessada por qualquer função.
  - Private: somente métodos desta classe podem utilizar o que é definido desta maneira.
  - Protected: somente métodos desta classe e de suas subclasses podem acessar.

OO – Classe Pessoa

```
class Pessoa {
    //propriedades
    protected $nome;
    protected $idade;
    //métodos
    public function __construct($nome, $idade){
        $this->nome = $nome;
        $this->idade = $idade;
        echo "Objeto Pessoa Construído<br>";
    }
    public function __destruct(){
        echo "Objeto Pessoa Destruído<br>";
}
```

OO – Classe Pessoa

```
public function __set($var, $valor){
    $this->$var = $valor;
}
public function __get($var){
    return $this->$var;
}
public function imprime(){
    echo "Nome: " . $this->nome . " Idade: " . $this->idade .
    "<br>"; }
}
```

OO – Aluno (Subclasse de Pessoa)

```
class Aluno extends Pessoa {
    protected $nota;
    public function __construct($nome, $idade, $nota){
        parent::__construct($nome, $idade);
        $this->nota = $nota;
        echo "Objeto Aluno Construído<br>";
    }
    public function __destruct(){
        echo "Objeto Aluno Destruído<br>";
    }
}
```

OO – Aluno (Subclasse de Pessoa)

```
public function __set($var, $valor){
    $this->$var = $valor;
}
public function imprime(){
    parent::imprime();
    echo "Nota: " . $this->nota . "<br>";
}
public function verificaSePassou($media = 7){
    return $this->nota >= $media ? "Parabens! Passou direto.<br>" :
    "Ahh! Ficou de final.<br>";
}
```

OO - Instância

```
$p = new Pessoa("Jose", 30); // Nova pessoa
$p->nome = "Jorge"; // Altera nome
$p->imprime(); // Imprime
$a = new Aluno(“Maria", 20, 0); // Novo aluno
$a->nota = 7.5; // Altera nota
$a->imprime(); // Imprime
echo $a->verificaSePassou(); // Chama método
$a2 = new Aluno(“Joao", 20, 5.5); // Novo aluno
$a2->imprime(); // Imprime
echo $a2->verificaSePassou(6); //Chama método
```

OO - Saída

Nome: Jorge Idade: 30

Nota: 7.5 // Nota da Maria

Parabens! Passou direto.

Nota: 5.5 // Nota do João

Ahh! Ficou de final.

Concorrência

- Diferentemente do GO, PHP não é uma linguagem multithread, mas é possível simular paralelismo utilizando multiprocesso. Quando um processo pai cria um processo filho ambos os processos são executados concorrentemente. Isso é póssivel em PHP através do PHP Process Control Functions (PNCTL). Para isso o pacote PHP deve ser compilado com a opção --enable-pcntl, disponível somente em sistemas linux.
- Essas função não deveriam ser usadas dentro de um servidor web e resultados inesperados podem ser obtidos quando isso é feito.

Concorrência

- Esse é um exemplo simples de criação de um processo em PHP:

```
$pid = pcntl_fork();
if($pid) {
    //o que deve ser rodado no processo pai
    echo "pai";
}
else {
    // o que deve ser rodado no processo filho
    echo "filho";
}
```

Concorrência

- Essas são as principais funções para criação e manipulação de processos:

int pcntl_alarm (interno $segundos)

- Cria um temporizador que envia um sinal SIGALRM para um processo. Qualquer nova chamada a pcntl_alarm cancela a anterior.

void pcntl_exec (string $path [, array $args [, array $envs]])

- Executa um programa no espaço de mémoria do processo conrrente. O primeiro argumento é o caminho para o programa. O segundo é um vetor de argumentos passados ao programa.

int pcntl_fork (void)

Concorrência

- PHP não possui suporte “built-in” para multithreading. Multithread em fato é implementado pela biblioteca PCNTL que não é funcional em todos os sistemas, porem boa parte das instalações atuais de PHP disponibilizam multithread.
- Você pode criar um while para criar e gerenciar threads, mas cuidado com as limitações de hardware, lembre-se que todos os limites são únicos para cada processo, em outras palavras, se você possui 10M de limite de alocação de memória este será estendido ao thread, ou seja, digamos que você tenha um processo pai e dois filhos, seu aplicativo pode então alocar até 30M de memória.

Concorrência

- Multithread não é muito útil para aplicativos construídos para rodarem sobre o apache, não adianta pensar em utilizar o multicore do processador posto que outro thread do próprio apache já estará utilizando aquela parte do processamento, é como tentar rodar mais rápido o processo 1 tirando o processador do processo 2, o processo 2 também precisa ser finalizado, porem um bom exemplo de como utilizar multithread em php é o phpUnit, você pode construir uma suite e separar seu processamento em várias instâncias para agilizar a conclusão gravando em disco os resultados.

Avaliação da Linguagem

- Redigibilidade.
- Portabilidade.
- Eficiência.
- Confiabilidade.

Referências

- http://php.net/manual/pt_BR/
- http://pt.wikipedia.org/wiki/PHP
- http://www.programabrasil.org
- http://pt.wikibooks.org/wiki/
- F. M.Varejão. Linguagens de Programação – Conceitos e Técnicas. Campus, 2004.

Dúvidas?