Introdução ao PHP

@dojomaringa

Apresentação

- Criada em 1994

Originalmente, criada apenas para suprir necessidades do seu criador, Ramus Lerdorf, no desenvolvimento de páginas pessoais.

- PHP4 lançada em 2000

Introduzida a Orientação a Objetos.

- PHP5 lançada em 2004

Reescrito toda a parte de Orientação a Objetos. Namespaces, garbage collector, funções anônimas e outras novidades.

Apresentação

> Tipagem **dinâmica** e fraca. Veloz e robusto. Estruturado e **orientado a objetos**. Pode ser executado em múltiplas plataformas. Sintaxe similar ao C/C++ e Perl e **open source**.

Wikipedia

Introdução ao PHP

1) Sintaxe básica

Um simples exemplo da sintaxe

2) Estruturas básicas

Tipagem, array, hash, condicionais, switch, for e foreach

3) Definição de classes e métodos

Paradigma estrutural e orientado a objetos

4) Testes unitários

SimpleTest

Sintaxe básica

```php
<?php

$string = “Hello World! \n”;

echo $string;

# Substitui World por Dojo Maringá
$string = str_replace(“World”, “Dojo Maringá”, $string);

echo $string;

// Adiciona “no CESUMAR”
$string = $string . “no CESUMAR \n”;

echo $string;

?>
```

Estruturas básicas

Tipagem

```php
<?php

$bar = “2”;
echo $bar . “/” . gettype($bar); # => 2/string

$bar = “2” + 1;
echo $bar . “/” . gettype($bar); # => 3/integer

$bar = 1;
echo $bar . “/” . gettype($bar); # => 1/integer

$bar = 1 . “2”;
echo $bar . “/” . gettype($bar); # => 12/string

?>
```

Estruturas básicas

Array e hash

```php
<?php

// Array
$bar = array(“John”, “Doe”);

?>
```

```php
/*
 * Array
 * (
 *     [0] => John
 *     [1] => Doe
 * )
 */
```

```php
<?php

// Hash
$bar = array(“firstname” => “John”, “lastname” => “Doe”);

?>
```

```php
/*
 * Array
 * (
 *     [firstname] => John
 *     [lastname] => Doe
 * )
 */
```

Estruturas básicas

Condicionais

```php
<?php

$foo = true;
$bar = false;

if ($foo) {
  echo “foo”;
}

if ($bar) {
  echo “bar”;
} else {
  echo “foo”;
}

?>
```

Estruturas básicas

Switch

```php
<?php

$bar = “John Doe”;

switch ($bar) {
  case “John Doe”:
    echo “Hey John!”;
    break;
  case true:
    echo “True story”;
    break;
 default:
    echo “Something”;
}

?>
```

Estruturas básicas

For e foreach

```php
<?php

$bar = array(“John”, “Mary”, “Doe”);

for ($i = 0; $i < count($bar); $i++) {
  echo $bar[$i];
}

foreach ($bar as $key => $value) {
  echo $key . “ => ” . $value;
}

?>
```

Classes e métodos

Paradigma estrutural

```php
<?php

function foo($a, $b) {
  return $a + $b;
}

$foo = foo(1, 2); # => 3

function bar($a, $b) {
  return $a - $b;
}

$bar = bar(2, 1); # => 1

?>
```

Classes e métodos

Paradigma orientado a objetos

```php
<?php

class Foo {
  private $name;

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

  public function name() {
    return $this->name;
  }
}

$foo = new Foo(“John Doe”);

echo $foo->name(); # => “John Doe”

?>
```

Testes unitários

SimpleTest

* Biblioteca elaborada
* Implementação **simplificada**
* Totalmente **prático**

SimpleTest

Testes unitários

SimpleTest

```
assertTrue($x)                  # Falha se $x for falso
assertFalse($x)                 # Falha se $x for verdadeiro
assertNull($x)                  # Falha se $x for definido
assertNotNull($x)               # Falha se $x não for definido
assertIsA($x, $t)               # Falha se $x não for uma classe ou tipo de $t
assertNotA($x, $t)              # Falha se $x for uma classe ou tipo de $t
assertEqual($x, $t)             # Falha se $x == $y for falso
assertNotEqual($x, $t)          # Falha se $x == $y for verdadeiro
assertWithinMargin($x, $y, $m)  # Falha se abs($x - $y) < $m for falso
assertOutsideMargin($x, $y, $m) # Falha se abs($x - $y) < $m for verdadeiro
assertIdentical($x, $y)         # Falha se $x == $y for falso ou de um tipo diferente
assertNotIdentical($x, $y)      # Falha se $x == $y for verdadeiro e tipos iguais
assertReference($x, $y)         # Falha a menos que $x e $y sejam a mesma variável
assertClone($x, $y)             # Falha a menos que $x e $y sejam cópias idênticas
assertPattern($p, $x)           # Falha a menos que a expressão reg. $p combine com $x
assertNoPattern($p, $y)         # Falha se a expressão reg. $p combine com $x
expectError($x)                 # Falha se o erro correspondente não ocorrer
expectException($x)             # Falha se a exceção não for lançada
ignoreException($x)             # Ignora qualquer exceção que seja lançada
assert($e)                      # Falha quando não atende as expectativas do objeto $e
```

Testes unitários

SimpleTest

```php
<?php

require_once("simpletest/autorun.php");

class SomeTests extends UnitTestCase {
  public function testOneAndOneMakesTwo() {
    $this->assertEqual(1 + 1, 2);
  }
}

?>
```

Testes unitários

SimpleTest

```php
<?php

require_once(“simpletest/autorun.php”);
require_once(“person.php”);

class PersonTests extends UnitTestCase {
  public $person;

  public function setUp() {
    $this->person = new Person();
  }

  public function testPersonCanSit() {
    $this->assertEqual($this->person->sit(), “Sit!”);
  }
}

?>
```

Perguntas?

Obrigado!

- @rogeriozambon
- /rogeriozambon