
PHP

Sumário

Introdução

O que é PHP?

PHP é uma linguagem que permite criar sites WEB dinâmicos, possibilitando uma interação com o usuário através de formulários, parâmetros da URL e links. A diferença de PHP com relação a linguagens semelhantes a **Javascript** é que o código PHP é executado no servidor, sendo enviado para o cliente apenas html puro. Desta maneira é possível interagir com bancos de dados e aplicações existentes no servidor, com a vantagem de não expor o código fonte para o cliente. Isso pode ser útil quando o programa está lidando com senhas ou qualquer tipo de informação confidencial.

O que diferencia PHP de um script CGI escrito em C ou Perl é que o código PHP fica embutido no próprio HTML, enquanto no outro caso é necessário que o script CGI gere todo o código HTML, ou leia de um outro arquivo.

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.

Como surgiu a linguagem PHP?

A linguagem PHP foi concebida durante o outono de 1994 por **Rasmus Lerdorf**. As primeiras versões não foram disponibilizadas, tendo sido utilizadas em sua *home-page* apenas para que ele pudesse ter informações sobre as visitas que estavam sendo feitas. A primeira versão utilizada por outras pessoas foi disponibilizada em 1995, e ficou conhecida como "**P**ersonal **H**ome **P**age Tools" (**ferramentas para página pessoal**). Era composta por um sistema bastante simples que interpretava alguns macros e alguns utilitários que rodavam "por trás" das *home-pages*: um livro de visitas, um contador e algumas outras coisas.

Em meados de 1995 o interpretador foi reescrito e ganhou o nome de **PHP/FI**. O "FI" veio de outro pacote escrito por Rasmus que interpretava dados de formulários HTML (**F**orm **I**nterpreter). Ele combinou os scripts do pacote *Personal Home Page Tools* com o FI e adicionou suporte a mSQL

(que originou, mais tarde, o MySQL), nascendo assim o PHP/FI, que cresceu bastante, e as pessoas passaram a contribuir com o projeto.

Estima-se que em 1996 PHP/FI estava sendo usado por cerca de 15.000 *sites* pelo mundo, e em meados de 1997 esse número subiu para mais de 50.000. Nessa época houve uma mudança no desenvolvimento do PHP. Ele deixou de ser um projeto de Rasmus com contribuições de outras pessoas para ter uma equipe de desenvolvimento mais organizada. O interpretador foi reescrito por **Zeev Suraski** e **Andi Gutmans**, e esse novo interpretador foram a base para a versão 3.

O lançamento do PHP4, ocorrido em 22/05/2000, trouxe muitas novidades aos programadores de PHP. Uma das principais foi o suporte a sessões, bastante útil pra identificar o cliente que solicitou determinada informação. Além das mudanças referentes a sintaxe e novos recursos de programação, o PHP4 trouxe como novidade um otimizador chamado Zend, que permite a execução muito mais rápida de scripts PHP. A empresa que produz o Zend promete para este ano o lançamento de um compilador de PHP. Códigos compilados serão executados mais rapidamente, além de proteger o código-fonte da aplicação.

Tags de Comentários: Como Documentar Uma Linha

Comentários de linhas de programa podem ser:

- **Barra e asterisco**, que é o estilo da linguagem C:

```
/*  isto e um comentário
    no php    */
```

- **Sinal de escopo**, também chamado de sustenido:

```
# isto é comentário de uma linha
      # e se quiser uma segunda linha aqui esta.
```

- **Duas barras seguidas**:

```
// comentário igual ao do item 2
      // mudou o caracter 'lasanha' por duas barras
```

Ao encontrar estes sinais, o PHP sabe que não deve interpretar as linhas envolvidas por eles por tratar-se de comentários.

Operadores no PHP

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, que é o ponto (**.**), que significa 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”, negação (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

De incremento e decremento

Operadores 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. Observe o exemplo:

```
$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
```

Acesso a Arquivos

Através do PHP é possível ter acesso aos arquivos do sistema, e até arquivos remotos. A seguir veremos alguns dos comandos utilizados para manipular arquivos no PHP.

Copiando Arquivos

Para fazer uma cópia de arquivo utilizando PHP basta utilizar a função copy, desde que o usuário tenha as permissões necessárias para isso. A sintaxe da função copy é a seguinte:

```
bool copy(string origem, string destino);
```

onde bool indica que a função retorna verdadeiro (true) ou falso (false) - verdadeiro em caso de sucesso ou falso em caso de falha na cópia; o parâmetro **`string origem`** indica que o arquivo de origem é passado na forma de uma string e **`string destino`** indica que o arquivo de destino da cópia também será uma string. As strings contendo origem e destino devem conter os caminhos completos.

Verificando o tamanho de um arquivo

A função **filesize** pode ser bastante útil na criação de um script que liste o conteúdo de um diretório, mas também é utilizada em conjunto com a função **fread**, que será vista mais adiante.

```
int filesize(string arquivo);
```

Esta função retorna um inteiro (**`int`**) com o tamanho do arquivo, em bytes, ou false em caso de erro. Como parâmetro, deve-se passar o nome do arquivo, sendo este do tipo string.

Verificando se um arquivo existe

Para evitar erros em tratamento de arquivos, em certos casos é aconselhável verificar se determinado arquivo existe. Para isso devemos utilizar a função **file_exists( )**:

```
int file_exists(string arquivo);
```

Esta função retorna apenas **`true`** ou **`false`**, não informando mais nada sobre o arquivo.

Limpando o cache

Por terem execução lenta, algumas funções que verificam o estado de arquivos utilizam um cache, ou seja, chamadas sucessivas da mesma função com relação ao mesmo arquivo não verificam se houve mudança no mesmo, retornando sempre o mesmo valor. Para eliminar esse cache, obrigando o PHP a reavaliar o valor de retorno de uma função, deve ser utilizada a seguinte função:

```
void clearstatcache();
```

A palavra "void" indica que a função não retorna valor algum. As funções `filesize` e `file_exists` utilizam cache.

Abrindo arquivos para leitura e/ou escrita

Para ler ou escrever num arquivo é preciso antes de qualquer coisa abri-lo. Para isso deve ser utilizada a função fopen, como visto a seguir:

```
int fopen(string arquivo, string modo, [int use_include_path]);
```

A função fopen retorna `false` em caso de erro, e um identificador do arquivo em caso de sucesso. Esse identificador será utilizado em outras funções que manipulam o conteúdo do arquivo. O primeiro argumento é uma string contendo o nome do arquivo; o segundo, o modo como o arquivo será aberto, que pode ser um dos seguintes:

`r`	Abre o arquivo com permissão apenas para leitura.
`r+`	Abre o arquivo com permissão para escrita e leitura, posicionando o ponteiro no início do mesmo.
`w`	Abre o arquivo com permissão apenas para escrita. Se o arquivo existir, todo o conteúdo é apagado. Se não existir, o PHP tenta criá-lo. O ponteiro é posicionado no início do arquivo
`w+`	Abre o arquivo com permissão para escrita e leitura. Se o arquivo existir, todo o conteúdo é apagado. Se não existir, o PHP tenta criá-lo. O ponteiro é posicionado no início do arquivo
`a`	Abre o arquivo com permissão apenas para escrita. Se o arquivo não existir, o PHP tenta criá-lo. O ponteiro é posicionado no final do arquivo
`a+`	Abre o arquivo com permissão para escrita e leitura. Se o arquivo não existir, o PHP tenta criá-lo. O ponteiro é posicionado no final do arquivo.

O ponteiro citado na tabela dos modos de abrir um arquivo refere-se à posição a partir de onde os dados serão lidos e/ou gravados. Para alterar a posição desse ponteiro, pode-se utilizar a função fseek:

```
int fseek(int fp, int posição);
```

onde fp é um identificador de arquivo, retornado da função fopen.

O terceiro parâmetro da função fopen, que pode ter valor `"0"` ou `"1"` indica se o **`include_path`** deverá ser utilizado para localizar o arquivo. O include_path é um parâmetro determinado no arquivo **php.ini** que indica exatamente em quais diretórios determinados arquivos serão procurados.

Além de abrir arquivos localmente, utilizando o sistema de arquivos, a função fopen também permite abrir arquivos remotos, utilizando os protocolos http ou ftp, da seguinte maneira:

- se a string como o nome do arquivo iniciar por "http://" (maiúsculas e minúsculas são iguais), uma conexão é aberta com o servidor e o arquivo contendo o texto de retorno será

aberto. Atenção: qualquer alteração feita no arquivo afetará apenas o arquivo temporário local. O original será mantido.

- se a string como o nome do arquivo iniciar por "ftp://" (maiúsculas e minúsculas são iguais), uma conexão é aberta com o servidor e o arquivo será aberto. Utilizando ftp o arquivo poderá ser aberto para leitura ou escrita, mas não simultaneamente.

Para encerrar a manipulação de um arquivo deve-se utilizar a função fclose, que tem o seguinte formato:

```
int fclose(int fp);
```

onde fp é o identificador do arquivo, retornado pela função fopen.

Lendo um arquivo

```
string fread(int fp, int tamanho);
```

Esta função retorna uma string com o conteúdo do arquivo. O segundo parâmetro determina até onde o arquivo será lido. Se o tamanho determinado for maior que o arquivo, não ocorre erro, tendo como retorno apenas o arquivo. Na maioria dos casos a função **`filesize`** é bastante útil, como no exemplo abaixo:

```
$meuarquivo = "c:/autoexec.bat";

$id = fopen($meuarquivo, "r");

$conteudo = fread($id,filesize($meuarquivo));
```

A função **fread** é "binary-safe", ou seja, pode ser usada para ler o conteúdo de um arquivo binário. Obviamente nesse caso é preciso saber exatamente onde utilizar o valor lido, para não obter resultados indesejados.

Escrevendo em um arquivo

```
int fwrite(int fp, string conteudo, [int tamanho]);
```

Esta função grava num arquivo o conteúdo do segundo parâmetro. Se o tamanho for fornecido e for menor que o tamanho da string, será feita a gravação apenas de uma parte da mesma, determinada pelo terceiro parâmetro.

Para demonstrar a leitura de um arquivo, utilizaremos um exemplo que necessita apenas de uma imagem do tipo GIF, que deve estar no mesmo diretório que nosso script de exemplo:

```
<?

$arquivo = "teste.gif"; /* este nome deve ser alterado para o nome do arquivo a
ser utilizado */

$id = fopen($arquivo, "r"); /* abre o arquivo para leitura */

$conteudo = fread($id,filesize($arquivo)); /* le o conteudo do arquivo e grava
na variavel $conteudo */

fclose($id); /* fecha o arquivo */

header("Content-type: image/gif"); /* esta linha envia um header ao browser
informando que o tipo de arquivo que está sendo enviado é uma imagem no formato
gif */

echo $conteudo; /* esta última linha envia ao browser o conteúdo do arquivo */

?>
```

Para que o exemplo funcione corretamente é preciso que o script seja apenas o que está listado, não podendo haver texto algum (nem mesmo espaço ou linha em branco) antes e depois do script. Visualisando o script pelo browser, teremos a imagem selecionada.

Capturando Data e Hora

O comando **`date`** captura e formata a data e a hora atual. A sintaxe do comando é:

```
string date ( string formato [, int timestamp ] )
```

A função retorna uma **string** de acordo com a string "formato" passada como parâmetro, usando o inteiro **`timestamp`** dado ou, na falta deste, a data e/ou hora atual. O parâmetro opcional *timestamp* é um inteiro longo que contém o números de segundos no formato chamado de **timestamp Unix**, que mostra o número de segundos decorridos desde a meia-noite de 1º de Janeiro de 1970 (January 1 1970 00:00:00 GMT) até a data especificada.

A captura de data atual é feita assim:

```
$varData=date('d/m/Y');
```

e a captura de hora atual é feita assim:

```
$varHora=date('h:m:s');
```

Abaixo, uma demonstração de funções para data e hora, usando funções como **getdate** e **microtime**:

```
<html>
<title>Título</title>
<body>
<br>
<?
     //data_hora.php
$data=date('d/m/Y');
echo "A data é: $data ";
$hora=date('h:i:s');
echo "<br> e a hora e $hora";
$completa=getdate(time('h:i:s'));
echo"<br>o getdata é ($completa[0])";
$micro=microtime();
echo"<br>e o microtime é ($micro)";

?>
</body>
</html
```

No navegador teremos

A data é: 19/06/2006
e a hora e 04:25:00
o getdata é (1150745100)
e o microtime é (0.15664900 1150745101)

Os principais caracteres válidos para *formato* estão descritos a seguir:

Caracteres para formato	Descrição	Exemplo de valores retornados
Para os dias	---	---
d	Dia do mês, 2 digitos com zero à esquerda	*01* até *31*
D	Uma representação textual de um dia, três letras	*Mon* até *Sun*
j	Dia do mês sem preenchimento de zero	*1* até *31*
z	O dia do ano (começando do 0)	*0* through *365*
Para a semana	---	---
l ('L' minúsculo)	A representação completa do dia da semana	*Sunday* até *Saturday*
w	Representação numérica do dia da semana	*0* (para domingo) até *6* (para sábado)
Para o mês	---	---
F	Uma representação completa de um mês, como January ou March (sempre em inglês)	*January* até *December*
M	Representação numérica de um mês, com zero à esquerda	*01* a *12*
M	Uma representação textual curta de um mês, em três letras	*Jan* a *Dec*
N	Representação numérica de um mês, sem zero à esquerda	*1* a *12*
T	Número de dias de um dado mês	*de 28* a *31*
Para o ano	---	---
L	Indica se estamos ou não em ano bissexto	Retorna *1* se estamos em ano bissexto; *0* caso contrário.
Y	Uma representação de ano completa, 4 dígitos	*1999, 2003, 2009, etc.*
y	Uma representação do ano com dois dígitos	*99*, *03, 09, etc.*
Para o tempo	---	---
a	Antes/Depois de meio-dia (em minúsculo)	*am* ou *pm*
A	Antes/Depois de meio-dia (em maiúsculo)	*AM* ou *PM*
h	Formato de 12 horas com zero à esquerda	*01* até *12*
H	Formato de 24 horas com zero à esquerda	*00* até *23*
i	Minutos com zero à esquerda	*00* até *59*
s	Segundos, com zero à esquerda	*00* até *59*

Caracteres não reconhecidos no formato serão impressos como são.

A Função Explode()

Esta é uma função muito útil para dividir uma string em partes, de acordo com um separador especificado. Sua sintaxe é:

```
array explode ( string separador , string texto [, int limite ] )
```

A função retorna uma matriz de strings, cada uma como uma substring da string passada como parâmetro em *texto*. Isso significa que, a cada vez que a função encontra, na string texto, o separador especificado, este novo membro encontrado torna-se uma nova entrada na matriz.

O parâmetro *limite* é opcional e, quando definido, limite o valor máximo de elementos permitidos no array retornado. Se *limite* for negativo, todos componentes, exceto o último, são retornados.

Como exemplo, vamos imaginar que você queira descobrir a qual domínio pertence um determinado endereço de e-mail. Portanto, desejo obter a string que vem depois do sinal de “arroba” (@). Sendo assim, arroba será nosso separador:

```
<?

$email =  “alexandre@terra.com.br”;

$minha matriz = explode(“@”,$email);

print  $minha_matriz[0] . “<BR>”;
print  $minha_matriz[1];

?>
```

O retorno do exemplo acima, no navegador, será:

alexandre
terra.com.br

Outro exemplo: digamos que precisemos pegar uma data no formato "*dd/mm/aaaa*" e tranformá-la em "*aaaa-mm-dd*", que é o padrão aceito pelo banco de dados MySQL para armazenamento de datas. Podemos usar **`explode()`** para já deixar a data (inserida num formulário pelo usuário, por exemplo) formatada para inclusão no banco. Neste caso, nosso separador seria a barra (/):

```
<?

// Variavel $data no padrão dd/mm/aaaa

$data = '01/09/2009';

// Variavel $aux pegando o resultado da separação pelo caracter '/'

$aux = explode(“/”,$data);
```

```
// Exibe a data completa na tela

echo "Data completa: ".$data."\n";

// Exibe a data no formato-padrão do MySQL (aaaa-mm-dd)

echo  "Novo formato: " . $aux[2] . "-" . $aux[1] . "-" . $aux[0];

?>
```

No navegador, teríamos:

Data completa: 01/09/2009
Novo formato: 2009-09-02

Impressão e Saída de Dados

As construções básicas da linguagem para impressão e saída são print e echo. Vejamos um exemplo:

```
<?php

//ECHO


echo "Linha exibida na Tela" , " : Em duas Frases";

echo"<br><br>";
echo ("Linha exibida na Tela" . "   Em duas Frases");

?>
```

No navegador teremos:

Linha exibida na Tela : Em duas Frases

Linha exibida na Tela Em duas Frases

Observe o ponto (**.**) para concatenação para unir strings.

```
<?php

//PRINT

      echo "Multiplicando o numero 3.1415 por 2";
      print "<br><br>";
      print ("3.1415")*2;     //equivalendo a string qdo.entre aspas
      print "<br><br>";
      print (3.1415)*2; //equivalendo a numérico qdo.sem aspas

?>
```

No navegador teremos:

Multiplicando o numero 3.1415 por 2

6.283

6.283

Incluindo Outros Arquivos

Podemos obrigar um determinado arquivo a incluir, dentro de si mesmo, outro arquivo PHP externo, usando as funções **include** e **require**. As sintaxes estão definidas a seguir:

- `include("/filepath/filename”)`
- `require("/filepath/filename”)`

Tanto **include** como **require** têm o efeito de juntar o conteúdo de outros arquivos no código PHP no ponto em que são chamadas. A única diferença entre elas é a maneira como falham se o arquivo chamado não puder ser localizado. A função **include** fará com que um aviso seja exibido, mas o processamento do script continuará, enquanto **require** produzirá um erro fatal caso o arquivo não seja localizado, parando todo o script.

Já as funções abaixo só permitirão a inclusão de um determinado arquivo uma única vez pelo script do PHP. Isto é útil, pois não permite redeclarar funções que se acontecer resultam em erro fatal automático:

- `include_once("/filepath/filename”)`
- `require_once("/filepath/filename”)`

Um exemplo importante para utilizar estas funções é o uso de cabeçalhos e rodapés para as paginas web de um site.

Usando If: Testes, Decisão e Desvios

Exemplo de um if :

```
if (3==2+1)
      print ("acertou, o resultado combina .<br>");
```

ou então:

```
if (3==2+1)
      {
      print ("acertou");
      print ("o resultado combina.<br>");
      }
```

Observação : neste segundo exemplo, o if pode conter um bloco de instruções, desde que dentro de **chaves** (**{ }**)

A seguir, um exemplo utilizando **if..else..** e **else...if**:

```
<?php

$primeiro=10;
$segundo=20;
$terceiro=30;

print ("===primeira parte da resposta===<br><br>");

if (($primeiro < $segundo) and
      ($terceiro > $segundo))

      { print ("primeiro menor que segundo ");
      print ("<br>e terceiro maior que segundo ");
      }
      else
      print ("algum valor nao confere");

      print ("<br>terminada a primeira seq.de if comparacao")
print ("<br><br>===segunda parte da resposta===<br>");


if ($primeiro<$segundo)
      print ("<br><br>primeiro menor que segundo");
elseif ($primeiro==$segundo)
      print ("<br>primeiro igual a segundo<br>");
elseif ($primeiro>$segundo)
      print ("<br>primeiro maior que segundo<br>");
print ("<br>terminada a  seq.de elseif comparacao");

?>
```

Usando Switch Para Facilitar Alguns Testes e Desvios

O **switch** é muito similar a uma série de comandos **if** numa mesma expressão. Em muitos casos, você pode querer comparar uma mesma variável ou expressão com diversos valores, e executar diferentes pedaços de código para cada um. É para isso que o switch existe.

Os exemplos a seguir mostram duas maneiras diferentes de escrever a mesma coisa, uma usando uma série de **if** e **else if** , e outra, usando o **switch**:

```
<?
if ($i == 0) {
    echo "i é 0";
} elseif ($i == 1) {
    echo "i é igual a 1";
} elseif ($i == 2) {
    echo "i é igual a 2";
}

///////agora, usando o switch ////////////////////////////

switch ($i) {
    case 0:
        echo "i é 0";
        break;
    case 1:
        echo "i é igual a 1";
        break;
    case 2:
        echo "i é igual a 2";
        break;
}
?>
```

O próximo exemplo mostra que o switch também permite utilizar strings:

```
<?
switch ($i) {
    case "maça":
        echo "i é uma maçã ";
        break;
    case "laranja":
        echo "i é uma laranja";
        break;
    case "bolo":
        echo "i é um bolo";
        break;
}
?>
```

É importante entender a estrutura do switch para evitar erros. O switch executa linha por linha. De cara, nenhum código é executado. Apenas quando o **`case`** que coincide com o valor testado é encontrado é que o PHP começa, então, a executar todas as demais linhas. O PHP continua a executar todos os códigos até o final do bloco switch. Isso quer dizer que, mesmo tendo já encontrado o `case` correto, ainda assim os outros serão lidos pelo PHP. Desse modo, se você não adicionar o comando **`break`** ao final de cada `case`, o PHP seguirá lendo todos eles. Veja o exemplo:

```
<?
switch ($i) {
    case 0:
        echo "i é 0";
    case 1:
        echo "i é igual a 1";
    case 2:
        echo "i é igual a 2";
}
?>
```

Aqui, se **$i** for igual a 0 (zero), o PHP, ainda assim, executará todos os demais comandos **echo**, exibindo todas as frases na tela; se **$i** for 1, o PHP executará os últimos dois commandos **echo**, imprimindo-os na tela. Portanto, não devemos nos esquecer de acrescentar o comando **break** ao final de cada echo evitando, assim, a execução das demais linhas.

Laços de Repetição

Um laço é uma forma de se produzir um conjunto de procedimentos tantas vezes quantas forem necessárias, segundo a lógica que se estiver usando. Os comandos de laço que veremos são:

for, **while** e **do-while**.

O laço For

A construção de loop mais complicada é o **for**:

```
for(expressão inicial; verificação de termino; expressão de final do loop)
    {
       instruções que compõem o for, entre colchetes.
    }
```

Veja programa a seguir:

```
<Html>
<head>
<title>tabela de divisoes</title>
<!--criado por ivan em maio de 2006-->
</head>
<body bgcolor=lightgreen>

<h2><i><pre>                        tabela de divisao</pre></i></h2>
<table border=25 align=center bgcolor=lightblue>
<?Php
      //for.php
      $start=1;
      $end=10;
      print ("<tr>"); // pula linhas na tabela
      print ("<th> </th>"); //desloca-se para a direita nas celulas

      for ($count=$start;$count<=$end;$count++)
            print ("<th>$count</th>");
      print ("</tr>");
      for ($count=$start;$count<=$end;$count++)
      {
         print ("<tr><th>$count</th>");
         for ($ct=$start;$ct<=$end;$ct++)
      {
         $result=$count/$ct;
         printf("<td>%.3f</td>"
            ,$result);
      }
      print ("</tr>\n");
      }

?>
</table>
<br>
<br>linha inserida apos a primeira tabela<br>
<br>definicao de uma segunda tabela<br><br>
<table border=10 align=left>
<?Php
$star=1;
```

```
$end=30;
      print ("<tr>");
      print ("<th> </th>");
   for ($count=$start;$count<=$end;$count++)

      print ("<th>$count</th>");

?>
</table>
</body>
</html>
```

No navegador teremos:

Tabela de Divisao

	1	2	3	4	5	6	7	8	9	10
1	1.000	0.500	0.333	0.250	0.200	0.167	0.143	0.125	0.111	0.100
2	2.000	1.000	0.667	0.500	0.400	0.333	0.286	0.250	0.222	0.200
3	3.000	1.500	1.000	0.750	0.600	0.500	0.429	0.375	0.333	0.300
4	4.000	2.000	1.333	1.000	0.800	0.667	0.571	0.500	0.444	0.400
5	5.000	2.500	1.667	1.250	1.000	0.833	0.714	0.625	0.556	0.500
6	6.000	3.000	2.000	1.500	1.200	1.000	0.857	0.750	0.667	0.600
7	7.000	3.500	2.333	1.750	1.400	1.167	1.000	0.875	0.778	0.700
8	8.000	4.000	2.667	2.000	1.600	1.333	1.143	1.000	0.889	0.800
9	9.000	4.500	3.000	2.250	1.800	1.500	1.286	1.125	1.000	0.900
10	10.000	5.000	3.333	2.500	2.000	1.667	1.429	1.250	1.111	1.000

linha inserida apos a primeira tabela

definição de uma segunda tabela

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

O laço while

```
while(condicao)

      instruções
```

Exemplo de um programa com **while**:

```
<?php


      $count=1;
      while($count <=10)
            {
            print ("contagem em $count<br>");
            $count=$count+1;
            }
?>
```

O laço do-while

A diferença entre **do-while** e o **while** é que o teste acontece no final do loop. Veja um exemplo de programa abaixo:

```
<?php

$count=8

do
      {
      print ("contador e $count<br>");
      $count=$count+1;
      }
      while ($count<=10)


?>
```

Uploads com formulários HTML

Os formulários HTML têm um tipo de componente utilizado em upload de arquivos e todos os elementos de formulários quando submetidos a scripts PHP criam variáveis com os mesmos nomes. Mas no caso do elemento `"file"`, o tratamento é diferente. Ao ser submetido o formulário, o arquivo uploadeado é gravado num arquivo temporário do disco, que será apagado ao final da execução do script. Além disso, quatro variáveis são criadas no contexto do script PHP:

- $meuarquivo ➔ nome do arquivo temporario criado;
- $meuarquivo_name ➔ nome original do arquivo selecionado pelo usuário;
- $meuarquivo_size ➔ tamanho do arquivo enviado;
- $meuarquivo_type ➔ tipo do arquivo, se esta informação for fornecida pelo browser

Definindo um Formulário

Por ser uma linguagem de marcação, a sintaxe do HTML na maioria dos casos exige uma "tag" de início e uma de final daquele bloco. É Exatamente isso que ocorre com a definição de um formulário: uma tag no início e outra no final, sendo que todos os elementos do formulário devem estar entre as duas tags. Isto torna possível a inclusão de mais de um formulário num mesmo html. As tags citadas são:

```
<form name="" action="" method="" enctype="">
```

Onde temos:

- `name`: o identificador do formulário. Utilizado principalmente em Scripts client-side (JavaScript);
- `action`: nome do script que receberá os dados do formulário ao ser submetido. Mais à frente estão abordadas as maneiras de tratar esses dados recebidos;
- `method`: método de envio dos dados: get ou post;
- `enctype`: formato em que os dados serão enviados. O default é urlencoded. Se for utilizado um elemento do tipo upload de arquivo (file) é preciso utilizar o tipo `multipart/form-data`.

Exemplo:

```
<form action="exemplo.php" method="post">

(textos e elementos do form)
```

```
</form>
```

Cada elemento do formulário deve possuir um nome que irá identificá-lo no momento em que o script indicado no ACTION for tratar os dados.

A tag <input>

Muitos elementos de um formulário html são definidos pela tag `<input>`. Cada tipo de elemento possui parâmetros próprios, mas todos possuem pelo menos dois parâmetros em comum: type, que define o tipo de elemento, e name, que como já foi dito define o nome daquele elemento.

Campo de Texto

```
<input type="text" name="" value="" size="" maxlength="">
```

O campo mais comum em formulários. Exibe na tela um campo para entrada de texto com apenas uma linha.

Parâmetros:

- Value – o valor pré-definido do elemento, que aparecerá quando a página for carregada;
- Size – O tamanho do elemento na tela, em caracteres;
- Maxlength – O tamanho máximo do texto contido no elemento, em caracteres;

Campo de Texto com Máscara

```
<input type="password" name="" value="" size="" maxlength="">
```

Tipo de campo semelhante ao anterior, com a diferença que neste caso os dados digitados são substituídos por asteriscos, e por isso é os mais recomendado para campos que devam conter senhas. É importante salientar que nenhuma criptografia é utilizada. Apenas não aparece na tela o que está sendo digitado.

Parâmetros:

- Value – o valor pré-definido do elemento, que aparecerá quando a página for carregada;
- Size – O tamanho do elemento na tela, em caracteres;
- Maxlength – O tamanho máximo do texto contido no elemento, em caracteres;

Checkbox

```
<input type="checkbox" name="" value="" checked>
```

Utilizado para campos de múltipla escolha, onde o usuário pode marcar mais de uma opção.

Parâmetros:

- Value – o valor que será enviado ao servidor quando o formulário for submetido, no caso do campo estar marcado
- Checked – O estado inicial do elemento. Quando presente, o elemento já aparece marcado;

Radio Button

```
<input type="radio" name="" value="" checked>
```

Utilizado para campos de múltipla escolha, onde o usuário pode marcar apenas uma opção. Para agrupar vários elementos deste tipo, fazendo com que eles sejam exclusivos, basta atribuir o mesmo nome a todos do grupo.

Parâmetros:

- Value – o valor que será enviado ao servidor quando o formulário for submetido, no caso do campo estar marcado
- Checked – O estado inicial do elemento. Quando presente, o elemento já aparece marcado;

Submit Button

```
<input type="submit" name="btEnviar" value="Enviar">
```

Utilizado para enviar os dados do formulário para o script descrito na seção "action" da definição do formulário

Parâmetros:

- Value – o texto que aparecerá no corpo do botão.

Reset Button

```
<input type="reset" name="ntResetar" value="Resetar">
```

Utilizado para fazer todos os campos do formulário retornem ao valor original, quando a página foi carregada. Bastante utilizado como botão "limpar", mas na realidade só limpa os campos se todos eles têm como valor uma string vazia.

Parâmetros:

- Value – o texto que aparecerá no corpo do botão.

Button

```
<input type="button" name="" value="">
```

Utilizado normalmente para ativar funções de scripts client-side (JavaScript, por exemplo). Sem essa utilização, não produz efeito algum

Parâmetros:

- Value – o texto que aparecerá no corpo do botão.

TextArea

```
<textarea cols="" rows="" name="" wrap="">texto</textarea>
```

Exibe na tela uma caixa de texto, com o tamanho definido pelos parâmetros "cols" e "rows".

Parâmetros:

- Cols – número de colunas do campo, em caracteres;
- Rows– número de linhas do campo, em caracteres;
- Wrap – Maneira como são tratadas as quebras de linha automáticas. O valor soft faz com que o texto "quebre" somente na tela, sendo enviado para o servidor o texto da maneira como foi digitado; O valor "hard" faz com que seja enviado para o servidor da maneira como o texto aparece na tela, com todas as quebras de linhas inseridas automaticamente; o valor "off" faz com que o texto não quebre na tela e nem quando enviado ao servidor.
- Value – O elemento do tipo textarea não possui o parâmetro "value". O valor pré-definido do campo é o texto que fica entre as tags <textarea> e </textarea>.

Select

```
<select name="" size="" multiple>

<option value="">texto</option>

</select>
```

Se o parâmetro "size" tiver o valor 1 e não houver o parâmetro "multiple", exibe na tela uma "combo box". Caso contrário, exibe na tela uma "select list".

Parâmetros:

- Size – número de linhas exibidas. Default: 1;
- Multiple – parâmetro que, se presente, permite que sejam selecionadas duas ou mais linhas, através das teclas Control ou Shift;
- option – Cada item do tipo "option" acrescenta uma linha ao select;
- value – Valor a ser enviado ao servidor se aquele elemento for selecionado. Default: o texto do item;
- text – valor a ser exibido para aquele item. Não é definido por um parâmetro, mas pelo texto que fica entre as tags <option> e </option>

Upload de Arquivos

```
<input type="file" name="" size="">
```

Exibe na tela do browser um campo de texto e um botão, que ao clicado abre uma janela para localizar um arquivo no disco. Para utilizar este tipo de componente, o formulário deverá utilizar o método "POST" e ter o parâmetro "**enctype**" com o valor **"multipart/form-data" .**

Parâmetros:

- Size – O tamanho do campo de texto exibido.

Exemplo de Upload

No exemplo que será mostrado aqui, "meu arquivo" (que chegará ao PHP na forma da variável *$meuarquivo*) é o nome de um elemento do formulário.

Para armazenar o conteúdo de um arquivo numa tabela da base de dados ou até num arquivo definitivo (neste segundo caso é mais simples utilizar a função `copy`) podemos utilizar o seguinte script, supondo que o campo do formulário tenha o nome "teste":

```
<?

$id = fopen($teste, "r"); /* abre o arquivo para leitura */

$teste_conteudo = fread($id,filesize($teste)); /* le o conteudo do arquivo e
grava na variavel $conteudo */

fclose($id); /* fecha o arquivo */

?>
```

Com o exemplo acima, teremos o conteudo do arquivo enviado armazenado na string `$teste_conteudo`, podendo assim ser armazenado onde for mais adequado. Vejamos agora um exemplo bem simples de gravação e leitura de um arquivo:

```
<?php

      //ARQUIVOS_CRIANDO.PHP

$teste="c:\xpto.txt";
$id = fopen($teste, "a"); /* abre o arquivo APENAS para escrita */


//captura de data:

$varData=date('d/m/Y');

//captura de hora:

//$varHora=date('h:m:s');
$varHora=date('h:i:s');


$conteudo=("GRAVOU em " . $varData . " no horario  " . $varHora);


fwrite($id, $conteudo,80);

fclose($id); /* fecha o arquivo */

?>
```

Em seguida, lendo o mesmo arquivo:

```
<?
      //ARQUIVOS_LENDO.PHP

      //LENDO

$teste="c:\xpto.txt";
$id = fopen($teste, "a+"); /* abre o arquivo para leitura */
$conteudo = fread($id,filesize($teste)); /* le o conteudo do arquivo e grava na
variavel $conteudo */
print 'arquivo...:';
print $teste;
print "<br>";
PRINT 'conteudo..:';
print $conteudo;

fclose($id); /* fecha o arquivo */



?>
```

No navegador teremos:

arquivo...:c:\xpto.txt
conteudo..:GRAVOU em 19/06/2006 no horario 04:32:40GRAVOU em 19/06/2006 no horario 04:32:43

Obs: O arquivo foi gravado duas vezes, portanto temos dois horários diferentes na seqüência de gravação: as 4:32:40 e as 4:32:43

POST: Passando Variáveis Para Outra Pagina/Formulário

Enviar dados de uma pagina para outra

A rotina abaixo (**workout_calc_var.html**) vai chamar a subseqüente (**wc_handler_var.php**). Veja os parâmetros de transferência da variável chamada **exercise**, que vai ser declarada como **$exercise** na rotina de **wc_handler_var.php**.

Rotina do arquivo "workout_calc_var.html"

```
<html>
<head>
<STYLE TYPE="text/css">
<!--
Body, p {color:black; font-family: verdana; font-size:10 pt}
H1      {color:black; font_family: arial; font_size: 12 pt}
-->
</style>
</head>
<body>
<table border=0 cellpadding=10 width=100%>
<tr>
<TD BGCOLOR="#F0F8FF" ALIGN=CENTER VALIGN=TOP WIDTH=150>
</TD>
<TD BGCOLOR="#FFFFFF" ALIGN=LEFT VALIGN=TOP WIDTH=83%
<H1>Workout calculator (passing a variable)</h1>
<p>Enter an exercise, and we'll tell you how long you'd have to do it<br>to burn
one pound on fat.</p>
<FORM METHOD="post" ACTION= "wc_handler_var.php">
<INPUT TYPE="text" size=50 name="exercise">
<br><br>
<input type="submit" name="submit" value="Burn, baby, burn!">
</FORM>
</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

Rotina wc_handler_var.php:

```
<?php
$exercise= $_post['exercise'];
?>
<html>
<head>
<style type="text/css">
<!--
body, p {color: black; font-family: verdana; font-size: 10 pt}
h1    {color: black; font-family: arial; font-size:12 pt}
-->
```

```
</style>
</head>
<body>

<table border=0 cellpadding=10 width=100%>
<tr>
<td bgcolor="#f0f8ff" align=center valign=top width=150>
</td>
<td bgcolor="#ffffff" align=left valign=top width=83%>
<h1>workout calculator handler, part 1 </h1>
<p>we've successfully passed the contents of the text input field,<br>
as a variable called "exercise" with a value of <b>

<?php
      echo $exercise;
?>

</b>.<br>
but before we can do anything interesting with it, we need to learn about
string.</p>
<br><br>
<font color=darkred size=6 name=arial>
<?php
      print ("observar que a variavel passada<br> para esta pagina
            foi a variavel dolarexercise com o valor de :<br> ==>  $exercise
<==");
?>
</font>
</td>
</tr>
</table>
</body>
</html>
```

Arrays e Funções de Arrays

Arrays do Php

Os arrays em PHP não são exatamente o mesmo que em outras linguagens. Normalmente entende-se um array como um vetor de índices de ordem linear. No PHP os arrays são associativos.

Exemplos:

Arrays de outras linguagens:
Array_comum[0]=”conteúdo de Índice Zero”
Array_comum[1]=”conteúdo de Índice Um”
Array_comum[2]=”conteúdo de Índice Dois”

Arrays do PHP
Array_comum[“Primeiro”]=”conteúdo de Índice Zero”
Array_comum[2]=”conteúdo de Índice Um”
Array_comum[“terceiro”]=”conteúdo de Índice Dois”

Como primeiro exemplo, utilizamos o array tradicional (estilo vetor):

```
<html>
<body>


<?php

// A Seguir 3 formas de carregar o vetor
// Neste exercicio, a terceira carga será res -
// ponsável pela exibição do resultado no
// navegador.

//====================================
//   carga de vetor como string
//====================================


$teste = array("valor1","valor2", "valor3","valor4", "valor5");
//====================================


//====================================
//   carga de vetor passo a passo
//====================================


$teste[0] = "valorX";
$teste[1] = "valor1";
$teste[2] = "valorY";
$teste[3] = "valor3";
$teste[4] = "valorZ";
$teste[5] = "valor5";
//====================================
```

```
//===================================
//   carga de vetor via FOR
//===================================

for ($count=0;$count<=4;$count++)   {
      $teste[$count]=("valor" . $count);
      }
//===================================


//===================================
//   impressao
//===================================

for ($count=0;$count<=4;$count++) {
      echo (" ");
      echo $teste[$count];
      echo (" ");
      }
//===================================
?>
</body>
</html>
```

No navegador teremos:

valor0 valor1 valor2 valor3 valor4

Veremos a seguir, exemplo deste array na estrutura do PHP com índice associativo:

```
$teste[0] = "valorX";
$teste['Primeiro'] = "valor1";
$teste['proximo'] = "valorY";
$teste[3] = "valor3";
$teste['quatro'] = "valorZ";
$teste[5] = "valor5";
```

Neste exemplo, o array tem os valores : "valorX","valor1",..... até "valor5" , cada um dos quais armazenado em associação com uma chave : 0,'Primeiro','proximo'....até 5 .

Portanto, se desejar imitar o comportamento de um array vetorial, simplesmente restrinja o seu código para utilizar somente inteiros seqüenciais como chave.

Array Vazio

Criar um array sem elementos pode ser útil para passar para uma função que espera um array como argumento.

```
$meuArray=array();
```

Funções que retornam arrays

A última forma de criar array em um script é chamando uma função que retorna uma array. A função range().

```
$meuArray=range(1,5);
```

equivale a `$meuArray=array(1,2,3,4,5);`

Agora passamos para o procedimento de recuperação de valores armazenados em Arrays:

Recuperando por Índice

A forma mais direta de recuperar um valor em um array é:

`$recuperado=$meuArray[1];` que nos forneceria o valor : 'maçã'

Recuperando por List

Utilizado para atribuir vários elementos do array a variáveis em sucessão.

Por exemplo:

```
$meuArray = array('maçã','laranja',banana','pêra');

list($primeira,$segunda)=$meuArray;
```

Nesta situação, a variável $primeira irá conter 'maça' e a variável $segunda irá conter 'laranja'.

Arrays Multidimensionais

Agora já podemos começar a estudar os arrays multidimensionais, ou arrays dentro de arrays. Imaginemos primeiro um quadro de dados:

Frutas	**Flores**	**Jóias**	**Animais**
Maçã	Rosa	Cristal	Cavalo
Laranja	Petúnia	Diamante	Gato
.	.	.	.
.	.	.	.
.	.	.	.
Banana	Margarida	Turmalina	Rato
Pêra	Lírio	Ouro	Leão

Em seguida, construímos o programa:

```
<?php
      //array_multiplo.php
      $matriz = array('frutas' => array(0=>'maçã',1=>'laranja') ,
                        'flores' => array(0=>'rosa', 1 =>'petúnia'),
                  'joias'=> array(0=>'cristal',1=>'diamante'),
                  'animais'=> array(0=>'cavalo',1=>'gato'));

      print "<br><br>";
      $index='frutas';
      print ($matriz[$index][1]);

      print "<br><br>";
      $index='flores';
      print ($matriz[$index][1]);

      print "<br><br>";
      $index='joias';
      print ($matriz[$index][0]);

      print "<br><br>";
      //este item esta setando direto no print o valor que viria de $index
      print ($matriz['animais'][1]);

?>
```

No navegador teremos:

laranja
petúnia
cristal
gato

Vejamos outro exercício para melhorar o entendimento sobre array múltiplo:

```
<?php
//array_multiplo2.php
      $matriz=array('1A4','3A12','6A24','9A36');

      for ($count=1;$count<=4;$count++)
            { $matriz['1A4'][$count]=$count;
              $matriz['3A12'][$count]=$count*3;
              $matriz['6A24'][$count]=$count*6;
              $matriz['9A36'][$count]=$count*9;

            }
      PRINT "<BR><BR>";
      print "1A4 CONTEM ";
      for ($count=1;$count<=4;$count++)
             { print "-";
              PRINT $matriz['1A4'][$count];
            }
      print "<br><br>";
      PRINT "3A12 CONTEM";
      for ($count=1;$count<=4;$count++)
              {print "-";
              PRINT $matriz['3A12'][$count];
            }
      print "<br><br>";
      PRINT "6A24 CONTEM";
```

```
      for ($count=1;$count<=4;$count++)
              {print "-";
              PRINT $matriz['6A24'][$count];
             }
      print "<br><br>";
      PRINT "9A36 CONTEM";
      for ($count=1;$count<=4;$count++)
             { print "-";
              PRINT $matriz['9A36'][$count];
             }
?>
```

No navegador teremos:

1A4 CONTEM -1-2-3-4
3A12 CONTEM-3-6-9-12
6A24 CONTEM-6-12-18-24
9A36 CONTEM-9-18-27-36

Inspecionando Arrays

Funções simples para consultar arrays:

is_array()	Aceita um argumento e retorna true se o argumento for uma array e false caso não seja.
count()	Aceita um array como argumento e retorna o número de elementos não vazios no array.
sizeof()	Idêntico a count().
In_array()	Aceita dois argumentos: um elemento(que pode ser um array) e um array(que poderia conter o elemento). Retorna true se o elemento estiver contido como um valor no array, false em caso contrário.
Isset($array[$key])	Aceita uma forma array[chave] e retorna true se a parte "chave" for uma chave válida para o array

Excluindo Elementos de Arrays

Excluir um elemento do array é simples como eliminar uma variável atribuída; basta usar função **unset( )**. Veja o exemplo:

My_array[0]='inicio';
My_array[1]='primeiro';
My_array[2]='segundo';

Em seguida, usamos: **`unset(my_array[1]);`**

Portanto restaria:

My_array[0]=’inicio’;
My_array[2]=’segundo’;

Observe que não permacene my_array[1]=’’, ou seja , um array vazio. Isto só ocorreria se tivéssemos feito **$my_array[1] = ’’**.

Iteração – Técnicas para Lidar Com Elementos do Array em Massa

Utilizando Funções

Primeiro criamos um array e armazenamos nele nomes de cidades em associação com índices numéricos. Ainda devemos armazenar os nomes dos países indexados pelo nome das cidades:

```
<?php
            //iteracao_cidades.php
            //array iteracao

            $cidade=array();
            $cidade[0]='Brasília';
            $cidade['Brasília']='Brasil';
            $cidade[1]='Buenos_Aires';
            $cidade['Buenos Aires']='Argentina';
            $cidade[2]='Montevidéu';
            $cidade['Montevidéu']='Uruguai';



      //==>Agora podemos usar o sistema de chave de array para extrair os dados.

            Function cidade_numero($numero,$nome)
            {
               if (isset($nome[$numero]))
               {
                  $a_cidade=$nome[$numero];
                  $pais=$nome[$a_cidade];
                  print("$a_cidade  localização==>  $pais<Br>");
                }
            }


            cidade_numero(0,$cidade);
            cidade_numero(1,$cidade);
            cidade_numero(2,$cidade);

?>
```

No navegador teremos:

Brasília localização==> Brasil
Buenos_Aires localização==> Argentina
Montevidéu localização==> Uruguai

Este método funciona quando conhecemos as chaves. Entretanto, quando precisamos imprimir tudo que um array contém sem conhecermos suas chaves, devemos usar outra construção: **`foreach`**.

Vamos utilizá-la então:

```
<?php
      //iteracao_cidades_foreach.php
      //Versão 1

            $cidade=array();
            $cidade[0]='Brasília';
            $cidade['Brasília']='Brasil';
            $cidade[1]='Buenos_Aires';
            $cidade['Buenos_Aires']='Argentina';
            $cidade[2]='Montevidéu';
            $cidade['Montevidéu']='Uruguai';



      Function imprime_tudo_foreach($cidade_enviada)
            {
            foreach($cidade_enviada as $nome){
                  print ("$nome<Br>");
                  }
            }
      print imprime_tudo_foreach($cidade);



?>
```

No navegador teremos:

Brasília
Brasil
Buenos_Aires
Argentina
Montevidéu
Uruguai

Observe na versão a seguir uma mudança na descrição da chave dentro dos parênteses do foreach:

```
<?php
      //iteracao_cidades_foreach_vs2.php
      //Versão 2

            $cidade=array();
            $cidade[0]='Brasília';
            $cidade['Brasília']='Brasil';
```

```
            $cidade[1]='Buenos_Aires';
            $cidade['Buenos_Aires']='Argentina';
            $cidade[2]='Montevidéu';
            $cidade['Montevidéu']='Uruguai';



      Function imprime_tudo_foreach($cidade_enviada)
            {
            foreach($cidade_enviada as $nome => $VALUE){
                  print ("$nome<Br>");
                  }
            }

      print imprime_tudo_foreach($cidade);



?>
```

No navegador teremos:

0
Brasília
1
Buenos_Aires
2
Montevidéu

Iterando com current() , next() , reset() ....

```
<?php
//PROGRAMA ==> testes_multiplos_iteracao.php


      //Preenchendo o array

            $cidade=array();
            $cidade[0]='Brasília';
            $cidade['Brasília']='Brasil';
            $cidade[1]='Buenos_Aires';
            $cidade['Buenos_Aires']='Argentina';
            $cidade[2]='Montevidéu';
            $cidade['Montevidéu']='Uruguai';
//=======================================================

      //Utilizando current() e next() - Registro atual e proximo
      //PASSAGEM POR VALOR

      function imprime_proxima($cidade)
      { //atenção-- não funciona bem em caso de valor false
        //veja a função each()

            $atual=current($cidade);
            if ($atual)
                  print("$atual<br>");
            else
```

```
                  print("Não ha nada para imprimir");
            while ($atual=next($cidade))
                  print("$atual<br>");
      }
      print("============================================================");
      Print "<br><br>";
      print ("<b>Veja acima desta linha os warnings (avisos) que
ocorreram<br>");
      print ("e observe que mais abaixo vamos comentar o motivo</b><br><br>");
      print ("<b>Listaremos agora o array cidades 2 vezes</b>");
      Print "<br><br>";
      imprime_proxima($cidade);
      print "<br>";
      imprime_proxima($cidade);
      print ("<B><br>Observe que listamos 2 vezes e o ponteiro voltou<br>");
      print ("para o inicio do array ao iniciar a 2 lista.<br></B>");

//=========================================================
      //PASSAGEM POR REFERENCIA

      print ("<B><br> A seguir listaremos passando os arrays por
referencia<Br>");
      print ("ao inves de passar por valor.Note o & (e comercial na <br>");
      print ("chamada da função<br>");
      print ("Esta execução está obsoleta, portanto junto com a lista <br>");
      print ("de cidades e paises, voce verá Warning (avisos)<br><br></B>");

      imprime_proxima(&$cidade);
      print "<br>";
      imprime_proxima(&$cidade);

//=========================================================
      //Utilizando reset() - Reiniciando o Array
      reset($cidade);
      print "<br><br>";
      print ("<b>Agora estamos utilizando reset()</b>");
      print "<br><br>";
      imprime_proxima(&$cidade);
      print "<br>";
      reset($cidade);
      imprime_proxima(&$cidade);

//=========================================================
      //Utilizando end() & prev() - Ordem Inversa



      function imprime_anterior($cidade)
      { //atenção-- não funciona bem em caso de valor false
        //veja a função each()

            $atual=end($cidade);
            if ($atual)
                  print("$atual<br>");
            else
                  print("Não ha nada para imprimir");
            while ($atual=prev($cidade))
                  print("$atual<br>");
      }


      print "<br><br>";
```

```
      print ("<b>Agora estamos utilizando end()</b>");
      print "<br><br>";
      imprime_anterior(&$cidade);
      print "<br>";

      print "<br><br><b>Imprimindo pela segunda vez</b><br>";
      imprime_anterior(&$cidade);
      print "<br>";

//=========================================================
      //Extraindo as chaves com key()

      function imprime CHAVES($cidade)
      { //veja a função each()
            reset($cidade);
            $atual=current($cidade);
            $chave=key($cidade);
            if ($atual)
                  print("chave atual :$chave; CONTÉM: $atual<br>");
            else
                  print("Não ha nada para imprimir");
            while ($atual=next($cidade))
            {     $chave=key($cidade);
                  print("chave atual :$chave;CONTÉM: $atual<br>");
            }
      }
      imprime_CHAVES($cidade);


//=========================================================
      // FUNÇÃO EACH() - VALORES VAZIOS

      PRINT "<BR><BR>";
      PRINT ("<B>A SEGUIR VAMOS FALSIFICAR UM VALOR DO ARRAY PARA MOSTRAR A");
      PRINT (" DIFERENÇA QUE OCORRE AO UTILIZARMOS A FUNÇÃO EACH()<br>");
      print ("Observe que a função não para prematuramente se um valor for falso
ou vazio</B><BR><BR>");

      //Preenchendo o array com um item falso

            $cidade=array();
            $cidade[0]='Brasília';
            $cidade['Brasília']='Brasil';
            //$cidade[1]='Buenos_Aires'; INCLUIMOS FALSE NESTE ITEM
            $cidade[1]="";
            $cidade['Buenos_Aires']='Argentina';
            $cidade[2]='Montevidéu';
            $cidade['Montevidéu']='Uruguai';



      function imprime_each($cidade)
      { //imprima confiavelmente tudo que existir no array

            reset($cidade);

            while ($celula=each($cidade))
            {     $atual=$celula['value'];
                  $chave=$celula['key'];
                  print("chave atual :$chave;CONTÉM: $atual<br>");
            }
      }
```

```
	imprime_each($cidade);

?>
```

No navegador teremos:

Inicio da página do navegador

Warning: Call-time pass-by-reference has been deprecated - argument passed by value; If you would like to pass it by reference, modify the declaration of imprime_proxima(). If you would like to enable call-time pass-by-reference, you can set allow_call_time_pass_reference to true in your INI file. However, future versions may not support this any longer. in **c:\arquivos de programas\easyphp1-8\www\programas_do_manual_php\testes_multiplos_iteracao.php** on line **52**

Warning: Call-time pass-by-reference has been deprecated - argument passed by value; If you would like to pass it by reference, modify the declaration of imprime_proxima(). If you would like to enable call-time pass-by-reference, you can set allow_call_time_pass_reference to true in your INI file. However, future versions may not support this any longer. in **c:\arquivos de programas\easyphp1-8\www\programas_do_manual_php\testes_multiplos_iteracao.php** on line **54**

Warning: Call-time pass-by-reference has been deprecated - argument passed by value; If you would like to pass it by reference, modify the declaration of imprime_proxima(). If you would like to enable call-time pass-by-reference, you can set allow_call_time_pass_reference to true in your INI file. However, future versions may not support this any longer. in **c:\arquivos de programas\easyphp1-8\www\programas_do_manual_php\testes_multiplos_iteracao.php** on line **62**

Warning: Call-time pass-by-reference has been deprecated - argument passed by value; If you would like to pass it by reference, modify the declaration of imprime_proxima(). If you would like to enable call-time pass-by-reference, you can set allow_call_time_pass_reference to true in your INI file. However, future versions may not support this any longer. in **c:\arquivos de programas\easyphp1-8\www\programas_do_manual_php\testes_multiplos_iteracao.php** on line **65**

Warning: Call-time pass-by-reference has been deprecated - argument passed by value; If you would like to pass it by reference, modify the declaration of imprime_anterior(). If you would like to enable call-time pass-by-reference, you can set allow_call_time_pass_reference to true in your INI file. However, future versions may not support this any longer. in **c:\arquivos de programas\easyphp1-8\www\programas_do_manual_php\testes_multiplos_iteracao.php** on line **91**

Warning: Call-time pass-by-reference has been deprecated - argument passed by value; If you would like to pass it by reference, modify the declaration of imprime_anterior(). If you would like to enable call-time pass-by-reference, you can set allow_call_time_pass_reference to true in your INI file. However, future versions may not support this any longer. in **c:\arquivos de programas\easyphp1-**

8\www\programas_do_manual_php\testes_multiplos_iteracao.php on line **95**
==========

**Veja acima desta linha os warnings (avisos) que ocorreram
e observe que mais abaixo vamos comentar o motivo**

Listaremos agora o array cidades 2 vezes

Brasília
Brasil
Buenos_Aires
Argentina
Montevidéu
Uruguai

Brasília
Brasil
Buenos_Aires
Argentina
Montevidéu
Uruguai

**Observe que listamos 2 vezes e o ponteiro voltou
para o inicio do array ao iniciar a 2 lista.**

**A seguir listaremos passando os arrays por referencia
ao inves de passar por valor.Note o & (e comercial na
chamada da função
Esta execução está obsoleta, portanto junto com a lista
de cidades e paises, voce verá Warning (avisos)**

Brasília
Brasil
Buenos_Aires
Argentina
Montevidéu
Uruguai

Não ha nada para imprimir

Agora estamos utilizando reset()

Brasília
Brasil
Buenos_Aires
Argentina
Montevidéu
Uruguai

Brasília
Brasil

Buenos_Aires
Argentina
Montevidéu
Uruguai

Agora estamos utilizando end()

Uruguai
Montevidéu
Argentina
Buenos_Aires
Brasil
Brasília

Imprimindo pela segunda vez
Uruguai
Montevidéu
Argentina
Buenos_Aires
Brasil
Brasília

chave atual :0; CONTÉM: Brasília
chave atual :Brasília;CONTÉM: Brasil
chave atual :1;CONTÉM: Buenos_Aires
chave atual :Buenos_Aires;CONTÉM: Argentina
chave atual :2;CONTÉM: Montevidéu
chave atual :Montevidéu;CONTÉM: Uruguai

A SEGUIR VAMOS FALSIFICAR UM VALOR DO ARRAY PARA MOSTRAR A DIFERENÇA QUE OCORRE AO UTILIZARMOS A FUNÇÃO EACH()
Observe que a função nao para prematuramente se um valor for falso ou vazio

chave atual :0;CONTÉM: Brasília
chave atual :Brasília;CONTÉM: Brasil
chave atual :1;CONTÉM:
chave atual :Buenos_Aires;CONTÉM: Argentina
chave atual :2;CONTÉM: Montevidéu
chave atual :Montevidéu;CONTÉM: Uruguai

Fim da página do navegador

Transformações de Arrays

Recuperando CHAVES E VALORES

Podemos recuperar as chaves de um array usando **array_keys( )**. A sintaxe desta função é

```
array array_keys ( array input [, mixed search_value])
```

Desse modo, podemos entender que **array_keys()** retorna as chaves (numéricas e string) do array passado como entrada (*array input*). Se o parâmetro opcional *search_value* for especificado, então apenas as chaves para esse valor serão retornadas. Do contrário, todas as chaves de *input* serão retornadas. Observe o exemplo:

```
<?php
$array = array(0 => 100, "cor" => "vermelho");
print_r(array_keys($array));

$array = array("azul", "vermelho", "verde", "azul", "azul");
print_r(array_keys($array, "azul"));

$array = array("cor" => array("azul", "vermelho", "verde"), "tamanho" =>
array("pequeno", "medio", "grande"));
print_r(array_keys($array));

?>
```

A saída deste programa seria:

```
Array
(
    [0] => 0
    [1] => cor
)
Array
(
    [0] => 0
    [1] => 3
    [2] => 4
)
Array
(
    [0] => cor
    [1] => tamanho
)
```

Portanto, a função array_keys() retorna as chaves de um array de entrada na forma de um novo array, onde armazena as chaves retornadas, iniciando em zero. Temos uma outra função, **array_values()**, que faz exatamente a mesma coisa, exceto pelo fato de armazenar os valores do array original. Vejamos um exemplo prático:

```
<?php

      //transformando_arrays.php

      //CRIANDO O ARRAY
      $pizza_requests=array('MICHELE' => 'MUSSARELA',
      'KATIA' => 'QUATRO QUEIJOS',
      'IVONE' => 'TOMATE SECO',
      'CAMILA' => 'MARGUERITA',
      'JACKELINE' => 'NAPOLITANA');

      function print_KV_each($x)
      {     reset ($x);
            while($array_cell = each($x))
            {
            $currente_value=$array_cell['value'];
            $currente_key=$array_cell['key'];
            print ("chave:$currente_key; Valor:$currente_value<bR>");

            }
      }


      //IMPRESSÃO
      PRINT ("Chaves do array:<br>");
      print_KV_each(array_keys($pizza_requests));

      print ("<br>Valores do Array:<br>");
      print_KV_each(array_values($pizza_requests));

?>
```

No navegador teremos:

Chaves do array:
chave:0; Valor:MICHELE
chave:1; Valor:KATIA
chave:2; Valor:IVONE
chave:3; Valor:CAMILA
chave:4; Valor:JACKELINE

Valores do Array:
chave:0; Valor:MUSSARELA
chave:1; Valor:QUATRO QUEIJOS
chave:2; Valor:TOMATE SECO
chave:3; Valor:MARGUERITA
chave:4; Valor:NAPOLITANA

Virando,Invertendo,Embaralhando

As funções **array_flip()** e **array_reverse()** transformam as chaves de um array em valores e vice-versa. Por exemplo:

```
<?php
      //virando_invertendo_arrays.php
```

```
     //PEDINDO PIZZAS :

     //CRIANDO O ARRAY
     $pizza_requests=array('MICHELE' => 'MUSSARELA',
     'KATIA' => 'QUATRO QUEIJOS',
     'IVONE' => 'TOMATE SECO',
     'CAMILA' => 'MARGUERITA',
     'JACKELINE' => 'NAPOLITANA');

     function print_KV_each($x)
     {     reset ($x);
          while($array_cell = each($x))
          {
          $currente_value=$array_cell['value'];
          $currente_key=$array_cell['key'];
          print ("chave:$currente_key; Valor:$currente_value<bR>");

          }
     }
     print "<br><br>";

     //IMPRESSÃO
     print ("<b>Invertendo  chave e valor</b><br>");
     PRINT ("Chaves do array:<br>");
     print_KV_each(array_flip($pizza_requests));

     print "<br><br>";


     print ("<b>Revertendo a lista</b><br>");
     PRINT ("Chaves do array:<br>");
     print_KV_each(array_reverse($pizza_requests));

?>
```

No navegador teremos:

Invertendo chave e valor
Chaves do array:
chave: MUSSARELA; Valor: MICHELE
chave: QUATRO QUEIJOS; Valor: KATIA
chave: TOMATE SECO; Valor: IVONE
chave: MARGUERITA; Valor: CAMILA
chave: NAPOLITANA; Valor: JACKELINE

Revertendo a lista
Chaves do array:
chave:JACKELINE; Valor:NAPOLITANA
chave:CAMILA; Valor:MARGUERITA
chave:IVONE; Valor:TOMATE SECO
chave:KATIA; Valor:QUATRO QUEIJOS
chave:MICHELE; Valor:MUSSARELA

Usando PHP com Banco de Dados

Estabelecendo conexões

Para acessar bases de dados num servidor MySQL, é necessário antes estabelecer uma conexão. Para isso, deve ser utilizado o comando `mysql_connect`, ou o `mysql_pconnect`. A diferença entre os dois comandos é que o mysql_pconnect estabelece uma conexão permanente, ou seja, que não é encerrada ao final da execução do script. As assinaturas dos dois comandos são semelhantes, como pode ser verificado a seguir:

```
int mysql_connect(string [host[:porta]] , string [login] , string [senha] );

int mysql_pconnect(string [host[:porta]] , string [login] , string [senha] );
```

O valor de retorno é um inteiro que identifica a conexão, ou falso se a conexão falhar. Antes de tentar estabelecer uma conexão, o interpretador PHP verifica se já existe uma conexão estabelecida com o mesmo host, o mesmo login e a mesma senha. Se existir, o identificador desta conexão é retornado. Senão, uma nova conexão é criada.

Uma conexão estabelecida com o comando `mysql_connect` é encerrada ao final da execução do script. Para encerrá-la antes disso deve ser utilizado o comando `mysql_close`, que tem a seguinte assinatura:

```
int mysql_close(int [identificador da conexão] );
```

Se o identificador não for fornecido, a última conexão estabelecida será encerrada.

Selecionando a base de dados

Depois de estabelecida a conexão, é preciso selecionar a base de dados a ser utilizada, através do comando `mysql_select_db`, que segue o seguinte modelo:

```
int mysql_select_db(string base, int [conexao] );
```

Novamente, se o identificador da conexão não for fornecido, a última conexão estabelecida será utilizada.

Realizando consultas

Para executar consultas SQL no MySQL, utiliza-se o comando `mysql_query`, que tem a seguinte assinatura:

```
int mysql_query(string query, int [conexao] );
```

Onde query é a expressão SQL a ser executada, sem o ponto-e-vírgula no final, e conexao é o identificador da conexão a ser utilizada. A consulta será executada na base de dados selecionada pelo comando **mysql_select_db**.

É bom lembrar que uma consulta não significa apenas um comando SELECT. A consulta pode conter qualquer comando SQL aceito pelo banco.

O valor de retorno é falso se a expressão SQL for incorreta, e diferente de zero se for correta. No caso de uma expressão SELECT, as linhas retornadas são armazenadas numa memória de resultados, e o valor de retorno é o identificador do resultado. Alguns comandos podem ser realizados com esse resultado:

Apagando o resultado

```
int mysql_free_result(int result);
```

O comando mysql_free-result deve ser utilizado para apagar da memória o resultado indicado.

Número de linhas

```
int mysql_num_rows(int result);
```

O comando mysql_num_rows retorna o número de linhas contidas num resultado.

Utilizando os resultados

Existem diversas maneiras de ler os resultados de uma query SELECT. As mais comuns serão vistas a seguir:

```
int mysql_result(int result, int linha, mixed [campo] );
```

Retorna o conteúdo de uma célula da tabela de resultados, onde:

- `result` é o identificador do resultado;
- `linha` é o número da linha, iniciado por 0 (zero);
- `campo` é uma string com o nome do campo, ou um número correspondente ao número da coluna. Se for utilizado um alias na consulta, este deve ser utilizado no comando **mysql_result.**

Este comando deve ser utilizado apenas para resultados pequenos. Quando o volume de dados for maior, é recomendado utilizar um dos métodos a seguir:

```
array mysql_fetch_array(int result);
```

Lê uma linha do resultado e devolve um array, cujos índices são os nomes dos campos. A execução seguinte do mesmo comando lerá a próxima linha, até chegar ao final do resultado.

```
array mysql_fetch_row(int result);
```

Semelhante ao comando anterior, com a diferença que os índices do array são numéricos, iniciando pelo 0 (zero).

Alterando o ponteiro de um resultado

```
int mysql_data_seek(int result, int numero);
```

Cada resultado possui um "ponteiro", que indica qual será a próxima linha lida com o comando **`mysql_fetch_row` (ou `mysql_fetch_array`).** Para alterar a posição indicada por esse ponteiro deve ser utilizada a função `mysql_data_seek`, sendo que o número da primeira linha de um resultado é zero.

Vejamos um exemplo simples de Conexão e acesso ao banco MYSQL:

```
<body bgcolor=gold>
<?php
      //conexao_db_query.php

//ESTA ROTINA, CONECTA AO BANCO, SELECIONA UMA TABELA
//E EXIBE SEUS DADOS.

//O BANCO USADO FOI O XTESTE E A TABELA CADASTRO.
      $conexao=mysql_connect("127.0.0.1","root","");
print ("CONEXAO RETORNA COM : $conexao");

      mysql_select_db("xteste",$conexao);
      $consulta="SELECT * from cadastro"; // where codigo like 2";
      $resultado= mysql_query($consulta, $conexao);
      print ("<BR>RESULTADO DA QUERY = :$resultado");
      print ("<br>");
      $xlinhas= mysql_num_rows($resultado);
      print ("linhas =  :$xlinhas<br>");
      //printf("NOME  : ", mysql_result($resultado,0,"nome"),"<BR>\n");

      echo ("<br><br>CADASTRO DE PESSOAS  <br><BR>");

      echo "<table border=1>\n";
      echo "<tr><td><font color=red>CODIGO</td><td><font color=blue>N  O  M  E
<td><font color=green>TELEFONE</TD></tr>\n";

      //while ($linhas = $xlinhas)
      for ($linha=1;$linha<=$xlinhas;$linha++)
      {
            $zlinha=mysql_fetch_row($resultado);
            printf("<tr><td><font color=red>$zlinha[0]</td>");
            printf("<td><font color=blue>$zlinha[1]</td>");
            printf("<td><font color=green>$zlinha[2]</td></tr>");
      }

      echo "</table>\n";
?>
```

Veja a seguir o resultado no navegador:

CONEXAO RETORNA COM : Resource id #2
RESULTADO DA QUERY = :Resource id #3
linhas = :3

CADASTRO DE PESSOAS

CODIGO N O M E TELEFONE
1 IVAN 38759449
2 LAIS 38759449
3 YURI 38011633

Enviando e-mails via PHP

A função **mail()** automaticamente envia uma mensagem de e-mail especificada para um determinado destinatário. Eis a sintaxe da função:

```
bool mail ( string to, string subject, string message [, string
            additional_headers [, string additional_parameters]] )
```

Em **`to`**, devemos informar o destinatário do e-mail. Destinatários múltiplos podem ser especificados colocando uma vírgula entre cada endereço. A mensagem deve ser passada no parâmetro **`message`** e o assunto em **`subject`**. Os parâmetros **`additional_headers`** e **`additional_parameters`** são opcionais.

A função **mail()** devolve **`true`** se o e-mail foi enviado com sucesso, e **`false`** caso contrário.

Atenção: A implentação do Windows de **mail()** difere bastante da implentação Unix. Primeiro, ele não usa um binário local para compor mensagens mas apenas opera com sockets diretos o que significa que uma *MTA* é necessária monitorando um socket de rede (que pode ser ou o localhost ou uma máquina remota). Segundo, os cabeçalhos personalizados como *From:*, *Cc:*, *Bcc:* e *Date:* são **not** interpretados por *MTA* em primeiro lugar, mas são analisados pelo *PHP*. *PHP* < 4.3 somente elementos suportados *Cc:* elemento de cabeçalho (e foi caso-sensitivo). *PHP* >= 4.3 suporta todos os elementos de cabeçalho mencionados e não mais caso-sensitivo.

Exemplo 1: Enviando e-mail.

```
<?php
mail(jon@exemplo.com.br, "meu assunto", "isto é um teste de e-mail");
?>
```

Se uma string como quarto argumento é passada, esta string é inserida no fim do cabeçalho. É usado tipicamente para adicionar cabeçalhos extras. Cabeçalhos extras múltiplos são separados com sinal de retorno e nova linha.

Exemplo 2: Enviando e-mail com cabeçalhos extras

```
<?php
mail("nobody@example.com", "the subject", $message,
     "From: webmaster@{$_SERVER['SERVER_NAME']}\r\n" .
     "Reply-To: webmaster@{$_SERVER['SERVER_NAME']}\r\n" .
     "X-Mailer: PHP/" . phpversion());
?>
```

O parâmetro **`additional_parameters`** pode ser usado para passar um parâmetro adicional para o programa configurado para usar quando enviar e-mail usando a definição de configuração **sendmail_path**. Por exemplo, isto pode ser usado para definir o endereço do remetente quando usar `sendmail` com a opção de configuração *`-f`*.

Bibliografia

Livros

- ✓ BARRETO, Mauricio Vivas de Souza, apostila do Projeto Supervisionado de Final de Curso da Universidade Federal de Sergipe - Centro de Ciências Exatas e Tecnologia do Departamento de Estatística e Informática, abril de 2000

- ✓ CONVERSE, Tim e PARK, Joyce, "PHP – A Bíblia", 2ª Edição, editora Campus, 2003

Internet

- ✓ MANUAL DO PHP
  http://br.php.net/manual/pt_BR/index.php
  Traz um manual online em inglês, português e diversos outros idiomas sobre o PHP