

PHP

<http://php.net/>

- É uma linguagem de scripting especialmente útil para gerar HTML
- É uma linguagem normalmente embebida num documento HTML
- É uma linguagem de scripting que corre no servidor: em condições normais o código PHP **nunca** sai do servidor!

Exemplo: HelloWorld.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01  
Transitional//EN"  
"http://www.w3.org/TR/html401/loose.dtd">  
<html>  
<head>  
<title> Hello World </title>  
<meta http-equiv="Content-Type" content="text/html;  
charset=iso-8859-1">  
</head>  
<body>  
    <?php  
        print "<p>Hello World!</p>" ;  
    ?>  
</body>  
</html>
```

PHP Opening and Closing Tags

```
<?php
```

```
echo 'if you want to serve XHTML  
or XML documents, do like this';
```

```
?>
```

```
<script language="php">
```

```
echo 'some editors (like  
FrontPage) don\'t like processing  
instructions';
```

```
</script>
```

```
<?= expression ?>
```

This is a shortcut for

```
<? echo expression ?>
```

Comentários

```
/* This is a multi-line comment  
    yet another line of comment  
*/
```

```
// This is a one-line comment
```

```
# This is a shell-style comment
```

echo, print, printf

- Echo pode imprimir simultaneamente vários tipos de dados

```
echo "Este stock tem" , 42, "items  
distintos" ;
```

- Print

```
print "Este stock tem 42 items  
distintos" ;
```

- printf

```
printf("Este %s %u %s distintos",  
"stock tem" , 42, "items") ;
```

Variáveis

```
$number = 45;
```

```
$vehicle = "autocarro";
```

```
$mensagem1 = "este $vehicle leva  
$number pessoas";
```

```
$mensagem2='este $vehicle leva  
$number pessoas';
```

```
print $mensagem1;
```

```
este autocarro leva 45 pessoas
```

```
print $mensagem2;
```

```
este $vehicle leva $number pessoas
```

Variáveis

- boolean

```
$foo = True;
```

- integer

```
$a = 1234;
```

```
$a = 0x1A;
```

- float

```
$b = 1.2e3;
```

- string

```
$c = "este texto vai aparecer em  
\n duas linhas";
```

```
$d = "este texto vai aparecer em "  
. "\n duas linhas";
```

```
$d = `este texto vai aparecer em `  
. ` \n uma linha`;
```

arrays

```
$numbers = array();  
$numbers = array(5,4,3,2,1);  
$numbers[5] = 0;
```

```
print $numbers[2];  
3
```

```
$words = array ("web", "database",  
"php");  
print $words[0];  
web
```

- arrays associativos

```
$comida = array('fruta' => 'pera',  
'vegetal' => 'batata');
```

```
$comida['peixe'] = "robalo" ;
```

```
print $comida['fruta'];  
pera
```


Variáveis Prédefinidas

- são arrays associativos que contêm todos os dados transmitidos pelo browser ao servidor web
- `$_SERVER` cabeçalhos HTTP que o browser envia
- `$_GET` variáveis de um formulário enviadas pelo método GET
- `$_POST` variáveis de um formulário enviadas pelo método POST
- `$_COOKIE` cookies enviados pelo browser
- `$_ENV` variáveis de ambiente
- `$_SESSION` variáveis de sessão

Estruturas de controle

- **if else**

```
if ($a > $b) {  
    echo "a is bigger than b";  
}  
else {  
    echo "a is NOT bigger than b";  
}
```

- **while**

```
$i = 1;  
while ($i <= 10) {  
    echo $i++;  
}
```

- **do while**

```
$i = 10;  
do {  
    echo $i;  
    $i = $i - 1;  
} while ($i > 0);
```

- **for**

```
for ($i = 1; $i <= 10; $i++) {  
    echo $i;  
}
```

foreach

- útil para extrair valores de arrays
- em arrays simples

```
$arr = array( "one", "two",  
"three" );
```

```
foreach ($arr as $value) {  
    echo "Value: $value<br />\n";  
}
```

- em arrays associativos

```
$a = array(  
    "one" => 1,  
    "two" => 2,  
    "three" => 3,  
    "seventeen" => 17  
);
```

```
foreach ($a as $k => $v) {  
    echo "$k => $v \n";  
}
```

Break e Continue

- **break**

```
$arr = array('one', 'two',  
'three', 'four', 'stop', 'five');  
while (list(, $val) = each($arr))  
{  
    if ($val == 'stop') {  
        break;  
    }  
    echo "$val<br />\n";  
}
```

- **continue**

```
while (list($key, $value) =  
each($arr)) {  
    if (!($key % 2)) { // skip odd  
members  
        continue;  
    }  
    do_something_odd($value);  
}
```

Switch

```
switch ($i) {  
    case "apple":  
        echo "it is apple";  
        break;  
    case "bar":  
        echo "it is bar";  
        break;  
    case "cake":  
        echo "it is cake";  
        break;  
    case default:  
        echo "it is not food";  
}
```

Require e Include

```
require 'prepend.php'; //fatal se  
não for encontrado
```

```
require_once("prepend.php");  
//inclui apenas uma vez
```

```
include("prepend.php"); //warning  
se não for encontrado
```

Funções

- **by value**

```
function add_some_extra($string)
{
    $string .= 'and something
extra.';
    return $string;
}
$str = 'This is a string, ';
extra_str=add_some_extra($str);
echo $str;
echo $extra_str;
```

- **by reference**

```
function add_some_extra(&$string)
{
    $string .= 'and something
extra.';
}
$str = 'This is a string, ';
add_some_extra($str);
echo $str;
```

Funções para arrays

```
$days = array("mon", "tue",  
"wed");  
print count($days)  
3
```

```
$pets= array("maria"=>"cao",  
"jose"=>"cao", "joao"=>"gato");  
$pet_frequency=array_count_values(  
$pets);  
print $pet_frequency["cao"];  
2
```

```
$var=array(10,5,37,42,1,-56);  
print max($var);  
print min($var);
```

```
$valor=19;  
if(in_array($valor, $var)) print  
"{ $valor } encontra-se no array  
\$var";
```


Funções para arrays (cont)

```
$owner="joao";  
if (array_key_exists($owner,  
$pets)) print "{$owner} tem um  
{$pets[$owner]} como bicho de  
estimação"  
  
print implode(array_key($pets), "  
");
```

maria, josé, joão

```
sort($var);  
rsort($var);
```

Funções para strings

```
print strlen("this is a string");  
//imprime 16
```

```
print str_pad("PHP", 6); //imprime  
PHP seguido de 3 espaços
```

```
strtolower("PHP and MySQL");  
strtoupper("PHP and MySQL");  
ucfirst("now is the time");  
ucwords("now is the time");
```

```
trim(" Tiger Land ");  
ltrim(" Tiger Land ");  
rtrim(" Tiger Land ");
```

```
strcmp("mouse", "mouse");  
// devolve 0  
strcmp("mouse", "Mouse");  
// devolve -1
```

```
$var="abcdefg";  
substr($var, 2, 3); //devolve cde  
substr($var, -5, 2); //devolve de
```

Funções para strings (cont)

```
$var="abcdefg";  
strpos($var, "d");  
3
```

```
print substr_replace($var, "DEF",  
2);  
//devolve abDEFfg
```

```
$var="old-age for the old";  
print str_replace("old", "new",  
$var);  
//devolve new-age for the new
```

```
$var="to be or not to be";  
$match=ereg('^to', $var); //true  
$match=ereg('be$', $var); //true
```

```
$pattern='(http://)?[a-zA-Z]+(\.[a-zA-Z]+)$';  
ereg($pattern, "www.google.com");  
//true
```

Datas e Tempo

```
print time()  
//prints timestamp e.g.,  
1064699133  
  
//create a timestamp 9:30  
18june1998  
mktime(9,30,0,6,18,1998);  
strtotime("18 june 1998");  
  
$var=mktime(9,30,0,6,18,1998);  
print date('d/m/Y', $var);  
//prints 18/6/1998  
print strftime("%D", $var);  
//C like; prints 06/18/1998  
  
setlocale(LC_TIME, 'pt');  
print date('I d F Y', time());  
//prints Terça 13 Março 2007
```

classes e objetos

```
class Cart {  
    var $items;    // Items in our shopping cart  
  
    // Add $num articles of $artnr to the cart  
  
        function add_item($artnr, $num) {  
            $this->items[$artnr] += $num;  
        }  
}  
  
$cart = new Cart;  
$cart->add_item("10", 1);  
  
$another_cart = new Cart;  
$another_cart->add_item("0815", 3);
```

Extended Classes

```
class Named_Cart extends Cart {  
    var $owner;  
  
    function set_owner ($name) {  
        $this->owner = $name;  
    }  
}
```

```
$ncart = new Named_Cart;    // Create a named cart  
$ncart->set_owner("kris");  // Name that cart  
print $ncart->owner;        // print the cart owners  
name  
$ncart->add_item("10", 1);  // (inherited  
functionality from cart)
```

Class Constructors

- Constructors são funções que são executadas quando se cria uma nova instância da classe.
- A função é uma constructor quando tem o mesmo nome da classe.

```
class Constructor_Cart extends Cart {  
    function Constructor_Cart($item = "10", $num = 1)  
{  
        $this->add_item ($item, $num);  
    }  
}
```

Class Constructors (cont.)

```
// Shop the same old boring stuff.  
$default_cart = new Constructor_Cart;  
  
// Shop for real...  
$different_cart = new Constructor_Cart("20", 17);
```


:: operator

- Utilizado para invocar funções e variáveis definidas em classes que não foram instanciadas

```
class A {  
    function example() {  
        echo "I am the original function .<br />\n";  
    }  
}
```

```
// there is no object of class A. this will print  
// I am the original function .<br />
```

```
A::example();
```