



PHP – Uma visão geral

Nicholas Figueiredo
Marini Favero

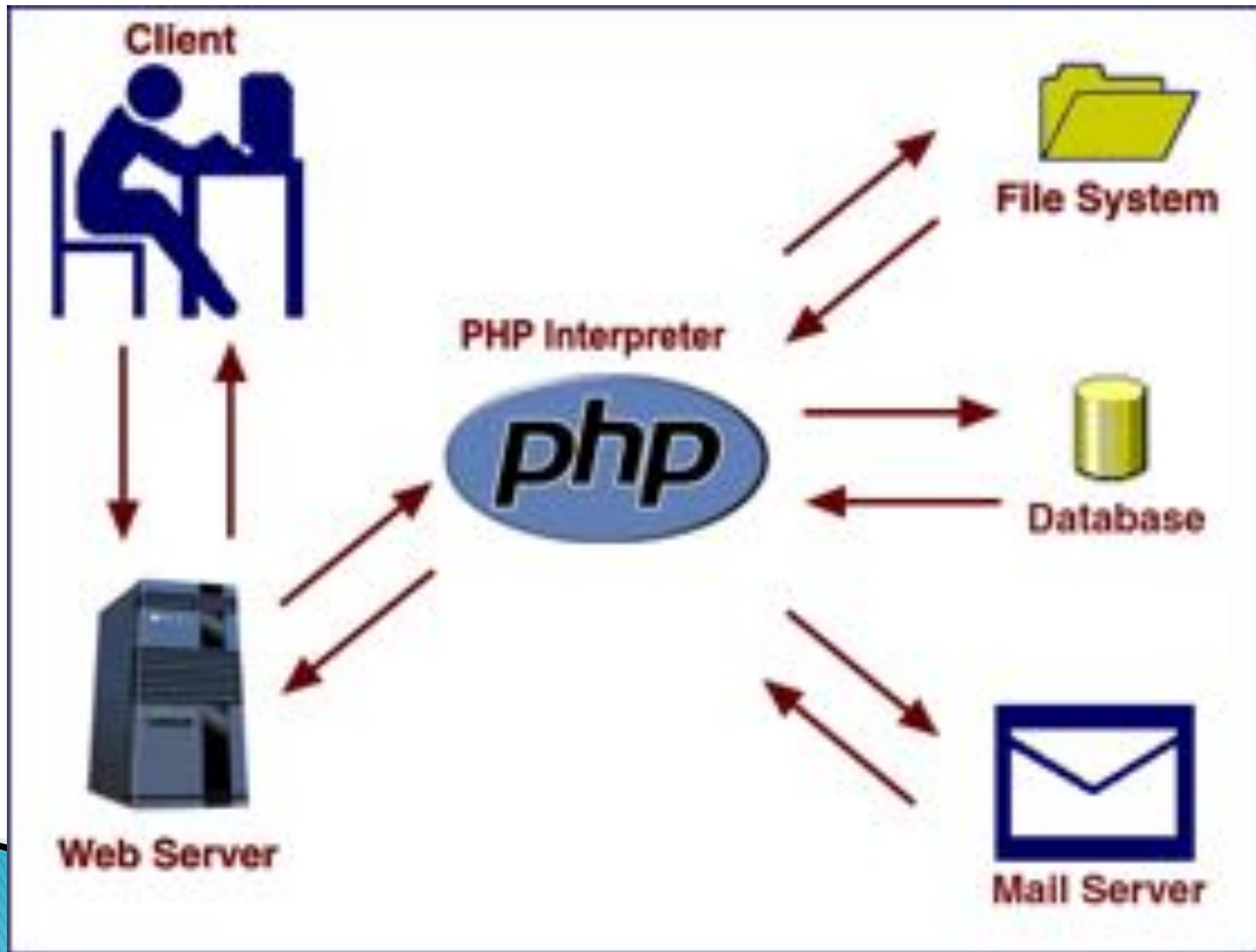
Introdução

- Criada em 1994 por Rasmus Lerdof
 - A primeira versão disponível foi em 1995 e ficou conhecida como “Personal Home Page Tools”
 - Era composta por um sistema que interpretava macros e utilitários que rodavam “por trás” das homepages.
- 

O que é PHP?

- É uma linguagem de script open source de uso geral.
 - Muito utilizada e especialmente guarnecida para o desenvolvimento de aplicações WEB embútil dentro do HTML.
 - É executado apenas no servidor, retornando apenas o HTML para o cliente.
- 

Como funciona?

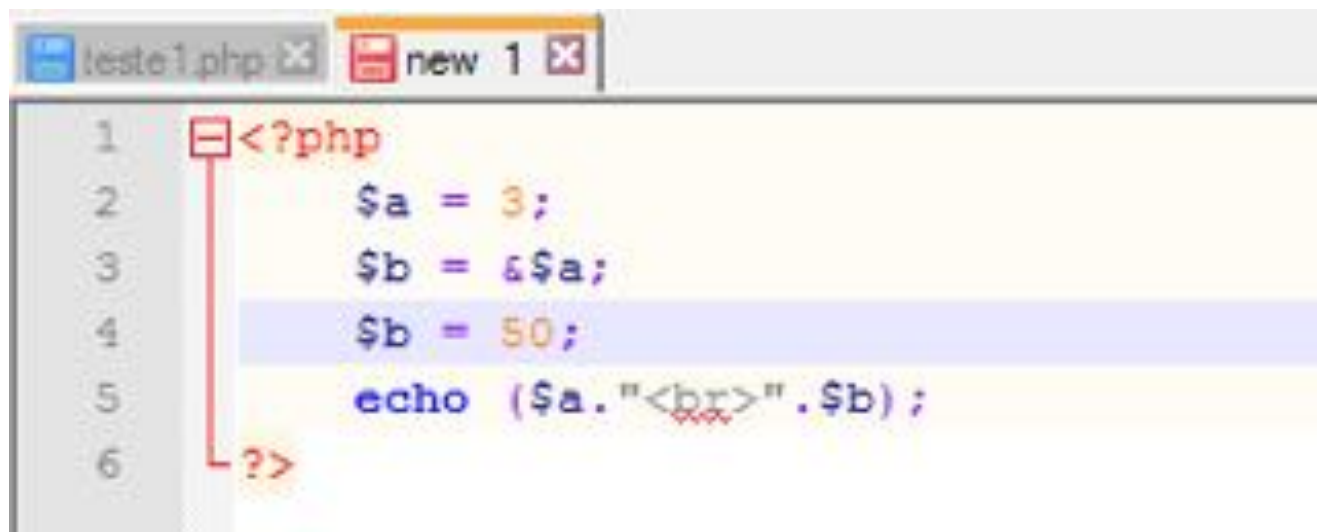


Propriedades da LP

- ▶ Legibilidade:
 - Boa, mas permite “goto”.
- ▶ Redigibilidade:
 - Excelente:
 - $\$a = \text{“cliente”}$;
 - $\$\$a = 31$;
 - Imprimindo $\$cliente$ obtém-se 31.
- ▶ Portabilidade:
 - LP muito portátil.

Propriedades da LP

- ▶ Confiabilidade
 - Apresenta problemas:
 - Permite manipular memória:



```
1 <?php
2     $a = 3;
3     $b = $a;
4     $b = 50;
5     echo ($a."<br>".$b);
6 ?>
```

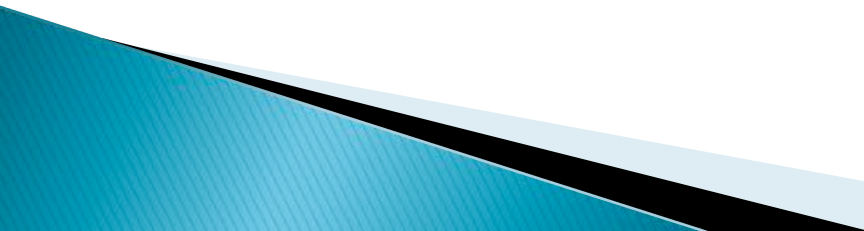
The image shows a screenshot of a code editor window with two tabs: 'teste1.php' and 'new 1'. The code is written in PHP and consists of six lines. Line 4, '\$b = 50;', is highlighted in blue. A red bracket on the left side of the editor spans from line 1 to line 6, indicating a block of code. The code defines two variables, \$a and \$b, and then echoes their values concatenated with a line break.

Propriedades da LP

- ▶ Não tem declaração de tipos

```
1 <?php
2     $a = 2;
3     $b = true;
4     echo("Oi, eu sou o Goku Super Saiyajin ".$a + $b);
5 ?>
6
```

Propriedades da LP

- ▶ PHP é uma linguagem interpretada.
 - ▶ Quanto ao paradigma PHP é estruturada e orientada a objetos.
 - ▶ Linguagem estruturada:
 - 1) Estruturas básicas de controle: sequência, condição e repetição.
 - 2) Subprogramação (ou modularização).
 - 3) Tipos Abstratos de Dados.
 - ▶ Tipagem dinâmica
- 

Amarrações

- ▶ Linguagem que permite orientação a objeto e polimorfismo.
- ▶ Amarração tardia.
- ▶ Problema com “self”, resolvido com late static binding.

```
1 <?php
2 class pessoa{
3     static protected $status = "status YAMA";
4
5     static public get$status(){
6         return self::$status; //mudar self para static
7     }
8 }
9 class zumbi {
10     static protected $status = "status YAMA";
11 }
12 echo zumbi::get$status().<br/>;
13
14
```

Identificadores

Palavras-chave do PHP

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


Identificadores

- ▶ PHP é case sensitive.
 - `$a != $A`.
- ▶ Escopo estático com blocos monolíticos.
- ▶ Declaração de constantes:
 - `define("F200", "alguma coisa"); //correto.`
 - Deve começar com uma letra ou sublinhado.
 - `define("2F00", "alguma coisa"); //errado.`

Identificadores

- ▶ Declaração de variável:
 - \$algumacoisa
 - Nomes de variáveis devem começar com letra ou sublinhado seguidos por letras, números ou sublinhados.

Valores e tipos de dados

- ▶ Tipagem dinâmica.
 - O interpretador infere o tipo.
 - ▶ PHP suporta 8 tipos primitivos:
- 

Valores e tipos de dados

São quatro tipos básicos:

- boolean
- integer
- float (número de ponto flutuante, ou também double)
- string

Dois tipos compostos:

- array
- object

E finalmente dois tipos especiais:

- resource
- NULL

Variáveis e constantes

- ▶ Escopo das variáveis:
 - A maior parte das variáveis do PHP tem somente escopo local. Este escopo local inclui os arquivos incluídos.

```
1  <?php
2      $a = 1;
3      include 'b.inc';
4  ?>
5
```

Variáveis e constantes

- ▶ Nova função, novo escopo local

```
1  <?php
2  $a = 1; //escopo global
3
4  function Teste()
5  {
6      echo $a; //erro, variavel nao definida
7  }
8
9  Teste();
10 ?>
```

- ▶ Solução:
 - global \$a;

Variáveis e constantes

- ▶ Variáveis estáticas:
 - Não perdem o valor quando a função terminar.

```
1 <?php
2 function Teste ()
3 {
4     $a = 0;
5     echo $a;
6     $a++; //nao faz sentido
7 }
8 ?>
```

Variáveis pré-definidas

PHP provê um grande número de variáveis pré-definidas para todos scripts. As variáveis representam tudo de variáveis externas à variáveis nativas de ambiente, última mensagem de erro à último cabeçalho recebido.

Veja também o FAQ intitulado "Como register_globals pode afetar?"

Índice

- Superglobais – Superglobais são variáveis nativas que estão sempre disponíveis em todos escopos
- \$_GLOBALS – Referencia todas variáveis disponíveis no escopo global
- \$_SERVER – Informação do servidor e ambiente de execução
- \$_GET – HTTP GET variáveis
- \$_POST – HTTP POST variables
- \$_FILES – HTTP File Upload variáveis
- \$_REQUEST – Variáveis de requisição HTTP
- \$_SESSION – Variáveis de sessão
- \$_ENV – Environment variables
- \$_COOKIE – HTTP Cookies
- \$php_errormsg – A mensagem de erro anterior
- \$_HTTP_RAW_POST_DATA – Informação não-tratada do POST
- \$http_response_header – Cabeçalhos de resposta HTTP
- \$argc – O número de argumentos passados para o script
- \$argv – Array de argumentos passados para o script

Variáveis superglobais

- ▶ Sempre disponíveis em todos escopos

- \$GLOBALS
- \$_SERVER
- \$_GET
- \$_POST
- \$_FILES
- \$_COOKIE
- \$_SESSION
- \$_REQUEST
- \$_ENV

Operadores

- Operadores Aritméticos
- Operadores de Atribuição
- Operador Bit-a-bit
- Operadores de Comparação
- Operadores de controle de erro
- Operadores de Execução
- Operadores de Incremento/Decremento
- Operadores Lógicos
- Operadores de String
- Operadores de Arrays
- Operadores de tipo

Operadores aritméticos

Operadores Aritméticos

Exemplo	Nome	Resultado
$-\$a$	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 $\$a$ por $\$b$.
$\$a \% \b	Módulo	Resto de $\$a$ dividido por $\$b$.

Operadores com atribuição

See the Arithmetic Operators page

(<http://www.php.net/manual/en/language.operators.arithmetic.php>)

Assignment	Same as:	
<code>\$a += \$b</code>	<code>\$a = \$a + \$b</code>	Addition
<code>\$a -= \$b</code>	<code>\$a = \$a - \$b</code>	Subtraction
<code>\$a *= \$b</code>	<code>\$a = \$a * \$b</code>	Multiplication
<code>\$a /= \$b</code>	<code>\$a = \$a / \$b</code>	Division
<code>\$a %= \$b</code>	<code>\$a = \$a % \$b</code>	Modulus

See the String Operators page(<http://www.php.net/manual/en/language.operators.string.php>)

<code>\$a .= \$b</code>	<code>\$a = \$a . \$b</code>	Concatenate
-------------------------	------------------------------	-------------

See the Bitwise Operators page (<http://www.php.net/manual/en/language.operators.bitwise.php>)

<code>\$a &= \$b</code>	<code>\$a = \$a & \$b</code>	Bitwise And
<code>\$a = \$b</code>	<code>\$a = \$a \$b</code>	Bitwise Or
<code>\$a ^= \$b</code>	<code>\$a = \$a ^ \$b</code>	Bitwise Xor
<code>\$a <<= \$b</code>	<code>\$a = \$a << \$b</code>	Left shift
<code>\$a >>= \$b</code>	<code>\$a = \$a >> \$b</code>	Right shift

Comandos

PRINCIPAIS COMANDOS PHP

<u>if</u>	<u>else</u>	<u>elseif</u>	<u>while</u>	<u>do while</u>
<u>for</u>	<u>foreach</u>	<u>break</u>	<u>continue</u>	<u>functions</u>
<u>date</u>	<u>strpos</u>	<u>str-replace</u>	<u>ltrim</u>	<u>rtrim</u>
<u>trim</u>	<u>substr</u>	<u>substr-replace</u>	<u>array</u>	<u>activex</u>

Comando break:

- Cancela a execução dos comandos for, foreach while, do..while ou switch.

Comando continue:

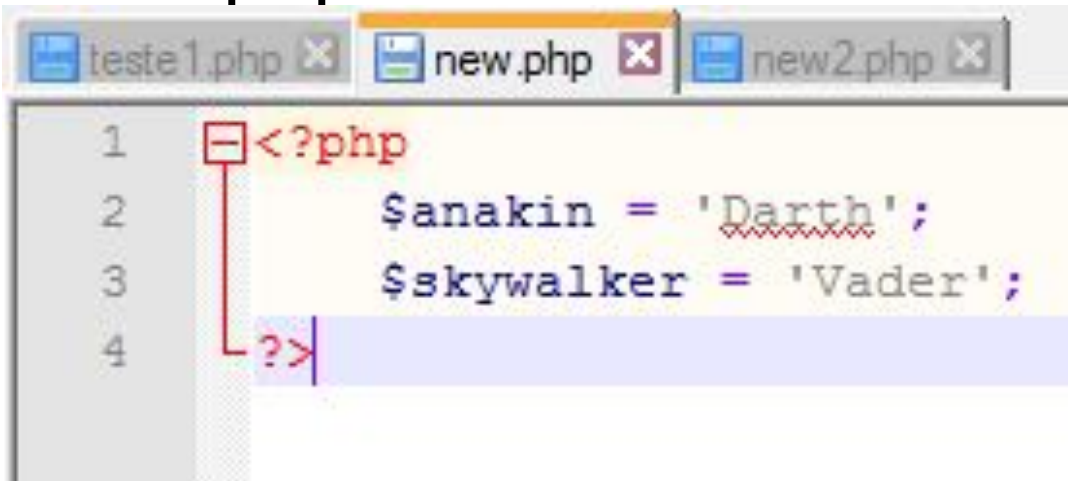
- Pula o resto da iteração atual e salta para o início da próxima iteração

Modularização

- ▶ Usuário pode criar funções.
- ▶ Um script pode “incluir” outro script e utilizar suas funções.
- ▶ Para incluir outro módulo:
 - `include.outro_modulo`
 - `require.outro_modulo`

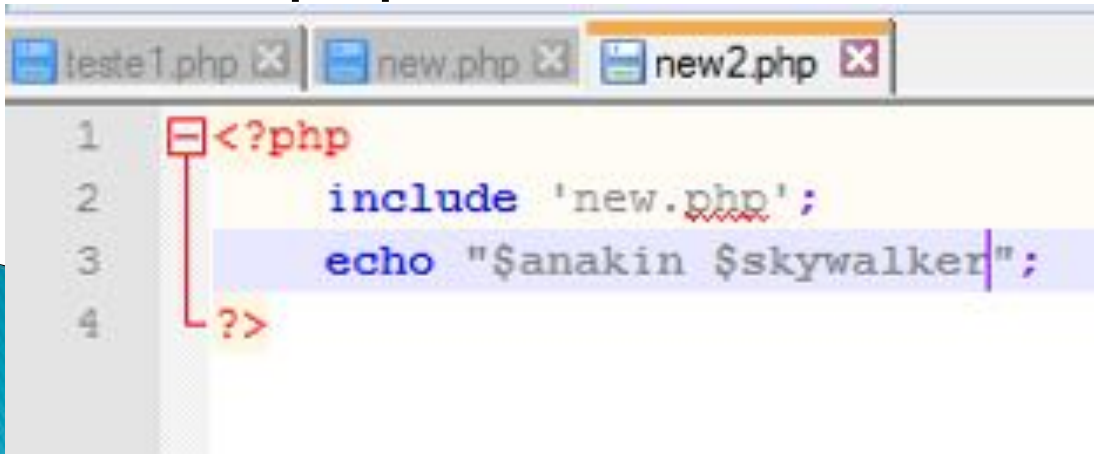
Exemplo include

▶ new.php



```
1 <?php
2     $anakin = 'Darth';
3     $skywalker = 'Vader';
4     ?>
```

▶ new2.php



```
1 <?php
2     include 'new.php';
3     echo "$anakin $skywalker";
4     ?>
```

Include/Require

- ▶ **Require:**
 - Tenta anexar o arquivo, se ocorrer erro aborta o script e gera uma mensagem de erro.
- ▶ **Include:**
 - Tenta anexar o arquivo, se ocorrer erro ele continua a execução do script e gera um warning.

Passagem de parâmetros

- ▶ Cópia ou referência.

```
1 <?php
2 function Duplica($parametro){
3     $parametro *= 2;
4 }
5 function Duplica_Altera(&$parametro){
6     $parametro *= 2;
7 }
8 $a = 3;
9 Duplica($a);
10 echo($a."<br>");
11 Duplica_Altera($a);
12 echo($a."<br>");
13 //Imprim 3 e 6.
14 ?>
```

Passagem de parâmetros

- ▶ Passagem por referência do tipo `passagem_de_parametro(&$a)` foi removida:

```
1 <?php
2     function Duplica($parametro){
3         $parametro *= 2;
4     }
5     function Duplica_Altera(&$parametro){
6         $parametro *= 2;
7     }
8     $a = 3;
9     Duplica($a);
10    echo($a."<br>");
11    Duplica_Altera($a);
12    echo($a."<br>");
13    //Imprime 3 e 6.
14 >
```

Passagem de parâmetros

- ▶ Atribuição default:

```
1 <?php
2     function Multiplica($parametro, $fator = 2) {
3         return $parametro * $fator;
4     }
5     $a = 3;
6     echo(Multiplica($a). "<br>");
7     echo(Multiplica($a, 4). "<br>");
8 ?>
```

Polimorfismo

- ▶ PHP não é uma linguagem fortemente tipada

```
1  <?php
2      $a = 3;
3      $b = "goiaba";
4      $c = "1";
5      echo ($a + $b."<br>");
6      echo ($a + $c."<br>");
7      echo ($b + $c);
8  ?>
```

- ▶ $\$a + \$b = 3$
- ▶ $\$a + \$c = 4$
- ▶ $\$b + \$c = 1$


Polimorfismo

- ▶ Ad-hoc
 - Coerção.
 - Sobrecarga.
- ▶ Inclusão:
 - Possui, por ser orientada a objetos.

Polimorfismo

```
1 <?php
2 class SimpleClass
3 {
4     // declaração de membro
5     public $var = 'um valor padrão';
6
7     // declaração de método
8     public function displayVar() {
9         echo $this->var;
10    }
11 }
12 ?>
13
14 <?php
15 class ExtendClass extends SimpleClass
16 {
17     // Redefine o método pai
18     function displayVar()
19     {
20         echo "Classe Herdeira\n";
21         parent::displayVar();
22     }
23 }
24
25 $extended = new ExtendClass();
26 $extended->displayVar();
27 ?>
```


Excessões

- ▶ Uma exceção pode ser disparada (thrown), ou ser pega (caught ou "caught").
 - ▶ Código pode ser rodeado em um block try, para facilitar a captura de exceções em potencial.
 - ▶ Cada bloco try, deve ter pelo menos um bloco catch correspondente.
- 

Excessões

- ▶ Pode-se criar uma classe de exceção herdando da classe Exception

```
<?php
class Exception {

    protected $message = 'Unknown exception'; // Mensagem da exceção
    protected $code = 0; // Código da exceção definido pelo usuário
    protected $file; // Arquivo gerador da exceção
    protected $line; // Linha geradora da exceção

    function __construct(string $message=NULL, int code=0);

    final function getMessage(); // Mensagem da exceção
    final function getCode(); // Código da exceção
    final function getFile(); // Arquivo gerador
    final function getTrace(); // um array com o backtrace()
    final function getTraceAsString(); // String formatada do trace

    /* Sobrecarregável */
    function _toString(); // String formatada para ser mostrada

}
?>
```

Disparo de exceção

```
<?php
function inverse($x) {
    if (!$x) {
        throw new Exception('Division by zero.');
```


throw new Exception('Division by zero.');

```
    }
    else return 1/$x;
}

try {
    echo inverse(5) . "\n";
    echo inverse(0) . "\n";
} catch (Exception $e) {
    echo "Exceção pega: ", $e->getMessage(), "\n";
}

// continua a execução
echo 'Hello World';
?>
```

Concorrência

- ▶ PHP tem suporte à threads
 - ▶ Modulo pthreads
 - ▶ A classe Thread permite a criação de uma thread simples.
 - ▶ Multithread não faz muito sentido em PHP, porque não se relaciona com sua aplicação.
- 

A classe Thread

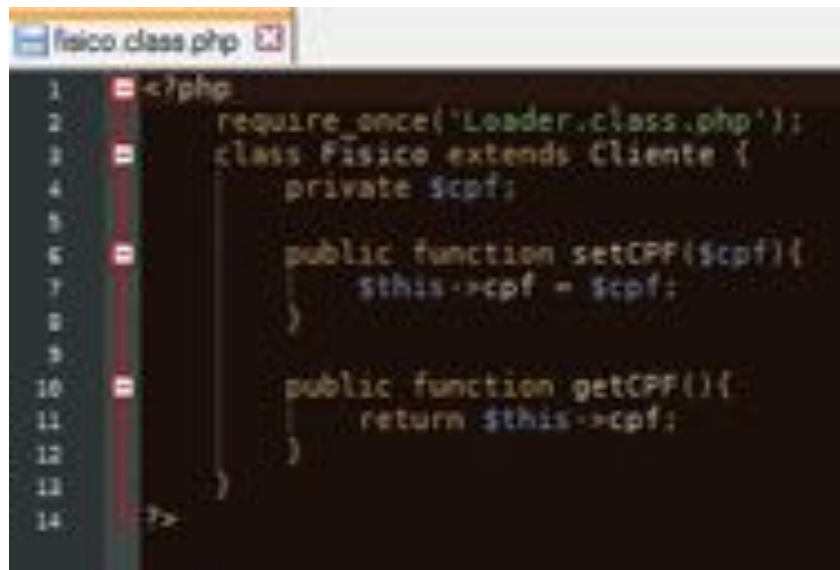
Thread {

/ Métodos */*

```
final public boolean chunk ( long $size , boolean $preserve )
final public long getCreatorId ( void )
final public static long getCurrentThreadId ( void )
final public long getThreadId ( void )
final public boolean isJoined ( void )
final public boolean isRunning ( void )
final public boolean isStarted ( void )
final public boolean isTerminated ( void )
final public boolean isWaiting ( void )
final public boolean join ( void )
final public boolean lock ( void )
final public boolean merge ( mixed $from [, mixed $overwrite ] )
final public boolean notify ( void )
final public boolean pop ( void )
abstract public void run ( void )
final public boolean shift ( void )
final public boolean start ( [ long $options ] )
final public mixed synchronized ( Closure $block [, mixed $... ] )
final public boolean unlock ( void )
final public boolean wait ( [ long $timeout ] )
```

Orientação a objeto

- ▶ A utilização de OO em PHP é bem simples.



```
1 <?php
2     require_once('Loader.class.php');
3     class Físico extends Cliente {
4         private $cpf;
5
6         public function setCPF($cpf){
7             $this->cpf = $cpf;
8         }
9
10        public function getCPF(){
11            return $this->cpf;
12        }
13    }
14
```

Orientação a objeto

- ▶ Quando é instanciado um novo objeto é sempre necessário fazer um 'include' do arquivo da classe aonde está sendo instanciado o objeto.
- ▶ No trabalho foi utilizado um auto load:

```
// Classes:  
require_once('classes/Loader.class.php');
```

```
$$classe = new Fisico();
```

```
<?php  
interface Loader {  
    spl_autoload_register('__autoload');  
    function __autoload($classe){  
        $filename = "./classes/".$classe.".class.php";  
        if(file_exists($filename)) require_once($filename);  
        else throw new Exception('Não foi possível carregar "' . $classe . '" via autoload');  
    }  
}  
?>
```


Como aprender PHP?

- ▶ Site com documentação completa sobre php:
- ▶ <http://php.net/>
- ▶ Site muito bom para referencias e como utilizar os scripts:
- ▶ <http://www.w3schools.com/>
- ▶ Servidor para testes / fácil de usar
- ▶ https://www.apachefriends.org/pt_br/index.html (Windows / Linux / os x)

Fim!



Dúvidas?