O Portal do conhecimento

http://apostilando.com

Curso PHP

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Curso de PHP

Apostila desenvolvida por Bruno Rodrigues Siqueira(bruno@netfly.com.br)

1. Introdução

O que é PHP?

PHP significa: *Hypertext Preprocessor*. Realmente, o produto foi originalmente chamado de “Personal Home Page Tools”; mas como se expandiu em escopo, um nome novo e mais apropriado foi escolhido por votação da comunidade. Você pode utilizar qualquer extensão que desejar para designar um arquivo PHP, mas os recomendados foram .php , .phtml. O PHP está atualmente na versão 4, chamado de PHP4 ou, simplesmente de PHP.

PHP é uma linguagem de criação de scripts embutida em HTML no servidor. Os produtos patenteados nesse nicho do mercado são as Active Server Pages da Microsoft, o Coldfusion da Allaire e as Java Server Pages da Sun. PHP é às vezes chamado de “o ASP de código−fonte aberto” porque sua funcionabilidade é tão semelhante ao produto/conceito, ou o que quer que seja, da Microsoft.

Exploraremos a criação de script no servidor, mais profundamente, nos próximos capítulos, mas, no momento, você pode pensar no PHP como uma coleção de supertags de HTML que permitem adicionar funções do servidor às suas páginas da Web. Por exemplo, você pode utilizar PHP para montar instantaneamente uma complexa página da Web ou desencadear um programa que automaticamente execute o débito no cartão de crédito quando um cliente realizar uma compra.

Falando estritamente, o PHP tem pouca relação com layout, eventos ou qualquer coisa relacionada à aparência de uma página da Web. De fato, a maior parte do que o PHP realiza é invisível para o usuário final. Alguém visualizando uma página de PHP não será capaz de dizer que não foi escrita em HTML, porque o resultado final do PHP é HTML.

O PHP é um módulo oficial do servidor http Apache, o líder do mercado de servidores Web livres que constitui aproximadamente 55 por cento da World Wide Web. Isso significa que o mecanismo de script do PHP pode ser construído no próprio servidor Web, tornando a manipulação de dados mais rápida. Assim como o servidor Apache, o PHP é compatível com várias plataformas, o que significa que ele executa em seu formato original em várias versões do UNIX e do Windows. Todos os projetos sob a égide da Apache Software Foundation – incluindo o PHP – são software de código−fonte aberto.

As várias versões do PHP foram aclamadas e premiadas nos últimos anos. O PHP3 foi o finalista em 1999 no LinuxWorld Editor´s Choice Awards (na categoria de biblioteca/ferramentas de programação) e

ganhou o segundo lugar, perdendo só para o ColdFusion, em 1998 no Cnet Builder.com Product Awards (na categoria de melhor ferramenta de script de servidor – eles deram bastante importância ao IDE), ao passo que a combinação PHP3/MySQL ganhou prêmio de banco de dados do ano no Web98. Nada mau para um software sem relações públicas, sem publicidade e sem uma significativa exposição na mídia.

História do PHP

Rasmus Lerdorf – engenheiro de software, membro da equipe Apache e o homem misterioso do ano – é o criador e a força motriz original por trás do PHP. A primeira parte do PHP foi desenvolvida para utilização pessoal no final de 1994. Tratava−se de um *wrapper* de PerlCGI que o auxiliava a monitorar as pessoas que acessavam o seu site pessoal. No ano seguinte, ele montou um pacote chamado de Personal Home Page Tools (também conhecido como PHP Construction Kit) em resposta à demanda de usuários que por acaso ou por relatos falados depararam−se com o seu trabalho. A versão 2 foi logo lançada sob o título de PHP/FI e incluía o Form Interpreter, uma ferramenta para analisar sintaticamente consultas de SQL.

Em meados de 1997, o PHP estava sendo utilizado mundialmente em aproximadamente 50.000 sites. Obviamente estava se tornando muito grande para uma única pessoa administrar, mesmo para alguém concentrado e cheio de energia como Rasmus. Agora uma pequena equipe central de desenvolvimento mantinha o projeto sobre o modelo de “junta benevolente” do código−fonte aberto, com contribuições de desenvolvedores e usuários em todo o mundo. Zeev Suraski e Andi Gutmans, dois programadores israelenses que desenvolveram os analisadores de sintaxe PHP3 e PHP4, também generalizaram e estenderam seus trabalhos sob a rubrica de Zend.com (Zeev, Andi, Zend, entendeu?).

O quarto trimestre de 1998 iniciou um período de crescimento explosivo para o PHP, quando todas as tecnologias de código−fonte aberto ganharam uma publicidade intensa. Em outubro de 1998, de acordo com a melhor suposição, mais de 100.000 domínios únicos utilizavam PHP de alguma maneira. Um ano depois, o PHP quebrou a marca de um milhão de domínios. Enquanto escrevo esta apostila, o número explodiu para cerca de dois milhões de domínios.

O que pode ser feito com PHP?

Basicamente, qualquer coisa que pode ser feita por algum programa CGI pode ser feita também com PHP, como coletar dados de um formulário, gerar páginas dinamicamente ou enviar e receber *cookies*.

PHP também tem como uma das características mais importantes o suporte a um grande número de bancos de dados, como dBase, Interbase, mSQL, mySQL, Oracle, Sybase, PostgreSQL e vários outros. Construir uma página baseada em um banco de dados torna−se uma tarefa extremamente simples com PHP.

Além disso, PHP tem suporte a outros serviços através de protocolos como IMAP, SNMP, NNTP, POP3 e, logicamente, HTTP. Ainda é possível abrir *sockets* e interagir com outros protocolos.

Tabela comparativa

Custos Comparativos				
Item	**ASP**	**Cold Fusion**	**JSP**	**PHP**
Desenvolvimento	US$ 0 – 480	US$ 395	US$ 0	US$ 0
Servidor	US$ 620	US$ 1.295	US$ 0 – 595	US$ 0
RDBMS	US$ 1.220 − 4220	US$ 0 − ~10.000	US$ 0 − ~10.000	US$ 0
Suporte de incidente	US$0 − 245	US$ 0 − 75	US$ 0 − 75	US$ 0

2. Sintaxe Básica

Delimitando o código PHP

O código PHP fica embutido no próprio HTML. O interpretador identifica quando um código é PHP pelas seguintes tags:

<?php

comandos

?>

<script language=”php”>

comandos

</script>

<?

comandos

?>

<%

comandos

%>

O tipo de *tags* mais utilizado é o terceiro, que consiste em uma “abreviação” do primeiro. Para utilizá−lo, é necessário habilitar a opção *short−tags* na configuração do PHP. O último tipo serve para facilitar o uso por programadores acostumados à sintaxe de ASP. Para utilizá−lo também é necessário habilitá−lo no arquivo de configuração do PHP (*php.ini*)

Separador de instruções

Para cada fim de linha de código tem que haver um ponto e vírgula, indicando ao sistema fim de instrução.

Exemplo.

<?

echo 'com ponto e vírgula' ;

?>

Linhas de comando, de controle, não precisam de ponto e vírgula.

Exemplo.:

```
<?

if ($x == $x){  //aqui não precisa de ponto e vírgula

echo 'com ponto e vírgula' ; //aqui precisa de ponto e vírgula

}

?>
```

Nomes de variáveis

Toda variável em PHP tem seu nome composto pelo caracter $ e uma string, que deve iniciar por uma letra ou o caracter “_”. **PHP é case sensitive**, ou seja, as variáveis $php e $PHP são diferentes. Por isso é preciso ter muito cuidado ao definir os nomes das variáveis. É bom evitar os nomes em maiúsculas, pois como veremos mais adiante, o PHP já possui alguma variáveis pré−definidas cujos nomes são formados por letras maiúsculas.

Comentários

Há dois tipos de comentários em código PHP:

Comentários de uma linha:

Marca como comentário até o final da linha ou até o final do bloco de código PHP − o que vier antes. Pode ser delimitado pelo caracter “#” ou por duas barras (//).

Exemplo:

```
<?  echo “teste”; #isto é um teste

echo “teste”; //este teste é similar ao anterior

?>
```

Comentários de mais de uma linha:

Tem como delimitadores os caracteres “/*” para o início do bloco e “*/” para o final do comentário. Se o delimitador de final de código PHP (?>) estiver dentro de um comentário, não será reconhecido pelo interpretador.

Exemplos:

```
<?
echo “teste”; /* Isto é um comentário com mais
de uma linha que funciona corretamente
*/
?>
```

3. Tipos

Tipos Suportados

PHP suporta os seguintes tipos de dados:

· Inteiro

· Ponto flutuante

· String

· Array

· Objeto

PHP utiliza checagem de tipos dinâmica, ou seja, uma variável pode conter valores de diferentes tipos em diferentes momentos da execução do script. 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 (ver adiante).

Inteiros (integer ou long)

Uma variável pode conter um valor inteiro com atribuições que sigam as seguintes sintaxes:

```
$php = 1234;     # inteiro positivo na base decimal

$php = −234;     # inteiro negativo na base decimal

$php = 0234;     # inteiro na base octal−simbolizado pelo 0

# equivale a 156 decimal

$php = 0x34;     # inteiro na base hexadecimal(simbolizado

# pelo 0x) − equivale a 52 decimal.
```

A diferença entre inteiros simples e long está no número de bytes utilizados para armazenar a variável. Como a escolha é feita pelo interpretador PHP de maneira transparente para o usuário, podemos afirmar que os tipos são iguais.

Números em Ponto Flutuante (double ou float)

Uma variável pode ter um valor em ponto flutuante com atribuições que sigam as seguintes sintaxes:

```
$php = 1.234;

$php = 23e4;     # equivale a 230.000
```

Strings

Strings podem ser atribuídas de duas maneiras:

a) utilizando aspas simples (') − Desta maneira, o valor da variável será exatamente o texto contido entre as aspas (com exceção de \\ e \' − ver tabela abaixo)

b) utilizando aspas duplas (") − Desta maneira, qualquer variável ou caracter de escape será expandido antes de ser atribuído.

Exemplo:

```
<?

$teste = "Brasil";

$php = '−−−$teste−−\n';

echo "$php";

?>
```

A saída desse script será "−−−$teste−−\n".

```
<?
```

```
$teste = "Brasil";

$php = "---$teste---\n";

echo "$php";

?>
```

A saída desse script será "---Brasil--" (com uma quebra de linha no final).

A tabela seguinte lista os caracteres de escape:

Sintaxe	**Significado**
\n	Nova linha
\r	Retorno de carro (semelhante a \n)
\t	Tabulação horizontal
\\	A própria barra (\ )
\$	O símbolo $
\’	Aspa simples
\”	Aspa dupla

Arrays

Arrays em PHP podem ser observados como mapeamentos ou como vetores indexados. Mais precisamente, um valor do tipo array é um dicionário onde os índices são as chaves de acesso. Vale ressaltar que os índices podem ser valores de qualquer tipo e não somente inteiros. Inclusive, se os índices forem todos inteiros, estes

não precisam formar um intervalo contínuo

Como a checagem de tipos em PHP é dinâmica, valores de tipos diferentes podem ser usados como índices de array, assim como os valores mapeados também podem ser de diversos tipos.

Exemplo:

```
<?

$cor[1] = “vermelho”;

$cor[2] = “verde”;

$cor[3] = “azul”;

$cor[“teste”] = 1;

?>
```

Equivalentemente, pode−se escrever:

```
<?

$cor = array(1 => “vermelho, 2 => “verde, 3 => “azul”, “teste => 1);

?>
```

Listas

As listas são utilizadas em PHP para realizar atribuições múltiplas. Através de listas é possível atribuir valores que estão num array para variáveis. Vejamos o exemplo:

Exemplo:

```
list($a, $b, $c) = array(“a”, “b”, “c”);
```

O comando acima atribui valores às três variáveis simultaneamente. É bom notar que só são atribuídos às variáveis da lista os elementos do array que possuem índices inteiros e não negativos. No exemplo acima as três atribuições foram bem sucedidas porque ao inicializar um array sem especificar os índices eles passam a ser inteiros, a partir do zero. Um fator importante é que cada variável da lista possui um índice inteiro e ordinal, iniciando com zero, que serve para determinar qual valor será atribuído. No exemplo anterior temos $a com índice 0, $b com índice 1 e $c com índice 2. Vejamos um outro exemplo:

```
$arr = array(1=>”um”,3=>”tres”,”a”=>”letraA”,2=>”dois);

list($a,$b,$c,$d) = $arr;
```

Após a execução do código acima temos os seguintes valores:

$a == null

$b == “um”

$c == “dois”

$d == “tres”

Devemos observar que à variável $a não foi atribuído valor, pois no array não existe elemento com índice 0 (zero). Outro detalhe importante é que o valor “tres” foi atribuído à variável $d, e não a $b, pois seu índice é 3, o mesmo que $d na lista. Por fim, vemos que o valor “letraA” não foi atribuído a elemento algum da lista, pois seu índice não é inteiro.

Os índices da lista servem apenas como referência ao interpretador PHP para realizar as atribuições, não podendo ser acessados de maneira alguma pelo programador. De maneira diferente do array, uma lista não pode ser atribuída a uma variável, servindo apenas para fazer múltiplas atribuições através de um array.

Objetos

Um objeto pode ser inicializado utilizando o comando *new* para instanciar uma classe para uma variável.

Exemplo:

```
class teste {
    function nada() {
        echo “nada”;
    }
}
```

```
$php = new teste;

$php −> nada();
```

A utilização de objetos será mais detalhada mais à frente.

Booleanos

PHP não possui um tipo booleano, mas é capaz de avaliar expressões e retornar *true* ou *false*, através do tipo integer: é usado o valor 0 (zero) para representar o estado *false*, e qualquer valor diferente de zero (geralmente 1) para representar o estado *true*.

Transformação de tipos

A transformação de tipos em PHP pode ser feita das seguintes maneiras:

Coerções

Quando ocorrem determinadas operações (“+”, por exemplo) entre dois valores de tipos diferentes, o PHP converte o valor de um deles automaticamente (coerção). É interessante notar que se o operando for uma variável, seu valor não será alterado.

O tipo para o qual os valores dos operandos serão convertidos é determinado da seguinte forma: Se um dos operandos for float, o outro será convertido para float, senão, se um deles for integer, o outro será convertido para integer.

Exemplo:

```
$php = “1”;          // $php é a string “1”

$php = $php + 1;   // $php é o integer 2

$php = $php + 3.7;// $php é o double 5.7

$php = 1 + 1.5       // $php é o double 2.5
```

Como podemos notar, o PHP converte string para integer ou double mantendo o valor. O sistema utilizado pelo PHP para converter de *strings* para números é o seguinte:

· É analisado o início da string. Se contiver um número, ele será avaliado. Senão, o valor será 0 (zero);

· O número pode conter um sinal no início (“+” ou “−“);

· Se a string contiver um ponto em sua parte numérica a ser analisada, ele será considerado, e o valor obtido será double;

· Se a string contiver um “e” ou “E” em sua parte numérica a ser analisada, o valor seguinte será considerado como expoente da base 10, e o valor obtido será double;

Exemplos:

```
$php = 1 + “10.5”;         // $php == 11.5
$php = 1 + “−1.3e3”;     // $php == −1299
$php = 1 + “teste10.5”;   // $php == 1
$php = 1 + “10testes”;    // $php == 11
$php = 1 + "  10testes";     // $php == 11
$php = 1 + "+  10testes";     // $php == 1
```

Transformação explícita de tipos

A sintaxe do *typecast* de PHP é semelhante ao C: basta escrever o tipo entre parênteses antes do valor

Exemplo:

```
$php = 15;          // $php é integer (15)
$php = (double) $php // $php é double (15.0)
$php = 3.9          // $php é double (3.9)
$php = (int) $php // $php é integer (3)
```

// o valor decimal é truncado

Os tipos de *cast* permitidos são:

(int), (integer) Þ muda para integer;

(real), (double), (float) Þ muda para float;

(string) Þ muda para string;

(array) Þ muda para array;

(object) Þ muda para objeto.

Com a função settype

A função settype converte uma variável para o tipo especificado, que pode ser “integer”, “double”, “string”, “array” ou “object”.

Exemplo:

$php = 15; // $php é integer

settype($php,double) // $php é double

4. Constantes

Constantes pré−definidas

O PHP possui algumas constantes pré−definidas, indicando a versão do PHP, o Sistema Operacional do servidor, o arquivo em execução, e diversas outras informações. Para ter acesso a todas as constantes pré−definidas, pode−se utilizar a função phpinfo(), que exibe uma tabela contendo todas as constantes pré−definidas, assim como configurações da máquina, sistema operacional, servidor http e versão do PHP instalada.

Definindo constantes

Para definir constantes utiliza−se a função define. Uma vez definido, o valor de uma constante não poderá mais ser alterado. Uma constante só pode conter valores escalares, ou seja, não pode conter nem um array nem um objeto. A assinatura da função define é a seguinte:

int **define**(string nome_da_constante, mixed valor);

A função retorna true se for bem−sucedida. Veja um exemplo de sua utilização a seguir:

```
define ("pi", 3.1415926536);

$circunf = 2*pi*$raio;
```

5. Operadores

Aritméticos

Só podem ser utilizados quando os operandos são números (integer ou float). Se forem de outro tipo, terão seus valores convertidos antes da realização da operação.

+	adição
−	subtração
*	multiplicação
/	divisão
%	módulo

de strings

Só há um operador exclusivo para strings:

.	concatenação

de atribuição

Existe um operador básico de atribuição e diversos derivados. Sempre retornam o valor atribuído. No caso dos operadores derivados de atribuição, a operação é feita entre os dois operandos, sendo atribuído o resultado para o primeiro. A atribuição é sempre por valor, e não por referência.

=	atribuição simples
+=	atribuição com adição
−=	atribuição com subtração
*=	atribuição com multiplicação
/=	atribuição com divisão
%=	atribuição com módulo
.=	atribuição com concatenação

Exemplo:

$a = 7;

$a += 2; // $a passa a conter o valor 9

bit a bit

Comparam dois números bit a bit.

&	“e” lógico
|	“ou” lógico
^	ou exclusivo
~	não (inversão)
<<	shift left

>>	shift right

Lógicos

Utilizados para inteiros representando valores booleanos

and	“e” lógico
or	“ou” lógico
xor	ou exclusivo
!	não (inversão)
&&	“e” lógico
||	“ou” lógico

Existem dois operadores para “e” e para “ou porque eles têm diferentes posições na ordem de precedência.

Comparação

As comparações são feitas entre os valores contidos nas variáveis, e não as referências. Sempre retornam um valor booleano.

==	igual a
!=	diferente de
<	menor que
>	maior que
<=	menor ou igual a
>=	maior ou igual a

Expressão condicional

Existe um operador de seleção que é ternário. Funciona assim:

(expressao1)?(expressao2):(expressao3)

o interpretador PHP avalia a primeira expressão. Se ela for verdadeira, a expressão retorna o valor de expressão2. Senão, retorna o valor de expressão3.

de incremento e decremento

++	incremento
−−	decremento

Podem ser utilizados de duas formas: antes ou depois da variável. Quando utilizado antes, retorna o valor da variável antes de incrementá−la ou decrementá−la. Quando utilizado depois, retorna o valor da variável já incrementado ou decrementado.

Exemplos:

$a = $b = 10; // $a e $b recebem o valor 10

$c = $a++; // $c recebe 10 e $a passa a ter 11

$d = ++$b; // $d recebe 11, valor de $b já incrementado

Ordem de precedência dos operadores

A tabela a seguir mostra a ordem de precedência dos operadores no momento de avaliar as expressões;

Precedência	Associatividade	Operadores
1.	Esquerda	,
2.	Esquerda	or
3.	Esquerda	xor
4.	Esquerda	and
5.	Direita	print
6.	Esquerda	= += −= *= /= .= %= &= != ~= <<= >>=
7.	Esquerda	? :
8.	Esquerda	||

9.	Esquerda	&&
	Esquerda	|
10.		
	Esquerda	^
11.		
	Esquerda	&
12.		
	não associa	== !=
13.		
	não associa	< <= > >=
14.		
	Esquerda	<< >>
15.		
	Esquerda	+ − .
16.		
	Esquerda	* / %
17.		
	Direita	! ~ ++ −− (int) (double) (string) (array) (object) @
18.		
	Direita	[
19.		
	não associa	new
20.		

6. Estruturas de Controle

As estruturas que veremos a seguir são comuns para as linguagens de programação imperativas, bastando, portanto, descrever a sintaxe de cada uma delas, resumindo o funcionamento.

Blocos

Um bloco consiste de vários comandos agrupados com o objetivo de relacioná−los com determinado comando ou função. Em comandos como if, for, while, switch e em declarações de funções blocos podem ser utilizados para permitir que um comando faça parte do contexto desejado. Blocos em PHP são delimitados pelos caracteres “{” e “}”. A utilização dos delimitadores de bloco em uma parte qualquer do código não relacionada com os comandos citados ou funções não produzirá efeito algum, e será tratada normalmente pelo interpretador.

Exemplo:

```
if ($x == $y)
    comando1;
comando2;
```

Para que comando2 esteja relacionado ao if é preciso utilizar um bloco:

```
if ($x == $y){
    comando1;
```

```
    comando2;
}
```

Comandos de seleção

Também chamados de condicionais, os comandos de seleção permitem executar comandos ou blocos de comandos com base em testes feitos durante a execução.

if

O mais trivial dos comandos condicionais é o if. Ele testa a condição e executa o comando indicado se o resultado for true (valor diferente de zero). Ele possui duas sintaxes:

```
if (expressão)
    comando;
```

```
if (expressão){
    comando1;
    comando2;
    comando3;
    comando4;
```

```
    comando5;
    comando1;
    comando2;
    comando3;
}
```

```
if (expressão):
    comando;
. . .
    comando;
endif;
```

Para incluir mais de um comando no if da primeira sintaxe, é preciso utilizar um bloco, demarcado por chaves.

O else é um complemento opcional para o if. Se utilizado, o comando será executado se a expressão retornar o valor false (zero). Suas duas sintaxes são:

```
if (expressão)
comando;
else
comando;
```

```
if (expressão):
    comando;
. . .
    comando;
else
    comando;
. . .
    comando;
endif;
```

A seguir, temos um exemplo do comando if utilizado com else:

```
if ($a > $b)
    $maior = $a;
else
    $maior = $b;
```

O exemplo acima coloca em $maior o maior valor entre $a e $b

Em determinadas situações é necessário fazer mais de um teste, e executar condicionalmente diversos comandos ou blocos de comandos. Para facilitar o entendimento de uma estrutura do tipo:

```
if (expressao1)
    comando1;
else
    if (expressao2)
        comando2;
    else
        if (expressao3)
            comando3;
```

```
else
    comando4;
```

foi criado o comando, também opcional elseif. Ele tem a mesma função de um else e um if usados seqüencialmente, como no exemplo acima. Num mesmo if podem ser utilizados diversos elseif's, ficando essa utilização a critério do programador, que deve zelar pela legibilidade de seu script.

O comando elseif também pode ser utilizado com dois tipos de sintaxe. Em resumo, a sintaxe geral do comando if fica das seguintes maneiras:

```
if (expressao1)
   comando;
[ elseif (expressao2)
   comando; ]
[ else
   comando; ]
```

```
if (expressao1) :
    comando;
. . .
    comando;
[ elseif (expressao2)
    comando;
. . .
    comando; ]
[ else
    comando;
. . .
    comando; ]
endif;
```

switch

O comando switch atua de maneira semelhante a uma série de comandos if na mesma expressão. Freqüentemente o programador pode querer comparar uma variável com diversos valores, e executar um código diferente a depender de qual valor é igual ao da variável. Quando isso for necessário, deve−se usar o comando switch. O exemplo seguinte mostra dois trechos de código que fazem a mesma coisa, sendo que o primeiro utiliza uma série de if's e o segundo utiliza switch:

```
if ($i == 0)
    print "i é igual a zero";
elseif ($i == 1)
    print "i é igual a um";
elseif ($i == 2)
    print "i é igual a dois";

switch ($i) {
    case 0:
        print "i é igual a zero";
    break;
    case 1:
        print "i é igual a um";
    break;
```

```
    case 2:
        print “i é igual a dois”;
    break;
}
```

É importante compreender o funcionamento do switch para não cometer enganos. O comando switch testa linha a linha os cases encontrados, e a partir do momento que encontra um valor igual ao da variável testada, passa a executar todos os comandos seguintes, mesmo os que fazem parte de outro teste, até o fim do bloco. por isso usa−se o comando break, quebrando o fluxo e fazendo com que o código seja executado da maneira desejada. Veremos mais sobre o break mais adiante. Veja o exemplo:

```
switch ($i) {
    case 0:
        print “i é igual a zero”;
    case 1:
        print “i é igual a um”;
    case 2:
        print “i é igual a dois”;
                        }
```

No exemplo acima, se $i for igual a zero, os três comandos “print” serão executados. Se $i for igual a 1, os dois últimos “print” serão executados. O comando só funcionará da maneira desejada se $i for igual a 2.

Em outras linguagens que implementam o comando switch, ou similar, os valores a serem testados só podem ser do tipo inteiro. Em PHP é permitido usar valores do tipo string como elementos de teste do comando switch. O exemplo abaixo funciona perfeitamente:

```
switch ($s) {
    case “casa”:
        print “A casa é amarela”;
    case “arvore”:
        print “a árvore é bonita”;
    case “lâmpada”:
        print “João apagou a lâmpada”;
}
```

comandos de repetição

while

O while é o comando de repetição (laço) mais simples. Ele testa uma condição e executa um comando, ou um bloco de comandos, até que a condição testada seja falsa. Assim como o if, o while também possui duas sintaxes alternativas:

while (<expressão>)

    <comando>;

while (<expressão>){

    <comando1>;

    <comando2>;

    <comando3>;

    <comando4>;

    <comando5>;

    <comando3>;

}

while (<expressão>):

<comando>;

. . .

<comando>;

endwhile;

A expressão só é testada a cada vez que o bloco de instruções termina, além do teste inicial. Se o valor da expressão passar a ser false no meio do bloco de instruções, a execução segue até o final do bloco. Se no teste inicial a condição for avaliada como false, o bloco de comandos não será executado.

O exemplo a seguir mostra o uso do while para imprimir os números de 1 a 10:

$i = 1;

while ($i <=10)

print $i++;

do... while

O laço do...while funciona de maneira bastante semelhante ao while, com a simples diferença que a expressão é testada ao final do bloco de comandos. O laço do...while possui apenas uma sintaxe, que é a seguinte:

do {

<comando>

. . .

<comando>

} **while** (<expressão>);

O exemplo utilizado para ilustrar o uso do while pode ser feito da seguinte maneira utilizando o do... while:

```
$i = 0;

do {

print ++$i;

} while ($i < 10);
```

for

O tipo de laço mais complexo é o for. Para os que programam em C, C++ ou Java, a assimilação do funcionamento do for é natural. Mas para aqueles que estão acostumados a linguagens como Pascal, há uma grande mudança para o uso do for. As três sintaxes permitidas são:

for (<inicialização>;<condição>;<incremento ou decremento>)

    <comando>;

for (<inicialização>;<condição>;<incremento ou decremento>){

    <comando>;

    <comando>;

    <comando>;

    <comando>;

    <comando>;

}

for (<inicialização>;<condição>;<incremento ou decremento>):

    <comando>;

. . .

<comando>;

endfor;

As três expressões que ficam entre parênteses têm as seguintes finalidades:

Inicialização: comando ou seqüência de comandos a serem realizados antes do inicio do laço. Serve para inicializar variáveis.

Condição: Expressão booleana que define se os comandos que estão dentro do laço serão executados ou não. Enquanto a expressão for verdadeira (valor diferente de zero) os comandos serão executados.

Incremento: Comando executado ao final de cada execução do laço.

Um comando for funciona de maneira semelhante a um while escrito da seguinte forma:

<inicialização>

while (<condição>) {

    comandos

...

    <incremento>

}

Quebra de fluxo

Break

O comando break pode ser utilizado em laços de do, for e while, além do uso já visto no comando switch. Ao encontrar um break dentro de um desses laços, o interpretador PHP para imediatamente a execução do laço, seguindo normalmente o fluxo do script.

while ($x > 0) {

...

    if ($x == 20) {

```
        echo “erro! x = 20”;

    break;

...

}
```

No trecho de código acima, o laço while tem uma condição para seu término normal ($x <= 0), mas foi utilizado o break para o caso de um término não previsto no início do laço. Assim o interpretador seguirá para o comando seguinte ao laço.

Continue

O comando continue também deve ser utilizado no interior de laços, e funciona de maneira semelhante ao break, com a diferença que o fluxo ao invés de sair do laço volta para o início dele. Vejamos o exemplo:

```
for ($i = 0; $i < 100; $i++) {

    if ($i % 2) continue;

        echo “ $i “;

}
```

O exemplo acima é uma maneira ineficiente de imprimir os números pares entre 0 e 99. O que o laço faz é testar se o resto da divisão entre o número e 2 é 0. Se for diferente de zero (valor lógico true) o interpretador encontrará um continue, que faz com que os comandos seguintes do interior do laço sejam ignorados, seguindo para a próxima iteração.

7. Funções

Definindo funções

A sintaxe básica para definir uma função é:

function **nome_da_função**([arg1, arg2, arg3]) {

    Comandos;

... ;

    [return <valor de retorno>];

}

Qualquer código PHP válido pode estar contido no interior de uma função. Como a checagem de tipos em PHP é dinâmica, o tipo de retorno não deve ser declarado, sendo necessário que o programador esteja atento para que a função retorne o tipo desejado. É recomendável que esteja tudo bem documentado para facilitar a leitura e compreensão do código. Para efeito de documentação, utiliza−se o seguinte formato de declaração de função:

tipo function nome_da_funcao(tipo arg1, tipo arg2, ...);

Este formato só deve ser utilizado na documentação do script, pois o PHP não aceita a declaração de tipos. Isso significa que em muitos casos o programador deve estar atento ao tipos dos valores passados como parâmetros, pois se não for passado o tipo esperado não é emitido nenhum alerta pelo interpretador PHP, já que este não testa os tipos.

Valor de retorno

Toda função pode opcionalmente retornar um valor, ou simplesmente executar os comandos e não retornar valor algum.

Não é possível que uma função retorne mais de um valor, mas é permitido fazer com que uma função retorne um valor composto, como listas ou arrays.

Argumentos

É possível passar argumentos para uma função. Eles devem ser declarados logo após o nome da função, entre parênteses, e tornam−se variáveis pertencentes ao escopo local da função. A declaração do tipo de cada argumento também é utilizada apenas para efeito de documentação.

Exemplo:

function **imprime**($texto){

```
    echo $texto;
}

imprime(“teste de funções”);
```

Passagem de parâmetros por referência

Normalmente, a passagem de parâmetros em PHP é feita por valor, ou seja, se o conteúdo da variável for alterado, essa alteração não afeta a variável original.

Exemplo:

```
function mais5($numero) {
$numero += 5;
}

$a = 3;
mais5($a); //$a continua valendo 3
```

No exemplo acima, como a passagem de parâmetros é por valor, a função mais5 é inútil, já que após a execução sair da função o valor anterior da variável é recuperado. Se a passagem de valor fosse feita por referência, a variável $a teria 8 como valor. O que ocorre normalmente é que ao ser chamada uma função, o interpretador salva todo o escopo atual, ou seja, os conteúdos das variáveis. Se uma dessas variáveis for passada como parâmetro, seu conteúdo fica preservado, pois a função irá trabalhar na verdade com uma cópia da variável.7 Porém, se a passagem de parâmetros for feita por referência, toda alteração que a função realizar no valor passado como parâmetro afetará a variável que o contém.

Há duas maneiras de fazer com que uma função tenha parâmetros passados por referência: indicando isso na declaração da função, o que faz com que a passagem de parâmetros sempre seja assim; e também na própria chamada da função. Nos dois casos utiliza−se o modificador “&”. Vejamos um exemplo que ilustra os dois casos:

```
function mais5(&$num1, $num2) {

$num1 += 5;

$num2 += 5;

}



$a = $b = 1;

mais5($a, $b); /* Neste caso, só $num1 terá seu valor alterado, pois a passagem por referência está definida na declaração da função. */
```

mais5($a, &$b); /* Aqui as duas variáveis terão seus valores alterados. */

Argumentos com valores pré−definidos (default)

Em PHP é possível ter valores *default* para argumentos de funções, ou seja, valores que serão assumidos em caso de nada ser passado no lugar do argumento. Quando algum parâmetro é declarado desta maneira, a passagem do mesmo na chamada da função torna−se opcional.

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

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:

```
function teste($figura = circulo, $cor) {
    echo “a figura é um “. $figura. “ de cor “ $cor;
}

teste(azul);
```

/* A função não vai funcionar da maneira esperada, ocorrendo um erro no interpretador. A declaração correta é: */

```
function teste2($cor, $figura = circulo) {
    echo “a figura é um “. $figura. “ de cor “ $cor;
}

teste2(azul);
```

/* Aqui a função funciona da maneira esperada, ou seja, imprime o texto: “a figura é um círculo de cor azul” */

Contexto

O contexto é o conjunto de variáveis e seus respectivos valores num determinado ponto do programa. Na

chamada de uma função, ao iniciar a execução do bloco que contém a implementação da mesma é criado um novo contexto, contendo as variáveis declaradas dentro do bloco, ou seja, todas as variáveis utilizadas dentro daquele bloco serão eliminadas ao término da execução da função.

Escopo

O escopo de uma variável em PHP define a porção do programa onde ela pode ser utilizada. Na maioria dos casos todas as variáveis têm escopo global. Entretanto, em funções definidas pelo usuário um escopo local é criado. Uma variável de escopo global não pode ser utilizada no interior de uma função sem que haja uma declaração.

Exemplo:

```
$php = “Testando”;

function Teste() {

echo $php;

}

Teste();
```

O trecho acima não produzirá saída alguma, pois a variável $php é de escopo global, e não pode ser referida num escopo local, mesmo que não haja outra com nome igual que cubra a sua visibilidade. Para que o script funcione da forma desejada, a variável global a ser utilizada deve ser declarada.

Exemplo:

$php = “Testando”;

function Teste() {

global $php;

echo $php;

}

Teste();

Uma declaração “global” pode conter várias variáveis, separadas por vírgulas. Uma outra maneira de acessar variáveis de escopo global dentro de uma função é utilizando um array pré−definido pelo PHP cujo nome é $GLOBALS. O índice para a variável referida é o próprio nome da variável, sem o caracter $. O exemplo acima e o abaixo produzem o mesmo resultado:

Exemplo:

$php = "Testando";

function Teste() {

```
echo $GLOBALS["php"]; // imprime $php

echo $php; // não imprime nada

}


Teste();
```

8. Variáveis

O 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. Veja o seguinte exemplo:

```
function Teste() {

$a = 0;

echo $a;

$a++;

}
```

O último comando da função é inútil, pois assim que for encerrada a execução da função a variável $a perde seu valor. Já no exemplo seguinte, a cada chamada da função a variável $a terá seu valor impresso e será incrementada:

```
function Teste() {

static $a = 0;
```

```
echo $a;

$a++;

}
```

O modificador static é muito utilizado em funções recursivas, já que o valor de algumas variáveis precisa ser mantido. Ele funciona da seguinte forma: O valor das variáveis declaradas como estáticas é mantido ao terminar a execução da função. Na próxima execução da função, ao encontrar novamente a declaração com static, o valor da variável é recuperado.

Em outras palavras, uma variável declarada como static tem o mesmo “tempo de vida” que uma variável global, porém sua visibilidade é restrita ao escopo local em que foi declarada e só é recuperada após a declaração.

Exemplo:

```
function Teste() {

echo "$a";

static $a = 0;

$a++;

}
```

O exemplo acima não produzirá saída alguma. Na primeira execução da função, a impressão ocorre antes da atribuição de um valor à função e, portanto o conteúdo de $a é nulo (string vazia). Nas execuções seguintes da função Teste() a impressão ocorre antes da recuperação do valor de $a e, portanto nesse momento seu valor ainda é nulo. Para que a função retorne algum valor o modificador static deve ser utilizado.

Variáveis Variáveis

O PHP tem um recurso conhecido como variáveis variáveis, que consiste em variáveis cujos nomes também são variáveis. Sua utilização é feita através do duplo cifrão ($$).

$a = “teste”;

$$a = “Mauricio Vivas”;

O exemplo acima e equivalente ao seguinte:

$a = “teste”;

$teste = “Mauricio Vivas”;

Variáveis enviadas pelo navegador

Para interagir com a navegação feita pelo usuário, é necessário que o PHP possa enviar e receber informações para o software de navegação. A maneira de enviar informações, como já foi visto anteriormente, geralmente é através de um comando de impressão, como o *echo*. Para receber informações vindas do navegador através de um *link* ou um formulário html o PHP utiliza as informações enviadas através da URL. Por exemplo: se

seu script php está localizado em “http://localhost/teste.php3” e você o chama com a url “http://localhost/teste.php3?php=teste”, automaticamente o PHP criará uma variável com o nome $php 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.

URLencode

-

O formato urlencode é obtido substituindo os espaços pelo caracter “+” e todos os outros caracteres não alfa−numéricos (com exceção de “_”) pelo caracter “%” seguido do código ASCII em hexadecimal.

Por exemplo: o texto “Testando 1 2 3 !!” em urlencode fica “Testando+1+2+3+%21%21”

O PHP possui duas funções para tratar com texto em urlencode. Seguem suas sintaxes:

string **urlencode**(string texto);

string **urldecode**(string texto);

Essas funções servem respectivamente para codificar ou decodificar um texto passado como argumento. Para entender melhor o que é um argumento e como funciona uma função, leia o tópico “funções”.

Variáveis de ambiente

O PHP possui diversas variáveis de ambiente, como a $PHP_SELF, por exemplo, que contém o nome e o path do próprio arquivo. Algumas outras contém informações sobre o navegador do usuário, o servidor http, a versão do PHP e diversas informações. Para ter uma listagem de todas as variáveis e constantes de ambiente e seus respectivos conteúdos, deve−se utilizar a função phpinfo().

Verificando o tipo de uma variável

Por causa da tipagem dinâmica utilizada pelo PHP, nem sempre é possível saber qual o tipo de uma variável em determinado instantes não contar com a ajuda de algumas funções que ajudam a verificar isso. A verificação pode ser feita de duas maneiras:

Função que retorna o tipo da variável

Esta função é a gettype. Sua assinatura é a seguinte:

string **gettype**(mixed var);

A palavra “mixed” indica que a variável var pode ser de diversos tipos.

A função gettype pode retornar as seguintes strings: “integer”, “double”, “string”, “array”, “object” e “unknown type”.

Funções que testam o tipo da variável

São as funções is_int, is_integer, is_real, is_long, is_float, is_string, is_array e is_object. Todas têm o mesmo formato, seguindo modelo da assinatura a seguir:

int **is_integer**(mixed var);

Todas essas funções retornam true se a variável for daquele tipo, e false em caso contrário.

Destruindo uma variável

É possível desalocar uma variável se ela não for usada posteriormente através da função unset, que tem a seguinte assinatura:

int **unset**(mixed var);

A função destrói a variável, ou seja, libera a memória ocupada por ela, fazendo com que ela deixe de existir. Se mais na frente for feita uma chamada á variável, será criada uma nova variável de mesmo nome e de conteúdo vazio, a não ser que a chamada seja pela função isset. Se a operação for bem sucedida, retorna true.

Verificando se uma variável possui um valor

Existem dois tipos de teste que podem ser feitos para verificar se uma variável está setada: com a função isset e com a função empty.

A função isset

Possui o seguinte protótipo:

int **isset**(mixed var);

E retorna true se a variável estiver setada (ainda que com uma string vazia ou o valor zero), e false em caso contrário.

A função empty

Possui a seguinte assinatura:

int **empty**(mixed var);

E retorna true se a variável não contiver um valor (não estiver setada) ou possuir valor 0 (zero) ou uma string vazia. Caso contrário, retorna false.

Arrays Multidimensionais

Arrays multidimensionais são arrays simples com um dos (ou todos) seus elementos sendo outro array e assim consecutivamente.

Exemplo:

$Campeao[5] = 123456789 ;

```
$Tricampeao[“casa”] = $Campeao;

$Tricampeao[“predio”] = 19191919;


$Brasil[1] = $Tricampeao;

$Brasil[2] = “Bicampeao”;

$Brasil[“copa”] = $Tricampeao;

$Brasil[4] = “Tetracampeao”;

$Brasil[“mundo”] = “Pentacampeao”;



echo $Campeao[5];                 // resultará 123456789    Array simples
echo $Brasil[1][“casa”][5] ;      // resultará 19191919     Array tridimensional
echo $Tricampeao[“casa”][5];      // resultará 123456789    Array bidimensional
echo $Brasil[“copa”][“predio”];   // resultará 19191919     Array bidimensional
```

9. Classes e Objetos

Classe

Uma classe é um conjunto de variáveis e funções relacionadas a essas variáveis. Uma vantagem da utilização é poder usufruir o recurso de encapsulamento de informação. Com o encapsulamento o usuário de uma classe não precisa saber como ela é implementada, bastando para a utilização conhecer a interface, ou seja, as funções disponíveis. Uma classe é um tipo e, portanto não pode ser atribuída a uma variável. Para definir uma classe, deve−se utilizar a seguinte sintaxe:

```
class Nome_da_classe {

    var $variavel1;

    var $variavel2;

    function funcao1 ($parâmetro) {

        /* === corpo da função === */

    }

}
```

Objeto

Como foi dito anteriormente, classes são tipos, e não podem ser atribuídas a variáveis. Variáveis do tipo de uma classe são chamadas de objetos, e devem ser criadas utilizando o operador new, seguindo o exemplo abaixo:

$variável = new $nome_da_classe;

Para utilizar as funções definidas na classe, deve ser utilizado o operador “−>”, como no exemplo:

$variável−>funcao1();

A variável $this

Na definição de uma classe, pode−se utilizar a variável $this, que é o próprio objeto. Assim, quando uma classe é instanciada em um objeto, e uma função desse objeto na definição da classe utiliza a variável $this, essa variável significa o objeto que estamos utilizando.

Como exemplo da utilização de classes e objetos, podemos utilizar a classe conta, que define uma conta bancária bastante simples, com funções para ver saldo e fazer um crédito.

```
class conta {

var $saldo;

function saldo() {

return $this−>saldo;

}
```

```
function credito($valor) {

$this−>saldo += $valor;

}

}


$minhaconta = new conta;

$minhaconta−>saldo(); // a variável interna não foi

// inicializada, e não contém

// valor algum

$minhaconta−>credito(50);

$minhaconta−>saldo(); // retorna 50
```

SubClasses

Uma classe pode ser uma extensão de outra. Isso significa que ela herdará todas as variáveis e funções da outra classe, e ainda terá as que forem adicionadas pelo programador. Em PHP não é permitido utilizar herança múltipla, ou seja, uma classe pode ser extensão de apenas uma outra.Para criar uma classe extendida, ou derivada de outra, deve ser utilizada a palavra reservada extends, como pode ser visto no exemplo seguinte:

```
class novaconta extends conta {
```

```
    var $numero;

    function numero() {

        return $this−>numero;

    }

}
```

A classe acima é derivada da classe conta, tendo as mesmas funções e variáveis, com a adição da variável $numero e a função numero().

Construtores

Um construtor é uma função definida na classe que é automaticamente chamada no momento em que a classe é instanciada (através do operador new). O construtor deve ter o mesmo nome que a classe a que pertence. Veja o exemplo:

```
class conta {

    var $saldo;


    function conta () {

        $this.saldo = 0;

    }
```

```
    function saldo() {
        return $this−>saldo;
    }
    function credito($valor) {
        $this−>saldo += $valor;
    }
}
```

Podemos perceber que a classe conta agora possui um construtor, que inicializa a variável $saldo com o valor 0.

Um construtor pode conter argumentos, que são opcionais, o que torna esta ferramenta mais poderosa. No exemplo acima, o construtor da classe conta pode receber como argumento um valor, que seria o valor inicial da conta.

Vale observar que para classes derivadas, o construtor da classe pai não é automaticamente herdado quando o construtor da classe derivada é chamado.

10. PHP avançado

Interagindo com o browser

PHP também permite interagir com informações do browser automaticamente. Por exemplo, o script a seguir mostra informações sobre o browser do usuário.

```
<html>

<head><title>Aprendendo PHP</title></head>

<body>


<? echo $HTTP_USER_AGENT; ?>


</body>

</html>
```

Esse código em um Internet Explorer 6.0 com sistema operacional Windows 98, geraria: *Mozilla/4.0 (compatible; MSIE 6.0; Windows 98)*

```
<html>

<head><title>Aprendendo PHP</title></head>
```

```
<body>


<?
if  (strpos($HTTP_USER_AGENT,"MSIE”) != 0) {
echo "Você usa Internet Explorer";
} else {
echo "Você não usa Internet Explorer";
}
?>


</body>
</html>
```

Neste exemplo, será apenas exibido um texto informando se está sendo utilizado o Microsoft Internet Explorer ou não, mas para outras funções poderia ser utilizado algo semelhante.

É bom notar o surgimento de mais uma função no código anterior: strpos(string1,string2). Essa função retorna a posição da primeira aparição de string2 em string1, contando a partir de zero, e não retorna valor algum se não ocorrer. Assim, para testar se a string $HTTP_USER_AGENT contém a string “MSIE”, basta testar se strpos devolve algum valor.

Utilizando formulários HTML

Ao clicar num botão “Submit” em um formulário HTML as informações dos campos serão enviadas ao servidor especificado para que possa ser produzida uma resposta. O PHP trata esses valores como variáveis, cujo nome é o nome do campo definido no formulário. O exemplo a seguir mostra isso, e mostra também como o código PHP pode ser inserido em **qualquer** parte do código HTML:

```
<html>

<head><title>Aprendendo PHP</title></head>

<body>


<?php

if ($texto != "")

echo "Você digitou \"$texto\"<br><br>";

?>


<form method=post action="<? echo $PATH_INFO; ?>">

<input type="text" name="texto" value="" size=10>

<br>
```

```
<input type="submit" name="sub" value="Enviar!">

</form>



</body>

</html>
```

Ao salvar o arquivo acima e carregá−lo no browser, o usuário verá apenas um formulário que contém um espaço para digitar o texto. Ao digitar um texto qualquer e submeter o formulário, a resposta, que é o mesmo arquivo PHP (indicado pela constante $PATH_INFO, que retorna o nome do arquivo) exibirá a mensagem "Você digitou <<mensagem>>".

Isso ocorre porque o código PHP testa o conteúdo da variável $texto. Inicialmente ele é uma string vazia, e por isso nada é impresso na primeira parte. Quando algum texto é digitado no formulário e submetido, o PHP passa a tratá−lo como uma variável. Como no formulário o campo possui o nome “texto”, a variável com seu conteúdo será $texto. Assim, no próximo teste o valor da variável será diferente de uma string vazia, e o PHP imprime um texto antes do formulário.

Cookies

Cookies são mecanismos para armazenar e consultar informações nos browsers dos visitantes da página. O PHP atribui cookies utilizando a função **setcookie**, que deve ser utilizada antes da tag <html> numa página.

O uso de cookies não é recomendado quando se trata de informações sigilosas. Os dados dos cookies são armazenados no diretório de arquivos temporários do visitante, sendo facilmente visualizado por pessoas mal intencionadas.

Além da opção “aceitar cookies” que pode ser desativada a qualquer momento pelo visitante.

Para uma transmissão de dados segura é recomendável o uso de sessões(ver adiante).

Setcookie(“nome_do_cookie”,”seu_valor”,”tempo_de_vida”,”path”,”domínio”,”conexão_segura”)

Nome_do_cookie = É o nome que, posteriormente, se tornará a variável e o que o servirá de referência para indicar o cookie.

Seu_valor = É o valor que a variável possuirá. Esse valor pode ser de todos os tipos.

Tempo_de_vida = É o tempo, em segundos, que o cookie existirá no computador do visitante. Uma vez excedido esse prazo o cookie se apaga de modo irrecuperável. Se esse argumento ficar vazio, o cookie se apagará quando o visitante fechar o browser.

Path = endereço da página que gerou o cookie – automático

Domínio = domínio ao qual pertence o cookie – automático

Conexão_segura = Indica se o cookie deverá ser transmitido somente em uma conexão segura HTTPS.

Sessão

Sessões são mecanismos muito parecidos com os tradicionais cookies. Suas diferenças são que sessões são armazenadas no próprio servidor e não expiram a menos que o programador queira apagar a sessão.

Existem algumas funções que controlam sessões e estão detalhadas no capítulo “12.Bibliotecas de funções”.

Aqui estão as funções de sessão mais usadas.

Nome da função	Argumentos
Session_start()	Não precisa de argumento
Session_register()	A variável sem o cifrão
Session_unregister()	A variável sem o cifrão
Session_is_registered()	A variável sem o cifrão

O *session_destroy()* só deve ser usado quando for da vontade do programador acabar com todos as sessões daquele visitante, portanto muito cuidado com essa função.

Require

A função **require** põe o conteúdo de um outro arquivo no arquivo php atual, antes de ser executado. Quando o interpretador do PHP ler este arquivo, ele encontrará todo o conteúdo dos “require´s” adicionado no arquivo corrente.

```
Require(“nomedoarquivo”);
```

Criando o hábito de usar essa função, o programador pode vir a encontrar um erro de arquivo já declarado. Para evitar isso é recomendável que sempre que a função **require** for utilizada ela seja substituída pela função **require_once**.

```
Require_once(“nome_do_arquivo“);
```

Include

A função **Include** é semelhante à função **require**, com a diferença que o código do arquivo incluído é processado em tempo de execução, permitindo que sejam usados “includes” dentro de estruturas de controle como **for** e **while**.

```
$arqs = array(‘a1.inc’,’a2.inc’,’a3.inc’) ;

for ($i=0;$i<count($arqs); $i++){

        include($arqs[$i]);
```

```
}
if ($x == $y){
        include($arquivo1);
}
else{
        include($arquivo2);
}
/*
        Note que quando se utiliza a função include
Dentro de estruturas é necessário a utilização das chaves
*/
```

Formulários Avançados

O PHP também entende as matrizes em forma de variáveis vindas de um formulário.

Exemplo:

```
<form action=”matrizes.php” method=“POST”>
        Nome : <input type=”text” name=dados[nome]><br>
```

```
Email : <input type=”text” name=dados[email]><br>

Esportes de preferência: <br>

<select multiple name=”esportes[]”>

        <option value=”futebol”>Futebol</option>

        <option value=”vôlei”>Vôlei</option>

</select><br>

<input type=”submit” value=”enviar”>


</form>
```

Depois de enviado, os campos se tornarão variáveis e campos como este “dados[nome]” se transformarão em arrays que não deixam de ser variáveis.

Arquivos Remotos

Nas funções que trabalham com arquivos, é possível utilizar URLs para acessar arquivos em outros servidores na Web ou no próprio servidor.

As funções aqui utilizadas serão detalhadas mais à frente no capítulo “Biblioteca de Funções”.

Exemplo:

Lendo

```
$f = fopen(“http://www.php.net”,”r”);
        if (!$f){
                echo “Erro ao abrir a URL.<br>”;
exit;
}
while (!feof($f)){
        $s = fgets($f,256);
        echo $s;
}
fclose($f);
```

Escrevendo

```
$f = fopen(“ftp://user:senha@site”,”w”);
        if (!$f){
                echo “Erro ao abrir a URL.<br>”;
exit;
}
```

```
else{
        fputs($f,”texto a ser escrito”);
        fputs($f,”mais texto a ser escrito”);
        fputs($f,”mais texto a ser escrito”);
        fputs($f,”mais texto a ser escrito”);
        fputs($f,”mais texto a ser escrito”);
        fclose;
}
```

Tratamento de erros

Existem quatro tipos(até a versão 4.0) de erros no PHP para indicar a gravidade do erro encontrado ou ocorrido. Eles são:

1. Erros de funções (function errors)
2. Avisos (warnings)
3. Erros de processamento (parser error)
4. Observações (notice)

As mensagens de erro são uma coisa com que os programadores devem prestar muita atenção, afinal nenhum programador quer por no ar um sistema que quando o primeiro visitante entra apareça uma mensagem de erro. Para evitar essas inconveniências use sempre um “@” antes da cada chamada as funções. Se a opção **track_errors** no arquivo ***php.ini*** estiver habilitada, a mensagem de erro poderá ser encontrada na variável global **$php_errormsg**.

A chamada da função ficaria assim:

```
@strtolower();
```

Essa função deixaria todos os caracteres em minúsculo, mas como não foi passado nenhum argumento essa função deveria exibir uma mensagem de erro.

11.SQL

O que é?

De acordo com Andrew Taylor, inventor original da linguagem SQL, o SQL não significa “Structured Query language” (ou qualquer outra coisa nesse sentido). Mas para o resto do mundo, esse é seu significado agora. Como você poderia esperar desse (não−)título, o SQL representa um método mais geral e mais estrito de armazenamento de dados que o padrão anterior de banco de dados não−relacionais no estilo dbm.

SQL não é ciência espacial. As quatro instruções básicas de manipulação de dados suportadas essencialmente por todos os bancos de dados de SQL são SELECT, INSERT, UPDATE e DELETE. SELECT recupera os dados do banco de dados, INSERT insere em uma nova entrada, UPDATE edita partes da entrada no lugar e DELETE exclui uma entrada completamente.

A maior parte da dificuldade reside em projetar bancos de dados. O projetista deve pensar muito seriamente sobre a melhor maneira de representar cada parte de dados e relacionamento para a utilização planejada. É um prazer programar com bancos de dados bem projetados, enquanto os pobremente projetados podem deixar você careca quando contemplar as numerosas conexões e horrorosas junções.

Os bancos de dados SQL são criados pelas chamadas instruções de estrutura de dados. As mais importantes são CREATE, ALTER e DROP. Como se poderia imaginar, CREATE DATABASE cria um novo banco de dados e CREATE TABLE define uma nova tabela dentro de um banco de dados. ALTER altera a estrutura de uma tabela. DROP é a bomba nuclear entre os comandos de SQL, uma vez que exclui completamente tabelas ou banco de dados inteiros.

O bom projeto de banco de dados também é uma questão de segurança. Empregando medidas profiláticas razoáveis, um banco de dados SQL pode aprimorar a segurança de seu site.

Esta apostila explicará somente o básico de SQL.

SELECT

SELECT é o comando principal de que você precisa para obter informações fora de um banco de dados de SQL. A sintaxe básica é extremamente simples:

SELECT campo1, campo2, campo3 FROM tabela WHERE condições

Em alguns casos, você desejará pedir linhas inteiras em vez de selecionar partes individuais de informações. Essa prática é obsoleta por razões muito boas(ela pode ser mais lenta que solicitar apenas os dados de que você precisa e pode levar a problemas se você reprojetar a tabela) , mas ainda é utilizada e, portanto, precisamos menciona−la. Uma linha inteira é indicada utilizando o símbolo do asteristico:

SELECT * FROM minha_tabela WHERE campo1 = “abcdef”

Subseleções

Antes de deixarmos o reino das instruções SELECT, devemos mencionar a subseleção. Essa é uma instrução como:

SELECT phone_number FROM table WHERE name = ‘SELECT name FROM table2 WHERE id = 1’;

As subseleções são mais uma conveniência que uma necessidade. Elas podem ser muito úteis se você estiver trabalhando com enormes lotes de dados; mas é possível obter o mesmo resultado com duas seleções

mais simples(embora isso será algo mais lento, mesmo no PHP4).

INSERT

O comando que você precisa para colocar novos dados em um banco de dados é o INSERT. Eis a sintaxe básica:

```
INSERT INTO tabela (campo1,campo2,campo3) VALUES($val1,$val2,$val3);
```

Obviamente as colunas e seus valores devem corresponder; se você misturar os seus itens de array, nada de bom acontecerá. Se algumas linhas não tiverem valores para alguns campos, você precisará utilizar um valor nulo, vazio ou auto−incrementado – e, em um nível mais profundo, você pode ter assegurado de antemão que os campos possam ser nulos ou auto−incrementáveis. Se isso não for possível, você simplesmente deve omitir qualquer coluna que você deseja que assuma o padrão de valor vazio em uma instrução de INSERT.

Um desdobramento da instrução INSERT básica é INSERT INTO... SELECT. Isso apenas significa que você pode inserir os resultados de uma instrução SELECT:

```
INSERT INTO customer(birthmonth, birthflower, birthstone) SELECT * FROM birthday_info WHERE birthmonth = $birthmonth;
```

Entretanto, nem todos os bancos de dados de SQL têm essa capacidade. Além disso, é preciso ser cuidadoso com esse comando porque você pode facilmente causar problemas para você mesmo. Em geral, não é uma boa idéia selecionar a partir do mesmo banco de dados em que você esta inserindo.

UPDATE

UPDATE é utilizado para editar informações que já estão no banco de dados sem excluir qualquer quantidade significativa. Em outras palavras, você pode seletivamente alterar algumas informações sem excluir um registro antigo inteiro e inserir um novo. A sintaxe é:

UPDATE table SET campo1 = “val1”, campo2 = “val2”,campo3=”val3” WHERE condição;

A instrução condicional é exatamente como uma condição SELECT, como WHERE id=15 ou WHERE gender=”F”.

DELETE

DELETE é bastante auto−explicativa: você a utiliza para excluir permanentemente as informações do banco de dados. A sintaxe é:

DELETE datapoint FROM table WHERE condition;

A coisa mais importante a lembrar−se é da condição – se você não configurar uma, excluirá cada entrada nas colunas especificadas do banco de dados, sem uma confirmação ou uma segunda chance em muitos casos.

O que há em comum entre: DELETE x UPDATE

Você deve lembrar−se de utilizar uma condição toda vez que atualizar ou excluir. Se você não fizer isso, todas as linhas na tabela sofrerão a mesma alteração ou exclusão. Mesmo os programadores mais experientes esquecem a condição, para seu grande constrangimento profissional. Cuidado para não esquecer a condição.

CREATE

CREATE é utilizado para fazer uma tabela ou banco de dados completamente novo. Realmente criar um banco de dados implica um pouco mais que apenas atribuir um nome.

CREATE DATABASE db_name;

Todo o trabalho está em definir as tabelas e as colunas. Primeiro você declara o nome da tabela e depois deve detalhar os tipos específicos de colunas dessa tabela no que é chamado de “definição de criação”.

CREATE TABLE table {

```
    ->id_col INT NOT NULL AUTO_INCREMENT PRIMARY KEY,

    ->col1 TEXT NULL INDEX,

    ->col2 DATE NOT NULL

};
```

Diferentes servidores de SQL têm tipos de dados e opções de definição ligeiramente diferentes, então a sintaxe de um não pode transferir exatamente para outro.

DROP

DROP pode ser utilizado para excluir completamente uma tabela ou banco de dados e todos os seus dados associados. Ele não é um comando mais sutil:

```
DROP TABLE table;

DROP DATABASE db_name
```

Obviamente é necessário muito cuidado com essa instrução.

ALTER

ALTER é a maneira de alterar uma estrutura da tabela. Você simplesmente indica que tabela está alterando e redefine suas especificações. Novamente, eis os produtos SQL diferem na sutileza.

```
ALTER TABLE table RENAME AS new_table;

ALTER TABLE new_table ADD COLUMN col3 VARCHAR(50) ;

ALTER TABLE new_table DROP COLUMN col2;
```

12. Bancos de dados compatíveis com o PHP:

Os bancos de dados atualmente suportados pelo PHP são: ***Adabas D, dBase, mSQL, InterBase, SyBase, Empress, MySQL, Velocis, FilePro, Oracle, dbm,***

Informix, PostgreSQL.

13. Biblioteca de Funções

Neste capítulo não estão detalhadas todas as funções do PHP, mas grande parte das funções mais utilizadas estão detalhadas neste capítulo.

Bibliotecas requeridas

Para que se possa utilizar essas funções é preciso ter instalado, no servidor, as seguintes bibliotecas:

Módulo do PHP	Onde encontrar
LDAP	ftp://ftp.openldap.org/pub/openldap/
	ftp://terminator.rs.itd.umich.edu/ldap/
	ftp://ftp.critical−angle.com/pub/cai/slapd/
Berkley DB2	http://www.sleepycat.com/
SNMP	http://www.ece.ucdavis.edu/ucd−snmp/
GD	http://www.boutell.com/gd/#buildgd
mSQL	http://www.hughes.com.au/
MySQL	http://www.mysql.com/
IMAP	ftp://ftp.cac.washington.edu/imap/
FreeType (libttf)	http://www.freetype.org
Zlib	http://www.cdrom.com/pub/infozip/zlib
Expat XML parser	http://www.jclark.com/xml/expat.html
PDFlib	http://www.ifconnection.de/~tm/
mcrypt	ftp://argeas.cs−net.gr/pub/unix/mcrypt/
mhash	http://sasweb.de/mhash/
t1lib	http://www.neuroinformatik.ruhr−uni−bochum.de/ini/PEOPLE/rmz/t1lib.html
Dmalloc	http://www.dmalloc.com/
Aspell	http://metalab.unc.edu/kevina/aspell/

Array

Array

Retorna um array de parâmetros. Índices podem ser atribuídos aos parâmetros através do operador.

Array array(...);

Exemplo:

$arr = array(“palavras” => array(“1” => “index”,”2” => “main“,”3” => “default”),”tradução” => array(“índice”,”principal”,”padrão”));

array_keys [PHP4]

Retorna os índices de um array.

Array array_keys(array matriz);

Array_merge [PHP4]

Retorna um array resultante da concatenação de dois ou mais arrays.

Array array_merge(array matriz1, array matriz2, [...]);

Array_pop[PHP4]

Retorna e remove o ultimo elemento de um array.

Mixed array_pop(array matriz1);

Array_push[PHP4]

Acrescente elementos no final de um array. Retorna o numero de elementos de uma array.

Int array_push(array matriz1, mixed elem1,[...]);

Array_shift[PHP4]

Retorna e remove o primeiro elemento de um array

Mixed array_shift(array array);

Array_slice[PHP4]

Retorna uma seqüência de elementos de um array.

Array array_slice(array matriz, int inicio, int [tamanho]) ;

Argumento	Descrição
Matriz	o array de onde serão copiados os elementos
Início	Posição inicial no array
tamanho	Número de elementos a serem retornados

Array_splice[PHP4]

Remove elementos de um array e, opcionalmente, substitui os elementos removidos por outros.

Array array_splice(array matriz, int início, int [tamanho], array [substituição]);

Argumento	Descrição
Matriz	o array de onde serão copiados os elementos
Início	Posição inicial no array entrada
Tamanho	Número de elementos a serem removidos
Substituição	Arrays dos elementos que substituirão os elementos removidos

Array_unshift[PHP4]

Acrescenta elementos no início de um array. Retorna o número de elementos de uma array.

Int array_unshift(array matriz, mixed var, [...]);

Array_values[PHP4]

Retorna os valores de um array

Array array_values(array matriz);

Array_walk

Aplica uma função para cada elemento de um array.

Int array_walk(array matriz, string nomefunc, mixed dadosextras);

Function nomefunc($item, $índice, $dadosextras)

Exemplo:

$matriz = array(“1”=>”monitor”,”2”=>”Placa de vídeo”, “3”=> “mouse”);

//esta é uma das funções chamadas pelo array_walk

function imprime_item($item, $índice,$dados){

echo “$índice = $item($dados)”;

}

//esta é uma das funções chamadas pelo array_walk

function atualizar_item(&$item, $índice, $dados){

$item = “$item ($dados)“;

}

array_walk($matriz,’imprime_item’,’impresso’);

array_walk($matriz,’atualizar_item’,’atualizado’); array_walk($matriz,’imprime_item’,’atualizado’);

Arsort

Ordena um array na ordem reversa, mantendo a associação dos índices com os elementos

Void arsort(array array);

Asort

Ordena um array na ordem, mantendo a associação dos índices com os elementos

Void asort(array array);

Compact[PHP4]

Cria um array de variáveis com seus valores. Os parâmetros podem ser nomes de variáveis ou arrays com os nomes de variáveis.

Array compact(string nomevar|string nomesvar,[...]) ;

Exemplo:

$depto = “Desenvolvimento”;

$funcionário = “João Fulaninho”;

$cargo = “Analista de Sistemas” ;

$atributos = array(“cargo”,”depto”);

$res = compact(“funcionário”,”atributos”);

O código acima irá criar o array (“funcionário”=>”João Fulaninho”,”cargo”=:”Analista de Sistemas”,”depto”=>”Desenvolvimento”).

Count

Retorna o numero de elementos em um array. Retorna 1 se a variável não for um array e 0 se a variável não estiver definida.

Int count(mixed matriz) ;

Current

Retorna o elemento corrente em um array

Mixed current(array matriz);

Each

Retorna o par “elemento e valor” atual de um array, avançando o cursor do array para o próximo item. É semelhante a um loop *for* que percorre os elementos de um array.

Array each(array matriz) ;

End

Aponta o indicador interno de um array para o último elemento.

End(array matriz);

Extract

Converte os pares “índice => valor” em variáveis “nome => valor”. É o inverso da função compact().

Void extract(array matriz, int [tipo_extracação], string [prefixo]) ;

Tipo de extração	Descrição
EXTR_OVERWRITE	Sobrescreva as variáveis existentes se houver colisão
EXTR_SKIP	Não sobrescreva as variáveis se houver colisão
EXTR_PREFIX_SAME	Se houver colisão com as variáveis existentes, coloque o prefixo no nome da variável
EXTR_PREFIX_ALL	Coloque o prefixo no nome de todas as variáveis.

In_array[PHP4]

Retorna true se *texto* existir no array matriz

Bool in_array(mixed texto, array matriz);

Key

Retorna o elemento índice da posição corrente de um array

Mixed key(array matriz);

Ksort

Ordena um array pelos índices, mantendo a associação dos índices com os elementos.

Int ksort(array matriz) ;

List

Atribui uma lista de variáveis em apenas uma operação

Void list(...);

Next

Retorna o próximo elemento de um array, ou false se não houver mais elementos.

Mixed next(array matriz);

Pos

Retorna o elemento corrente de um array

Mixed pos(array matriz);

Prev

Retorna o elemento anterior ao elemento corrente de um array ou falso se não houver mais elementos.

Mixed prev(array matriz);

Range

Retorna um array contendo uma seqüência de números inteiros no intervalo especificado.

Array range(int lim_inferior, int lim_superior) ;

Reset

Aponta o indicador interno de um array para seu primeiro elemento.

Mixed reset(array matriz);

Rsort

Ordena um array na ordem inversa

Void rsort(array matriz);

Shuffle

Embaralha os elementos de um array

Void shuffle(array matriz);

Sizeof

Retorna o número de elementos de um array.

Int sizeof(array matriz);

Sort

Ordena um array

Void sort(array matriz);

Uasort

Ordena um array utilizando uma função definida pelo programador, mantendo a associação dos índices. A função de comparação deverá retornar um inteiro menor que zero, igual a zero, ou maior que zero, se o primeiro argumento for respectivamente menor que, igual a, ou maior que o segundo argumento.

Void uasort(array matriz, function func_compara) ;

Uksort

Ordena um array pelos índices, utilizando uma função definida pelo programador. A função de comparação deverá retornar um inteiro menor que zero, igual a zero, ou maior que zero, se o primeiro argumento for respectivamente menor que, igual a, ou maior que o segundo argumento.

Void uksort(array matriz, function func_compara);

Usort

Ordena um array pelos valores, utilizando uma função definida pelo programador. A função de comparação deverá retornar um inteiro menor que zero, igual a zero, ou maior que zero, se o primeiro argumento for respectivamente menor que, igual a, ou maior que o segundo argumento.

Void usort(array matriz, function func_compara);

Matemática para números inteiros

Bcpow

Eleva o primeiro número à potência do segundo número. O parâmetro adicional escala significa o número de dígitos após o ponto decimal.

String bcpow(string x, string y, int [escala]);

Bcscale

Muda o valor padrão do parâmetro escala para todas as funções matemáticas subseqüentes que não tenham um valor definido para escala.

String bcscale(int scala)

Datas

Checkdate

Retorna true se uma data for válida; caso contrário, retorna false

Int checkdate(int mês, int dia, int ano);

Date

Retorna uma data no formato especificado

String date(string formato,int[datahora]);

Argumento	Descrição
Data	Data/hora a ser formatada. Se não for especificada utilizará a data/hora corrente
formato	String com caracteres de formatação de datas

Caractere	Significado
a	“am” ou “pm”
A	“AM” ou “PM”
d	Dia em formato numérico. De “01” a “31”.
D	Dia da semana, textual, 3 letras. Ex.: “Fri”
F	Mês, textual, formato longo. Ex.: ”January”.
h	Hora no formato de 12 horas. De “01” a “12”.
H	Hora no formato de 24 horas. De “00” a “23”.
g	Hora no formato de 12 horas, sem os zeros à esquerda. De “1” a “12”.
G	Hora no formato de 24 horas, sem os zeros à esquerda. De “0” a “23”.
i	Minutos. De “00” a “59”.
j	Dia do mês sem os zeros à esquerda. De “1” a “31”.
l	Dia da semana, textual, longo. Ex.: “Friday”.
L	Booleano que indica se o ano é bissexto.”0” ou “1”.
m	Mês. De “01” a “12”.
n	Mês sem zeros à esquerda. De “1” a “12”.
M	Mês, textual, 3 letras. Ex.: “Jan”.
s	Segundos. De “00” a “59”.
S	Sufixo ordinal em inglês, textual, 2 caracteres. Ex.: “th“,”nd”.
t	Número de dias do mês determinado.Ex.:”28” a “31”.
U	Número de segundos desde o “epoch”.

w	Dia da semana, numérico. De “0”(domingo) a “6” (sábado) .
Y	Ano com 4 dígitos. Ex.: “2002”.
y	Ano com 2 dígitos. Ex.: “02”.
z	Dia do ano. De “0” a “365”.
Z	Fuso horário em segundos. De “−43200” a “43200”.

Getdate

Retorna um array associativo contendo todas as informações de uma data/hora específica.

Array getdate(int data/hora);

Índices da matriz retornada.

“seconds”	Segundos
“minutes”	Minutos
“hours”	Horas
“mday”	Dia do mês
“wday”	Dia da semana no formato decimal
“mon”	Mês no formato decimal
“year”	Ano no formato decimal
“yday”	Dia do ano no formato decimal
“weekday”	Dia da semana no formato decimal
“month”	Mês no formato texto

Gettimeofday

Retorna um array associativo contendo as informações da hora corrente

Array gettimeofday(void);

Os índices da matriz são:

“sec”	Segundos
“usec”	Microsegundos
“minuteswest”	Minutos a oeste de Greenwich
“dsttime”	Tipo da correção dst

Gmdate

Retorna uma data/hora GMT no formato UNIX. É idêntica à função mktime(), exceto que a hora retornada é a hora de Greenwich.

String gmdate(string formato, int datahora);

Gmmktime

Retorna a data/hora GMT no formato UNIX. É idêntica à função mktime(), exceto que a data passada no parâmetro representa uma data GMT.

Int gmmktime(int hora, int minuto, int segundo, int mês, int dia, int ano, int [dsf]);

Gmstrftime

Retorna uma data/hora GMT/CUT no formato especificado. É semelhante à função strftime(), exceto que a hora retornada é a hora de Greenwich.

String gmstrftime(string formato, int datahora);

Microtime

Retorna uma string contendo a data/hora atual no formato “msec sec” do UNIX, que significa o número de milisegundos e depois os segundos desde o UNIX Epoch(01/01/1970 00:00:00 GMT).

String microtime(void);

Mktime

Retorna uma data/hora no formato UNIX

Int mktime(int hora, int minuto, int segundo, int mês, ine dia, int ano, int [dsf]);

Strftime

Retorna uma data no formato especificado. Os nomes por extenso dos meses e dos dias da semana dependem da configuração feita com a função setlocale().

String strftime(string formato, int datahora);

Argumento	Descrição
Data	Data/hora a ser formatada. Se não especificada, utilizará a data/hora corrente
Formato	String com caracteres de formatação de datas

Caractere	Significado
%a	Dia da semana abreviado
%A	Dia da semana completo
%b	Nome abreviado do mês
%B	Nome completo do mês
%c	Representação preferida de data e hora
%d	Dia do mês no formato decimal (00−31)
%H	Hora no formato decimal de 24 horas (00−23)
%l	Hora no formato decimal de 12 horas (00−12)
%j	Dia do ano no formato decimal (001−366)
%m	Mês em formato decimal (1−12)
%M	Minutos no formato decimal
%p	‘am’ ou ‘pm’, dependendo da hora especificada
%S	Segundos no formato decimal
%U	Número da semana do ano atual no formato decimal, começando no primeiro domingo como o primeiro dia da primeira semana
%W	Número da semana do ano atual no formato decimal, começando na primeira segunda−feira como o primeiro dia da primeira semana
%w	Dia da semana no formato decimal, sendo o domingo igual a 0.
%x	Representação preferida da data, sem a hora
%X	Representação preferida da hora, sem a data
%y	Ano no formato decimal, sem o século (00−99)
%Y	Ano no formato decimal, com o século.
%Z	Zona de fuso horário, ou nome, ou abreviação.
%%	Uma caractere literal “%”.

Time

Retorna a data/hora no formato UNIX, que corresponde ao número de segundos desde o Unix Epoch(01\01\1970 00:00:00 GMT)

Int time(void);

Diretório

Chdir

Altera o diretório correspondente do PHP para o diretório especificado. Retorna true se tiver sucesso; caso contrário retorna false

Int chdir(string diretório);

Classe dir

É uma classe para leitura de diretórios. O diretório determinado é aberto. Esta classe possui duas propriedades que estarão disponíveis após a abertura do diretório: a propriedade handle, que pode ser usada com outras funções como readdir(), rewinddir() e closedir() e a propriedade path, que contém o caminho do diretório que foi aberto. Três métodos estão disponíveis: read, rewind e close.

New dir(string diretório) ;

Closedir

Encerra a associação de um identificador(handle) com um diretório.

Void closedir(int handle_dir) ;

Opendir

Retorna um handle de diretório para ser usado nas funções closedir(), readdir() e rewinddir().

Int opendir(string diretório);

Readdir

Retorna o próximo nome de arquivo do diretório, na ordem em que estiverem armazenadas.

String readdir(int handle_dir);

Rewinddir

Retorna o ponteiro à posição inicial do diretório

Void rewinddir(int handle_dir);

Execução de Programas

Escapeshellcmd

Retira quaisquer caracteres de um string que poderiam ser utilizados para enganar um comando Shell para executar comandos arbitrários. Esta função normalmente é usada para verificar os dados fornecidos pelo usuário antes de serem passados para as funções exec() ou system().

String escapeshellcmd(string comando);

Exec

Executa um comando externo e mostra a última linha do resultado do comando.

String exec(string comando, string [array], int [variável_ref]) ;

Argumento	Descrição
Comando	Comando externo a ser executado
Array	Array contendo as linhas do resultado
Variável_ref	Variável que conterá o código de retorno do comando executado

Passthru

Executa um comando externo e mostra todos os resultados.

String passthru(string comando, int [variável_ref]);

Argumento	Descrição
Comando	Comando externo a ser executado

Variável_ref	Variável que conterá o código de retorno do comando executado

System

Executa um comando externo e mostra os resultados.

String system(string comando, int [variável_ref]);

Argumento	Descrição
Comando	Comando externo a ser executado
Variável_ref	Variável que conterá o código de retorno do comando executado

Sistema de arquivos do servidor

Basename

Retorna o nome do arquivo em um path. No windows, ambos os caracteres, / e \, são usados como separadores de diretórios. Em outros ambientes é utilizado somente o caractere /.

String basename(string path);

Chgrp

Muda o grupo ao qual um arquivo pertence em ambientes UNIX. Em ambientes windows não faz nada, retornando sempre true.

Ine chgrp(string nome_do_arquivo, mixed grupo) ;

Chmod

Altera as permissões de um arquivo em ambientes UNIX. Retorna true em caso de sucesso; caso contrário, retorna false.

Em ambientes windows esta função não faz nada, sempre retornando true

Int chmod(string nome_do_arquivo, int modo) ;

Argumento	Descrição
Nome_arquivo	Nome do arquivo
Modo	O valor passado como parâmetro deve ser do tipo octal. Ex.: chmod(“/algumdiretorio/algumarquivo”,0755);

Chown

Altera o proprietário de um arquivo em ambientes UNIX. Somente o root pode alterar o proprietário de um arquivo. Retorna true em caso de sucesso; caso contrário, retorna false. No windows esta função na faz nada sempre retornando true.

Int chown(string nome_arquivo, mixed proprietário);

Clearstatcache

Apaga o cache de status dos arquivos utilizados em tempo de execução no PHP

Void clearstatcache(void);

Copy

Copia um arquivo. Retorna true se tiver sucesso; caso contrário, retorna false.

Int copy(string nome_arquivo, string arquivo_destino);

Delete

Apaga um arquivo do servidor

Void delete(string nome_arquivo) ;

Dirname

Retorna o nome de um diretório em um path. Veja a função basename().

String dirname(string path);

Diskfreespace

Retorna o número de bytes disponíveis em um disco.

Int diskfreespace(string nome_arquivo);

Fclose

Fecha um arquivo. Retorna true se tiver sucesso; caso contrário, retorna false.

Int fclose(int fp);

Feof

Retorna true caso o ponteiro de arquivo esteja no fim do arquivo ou ocorra um erro; caso contrário, retorna false.

Int feof(fp);

Fgetc

Retorna um caractere apontado pelo ponteiro do arquivo. Retorna false se for EOF.

String fgetc(int fp);

Fgetcsv

Lê uma linha do arquivo a partir do ponteiro e coloca em um array os campos lidos. Linhas em branco serão convertidas em arrays compreendendo um único campo nulo.

Array fgetcsv(int fp, int tamanho, string [delimitador]);

Fgets

Retorna uma string do arquivo a partir do ponteiro de arquivo até que (tamanho − 1) bytes sejam lidos do arquivo. A leitura termina quando encontra o fim de uma linha ou quando o arquivo termina.

String fgets(int fp, int tamanho);

Fgetss

Lê strings do arquivo como a função fgets(), tirando todas as tags HTML do texto lido.

String fgetss(int fp, int tamanho) ;

File

Lê um arquivo inteiro e coloca seu conteúdo em um array. Cada elemento do array corresponderá a cada linha do arquivo lido, sem tirar qualquer informação.

Array file(string nome_arquivo);

File_exists

Retorna true se um determinado arquivo existe; caso contrário, retorna false. Os resultados desta função são armazenados e reutilizados. Para atualizar esta informação utilize a função clearstatcache().

Int file_exists(string nome_arquivo);

Fileatime

Retorna a data/hora do último acesso ao arquivo, ou false se ocorrer um erro. Para atualizar esta informação utilize a função clearstatcache().

Int fileatime(string nome_arquivo);

Filectime

Retorna a data/hora da última mudança do inode de um arquivo, ou false se ocorrer um erro. Para atualizar esta informação utilize a função clearstatcache().

Int filectime(string nome_arquivo);

Filegroup

Retorna o nome do grupo ao qual pertence o arquivo, ou false em caso de erro. Os resultados desta função são armazenadas e reutilizadas. Para utilizar esta informação utilize a função clearstatcache().

Int filegroup(string nome_arquivo);

Fileinode

Retorna o número do inode de um arquivo, ou false em caso de erro. Os resultados desta função são armazenadas e reutilizadas. Para utilizar esta informação utilize a função clearstatcache().

Int fileinode(string nome_arquivo);

Filemtime

Retorna a data/hora da última alteração de um arquivo, ou false se ocorrer um erro. Para atualizar esta informação utilize a função clearstatcache().

Int filemtime(string nome_arquivo);

Fileowner

Retorna o número de ID do dono de um arquivo, ou false em caso de erro. Os resultados desta função são armazenadas e reutilizadas. Para utilizar esta informação utilize a função clearstatcache().

Int fileowner(string nome_arquivo);

Fileperms

Retorna as permissões de um arquivo, ou false em caso de erro. Os resultados desta função são armazenadas e reutilizadas. Para utilizar esta informação utilize a função clearstatcache().

Int fileperms(string nome_arquivo);

Filesize

Retorna o tamanho de um arquivo, ou false em caso de erro. Os resultados desta função são armazenadas e reutilizadas. Para utilizar esta informação utilize a função clearstatcache().

Int filesize(string nome_arquivo);

Filetype

Retorna o tipo de um arquivo. Os valores possíveis são fifo, char, dir, block, link, file e unknown. Retorna false em caso de erro. Os resultados desta função são armazenadas e reutilizadas. Para utilizar esta informação utilize a função clearstatcache().

String filetype(string nome_arquivo);

Flock

Altera as opções de compartilhamento de um arquivo aberto, fazendo com que todos os programas que precisem usar esse arquivo usem as mesmas opções de compartilhamento. Retorna true se tiver sucesso; caso contrário, retorna false.

Bool flock(int fp, int operacao);

Argumento	Descrição
Operação	Tipo de operação
1	Modo compartilhado para leitura
2	Modo exclusivo para escrita
3	Modo livre (compartilhado ou exclusivo)
4	Adicione 4 para não bloquear o arquivo enquanto estiver alterando o modo

Fopen

Abre um arquivo ou uma URL. A função retornará false se ocorrer erro.

Int fopen(string nome_arquivo, string modo);

Argumento	Descrição
Nome_arquivo	Nome do arquivo. A ação a ser tomada dependerá de como o nome é iniciado. Veja a baixo as opções.
“http://”	Abre um conexão HTTP 1.0 com o servidor especificado e um ponteiro de arquivo é retornado ao início do texto da resposta
“ftp://”	Abre um conexão FTP com o servidor especificado e um ponteiro de arquivo é retornado. O servidor deve suportar o modo passivo de FTP. É possível abrir os arquivos para leitura ou para escrita, mas não ambos ao mesmo tempo
outro	Se o nome começar com qualquer outra coisa, o arquivo será aberto no sistema de arquivos e um ponteiro de arquivo será retornado.
Modo	Modo de abertura. Adicionalmente pode−se colocar a letra “b” no parâmetro modo, informando que a ser processado é um arquivo binário
“r”	Somente leitura, a partir do inicio do arquivo.
“r+”	Leitura e escrita, a partir do inicio do arquivo.
“w”	Somente escrita. A partir do inicio do arquivo e apagando todo o conteúdo do arquivo. Se o arquivo não existir, a função tentará criá−lo.
“w+”	Para leitura e escrita. A partir do inicio do arquivo e apagando todo o conteúdo do arquivo. Se o arquivo não existir, a função tentará criá−lo.
“a”	Somente escrita. A partir do inicio do arquivo. Se o arquivo não existir, a função tentará criá−lo.
“a+”	Para leitura e escrita. A partir do inicio do arquivo. Se o arquivo não existir, a função tentará criá−lo.

Fpassthru

Lê o conteúdo do arquivo, do ponteiro ate o fim do arquivo, mostrando os dados e fechando o arquivo logo em seguida.

Int passthru(int fp);

Fputs

Escreve o conteúdo de uma string em um arquivo

Int fputs(int fp, string str, int [tamanho]);

Fread

Lê bytes de um arquivo

String fread(int fp, int [tamanho]);

Fseek

Muda a posição do ponteiro de arquivo. Só pode ser usada em arquivos do sistema de arquivos local. Retorna 0 se tiver sucesso; caso contrário, retorna –1.

Int fseek(fp, int offset);

Ftell

Retorna a posição do ponteiro do arquivo

Int ftell(int fp);

Fwrite

Escreve um número especifico de bytes de uma string em um arquivo

Int fwrite(int fp, string str, int [numbytes]);

Is_dir

Retorna true se o nome do arquivo especificado existir e for um diretório. Os resultados desta função são armazenadas e reutilizadas. Para atualizar esta informação utilize clearstatcache().

Bool is_dir(string nome_arquivo);

Is_executable

Retorna true se o nome do arquivo especificado existir e for um arquivo executável. Os resultados desta função são armazenadas e reutilizadas. Para atualizar esta informação utilize clearstatcache().

Bool is_executable(string nome_arquivo);

Is_file

Retorna true se o nome do arquivo especificado existir e for um arquivo. Os resultados desta função são armazenadas e reutilizadas. Para atualizar esta informação utilize clearstatcache().

Bool is_file(string nome_arquivo);

Is_link

Retorna true se o nome do arquivo especificado existir e for um link simbólico. Os resultados desta função são armazenadas e reutilizadas. Para atualizar esta informação utilize clearstatcache().

Bool is_link(string nome_arquivo);

Is_readable

Retorna true se o nome do arquivo especificado existir e puder ser lido. Os resultados desta função são armazenadas e reutilizadas. Para atualizar esta informação utilize clearstatcache().

Bool is_readable(string nome_arquivo);

Is writeable

Retorna true se o nome do arquivo especificado existir e puder ser escrito. Os resultados desta função são armazenadas e reutilizadas. Para atualizar esta informação utilize clearstatcache().

Bool is_writeable(string nome_arquivo);

Link

Cria um hard link

Int link(string target, string link);

Linkinfo

Retorna informações sobre um link

Int linkinfo(string path);

Mkdir

Cria um diretório com o modo especificado em formato octal

Retorna true se tiver sucesso; caso contrário, retorna false.

Int mkdir(string pathname, modo);

Pclose

Fecha um ponteiro de arquivo aberto por popen(), retornando o status do processo executado.

Int pclose(int fp);

Popen

Abre um ponteiro para um processo de arquivo

Int popen(string comando, string modo);

Readfile

Lê um arquivo e o envia para a saída padrão do sistema; retorna o número de bytes lidos

Int readfile(string nome_arquivo);

Readlink

Retorna o alvo(target) de um link simbólico

String readlink(string path);

Rename

Altera o nome de um arquivo. Retorna true se tiver sucesso; caso contrário false

Int rename(string nome_anterior, string nome_novo);

Rewind

Muda a posição do ponteiro para o inicio do arquivo

Int rewind(int fp);

Rmdir

Apaga um diretório vazio

Int rmdir(string nome_diretorio);

Set_file_buffer

Define o tamanho do buffer de escrita de arquivos (default = 8192 bytes). Se definido como 0, então as operações de escrita não serão colocadas em buffer.

Stat

Retorna em um array as informações sobre um arquivo

Array stat(string nome_arquivo) ;

Elemento	Conteúdo
1	Device
2	Inode
3	Modo de proteção inode

4	Número de links
5	Identificação do usuário proprietário do arquivo
6	Identificação do grupo proprietário do arquivo
7	Tipo de dispositivo Inode(não disponível no windows)
8	Tamanho(em bytes)
9	Data/hora do último acesso
10	Data/hora da última alteração no arquivo
11	Data/hora da última alteração de estado do arquivo
12	Tamanho de bloco para filesystem I/O (não disponível no windows)
13	Número de blocos alocados

Symlink

Cria um link simbólico entre o alvo existente e o nome especificado

Int symlink(string target, string link);

Tempnam

Retorna um nome de arquivo temporário único no diretório especificado com um prefixo determinado pelo parâmetro prefixo

String tempnam(string diretório, string prefixo);

Touch

Altera a data/hora de modificação de um arquivo. Se o arquivo não existir, será criado. Retorna true se tiver sucesso; caso contrário, retorna false.

Int touch(string nome_arquivo, int time);

Umask

Altera o umask atual. Quando o PHP é um módulo do servidor, o umask é restaurado no fim de cada request.

Int umask(int mask) ;

Unlink

Elimina um nome de arquivo(link)

Int unlink(string nome−arquivo);

Opções e informações do PHP

Error_log

Envia uma mensagem de erro para o registro de erros do servidor web, para uma porta TCP ou para um arquivo

Int error_log(string mensagem, int tipo_mensagem, string [destino], string [headersextras]);

Argumento	Descrição
Mensagem	Mensagem de erro
Tipo_mensagem	Tipo da mensagem
0	A mensagem é enviada para o log do php

1	A mensagem é enviada por email para o endereço indicado no argumento destino
2	A mensagem é enviada através de uma conexão remota de debugging do PHP.
3	A mensagem é adicionada ao arquivo indicado no argumento destino.
Destino	Destino da mensagem. Depende do tipo da mensagem
Headersextras	Argumento opcional utilizado quando o argumento tipo_mensagem for 1.

Error_reporting

Define quais error devem ser reportados pelo PHP

Int error_reporting(int [nível]);

Argumento	Descrição
Nível	Valor obtido com a soma dos códigos dos erros que devem ser reportados
1	E_ERROR – Erros que o PHP não pode ignorar, como falhas de alocação de memória, etc.
2	E_WARNING – Error que o PHP suspeita poderem causar um erro mais grave
4	E_PARSE – Erro de processamento do programa
8	E_NOTICE – Avisos de que certa parte de um código pode causar um erro mais grave
16	E_CORE_ERROR – Erros graves com os E_ERROR, mas são gerados pela base PHP.
32	E_CORE_WARNING – Possíveis erros no código, mas gerados pela base PHP

Extension_loaded

Retorna true se uma extensão foi carregada. Utilize a função phpinfo() para ver os nomes das extensões.

Bool extension_loaded(string nome_extensão);

Get_cfg_var

Retorna o valor de uma opção de configuração do PHP, ou false se ocorrer um erro

String get_cfg_var(string nome_variável) ;

Get_current_user

Retorna o nome do proprietário do script PHP corrente

String get_current_user(void);

Get_magic_quotes_gpc

Retorna o status da opção magic_quotes_gpc do arquivo de configuração php.ini (0 para off, 1 para on).

Long get_magic_quotes_gpc(void) ;

Get_magic_quotes_runtime

Retorna o status da opção magic_quotes_runtime do arquivo de configuração php.ini (0 para off, 1 para on).

Long get_magic_quotes_runtime(void) ;

Getenv

Retorna o valor de uma variável de ambiente, ou false se ocorrer um erro. Para ver a lista de variáveis de ambiente utilize a função phpinfo().

String getenv(string nome_variável);

Getlastmod

Retorna a data/hora da última modificação da página corrente ou false se ocorrer um erro

Int getlastmod(void);

Getmyinode

Retorna o inode do script corrente, ou false se ocorrer um erro.

Int getmyinode(void);

Getmypid

Retorna o ID de processo corrente do PHP, ou false se ocorrer um erro

Int getmypid(void);

Getmyuid

Retorna o ID do usuário do script PHP corrente, ou false ocorrer um erro

int getmyuid(void) ;

Getrusage

Retorna informações da utilização dos recursos atuais em um array

Array getrusage(int [who]);

Phpinfo

Retorna diversas informações sobre o PHP

Int phpinfo(void);

Phpversion

Retorna a versão do PHP instalada.

String phpversion(void);

Putenv

Atribui um valor a uma variável de ambiente

Void putenv(string setting);

Set_magic_quotes_runtime

Altera a configuração atual da opção magic_quotes_runtime do arquivo de configuração php.ini (0 para off, 1 para on)

Long set_magic_quotes_runtime(int nova_conf);

Set_time_limit

Limita o tempo Maximo de execução do script. Se definido como 0(zero) o script será executado indefinidamente.

Void set_time_limit(int num_segundos);

Matemática

Abs

Retorna o valor absoluto de um número.

Mixed abs(mixed número);

Acos

Retorna o arco−coseno de um ângulo em radianos

Float acos(float arg);

Asin

Retorna o arco−seno de um ângulo em radianos

Float asin(float arg);

Atan

Retorna o arco−tangente de um ângulo em radianos

Float atan(float arg);

Atan2

Retorna o arco−tangente de duas variáveis

Float atan2(float y, float x);

Base_convert

Retorna uma string com um número convertido para outra base numérica.

String base_convert(string número, int base_ant, int nova_base);

Bindec

Retorna um número convertido de binário para decimal

Int bindec(string string_binário);

Argumento	Descrição
String_binário	String contendo a representação binária de um número

Ceil

Retorna o próximo número inteiro maior ou igual ao número especificado

Int ceil(float número);

Cos

Retorna o co−seno de um ângulo em radianos

Float cos(float arg);

Decbin

Retorna uma string contendo a representação binária de um número

String decbin(int número);

Dechex

Retorna uma string contendo a representação hexadecimal de um número

String dechex(int número);

Decoct

Retorna uma string contendo a representação octal de um número

String decoct(int número);

Exp

Retorna a constante e elevada à potência especificada

Float exp(float arg);

Floor

Retorna o próximo número inteiro menor ou igual ao número especificado

Int floor(float número);

Getrandmax

Retorna o valor máximo que pode ser gerado pela função rand()

Int getrandmax(void);

Hexdec

Retorna um número convertido de hexadecimal para decimal.

Int hexdec(string string_hexa);

Log

Retorna o logaritmo natural de um número

Float log(float arg) ;

Log10

Retorna o logaritmo base 10 de um número

Float log10(float arg) ;

Max

Retorna o maior dentre os especificados

Mixed max(mixed arg1, mixed arg2, mixed argn);

Min

Retorna o menor dentre os especificados

Mixed min(mixed arg1, mixed arg2, mixed argn);

Mt_rand

Gera um número aleatório mais confiável no intervalo especificado

Int mt_rand(int [limite_inf], int [limite_sup]);

Mt_srand

Altera a semente do gerador de números aleatórios para a função mt_rand()

Void mt_srand(int semente) ;

Mt_getrandmax

Retorna o maior valor possível que a função mt_rand() pode retornar

Int mt_getrandmax(void);

Number_format

Retorna a versão formatada de um número, colocando os separadores de milhares e o separador decimal

String number_format(float número, int decimais, string sep_dec, string sep_milhar);

Octdec

Retorna um numero convertido de octal para decimal

Int octdec(string octal_string);

Pi

Retorna o valor da constante pi

Double pi(void);

Pow

Retorna o resultado de uma base elevada a um expoente

Float pow(float base, float expoente);

Rand

Retorna um número aleatório dentro de um intervalo especificado

Int rand(int [limit_inf], int [limi_sup]);

Round

Retorna o valor de um número arredondado para o número inteiro mais próximo. Se o número estiver exatamente entre dois números inteiros, o resultado será sempre o número inteiro par

Double rand(double número);

Sin

Retorna o seno de um ângulo em radianos

Float sin(float arg);

Sqrt

Retorna a raiz quadrada

Float sqrt(float arg);

Srand

Altera a semente do gerador de números aleatórios para a função

Void srand(int semente);

Tan

Retorna a tangente de um ângulo em radianos

Float tan(float arg);

Criptografia

Mcrypt_cbc

Criptografa/descriptografa dados no modo CBC

Int mcrypt_cbc(int cifra, string chave, string dados, int modo, string vetor_ini);

Mcrypt_cfb

Criptografa/descriptografa dados no modo CFB

Int mcrypt_cfb(int cifra, string chave, string dados, int modo, string vetor_ini);

Mcrypt_create_iv

Cria um vetor de inicialização (vetor_ini) de uma origem aleatória, que é utilizado como argumento nas funções de criptografia.

String mcrypt_create_iv(int tamanhoiv, int origemiv);

Mcrypt_ecb

Criptografa/descriptografa dados no modo ECB

Int mcrypt_ecb(int cifra, string chave, string dados, int modo, string vetor_ini);

Mcrypt_get_cipher_name

Retorna o nome da cifra especificada, ou false se a cifra não existir

String mcrypt_get_cipher_name(int cifra);

Mcrypt_get_block_size

Retorna o tamanho (em bytes) do bloco de uma cifra

Int mcrypt_get_block_size(int cifra);

Mcrypt_get_key_size

Retorna o tamanho (em bytes) da chave de uma cifra

Int mcrypt_get_key_size(int cifra);

Mcrypt_ofb

Criptografa/descriptografa dados no modo OFB

Int mcrypt_ofb(int cifra, string chave, string dados, int modo, string vetor_ini);

Funções diversas

Connection_aborted

Retorna true se um cliente desconectou

Int connection_aborted(void);

Connection_status

Retorna o status de uma conexão

Int connection_status(void);

Connection_timeout

Retorna true se o tempo de execução do script foi excedido

Int connection_timeout(void);

Dl

Carrega, em tempo de execução, uma extensão do PHP definida em library. A diretiva de configuração extension_*dir* do arquivo de configuração php.ini define em qual diretório o PHP deve procurar as extensões carregadas dinamicamente

Int dl(string library) ;

Eval

Processa uma string como código PHP

Void eval(string string_código);

Die

Imprime uma mensagem e termina o script PHP

Void die(string mensagem);

Exit

Termina o script PHP atual

Void exit(void);

Function_exists

Retorna true se uma determinada função foi definida; caso contrário, retorna false

Int function_exists(string nome_função);

Ignore_user_abort

Configura o processador do PHP para não cancelar o script caso o cliente desconecte antes da finalização do mesmo

Int ignore_user_abort(int [config]);

Iptcparse

Retorna uma lista das tags de um bloco binário IPTC (http://www.xe.net/iptc/)

Array iptcparse(string bloco_iptc);

Leak

Limpa a memória desperdiçada

Void leak(int numbytes);

Mail

Envia um email para um ou mais destinatários

Bool mail(string dest, string assunto, string mensagem, string [headers_adic]);

Argumento	Descrição
Dest	Endereço de email dos destinatários(separados por “,”)
Assunto	Assunto do email
Mensagem	Conteúdo do email
Headers_adic	Especifica headers adicionais que devem ser inseridas no fim do header padrão. Múltiplos headers podem ser especificados e devem ser separados por newline(“\n”)

Exemplo:

$destino = “NomeDestino <endereço@dominio.com.br>”;

$remetente = “NomeRemetente <remetente@dominio.com.br>”;

$assunto = “assunto do email”;

mail($destino,”$assunto”, $mensagem, “From: $remetente \n”);

Pack

compacta dados em uma string binária

string pack(string formato, mixed [args]...) ;

Register_shutdown_function

Registra uma função para execução ao término do script

Int register_shutdown_function(string nome_funcao);

Serialize

Gera uma apresentação armazenável de um valor

String serialize(mixed valor);

Sleep

Atrasa a execução por um tempo determinado (em segundos)

Void sleep(int num_segundos)

Unpack

Descompacta dados numa string binária

Array unpack(string formato, string data);

Unserialize

Gera um valor válido a partir de uma string em representação para armazenameto

Mixed unserialize(string str);

Uniquid

Gera um identificador único baseado na hora atual em microsegundos

Int uniquid (string prefixo);

Usleep

Atrasa a execução por um tempo determinado (em microsegundos)

Void usleep(int num_microsegundos);

Rede

Checkdnsrr

Verifica se um nome de servidor ou endereço de Ip existe nos registros de DNS

Int checkdnsrr(string host, string [tipo])

Closelog

Fecha uma conexão ao histórico do sistema (system logger)

Int closelog(void) ;

Debugger_on

Habilita o depurador interno do PHP

Int debugger_on(string endereço);

Debugger_off

Desabilita o depurador interno do PHP

Int debugger_off(void);

Fsockopen

Abre uma conexão com um servidor

Int fsockopen(string hostname, int port, int [errno], string [errstr], double [timeout]);

Gethostbyaddr

Retorna o nome do servidor de um endereço IP

String gethostbyaddr(string endereço_IP);

Gethostbyname

Retorna o endereço IP de um servidor

String gethostbyname(string hostname);

Openlog

Abre uma conexão ao histórico do sistema(system logger) para um programa

Int openlog(string ident, int opção, int facilidade);

Pfsockopen

Abre uma conexão persistente com um servidor

Int pfsockopen(string hostname, int port, int [errno], string [errstr], double [timeout]);

Set_socket_blocking

Define o modo de conexão com o servidor (bloqueado ou não bloqueado)

Int set_socket_blocking(int socket, int modo);

Syslog

Gera uma mensagem de histórico (log) do sistema

Int syslog(int prioridade, string mensagem);

Expressões regulares

Ereg

Expressão regular de busca/comparação.

Int ereg(string expr_regular, string texto, array [regs]);

Argumento	Descrição
Expr_regular	Expressão regular
Texto	Texto para busca
Regs	Opções diversas. Veja “man regex”

Ereg_replace

Expressão de busca/substituição. Retorna a string modificada

String ereg_replace(string expr_regular, string texto_subst, string texto) ;

Argumento	Descrição
Expr_regular	Expressão regular
Texto_subst	Texto para substituição
Texto	Texto a ser substituído

Eregi

Similar à função ereg(), exceto que não é sensível a letras maiúsculas e minúsculas

Int eregi(string expr_regular, string texto, array [regs]);

Eregi_replace

Similar à função ereg_replace(), exceto que não é sensível a letras maiúsculas e minúsculas

String eregi_replace(string expr_regular, string texto_subst, string texto) ;

Split

Expressão regular para dividir uma string em partes e colocar as partes em um array

Array split(string expr_regular, string texto, int [limite]);

Argumento	Descrição
Expr_regular	Expressão regular
Texto	String a ser dividida
Limite	Limitador (caractere que caracteriza os pontos

de divisão na string)

Sql_regcase

Retorna uma expressão regular para busca não−sensível a letras maiúsculas e minúsculas

String sql_regcase(string string);

Tratamento de sessões

Session_decode[PHP4]

Decodifica os dados de uma sessão em uma string

Bool session_decode(string dados);

Session_destroy[PHP4]

Destrói os dados registrados associados à sessão atual

Bool session_destroy(void);

Session_encode[PHP4]

Codifica os dados de uma sessão

Bool session_encode(void) ;

Session_start[PHP4]

Inicializa os dados de uma sessão

Bool session_start(void);

Session_id[PHP4]

Retorna ou muda o identificador da sessão atual

String session_id(string [id]);

Session_is_registered[PHP4]

Descobre se uma variável foi registrada na sessão

Bool session_is_registered(string nome);

Session_module_name[PHP4]

Retorna ou muda o nome do módulo da sessão atual

String session_module_name(string [módulo]);

Session_name[PHP4]

Retorna ou muda o nome da sessão atual

String session_name(string [nome]);

Session_register[PHP4]

Registra uma variável com a sessão atual

Bool session_register(string nome);

Session_save_path[PHP4]

Retorna ou muda o path de gravação da sessão atual

String session_save_path(string [path]);

Session_unregister[PHP4]

Descarta uma variável da sessão atual

Bool session_unregister(string nome);

Strings

Addslashes

Coloca barras invertidas antes dos caracteres especiais: apóstrofo(‘), aspas(“), barra invertida (\) e NUL

String addslashes(string str);

Bin2hex

Converte dados binários em representação hexadecimal

String bin2hex(string str);

Chop

Remove espaços em branco em seqüência

String chop(string str);

Chr

Retorna um caractere especifico dado um código ASCII

String chr(int cód_ascii);

Chunk_split

Divide uma string em pedaços de tamanho “compr”, colocando a string “fim” no fim de cada pedaço e retornando tudo em uma string. O parâmetro “compr” tem valor padrão de 76 e o parâmetro “fim” tem valor padrão de “\r\n” caso esses não sejam especificados.

String chunck_split(string texto, int [compr], string [fim]);

Convert_cyr_string

Converte uma string de um conjunto de caracteres cirílicos em outro

String convert_cyr_string(string str, string de, string para);

Crypt

Retorna uma string criptografada através do modo DES.

String crypt(string str, string [salf]);

Echo

Imprime uma ou mais strings

Echo (string arg1, string argn....);

Explode

Retorna um array contendo as partes da string com valores separados por um separador

Array explode(string separador, string str) ;

Flush

Envia todo o buffer de impressão ao browser cliente

Void flush(void);

Get_meta_tags

Extrai todas as tags <meta> de um arquivo e retorna um array

Array get_meta_tags(string nome_arquivo, int [use_include_path]) ;

Htmlentities

Converte todos os caracteres aplicáveis em tags HTML

String htmlentities(string str);

Htmlspecialchars

Converte caracteres especiais em tags HTML

String htmlspecialchars(string str);

Caracter	Descrição
‘&’	‘&’
‘”’	‘"’
‘<’	‘<’
‘>’	‘>’

Implode

Retorna uma string contendo a representação string de todos os elementos de um array separados pelo argumento glue

String implode(string glue, array fatias);

Join

O mesmo que implode()

String join(string glue, array fatias);

Ltrim

Retorna uma string sem os espaços iniciais

String ltrim(string str);

Md5

Retorna o hash MD5 de uma string

String md5(string str);

Nl2br

Retorna uma string com tags
 inseridos após os caracteres de nova linha

String nl2br(string str);

Ord

Retorna o valor ASCII do primeiro caractere de uma string

Int ord(string str);

Parse_str

Converte uma string de parâmetros no formato das URLs vindas de formulários HTML em variáveis

Void parse_str(string str);

Exemplo:

```
$str = “form=firm&operation=insert&index=0”;

parse_str($str);

echo $form;   //imprime “firm”

echo $operation;   //imprime “insert”

echo $index;   //imprime “0”
```

Print

imprime uma string

```
print(string arg);
```

Printf

Imprime uma string formatada

```
Int printf(string formato, mixed [args]...);
```

Quoted_printable_decode

Converte uma string imprimível em uma string de 8 bits

```
String quoted_printable_decode(string str);
```

Quotemeta

Coloca uma barra invertida antes dos caracteres meta (\\ + * ? [^] ($)) de uma string

String quotemeta(string str);

Rawurldeconde

Decodifica uma URL, retornando uma URL com os caracteres na forma literal

String rawurldecode(string str);

Rawurlenconde

Codifica uma URL, retornando uma URL com os caracteres não−alfanuméricos convertidos em códigos

String rawurlencode(string str);

Setlocale

Altera as informações de localidade

String setlocale(string categoria, string locale);

Similar_text

Calcula a similaridade entre duas strings

Int similar_text(string str1, string str2, double [porcentagem]);

Soundex

Calcula a chave soundex de uma string

String soundex(string str);

Sprintf

Retorna uma string formatada

String Sprintf(string formato, mixed [args]);

Strchr

O mesmo que strstr()

String strchr(string haystack, string needle);

Strcmp

Compara duas strings, retornando um valor: < 0 se str1 for menor que str2; > 0 se str1 for maior que str2, e 0 se elas forem iguais

Int strcmp(string str1, string str2);

Strcspn

Retorna o comprimento do segmento inicial de str1 que não contem nenhum dos caracteres de str2

Int strcspn(string str1, string str2);

Strip_tags

Retorna uma string sem as tags HTML e PHP

String strp_tags(string str);

Stripslashes

Apaga as barras invertidas de caracteres específicos

String stripslashes(string str);

Strlen

Retorna o comprimento de uma string

Int strlen(string str);

Strpos

Retorna a posição da primeira ocorrência de uma string(str2) dentro de outra (str1)

Int strpos(string str1, string str2, int [offset]);

Strrpos

Retorna a posição da última ocorrência de uma string(str2) dentro de outra(str1)

Int strrpos(string str1, char str2);

Strrchr

Retorna a parte do texto que vai do fim da string até a primeira ocorrência do caracter “car”. Retorna false se não houver nenhuma ocorrência de car

String strrchr(string texto, string car);

Strrev

Inverte uma string

String strrev(string str);

Strspn

Retorna o comprimento do segmento inicial de str1 que consiste inteiramente de caracteres em str2

Int strspn(string str1, string str2);

Strstr

Retorna a parte de str1 que vai da primeira ocorrência de str2 em str1 até o fim de str1

String strstr(string str1, string str2);

Strtok

Separa uma string em partes divididas por arg2

String strtok(string arg1, string str2) ;

Strtolower

Transforma as letras em uma string para minúscula.

String strtolower(string str) ;

Strtoupper

Transforma as letras em uma string para maiúscula.

String strtoupper(string str) ;

Str_replace

Substitui todas as ocorrências de uma substring por outra

String str_replace(string str_ant, string novo_str, string str) ;

Argumento	Descrição
Str_ant	Substring a ser substituída
Novo_str	Substring que substituirá a anterior
Str	String original

Strtr

Traduz todas as ocorrências de cada caractere em *de* pelo caractere correspondente em *para* na string str

String strtr(string str, string de, string para);

Substr

Retorna parte de uma string

String substr(string.str, int inicio, int [tamanho]);

Argumento	Descrição
Inicio	posição inicial
Tamanho	Número de caracteres a serem retornados

Trim

Apaga os espaços em branco do inicio e fim de uma string.

String trim(string str);

Ucfirst

Transforma o primeiro caractere de uma string em maiúsculo

String ucfirst(string str) ;

Ucwords

Transforma o primeiro caractere de cada palavra de uma string em maiúsculo

String ucwords(string str);

Funções para variáveis

Doubleval

Retorna o valor em ponto flutuante de uma variável

Double doubleval(mixed var) ;

Empty

Retorna false se var estiver atribuída; caso contrário retorna true

Int empty(mixed var) ;

Gettype

Retorna o tipo de uma variável

String gettype(mixed var);

Intval

Retorna o valor em inteiros de uma variável, utilizando uma base especificada. O padrão da base é 10

Int intval(mixed var, int [base])

Is_array

Retorna true se a variável for do tipo array

Int is_array(mixed var);

Is_double

Retorna true se a variável for do tipo double

Int is_double(mixed var);

Is_float

Retorna true se a variável for do tipo float

Int is_float(mixed var);

Is_int

Retorna true se a variável for do tipo inteiro

Int is_int(mixed var);

Is_integer

O mesmo que is_int()

Int is_integer(mixed var);

Is_long

O mesmo que is_int()

Int is_long(mixed var);

Is_object

Retorna true se a variável for um objeto

Int is_object(mixed var);

Is_real

Retorna true se a variável for do tipo real

Int is_real(mixed var);

Is_string

Retorna true se a variável for do tipo string

Int is_string(mixed var);

Isset

Retorna true se uma variável existir

Int isset(mixed var);

Settype

Altera o tipo de uma variável. Retorna true se tiver sucesso; caso contrário, retorna false.

Int settype(string var, string tipo);

Tipos permitidos
Integer
Double
String
Array
Objeto

Strval

Retorna o valor em string de uma variável

String strval(mixed var);

Unset

Exclui uma variável

Int unset(mixed var);

14. Referências na Internet

Site oficial do PHP	http://www.br.php.net http://www.php.net
Site do projeto Zend que originou o PHP4	http://www.zend.org
Site do servidor web Apache, que é amplamente utilizado e compatível com o PHP	http://www.apache.org
Site da lista principal de discussão de PHP no Brasil	http://www.allfinder.com.br/php
Site com diversos exemplos de PHP. Ótimo para iniciantes	http://www.weberdev.com
Site com diversas classes em PHP para utilizar	http://www.thewebmasters.net/php
Site com diversos artigos e tutoriais	http://www.phpbuilder.com
Artigos e informações sobre PHP para WebMasters	http://www.devshed.com/Server_Side/PHP
Tutorial de PHP com MySQL	http://www.hotwired.com/webmonkey/99/21/index2a.html
Como criar um site de busca, como o Yahoo!, em PHP e MySQL.	http://webreference.com/perl/xhoo/php1
Diversos projetos, tutoriais e informações sobre o PHP	http://www.phpwizard.net
PHP Knowledge Base	http://e−gineer.com/phpkb