Seminário PHP

Linguagens de Programação 2018/2

Breno Scalzer Coimbra

Bruno Frigeri Pirajá

Franco Schmidt Rossi

Gabriel Castro de Rezende

Rayane Nascimento

Rhuan Souza Caetano

Introdução

História

- A linguagem foi criada em 1994 por Rasmus Lerdof, utilizando um simples conjunto de binários escrito em linguagem de programação C.

- Inicialmente chamava-se PHP/FI(Personal Home Page/Forms Interpreter)

- Ao longo do tempo, mais funcionalidades foram desejadas, e Rasmus reescreveu o PHP Tools, produzindo implementação mais rica, sendo capaz de interações com Banco de Dados, e, fornecendo estrutura no qual usuários poderiam desenvolver formulários web.

- Logo depois, é liberada outra versão, dessa vez, a linguagem foi desenvolvida para, deliberadamente, ser parecida com C, tornando-a fácil para desenvolvedores habituados com a linguagem.

Atualmente

- PHP 7.
- Na maioria das vezes utilizado para desenvolvimento web mesclado ao código HTML.
- Suporte a inúmeros bancos de dados
- Seu propósito principal é de implementar soluções web velozes, simples e eficientes.

O que o PHP pode fazer?

- Basicamente, qualquer coisa. O PHP é focado principalmente nos scripts do lado do servidor (server-side). Então podemos: coletar dados de formulários, gerar páginas com conteúdo dinâmico ou enviar e receber cookies.

- Server-Side: é o principal campo de atuação do PHP. As informações são processadas por um servidor web que interpreta e retorna o resultado a ser exibido no browser.

- Scripts de linha de comando: é possível fazer um script PHP para executá-lo sem um servidor ou navegador. A única coisa necessária é o interpretador PHP.

Referência da Linguagem

Sintaxe Básica

- Quando o PHP interpreta um arquivo ele procura pelas tags de abertura e fechamento, **<?php** e **?>**, respectivamente. Caso um arquivo seja puramente PHP, é permitida a omissão da tag de fechamento.

```php
<?php
        echo "Hello World";
?>
```

- Separação de instruções.

```php
<?php
        echo "Hello World";
        eCHo "Hello World";
        ECHO "Hello World";
?>
```

- O PHP suporta comentários de algumas linguagens, como C, C++, Shell.

- Case sensitivity.

Sintaxe Básica

- Separação de instruções.

```
<?php
        echo “Hello World”;
?>

<?php echo “Hello World” ?>

<?php echo “Hello World”;
```

- Comentários

```
<?php
        echo “Hello World”;
    //Comentário utilizado em C

    /* Comentário
    de linhas múltiplas */

    #Comentário utilizado em Shell
?>
```

Variáveis

- As variáveis em PHP são representadas com um cifrão ($), seguido do nome da variável. Nomes de variáveis devem começar com letra ou sublinhado seguidos por letras, números ou sublinhados.

- Diferente de algumas linguagens, na linguagem PHP não é necessário a inicialização de variáveis, sendo a inicialização um padrão, como por exemplo, para uma variável do tipo inteiro temos zero como valor inicial e para uma variável do tipo string são definidas como vazia.

- Escopo das variáveis pode ser global, estático ou local.

Escopo de Variáveis

- O escopo de uma variável é o contexto onde foi definida.

```php
<?php
        $a = 1; //Global

        function Exemplo(){
                $b = 2; //Local
                static $c = 3; //Estática
        }
?>
```

Variáveis Globais, Estáticas e Locais

- Variáveis globais podem ser diretamente acessadas fora de funções ou dentro com o uso de global.

- A linguagem também guarda todas as variáveis globais dentro de uma array.

```php
<?php
        $a = 1;
        $b = 2;

        function Soma(){
        $GLOBALS['b'] = $GLOBALS['a'] +
$GLOBALS['b'];

        }
        Soma();
        echo $b;
?>
```

```php
<?php
        $in = 1;
        function Exemplo(){
                global $in;
                echo $in;
        }
        Exemplo();
?>
```

Variáveis Globais, Estáticas e Locais

- A maioria das variáveis do PHP tem somente escopo local. Este escopo local inclui os arquivos incluídos e requeridos.

- Uma variável estática existe somente no escopo local da função, mas não perde seu valor quando o nível de execução do programa deixa o escopo.

```php
<?php
        $a = 1;
        include 'b.inc';
?>
```

```php
<?php
        function Exemplo(){
                static $a;
                echo $a;
                $a++;
        }
        Exemplo();
?>
```

Palavras-chave

__halt_compiler()	abstract	and	array()	as
break	callable (a partir do PHP 5.4)	case	catch	class
clone	const	continue	declare	default
die()	do	echo	else	elseif
empty()	enddeclare	endfor	endforeach	endif
endswitch	endwhile	eval()	exit()	extends
final	finally (a partir do PHP 5.5)	for	foreach	function
global	goto (a partir do PHP 5.3)	if	implements	include
include_once	instanceof	insteadof (a partir do PHP 5.4)	interface	isset()
list()	namespace (a partir do PHP 5.3)	new	or	print
private	protected	public	require	require_once
return	static	switch	throw	trait (a partir do PHP 5.4)
try	unset()	use	var	while
xor	yield (a partir do PHP 5.5)			

Constantes

- As constantes globais são aproveitadas por todo script.
- Para se definir uma constante temos: define(nome, valor, case-insensitivity)

```php
<?php
	define(“nome”, “PHP”, true);

	function Exemplo(){
		echo nome;
	}

	Exemplo();
?>
```

Tipos

- PHP:
  - Tipagem dinâmica;
  - Fracamente tipada;
  - Copy-on-write;
  - Contagem de referências (coletor de lixo);
  - Não suporta definição de tipo explícita na declaração de variáveis;
  - Aceita conversão explícita de tipos;

Tipos

- Suporta oito tipos de dados primitivos divididos em três grupos:

1-Tipos básicos

- integer
- float/double
- string
- boolean

2- Tipos compostos

- array
- object

3- Tipos especiais

- resource
- NULL

Tipos

- Valores considerados false: 0, 0.0, NULL, string ou array vazios ou “0”.
- String:

```php
<?php
echo 'Hello,World!\n';
echo "Hello, World!\n";
?>
```

```php
<?php
$str = <<<EOD
Hello
World
EOD;
?>
```

Tipos

Tipo array:

- Pode ser tratado como um array, uma lista (vetor), hashtable, dicionário, coleção, pilha, fila dentre outros;
- Existe a possibilidade dos valores do array serem outros arrays;
- A chave é opcional, podendo ser um valor inteiro ou uma string.

```php
<?php
$a = array(1,2,3);
$b = array(1 => "Um",2 => "Dois",3 =>"Tres");
echo "$a[0]\n";
echo "$b[2]\n";
?>
```

Constantes Mágicas

- PHP tem nove constantes ‘mágicas’, que mudam dependendo de onde são utilizadas. Todas essas constantes são resolvidas em tempo de compilação, ao contrário das constantes regulares que são resolvidas em tempo de execução, e são case-insentitive.

Nome	Descrição
__LINE__	O número da linha corrente do arquivo.
__FILE__	O caminho completo e nome do arquivo com links simbólicos resolvidos. Se utilizado dentro de um include, o nome do arquivo incluído será retornado.
__DIR__	O diretório do arquivo. Se usado dentro de um include, o diretório do arquivo incluído é retornado. É equivalente a *dirname(__FILE__)*. O nome do diretório não possui barra no final, a não ser que seja o diretório raiz.
__FUNCTION__	O nome da função.
__CLASS__	O nome da classe. O nome da classe inclui o namespace em que foi declarado (por exemplo, *Foo\Bar*). Note que a partir do PHP 5.4, __CLASS__ também funcionará em traits. Quando utilizada em um método trait, __CLASS__ é o nome da classe que está utilizando a trait.
__TRAIT__	O nome do trait. O nome do trait inclui o namespace em que foi declarado (por exemplo, *Foo\Bar*).
__METHOD__	O nome do método da classe.
__NAMESPACE__	O nome do namespace corrente.

Armazenamento

Coletor de lixo

- PHP utiliza principalmente o método de contagem de referências;

- Se uma variável cair fora do escopo e não for mais usada em qualquer outro local do código atualmente executado, ela será coletada automaticamente. Você pode forçar isso mais cedo usando *unset()* para finalizar o escopo de variáveis antecipadamente;

Coletor de lixo

- Se uma variável é parte de uma referência cíclica (array), onde A aponta para B e B de volta para A, então a variável só pode ser limpa pelo coletor de lixo de ciclos do PHP. É acionado sempre que 10000 objetos ou matrizes cíclicas estão atualmente na memória e um deles está fora do escopo. O coletor é ativado por padrão em todas as solicitações, mas pode ser alternado com as funções *gc_enable()* e *gc_disable()*;.

- Se você chamar a função *gc_collect_cycles()*, então a coleta de referências cíclicas será acionada explicitamente, mesmo que você ainda não tenha 10000 deles na memória.

Serialização / Memória secundária

- Serialização de objetos é um conceito que se expande em quase todas as linguagens de programação, o principal objetivo desta é transformar um objeto em uma forma binária ou mesmo em formato de texto (como XML, por exemplo) para poder transmiti-lo via rede ou armazenar seu conteúdo sem perda de dados;

- PHP armazena em memória secundária através das funções se*rialize()* e *unserialize()*.

Expressões e comandos

Operadores

Precedência dos operadores

Associação	Operadores	Informação Adicional
não associativo	*clone new*	clone e new
esquerda	*[*	array()
direita	**	aritmética
direita	++ -- ~ (int) (float) (string) (array) (object) (bool) @	types e incremento/decremento
não associativo	*instanceof*	tipos
direita	!	lógicos
esquerda	* / %	aritmética
esquerda	+ - .	aritmética e string
esquerda	<< >>	bits
não associativo	< <= > >=	comparação
não associativo	== != === !== <> <=>	comparação
esquerda	&	bits e referências
esquerda	^	bits
esquerda	|	bits
esquerda	&&	lógicos
esquerda	||	lógicos
direita	??	comparação
esquerda	? :	ternário
direita	= += -= *= **= /= .= %= &= |= ^= <<= >>=	atribuição
esquerda	*and*	lógicos
esquerda	*xor*	lógicos
esquerda	*or*	lógicos

Operadores

Operadores Aritméticos

Exemplo	Nome	Resultado
+$a	Identidade	Conversão de *$a* para int ou float conforme apropriado.
-$a	Negação	Oposto de *$a*.
$a + $b	Adição	Soma de *$a* e *$b*.
$a - $b	Subtração	Diferença entre *$a* e *$b*.
$a * $b	Multiplicação	Produto de *$a* e *$b*.
$a / $b	Divisão	Quociente de *$a* e *$b*.
$a % $b	Módulo	Resto de *$a* dividido por *$b*.
$a ** $b	Exponencial	Resultado de *$a* elevado a *$b*. Introduzido no PHP 5.6.

Operadores

Além do operador básico de atribuição ($a = 3;), temos alguns operadores de atribuição úteis durante o desenvolvimento:

```
$a += $b      $a = $a + $b
$a -= $b      $a = $a - $b
$a *= $b      $a = $a * $b
$a /= $b      $a = $a / $b
$a %= $b      $a = $a % $b
```

Operadores

Operadores Bit-a-bit

Exemplo	Nome	Resultado
$a & $b	E (AND)	Os bits que estão ativos tanto em *$a* quanto em *$b* são ativados.
$a | $b	OU (OR inclusivo)	Os bits que estão ativos em *$a* ou em $b são ativados.
$a ^ $b	XOR (OR exclusivo)	Os bits que estão ativos em *$a* ou em *$b*, mas não em ambos, são ativados.
~ $a	NÃO (NOT)	Os bits que estão ativos em *$a* não são ativados, e vice-versa.
$a << $b	Deslocamento à esquerda	Desloca os bits de *$a* $b passos para a esquerda (cada passo significa "multiplica por dois")
$a >> $b	Deslocamento à direita	Desloca os bits de *$a* $b passos para a direita (cada passo significa "divide por dois")

Operadores

Assim como nos operadores aritméticos, podemos combinar os operadores bit-a-bit com o operador de atribuição:

```
$a &= $b      $a = $a & $b
$a |= $b      $a = $a | $b
$a ^= $b      $a = $a ^ $b
$a <<= $b      $a = $a << $b
$a >>= $b      $a = $a >> $b
```

Operadores

Operadores de comparação

Exemplo	Nome	Resultado
$a == $b	Igual	Verdadeiro (**TRUE**) se $a é igual a $b.
$a === $b	Idêntico	Verdadeiro (**TRUE**) se $a é igual a $b, e eles são do mesmo tipo.
$a != $b	Diferente	Verdadeiro se $a não é igual a $b.
$a <> $b	Diferente	Verdadeiro se $a não é igual a $b.
$a !== $b	Não idêntico	Verdadeiro de $a não é igual a $b, ou eles não são do mesmo tipo (introduzido no PHP4).
$a < $b	Menor que	Verdadeiro se $a é estritamente menor que $b.
$a > $b	Maior que	Verdadeiro se $a é estritamente maior que $b.
$a <= $b	Menor ou igual	Verdadeiro se $a é menor ou igual a $b.
$a >= $b	Maior ou igual	Verdadeiro se $a é maior ou igual a $b.
$a <=> $b	Spaceship (nave espacial)	Um integer menor que, igual a ou maior que zero quando *$a* é, respectivamente, menor que, igual a ou maior que *$b*. Disponível a partir do PHP 7.

Operadores

O php suporta, também, um operador de execução, representado por acentos graves (``):

```php
<?php
$output = `ls -al`;
echo "<pre>$output</pre>";
?>
```

O PHP tentará executar o conteúdo dentro dos acentos graves como um comando do shell.

Operadores

Operadores de Incremento/Decremento

Exemplo	Nome	Efeito
++$a	Pré-incremento	Incrementa $a em um, e então retorna $a.
$a++	Pós-incremento	Retorna $a, e então incrementa $a em um.
--$a	Pré-decremento	Decrementa $a em um, e então retorna $a.
$a--	Pós-decremento	Retorna $a, e então decrementa $a em um.

Operadores Lógicos

Exemplo	Nome	Resultado
$a and $b	E	Verdadeiro (**TRUE**) se tanto $a quanto $b são verdadeiros.
$a or $b	OU	Verdadeiro se $a ou $b são verdadeiros.
$a xor $b	XOR	Verdadeiro se $a ou $b são verdadeiros, mas não ambos.
! $a	NÃO	Verdadeiro se $a não é verdadeiro.
$a && $b	E	Verdadeiro se tanto $a quanto $b são verdadeiros.
$a || $b	OU	Verdadeiro se $a ou $b são verdadeiros.

Operadores

Operadores de Incremento/Decremento

Exemplo	Nome	Efeito
++$a	Pré-incremento	Incrementa $a em um, e então retorna $a.
$a++	Pós-incremento	Retorna $a, e então incrementa $a em um.
--$a	Pré-decremento	Decrementa $a em um, e então retorna $a.
$a--	Pós-decremento	Retorna $a, e então decrementa $a em um.

Operadores Lógicos

Exemplo	Nome	Resultado
$a and $b	E	Verdadeiro (**TRUE**) se tanto $a quanto $b são verdadeiros.
$a or $b	OU	Verdadeiro se $a ou $b são verdadeiros.
$a xor $b	XOR	Verdadeiro se $a ou $b são verdadeiros, mas não ambos.
! $a	NÃO	Verdadeiro se $a não é verdadeiro.
$a && $b	E	Verdadeiro se tanto $a quanto $b são verdadeiros.
$a || $b	OU	Verdadeiro se $a ou $b são verdadeiros.

Estruturas de controle

IF / ELSE

- Executa um bloco de código a partir de uma condição.

SWITCH

- Compara uma variável com várias condições e toma decisões para cada uma delas.

WHILE

- Executa o mesmo bloco de código enquanto uma condição for verdadeira.

```php
<?php

$a = 2;
$b = 5;

if ($a == $b) {
  /* bloco de código a ser executado */
}

switch ($a) {
  case 2:
    /* bloco de código a ser executado caso $a seja 2*/
    break;
  case 3:
    /* bloco de código a ser executado caso $a seja 3*/
    break;
  default:
    /* bloco de código a ser executado caso $a seja
diferente de 2 e de 3*/
    break;
}

while ($a <= $b) {
  $a++;
}

?>
```

Estruturas de controle

DO WHILE

- Executa o mesmo bloco de código enquanto uma condição for verdadeira, porém, diferente do while, a verificação acontece ao final do bloco.

FOR

- Executa o mesmo bloco de código enquanto uma condição for verdadeira, porém, diferente do while, inicializa uma variável e executa um código ao final de cada iteração.

```php
<?php

$a = 2;
$b = 5;

do {
  $a++;
} while ($a <= $b)

for ($a = 0; $a < $b; $a++) {
  /* bloco de código a ser executado*/
}

for ($a = 0; $a < 10; $a++) :
  /* bloco de código a ser executado*/
endfor;


?>
```

Estruturas de controle

FOREACH

- Executa um bloco de código para cada item de um array, utilizando uma variável referência.

CONTINUE

- Faz o loop ignorar a iteração atual, passando para a próxima.

BREAK

- Termina a execução do loop atual.

GOTO

- Desvio incondicional, uma má prática de programação.

```php
<?php

$a = 1;
$array = array(1, 2, 3);

foreach ($array as &$item) {
  $item++;
}


while ($a > 0) {
  $a++;
  if ($a < 10) {
    continue;
  }
  break;
}

$a = 1;

while ($a > 0) {
  $a++;
  if ($a == 10) {
    goto end;
  }
}

end :
echo 'end';

?>
```

Modularização

Subprogramas e parâmetros

- Funções podem conter qualquer código válido PHP (incluindo outras funções, ou classes)
- Identificadores de funções seguem as regras de nomeação normal de PHP
  - Porém, não são “case-sensitive”
- Podem ser referenciadas antes de serem definidas, exceto quando forem definidas *condicionalmente*
  - Assim como as classes, têm escopo global, mesmo quando definidas condicionalmente
- PHP não suporta *overloading* de funções
- Podem ser recursivas
- Podem ter número variável de argumentos, tanto quanto argumentos *default*

```php
<?php

$definirfoo = true;

/* foo () ainda nao pode ser chamada, porque nao
existe ainda; porem, bar () pode*/

bar();

if ($definirfoo) {
  function foo()
  {
    echo "Nao existe ate a excecucao do programa\n";
  }
}

/* foo () agora e valida, porque $definirfoo == true
*/

if ($definirfoo) foo();

function bar()
{
  echo "Existe imeadiatamente apos o inicio do
programa\n";
}

?>
```

Exemplo de definição condicional

Subprogramas e parâmetros

- Argumentos podem ser passados por cópia ou por referência
  - Argumentos com valores *default* devem ficar depois de todos os outros argumentos
  - Objetos são passados por referência
- São avaliados com correspondência posicional
- É permitida a declaração de tipos em argumentos de funções.
  - Levanta exceção caso o argumento real não seja do tipo declarado

```php
<?php

$x = 1;

echo "$x\n";        /* $x = 1, imprime: 1 */
bar($x);  /* $x agora vale 2, imprime: 2 */
bar($x, 3);         /* $x agora vale 3,
imprime: 3 */

/* bar($x, "1") */
/* o codigo no comentario acima levantaria
uma excecao do tipo "TypeError" */

function bar(&$y, int $def = 2)
{
  $y = $def;
  echo "$y\n";
}



?>
```

Exemplo de passagem por referência e de argumento com valor *default*

Tipos abstratos de dados

- Os tipos abstratos de dados implementáveis são as classes
- Identificadores de classes seguem as regras de nomeação normal de PHP
- A variável "*$this*", dentro de um método, é referência ao objeto chamador do método.
  - Indefinida fora de contexto de objeto
- Sintaxe parecida com classe e objetos no monte, em C++.
  - Usa-se "*new*" para criar uma instância de classe, "->" para chamar um método a partir de um objeto e "::" a partir da classe
- O operador "*new*" também cria instâncias de classes cujos nome estão dentro de uma string

```php
<?php
class A {
    public $var = 'um valor default';
    public function foo() {
        if (isset($this)) {
            echo '$this esta definida (';
            echo get_class($this);
            echo ")\n";
        } else {
            echo "\$this nao esta definida.\n";
        }
    }
}
class B extends A {
    public function bar() {
        A::foo();
    }
}

$a = new A();
$a->foo();                  /* $this esta definida (A) */

A::foo();                   /* $this nao esta definida. */

$bclass = "B";
$b = new $bclass();
$b->bar();                  /* $this nao esta definida. */

B::bar();                   /* $this nao esta definida. */
?>
```

Exemplos de construção de classe, herança, instanciação a partir de string e demonstração de casos em que "*$this*" está ou não definida.

Tipos abstratos de dados

- PHP não suporta herança múltipla
- A visibilidade de um método ou atributo de uma classe pode ser definido como “*public*”, “*protected*” e “*private*”, e tem o mesmo significado que essas palavras chave em C++
  - Caso a visibilidade não tenha sido declarada, o membro é considerado público
- É possível definir uma classe como abstrata declarando um ou mais métodos da classe como abstratos (palavra chave “*abstract*”)
- *Traits* equivalem à *interface* de Java, e é a maneira utilizada para contornar a ausência de herança múltipla da linguagem.

```php
<?php
class ClasseBase {
    abstract public function foo();
}

trait Interface {
    public function bar() {
        parent::foo();
        echo 'bar';
    }
}

class FooAndBar extends ClasseBase {
    use Interface;
    public function foo() {
        echo 'foo & ';
    }
}

$o = new FooAndBar();
$o->bar();          /* foo & bar */
?>
```

Exemplos de palavras chave de visibilidade, implementação de *Trait*, e de classe abstrata.

Pacotes e espaços de nome

- *Namespaces* são declarados utilizando a palavra chave *namespace*
  - Afeta apenas classes, *traits*, funções e constantes
  - Os *namespaces* “PHP” e “php” são ambos reservados.

```php
/* MinhaClasse.php */
<?php
namespace meu\nome;

class MinhaClasse {
    public function foo() {
        echo 'foo';
    }
}
?>

/* outro arquivo */
<?php
require_once 'MinhaClasse.php'

$a = new meu\nome\MinhaClasse ();
$a-> foo();                  /* foo */
?>
```

Exemplo de sintaxe de *namespace*

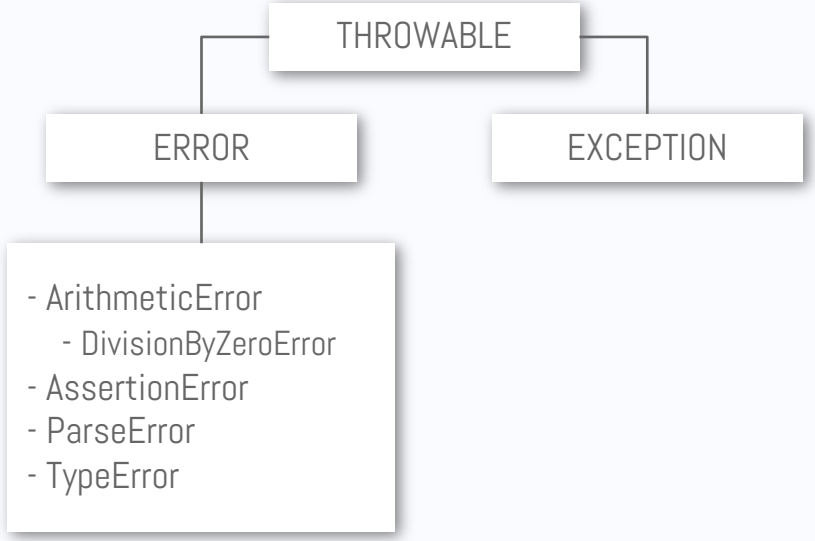
Compilação separada

- Não é possível compilar código PHP separadamente (pelo menos a linguagem não foi feita para isso).

Exceções

Exceções

- Error e Exception
- Erro tratado como exceção
- catch (Exception $e)



Exceções

- Envolvidas pelo bloco try
- Podem ser lançadas (throw) e capturadas (catch). Finally
- Múltiplos blocos catch, podendo ser relançadas
- Bloco seguinte do try não é executado
- Função *set_exception_handler()*
- *“Fatal error: Uncaught exception ‘Exception’”*
- Exceções do usuário devem estender a classe Exception

```php
<?php
class customException extends Exception {
  public function errorMessage() {
    $errorMsg = 'Error';
    return $errorMsg;
  }
}
$email = "someone@example...com";
try {
  if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE) {
    throw new customException($email);
  }
}
catch (customException $e) {
  echo $e->errorMessage();
}
?>
```

Funcionalidades

Sessão

- Armazenar informações a curto prazo.
- Inicia-se com session_start(), termina com session_destroy();
- Em geral dura até o navegador ser fechado.

```php
<?php
  // Começa a sessao
  session_start();
  $_SESSION["favcolor"] = "green";
?>


// Outra pagina
<?php
  session_start();
  // Variavel que foi iniciada em outra
pagina
  echo "Favorite color is " .
$_SESSION["favcolor"] . ".<br>";
  session_destroy();
?>
```

Cookie

- Semelhante a sessão
- Inicia-se com setcookie(), termina com unset() ou após expirar o tempo
- Permanece após navegador ser fechado, pelo tempo determinado

```php
<?php
  setcookie("name","value",time()+$int);
  echo $_COOKIE["your cookie name"];
?>
```

POST

- Método HTTP Post / GET

```
<form action="pagina.php"
method="post">
  Name: <input type="text"
name="name"><br>
  E-mail: <input type="text"
name="email"><br>
  <input type="submit">
</form>

//Arquivo pagina.php
<?php
  echo $_POST["email"];
?>
```

Banco de Dados

- Extensões para conexão com bancos de dado
- MySQL (mysqli), PostgreSQL

```php
<?php
  $mysqli = new mysqli("localhost",
"my_user", "my_password", "world");
  $mysqli->query("CREATE TEMPORARY TABLE
myCity LIKE City") === TRUE)
  $result = $mysqli->query("SELECT Name
FROM City LIMIT 10")
  printf("Select returned %d rows.\n",
$result->num_rows);

  $result->close();
  $mysqli->close();
?>
```

Concorrência

- Não é nativo da linguagem.
- O multithreading em PHP pode ser feito através da API Pthreads.
  - Para seu uso seguro, ela deve ser usada a partir da versão 7.2 do PHP.
  - Ela não pode ser usada no contexto de um servidor web.
- O gerenciamento de processos e semáforos pode ser efetuado através de extensão, porém, não estão disponíveis para distribuições Windows.

Concorrência

```php
<?php
class AsyncOperation extends Thread {

    public function __construct($arg) {
        $this->arg = $arg;
    }

    public function run() {
        if ($this->arg) {
            $sleep = mt_rand(1, 10);
            printf(' %s  -start -sleeps %d' . "\n", $this->arg, $sleep);
            sleep($sleep);
            printf(' %s  -finish' . "\n", $this->arg);
        }
    }
}

// Cria uma array
$stack = array();

//Inicializa Multiplas Threads
foreach ( range("A", "D") as $i ) {
    $stack[] = new AsyncOperation($i);
}

// Inicia Threads
foreach ( $stack as $t ) {
    $t->start();
}
```

```
A  -start -sleeps 5
B  -start -sleeps 3
C  -start -sleeps 10
D  -start -sleeps 2
D  -finish
B  -finish
A  -finish
C  -finish
```

Avaliação

Critérios gerais	C	C++	Java	PHP
Aplicabilidade	**Sim**	**Sim**	**Parcial**	**Parcial**
Confiabilidade	Não			
Aprendizado	Não			
Eficiência	Sim	Sim	Parcial	Sim
Portabilidade	Não	Não	Sim	Sim

Apesar de ser de propósito geral é voltado para aplicações WEB.

Avaliação

Critérios gerais	C	C++	Java	PHP
Aplicabilidade	Sim	Sim	Parcial	Parcial
Confiabilidade	**Não**	**Não**	**Sim**	**Parcial**
Aprendizado	Não			
Eficiência	Sim			
Portabilidade	Não	Não	Sim	Sim

Assim como Java, PHP possui coletor de lixo. Porém existem questões como incompatibilidade entre versões e tipagem fraca.

Avaliação

Critérios gerais	C	C++	Java	PHP
Aplicabilidade	Sim	Sim	Parcial	Parcial
Confiabilidade	Não	Não	Sim	Parcial
Aprendizado	**Não**	**Não**	**Não**	**Não**
Eficiência	Sim			
Portabilidade	Não			

Apresenta diversos recursos além de diferentes paradigmas.

Avaliação

Critérios gerais	C	C++	Java	PHP
Aplicabilidade	Sim	Sim	Parcial	Parcial
Confiabilidade	Não			
Aprendizado	Não			
Eficiência	**Sim**	**Sim**	**Parcial**	**Sim**
Portabilidade	Não	Não	Sim	Sim

Permite o programador decidir se o Coletor de Lixo fica ativado, não tem controle de índice de vetor.

Avaliação

Critérios gerais	C	C++	Java	PHP
Aplicabilidade	Sim	Sim	Parcial	Parcial
Confiabilidade	Não	Não	Sim	Parcial
Aprendizado	Não	Não	Não	Não
Eficiência	Sim			
Portabilidade	**Não**	**Não**	**Sim**	**Sim**

Característica fundamental da LP.

Avaliação

Critérios gerais	C	C++	Java	PHP
Método de Projeto	**Estruturado**	**Estruturado e OO**	**OO**	**Estruturado, OO e Funcional**
Evolutibilidade	Não	Parcial	Sim	Parcial
Reusabilidade	Parcial	Sim	Sim	Sim
Integração	Sim	Sim	Parcial	Sim
Custo	Depende da ferramenta	Depende da ferramenta	Depende da ferramenta	Depende da ferramenta

Avaliação

Critérios gerais	C	C++	Java	PHP
Método de Projeto	Estruturado	Estruturado e OO	OO	Estruturado, OO e Funcional
Evolutibilidade	**Não**	**Parcial**	**Sim**	**Parcial**
Reusabilidade	Parcial			
Integração	Sim			
Custo	Depende da ferramenta	Depende da ferramenta	Depende da ferramenta	Depende da ferramenta

Possui características que atrapalham a legibilidade, porém melhora esse aspecto utilizando a orientação a objeto.

Avaliação

Critérios gerais	C	C++	Java	PHP
Método de Projeto	Estruturado	Estruturado e OO	OO	Estruturado, OO e Funcional
Evolutibilidade	Não	Parcial	Sim	Parcial
Reusabilidade	**Parcial**	**Sim**	**Sim**	**Sim**
Integração	Sim			
Custo	Depende da ferramenta	ferramenta	ferramenta	ferramenta

Oferece conceito de classes, polimorfismo universal e a utilização e criação de frameworks.

Avaliação

Critérios gerais	C	C++	Java	PHP
Método de Projeto	Estruturado	Estruturado e OO	OO	Estruturado, OO e Funcional
Evolutibilidade	Não			
Reusabilidade	Parcial			
Integração	**Sim**	**Sim**	**Parcial**	**Sim**
Custo	Depende da ferramenta	Depende da ferramenta	Depende da ferramenta	Depende da ferramenta

Pode invocar códigos compilados e recursos nativos para interagir com banco de dados.

Avaliação

Critérios gerais	C	C++	Java	PHP
Método de Projeto	Estruturado	Estruturado e OO	OO	Estruturado, OO e Funcional
Evolutibilidade	Não	Parcial	Sim	Parcial
Reusabilidade	Parcial			
Integração	Sim			
Custo	**Depende da ferramenta**	**Depende da ferramenta**	**Depende da ferramenta**	**Depende da ferramenta**

É de domínio público. Existem inúmeras ferramentas gratuitas e pagas. Depende da escolha da equipe de desenvolvimento.

Avaliação

Critérios gerais	C	C++	Java	PHP
Escopo	**Sim**	**Sim**	**Sim**	**Sim**
Expressões e Comandos	Sim			
Tipos primitivos e compostos	Sim			
Gerenciamento de memória	Programador	Programador	Sistema	Programador / Sistema
Persistência dos dados	Biblioteca de funções	Biblioteca de classes e funções	JDBC, biblioteca de classes, serialização	Biblioteca de classes e serialização

Requer a definição explícita de entidades, associando-as a um escopo de visibilidade.

Avaliação

Critérios gerais	C	C++	Java	PHP
Escopo	Sim	Sim	Sim	Sim
Expressões e Comandos	**Sim**	**Sim**	**Sim**	**Sim**
Tipos primitivos e compostos	Sim			
Gerenciamento de memória	Programador	Programador	Sistema	Programador / Sistema
Persistência dos dados	Biblioteca de funções	Biblioteca de classes e funções	JDBC, biblioteca de classes, serialização	Biblioteca de classes e serialização

Oferece uma ampla variedade de expressões e comandos.

Avaliação

Critérios gerais	C	C++	Java	PHP
Escopo	Sim	Sim	Sim	Sim
Expressões e Comandos	Sim	Sim	Sim	Sim
Tipos primitivos e compostos	**Sim**	**Sim**	**Sim**	**Sim**
Gerenciamento de memória	Programador			
Persistência dos dados	Biblioteca de funções	Biblioteca de classes e funções	SGBD, biblioteca de classes, serialização	Biblioteca de classes e serialização

Oferece uma variedade de tipos primitivos, entretanto PHP não oferece tipo enumerado.

Avaliação

Critérios gerais	C	C++	Java	PHP
Escopo	Sim	Sim	Sim	Sim
Expressões e Comandos	Sim	Sim	Sim	Sim
Tipos primitivos e compostos	Sim			
Gerenciamento de memória	**Programador**	**Programador**	**Sistema**	**Programador / Sistema**
Persistência dos dados	Biblioteca de funções	Biblioteca de classes e funções	JDBC, biblioteca de classes, serialização	Biblioteca de classes e serialização

O programador pode escolher habilitar ou não o Coletor de Lixo.

Avaliação

Critérios gerais	C	C++	Java	PHP
Escopo	Sim	Sim	Sim	Sim
Expressões e Comandos	Sim	Sim	Sim	Sim
Tipos primitivos e compostos	Sim	Sim	Sim	Sim
Gerenciamento de memória	Programador			
Persistência dos dados	**Biblioteca de funções**	**Biblioteca de classes e funções**	**JDBC, biblioteca de classes, serialização**	**Biblioteca de classes e serialização**

Possui serialização e interface para facil interação com banco de dados.

Avaliação

Critérios gerais	C	C++	Java	PHP
Passagem de parâmetros	Lista variável e por valor	Lista variável, default, por valor e por referência	Lista variável, por valor e por cópia de referência	Lista variável, por valor e cópia de referência
Encapsulamento e proteção	Parcial			
Sistema de tipos	Não			
Verificação de tipos	Estática	Estática / Dinâmica	Estática / Dinâmica	Dinâmica

Faz passagem por valor, cópia de referência e lista variável.

Avaliação

Critérios gerais	C	C++	Java	PHP
Passagem de parâmetros	Lista variável e por valor	Lista variável, default, por valor e por referência	Lista variável, por valor e por cópia de referência	Lista variável, por valor e cópia de referência
Encapsulamento e proteção	**Parcial**	**Sim**	**Sim**	**Sim**
Sistema de tipos	Não	Oferece mecanismo de classes e visibilidade.		
Verificação de tipos	Estática	Dinâmica	Dinâmica	

Avaliação

Critérios gerais	C	C++	Java	PHP
Passagem de parâmetros	Lista variável e por valor	Lista variável, default, por valor e	Lista variável, por valor e por cópia	Lista variável, por valor e cópia de
Encapsulamento e proteção	Parcial			
Sistema de tipos	**Não**	**Parcial**	**Sim**	**Não**
Verificação de tipos	Estática	Estática / Dinâmica	Estática / Dinâmica	Dinâmica

Usa inferência de tipos e violações só são detectadas em tempo de execução.

Avaliação

Critérios gerais	C	C++	Java	PHP
Passagem de parâmetros	Lista variável e por valor	Lista variável, default, por valor e por referência	Lista variável, por valor e por cópia de referência	Lista variável, por valor e cópia de referência
Encapsulamento e proteção	Parcial	Sim	Sim	Sim
Sistema de tipos	Não			
Verificação de tipos	**Estática**	**Estática / Dinâmica**	**Estática / Dinâmica**	**Dinâmica**

Faz verificação de tipos por meio de inferência em tempo de execução.

Avaliação

Critérios gerais	C	C++	Java	PHP
Polimorfismo	**Coerção e sobrecarga**	**Todos**	**Todos**	**Todos**
Exceções	Não			
Concorrência	Não (biblioteca de funções)	funções)		de funções para Threads)

Possui todos, porém não permite sobrecarga de funções.

Avaliação

Critérios gerais	C	C++	Java	PHP
Polimorfismo	Coerção e sobrecarga	Todos	Todos	Todos
Exceções	**Não**	**Parcial**	**Sim**	**Sim**
Concorrência	Não (biblioteca de funções)	Oferece um sistema de tratamento de exceções.		ões ra

Avaliação

Critérios gerais	C	C++	Java	PHP
Polimorfismo	Coerção e sobrecarga	Todos	Todos	Todos
Exceções	Não	Recursos são oferecidos através de Extensões.		
Concorrência	**Não (biblioteca de funções)**	**Não (biblioteca de funções)**	**Sim**	**Parcial(Extensões de funções para Threads)**

Referências Bibliográficas

- https://php.net/manual/pt_BR/
- https://www.w3schools.com/php/
- https://tideways.com/profiler/blog/
- https://inf.ufes.br/~vitorsouza/wp-content/uploads/academia-br-lp-slides/