Desenvolvendo aplicações com PHP e MySQL

Sumário

Notas do Autor

Este guia não tem a pretensão de exaurir o assunto, mas servir de fonte de consulta a iniciantes em programação que querem desenvolver aplicações para Web.

É muito importante que o leitor teste os exemplos, faça os exercícios propostos e refaça os exercícios resolvidos.

Os pré-requisitos para a leitura deste guia são: lógica de programação e familiarização com HTML.

Estima-se que pouco mais de 10% dos domínios na Internet possuem o PHP instalado, daí uma grande oportunidade a esses estudantes e profissionais. Esta é a primeira versão do guia.

Daniel Moreira dos Santos
Técnico em Informática e Licenciando em Matemática

Contatos:
daniel-htm@hotmail.com
http://www.danielmoreira.wordpress.com

Dúvidas, sugestões e críticas são bem-vindas.

12 Setembro de 2009

Devido a outras ocupações voltei a escrever o guia e até mesmo a usar o PHP apenas neste ano de 2010. Hoje, dia 29 de Julho de 2010, publico a primeira versão, que chamarei de versão 0 (zero). Espero em breve atualizá-la com idéias que já tenho em mente e também com sugestões de leitores e amigos. Fico feliz de poder contribuir de alguma forma para o seu aprendizado. O segredo é dividir para conquistar!

Daniel Moreira dos Santos

29 de Julho de 2010

Introdução

O nome PHP é um acrônimo que significa “PHP Hypertext Preprocessor”. PHP é uma linguagem de script interpretada utilizada em maioria para desenvolver aplicações Web embutida em um HTML, mas também pode ser usada para o desenvolvimento de aplicações desktop GUI (veja sobre PHP-GTK) e scripts de linha de comando.

O PHP que temos hoje é o sucessor do PHP/FI, que significa Personal Home Page/Forms Interpreter. O PHP/FI foi criado por Rasmus Lerdorf por volta de 1995 com o intuito de controlar os acessos a sua home page (currículo on line).

Você pode consultar mais sobre a história do PHP na página oficial http://www.php.net. Atualmente o PHP está na versão 5. Esta versão pode ser encontrada no site assim como a documentação em português.

A princípio daremos uma ênfase maior ao desenvolvimento de aplicações para Web. Para tal, você precisa ter instalado em sua máquina um cliente (browser) Web, um servidor Web (usaremos o Apache) e evidentemente o interpretador PHP. Caso não tenha você pode instalá-los separadamente ou baixar os pacotes pré-configurados, você pode alugar um servidor que possua isso instalado se preferir.

Como a configuração não é nosso objetivo mostraremos onde e como baixar esses pacotes no Linux e Windows. Grande parte das aplicações Web faz interação com um banco de dados, nós utilizaremos o banco de dados MySQL, já que é muito utilizado e gratuito.

O PHP é uma linguagem de script Server-Side, isto é, um conjunto de códigos que serão interpretados no servidor. Uma característica é que o código de um arquivo.php não é exibido como acontece com um código HTML ou Java script que são chamados linguagens Client-Side, são interpretadas no cliente Web.

Instalando o VertrigoServ no Windows

Como dito anteriormente, precisamos instalar o Apache (servidor Web HTTP), PHP e o MySQL (Sistema Gerenciador de Banco de Dados). Para tal vamos instalar O VertrigoServ, que é um projeto desenvolvido para facilitar a instalação desses componentes.

O VertrigoServ ainda inclui SQLite, SQLiteManager, ZendOptimizer, Smarty e o PhpMyAdmin. Esse último nós utilizaremos algumas vezes para manusearmos o nosso banco MySQL.

Para baixá-lo abra a página do projeto http://vertrigo.sourceforge.net/?lang=br e procure pela versão mais recente. Depois de instalar, inicie a aplicação. Note que na barra de tarefas (à direita) aparecerá um ícone com um detalhe verde significando que tudo está funcionando (Apache, PHP e MySQL).

Clique sobre o ícone e em seguida sobre WWW Folder. Esta pasta é onde os seus arquivos.php deverão estar. Crie uma pasta com o nome do seu projeto, para visualizar seus arquivos .php abra o seu navegador preferido e digite http://localhost/nome_da_sua_pasta/nome_do_arquivo.php.

Você pode acessar o PhpMyAdmin em Tools (ou Ferramentas) e depois clicando sobre PhpMyAdmin.
Pronto, isso é tudo que precisaremos no momento!

Instalando o Xampp (Apache, MySQL, PHP, Perl) no Linux.

Abra o seu Web browser e digite http://sourceforge.net/project/showfiles.php?group_id=61776, essa é a página do projeto. Procure pela versão mais nova para Linux. Agora abra o seu terminal e digite:

sudo tar xvfz xampp-linux-1.6.4.tar.gz -C /opt

Agora vamos iniciar as aplicações:

sudo /opt/lampp/lampp start

Pronto. Está tudo instalado no diretório /opt/lampp com a configuração padrão. Você pode instalar em outro diretório se preferir.

Para ver seus arquivos.php através de um Web browser coloque-os na pasta /opt/lampp/htdocs. No navegador digite: http://localhost/nome_da_pasta/nome_do_arquivo.php

Um primeiro Script

Aqui vamos mostrar um primeiro script em PHP que imprimirá a mensagem "Olá Mundo!" no document do seu browser.

```
1  <html>
2  <head><title>Primeiro Script PHP</title></head>
3  <body>
4  <?php
5  echo "Olá Mundo"
6  ?>
7  </body>
8  </html>
```

Este não é um script muito útil, pois sua saída será a mesma de: * Olá Mundo *. Mas a intenção é de mostrar como ele aparece em um código HTML.

Note que um script em PHP começa com *<?php* e termina com *?>*. O que estiver entre esses delimitadores de código não aparece no lado cliente como um código. Clique com o botão direito do mouse sobre o document, em seguida clique em exibir código-fonte, note que você não consegue visualizar o código PHP.

A função *echo* imprime uma string na tela, nesse caso imprimimos "Olá Mundo".

Sintaxe Básica

Como falado anteriormente, um script em PHP inicia com a tag *<?php* e termina com a tag *?>*. Essas são as tags mais utilizadas para delimitar um código PHP, mas você também pode usar *<?* e *?>* desde que a opção short_open_tag esteja ativada no seu php.ini ou também se o PHP foi configurado com a opção --enable-short-tags.

Outra opção, porém menos comum, é a de usar *<script language-"php">* e *</script>*. Como curiosidade você poderia pesquisar sobre como usar tags asp no php.

Uma boa prática de programação é usar ponto e vírgula (;) no final de cada comando. O ponto e vírgula diz ao PHP que a instrução chegou ao fim, e se não houver erro ele executa a próxima instrução.

Você pode omitir o último ponto e vírgula (;) do bloco de código, pois a tag *?>* diz ao php que é o fim da instrução e do bloco.

Tipos

O PHP tem suporte a oito tipos primitivos: boolean, integer, float (double), string, array, object, resource e null. O PHP é uma linguagem dita fracamente tipada, pois o tipo da variável depende do contexto em que ela está inserida, isto é decidido em tempo de execução.

A função *var_dump()* checa o tipo e o valor de uma variável ou expressão. Mas comumente queremos saber apenas o tipo de uma variável, então usamos as funções *is_int(), is_float, is_string() e is_bool()*.

Assim como em outras linguagens podemos "forçar" um tipo em uma variável. Para fazermos isso usamos o que chamamos de *casting* ou a função *settype()*.

Para converter um tipo no PHP escrevemos o nome do tipo que queremos moldar a variável entre parêntesis e em seguida o nome da variável. Veja a sintaxe do casting no exemplo abaixo:

```php
<?php
$var1=1; //a variável $var1 é um inteiro
$var2=(bool)$var1; //a variável $var2 é um booleano
?>
```

Moldagens possíveis:

(int), (integer) - molde para inteiro.
(bool), (boolean) - converte para booleano.
(float), (double), (real) - converte para número de ponto flutuante.
(string) - converte para string
(binary) - converte para string binária
(array) - converte para array (object) - converte para objeto

Booleano

Booleano, *bool ou boolean,* é o tipo mais simples. Pode assumir apenas dois valores: *true* ou *false*. Esses são os chamados tipos lógicos. Observe que as palavras *true* e *false* são insensitivas, sendo assim é o mesmo que escrever *True* e *False*.

Usualmente o tipo booleano é utilizado quando retornado de uma expressão envolvendo algum operador lógico: ==, >=, <=, >, <, !=.

Como vimos acima, podemos converter um valor qualquer para boolean explicitamente utilizando casting ou atribuindo esse tipo a uma variável. Entretanto se uma estrutura de controle, operador ou função necessitar de um argumento booleano, esse será enviado para o PHP mesmo que a variável seja de um outro tipo qualquer.

Isso ocorre porque o PHP decide o tipo da variável em tempo de execução, isso significa que uma variável pode ser boolean dependendo do contexto.

Em uma conversão para booleano esses valores se tornam *false*:

1. O próprio booleano FALSE.
2. O inteiro 0 (zero) o ponto flutuante 0.0 (zero).
3. Uma string vazia e a string "0".
4. Um array sem elementos.
5. Um objeto sem elementos membros (somente PHP 4).
6. O tipo especial NULL (incluindo variáveis não definidas).
7. Objeto SimpleXML criado para tags vazias.

Qualquer valor diferente é assumido como *true*.

Inteiro

O tipo inteiro é o mesmo que conhecemos na matemática quando falamos de conjuntos numéricos. Um inteiro é um elemento do conjunto Z={..., -2, -1, 0, 1, 2, ...}. No entanto, a memória do computador é limitada, por isso existe um valor máximo para um inteiro no computador.

O tamanho máximo depende da plataforma, sendo um numero aproximado a 2 bilhões que é um número de 32 bits com sinal.

Como observado no manual do PHP, ele não suporta inteiros sem sinal. O tamanho do inteiro pode ser determinado por PHP_INT_SIZE, o valor máximo para PHP_INT_MAX desde o PHP 4.4.0 e PHP 5.0.5.

Os inteiros podem ser representados em notação decimal (ou base 10), octal (ou base 8) e hexadecimal (ou base 16). Para que o PHP entenda que um número está sendo representado em uma outra base numérica é preciso que você explicite isso.

Preceda o número com um *0* para indicar escrita em notação octal e com *0x* para indicar notação hexadecimal. Veja esse exemplo:

```php
<?php
$a=100; //número decimal
$a=-169; // número decimal negativo
$a=0123; // número octal (equivalente a 83 em decimal)
$a=0x1A; //número hexadecimal (equivalente a 26 em decimal)
?>
```

Uma observação importante é que se você usa um número maior que o limite suportado pelo tipo inteiro, o PHP interpreta como um float.

Não existe um operador de divisão inteira no PHP como o DIV em algumas linguagens. Entretanto você pode usar o casting que vai truncar o número real restando apenas os dígitos inteiros ou você pode usar a função *round()*.

Para converter um valor para o tipo inteiro você pode usar os modificadores *(int)* e *(integer)* como dito anteriormente ou então deixar com o que o PHP converta como acontece com as conversões para booleano. Isso acontece desde que algum operador, estrutura de controle ou função necessite de um argumento inteiro.

O valor lógico *true* é convertido para inteiro como *1,* e o valor lógico *false* é convertido com *0.*

Quando um ponto flutuante é moldado para inteiro através do casting ele é truncado. Se o número estiver além dos limites de um inteiro o resultado é indefinido.

Ponto Flutuante

O tipo *float*, *double* ou em português, *ponto flutuante*, é uma representação de um número real. Mas como já mencionado, o computador possui uma memória finita, o que implica que um float é uma aproximação limitada do número.

O tamanho de um *float* é dependente da plataforma assim como o tamanho de um *int*, sendo o máximo de *~1.8e308* com uma precisão de 14 casas decimais (número de 64 bits).

Por problemas de perda de precisão, nunca podemos ter certeza se um dado número de ponto flutuante realmente é exato. Na maioria das vezes ele é uma aproximação suficientemente boa para algum propósito específico.

O PHP vem com algumas funções de precisão arbitrária que podem te ajudar se precisar de uma precisão maior do que a fornecida.

Para converter um boolean ou um float, primeiro a conversão é feita para um int e então segue a regra de conversão de inteiros para ponto flutuante. O caso do tipo primitivo string é especial e veremos adiante.

Veja um exemplo de representações de ponto flutuante:

```php
<?php
$a = 1.234;
$b = 1.2e3;
$c = 7E-10;
?>
```

String

Uma *string* é uma cadeia de caracteres. No PHP uma string não possui um tamanho máximo, isto é, pode ser uma cadeia tão longa quanto se queira.

Você pode representar uma string com ' ' (apóstrofo) ou “ ” (aspas duplas).

O PHP entende mais seqüências de escape para caracteres especiais quando a string é delimitada por aspas.

Veja abaixo algumas seqüências de escape extraídas do manual:

\n fim de linha (linefeed ou LF ou 0x0A (10) em ASCII)
\r retorno de carro (carriage return ou CR ou 0x0D (13) em ASCII)
\t TAB horizontal (HT ou 0x09 (9) em ASCII)
\v TAB vertical (VT ou 0x0B (11) em ASCII) (desde o PHP 5.2.5)
\f form feed (FF ou 0x0C (12) em ASCII) (desde o PHP 5.2.5)
\\ contra barra ou barra invertida
\$ sinal de cifrão
\" aspas
\[0-7]{1,3} a seqüência de caracteres batendo a expressão regular dos caracteres em notação octal
\x[0-9A-Fa-f]{1,2} a seqüência de caracteres batendo a expressão regular de um caractere em notação hexadecimal.

Veja este exemplo do uso de strings retirado do manual:

```php
<?php
echo 'isto é uma string comum';
echo 'Você pode incluir novas linhas em strings,dessa maneira que estarátudo bem';
// Imprime: Arnold disse uma vez: "I\'ll be back"
echo 'Arnold once said: "I\'ll be back"';
// Imprime: Você tem certeza em apagar C:\*.*?
echo 'Você tem certeza em apagar C:\\*.*?';
// Imprime: Você tem certeza em apagar C:\*.*?
echo 'Você tem certeza em apagar C\*.*?';
// Imprime: Isto não será substituido: \n uma nova linha
echo 'Isto não será substituido: \n uma nova linha';
// Imprime: Variaveis $também não $expandem
echo 'Variaveis $também não $expandem';
?>
```

Array

Um array é um conjunto de variáveis indexadas em uma lista. No PHP, um array é um tipo bastante complexo, você pode acessar um valor da lista através de um índice que pode ser um inteiro ou até uma string.
Um array pode armazenar valores de diversos tipos, como, inteiro, float, booleano, string e até mesmo outro array introduzindo o conceito de array multidimensional. Um array de uma dimensão é comumente chamado de vetor.
Podemos simular árvores binárias como array de arrays, mas esse conceito foge da nossa pauta inicial.
Um array pode ser construído com o método *array()* do PHP. Veja a sintaxe e logo abaixo um exemplo ilustrativo do uso do construtor:

array(
índice => valor, índice=>valor, ...
)
Como falamos anteriormente, o índice pode ser um inteiro ou uma string. Valor pode ser de qualquer tipo.

```php
<?
$inf = array("nome" => "daniel", "idade" => "20", "solteiro" => true, 4 => 3.89);

echo $inf["nome"]; //imprime daniel
echo $inf[4]; // imprime 3.89
?>
```

É importante notar algumas coisas, se você indexar um valor com um float, isto é, usá-lo como um índice no seu array, ele será convertido para inteiro.

É possível criar um array omitindo os índices, isto faz com que o PHP fixe os índices como inteiros. Se todos os índices forem omitidos o PHP inicializa o primeiro índice com 0 e o restante de 1 em 1 em ordem crescente. Se forem omitidos alguns índices apenas, o PHP pega o maior índice inteiro acrescentado de um e indexa o novo valor da lista.

```php
<?
array(10, 20, 30, 40);// os indices são: 0, 1, 2, 3
array(5 => 43, 6 => 32, 7 => 56, 12); // aqui o último índice é 8
?>
```

Veja este exemplo de um array multidimensional:

```php
<?
// declaramos um array como um valor dentro de outro array
// note que "1" é um inteiro e não uma string, "01" é uma string
$inf = array("1" => array(1 => 5.1, 3 => 5.2, 5 => 5.3));

echo $inf[1][1];  // 5.1
echo $inf[1][3];  // 5.2
echo $inf[1][5];  // 5.3
?>
```

Existem outras observações a serem feitas sobre o tipo array. O uso do valor lógico *true* como índice será interpretado pelo PHP como o inteiro 1. Analogamente, *false* será interpretado como 0 inteiro.

Usar NULL para indexar um valor no array é interpretado como uma string vazia e usar uma string vazia como chave irá criar (ou sobrescrever) uma chave com uma string vazia e seu valor.

Um aviso importante: você não pode usar um array ou outro objeto como índice, isso fará com que o PHP exiba a seguinte mensagem de erro: “Illegal offset type.”.

Também existe a possibilidade de indexar valores em um array com colchetes explicitando o índice e o valor a ser atribuído. Veja o próximo exemplo:

```
<?
$inf[]="PHP"; // índice omitido, atribuida a string "PHP"
$inf[2]=1.25; // índice 2, atribuido um valor float
$inf["PHP"]=10; // índice uma string "PHP",atribuido um valor inteiro
?>
```

Variáveis

Uma variável, tanto no PHP como em qualquer outra linguagem, é uma referência à uma região limitada da memória que contem informação de um determinado tipo de dado.

Toda variável no PHP deve iniciar com $, isto se torna muito bom quando necessário identificar variáveis em um código muito extenso.

O PHP é case sensitive, isto é, há distinção no uso de letras maiúsculas e minúsculas. Logo, uma variável $Variavel é diferente de $variavel, $vArIaVeL, $VARIAVEL e etc.

O nome de uma variável no PHP pode ter qualquer tamanho, desde que obedeça algumas regras. Um nome de variável sempre deve iniciar com uma letra ou sublinhado, evidentemente depois do $, seguido de qualquer seqüência de letras, algarismos e sublinhados. Veja o próximo exemplo sobre o uso de variáveis no PHP:

```
<?php
$adn="uma string"; //nome de variável válido, começa com letra
$_d45 = 45; //nome de variavel válido, começa com _ (sublinhado)
$34_dan = $_d45 + 3.476; // nome de variável inválido, começa com número
$Var = 1.67;
$var = "isso vale:"; //$Var!=$var
$var = $var." ".$Var; // $var possui: isso vale 1.67
?>
```

Como na linguagem C é possível atribuir uma variável a outra por valor e atribuir por referência. Uma atribuição por valor significa que uma cópia da variável atribuída é passada para a outra variável. Quando o valor de uma das duas é modificado o da outra permanece inalterado.

Quando é feita uma atribuição por referência a segunda variável referencia (aponta) para a variável original. Logo é possível alterar o valor da variável original utilizando a variável que recebeu a

atribuição. Para fazer referência é necessário o uso do & antes da variável original. Veja este exemplo:

```php
<?php
$valor_1=10; //atribui o inteiro 10 a $valor_1
$valor_2=$valor_1; //atribuição por valor: atribui $valor_1 (que vale 10) a $valor_2
$valor_2++; //incrementa de 1 $valor_2
echo "Valor 1: ".$valor_1." e Valor 2: ".$valor_2; //imprime: Valor 1: 10 e Valor 2: 11

$Nvalor_1=10; //atribui o inteiro 10 a $Nvalor_1
$Nvalor_2=&$Nvalor_1; //atribuição por referência: faz $Nvalor_2 referenciar o mesmo valor que $Nvalor_1
$Nvalor_2++; //incrementa de 1 $Nvalor_1
echo "Valor 1: ".$Nvalor_1." e Valor 2: ".$Nvalor_2; //imprime: Valor 1: 11 e Valor 2: 11
?>
```

Como mostrado no exemplo acima, só é possível atribuir por referência variáveis com nome.

As variáveis não precisam ser inicializadas no PHP, falaremos um pouco sobre boas práticas de programação e riscos que isso representa no Capítulo de Segurança.

Quando não inicializadas elas recebem um valor padrão do tipo delas, que como dissemos, é decidido em tempo de execução de acordo com o contexto - FALSE, zero, string vazia ou null.

Constantes

Pela própria definição da palavra, uma constante não é uma variável. Uma variável é um identificador para um valor que pode ser alterado, e no PHP até receber um valor de outro tipo. Uma constante é um identificador de um único valor que não é alterado no decorrer do script.

O nome ou identificador de uma constante é sempre maiúsculo por padrão. Como qualquer outro rótulo no PHP, o nome de uma constante válida começa com uma letra ou sublinhado, seguido por uma seqüência de letras, números e/ou sublinhados. Veja este exemplo:

```php
<?php
define("CONS", "constante"); //Nome de constante válido
define("CONS2", "outra constante"); //Nome de constante válido
define("CONS_3", "outra"); //Nome de constante válido
define("2CONS", "alguma coisa"); //Nome de constante inválido, começa com número
?>
```

A forma de definir uma constante é utilizando a função *define()*. Uma constante não pode ser alterada depois de sua definição.

Constantes podem ser boolean, int, float ou string. Uma constante não pode ser um objeto, que estudaremos adiante.

Para acessar o valor de uma constante basta escrever seu nome ou utilizar a função *constant()*. Uma observação importante é que você não pode utilizar o caractere $ no nome de uma constante.

Expressões

Utilizamos expressões em todos os exemplos dados acima. Uma expressão é uma sentença com valor. As expressões podem ser separadas em expressões de atribuição e expressões de comparação.

A forma mais simples de aparecer uma expressão é em uma atribuição de um valor constante a uma variável, mas uma expressão aparece em diversas outras formas como comparações retornando um valor lógico. Veja o exemplo abaixo:

```php
<?php
function getStr($numero){
  if($numero)
    return "nao_zero";
    return "zero";
}
?>
<?php
$valor_1=2.3; //atribui 2.3 a $valor_1
$valor_2=$valor_1; //o mesmo que atribuir a constante 2.3 a $valor_2
$a++; //incremento de 1 a $a
$a--; //decrimento de 1 a $a
$b=getStr($a); //retorna uma string para $b
if($a==0){ //compara se $a é igual 0
  echo "a igual a zero";
}
?>
```

O PHP avalia uma atribuição da direita para a esquerda, portanto é válido e muito comum escrevermos *$a=$b=45;*. O que acontece é que o PHP atribui a constante 45 a $b e a $a.

No exemplo acima utilizamos os operadores incremento e decremento. Um incremento é uma atribuição do tipo *$a++;,* que é o mesmo que *$a=$a+1;*. Analogamente um decremento é subtrair 1 do valor original da variável, portanto *$a--;* é o mesmo que *$a=$a-1;*

Existem vários operadores de comparação, todos eles são utilizados em expressões. Vamos citá-los aqui e vê-los no próximo capítulo com mais detalhe.

Operador de atribuição =, operador de incremento ++, operador de decremento --, operadores de comparação: > (maior que), >= (maior ou igual a), < (menor que), <= (menor ou igual a), == (igual), != (diferente), === (igual a e do mesmo tipo), !== (diferente de ou não do mesmo tipo).

Note que existem formas diferentes de escrever a mesma coisa, por exemplo, *$a+=1;* isso é interpretado pelo PHP como o valor de $a incrementado de 1.
Um outro operador bem conhecido, mas talvez não tão utilizado seja o operador condicional ternário. A segunda sub-expressão é avaliada e tem seu valor retornado se a primeira sub-expressão for verdadeira, se a primeira sub-expressão for falsa a terceira sub-expressão é avaliada e seu valor retornado.

Daremos um exemplo do seu uso mais adiante.

Operadores

Um operador é algo que dá vida a uma expressão lógica, em outras palavras, ele avalia uma expressão de um ou mais valores e retorna um outro valor.

Existem os operadores unários que operam em apenas um valor, os operadores binários que operam em dois valores e o operador ternário que opera em três valores.

Como na matemática, no PHP existe uma precedência de operadores, isto é, quem será o primeiro a ser avaliado quando vários operadores estão na mesma expressão.

Veja as tabelas abaixo retiradas do manual do PHP:

Precedência dos operadores

Associação	Operador	Informação adicional
não associativo	clone new	clone e new
esquerda	[	array()
não associativo	++ --	incremento/decremento
não associativo	~ - (int) (float) (string) (array) (object) (bool) @	tipos
não associativo	instanceof	tipos
direita	!	lógico
esquerda	* / %	aritmético
esquerda	+ - .	aritmético e string
esquerda	<< >>	Bit-a-bit
não associativo	< <= > >= <>	comparação
não associativo	== != === !==	comparação
esquerda	&	Bit-a-bit e referências
esquerda	^	Bit-a-bit
esquerda	|	Bit-a-bit
esquerda	&&	lógico
esquerda	||	lógico
esquerda	? :	ternário
direita	= += -= *= /= .= %= &= |= ^= <<= >>=	atribuição
esquerda	and	lógico
esquerda	xor	lógico
esquerda	or	lógico
esquerda	,	muitos usos

Operadores Aritméticos

Exemplo	Nome	Resultado
-$a	Negação	Oposto de $a.
$a + $b	Adição	Soma de $a e $b.
$a - $b	Subtração	Diferença entre $a e $b.
$a * $b	Multiplicação	Produto de $a e $b.
$a / $b	Divisão	Quociente de $a por $b.
$a % $b	Módulo	Resto de $a dividido por $b.

Operadores de comparação

Exemplo	Nome	Resultado
$a == $b	Igual	Verdadeiro (**TRUE**) se $a é igual a $b.
$a === $b	Idêntico	Verdadeiro (**TRUE**) se $a é igual a $b, e eles são do mesmo tipo (introduzido no PHP4).
$a != $b	Diferente	Verdadeiro se $a não é igual a $b.
$a <> $b	Diferente	Verdadeiro se $a não é igual a $b.
$a !== $b	Não idêntico	Verdadeiro de $a não é igual a $b, ou eles não são do mesmo tipo (introduzido no PHP4).
$a < $b	Menor que	Verdadeiro se $a é estritamente menor que $b.
$a > $b	Maior que	Verdadeiro se $a é estritamente maior que $b.
$a <= $b	Menor ou igual	Verdadeiro se $a é menor ou igual a $b.
$a >= $b	Maior ou igual	Verdadeiro se $a é maior ou igual a $b.

Operadores de Incremento/Decremento

Exemplo	Nome	Efeito
++$a	Pré-incremento	Incrementa $a em um, e então retorna $a.
$a++	Pós-incremento	Retorna $a, e então incrementa $a em um.
--$a	Pré-decremento	Decrementa $a em um, e então retorna $a.
$a--	Pós-decremento	Retorna $a, e então decrementa $a em um.

Operadores Lógicos

Exemplo	Nome	Resultado
$a and $b	E	Verdadeiro (**TRUE**) se tanto $a quanto $b são verdadeiros.
$a or $b	OU	Verdadeiro se $a ou $b são verdadeiros.
$a xor $b	XOR	Verdadeiro se $a ou $b são verdadeiros, mas não ambos.
! $a	NÃO	Verdadeiro se $a não é verdadeiro.
$a && $b	E	Verdadeiro se tanto $a quanto $b são verdadeiros.
$a || $b	OU	Verdadeiro se $a ou $b são verdadeiros.

Operadores de array

Exemplo	Nome	Resultado
$a + $b	União	União de $a e $b.
$a == $b	Igualdade	**TRUE** se $a e $b tem os mesmos pares de chave/valor.
$a === $b	Identidade	**TRUE** se $a e $b tem os mesmos pares de chave/valor na mesma ordem e do mesmo tipo.
$a != $b	Desigualdade	**TRUE** se $a não é igual a $b.
$a <> $b	Desigualdade	**TRUE** se $a não é igual a $b.
$a !== $b	Não identidade	**TRUE** se $a não é idêntico a $b.

Estruturas de Controle

Basicamente as estruturas de controle são divididas em dois tipos: Estruturas condicionais e as estruturas de repetição (loops). As estruturas de controle aglomeram instruções que deverão ser executadas dada uma condição ou que serão executadas um número finito de vezes.

If

O if, ou "se" em português, é uma estrutura de controle do tipo condicional, isto é, um dado bloco de comandos será executado "se" certa condição for verificada. Note que já usamos esta estrutura de controle anteriormente. Para ficar mais claro vamos ver um exemplo:

```
<?
$valor=10;
if($valor==10){
  echo "Valor=".$valor;
}
?>
```

Neste exemplo $valor é igual a 10, se $valor igual a 10 então a próxima instrução é para imprimir a string "Valor=10".
A sintaxe básica da estrutura condicional if é:

```
if(condição){
  bloco_de_comandos
}
```

If else

O if else (se senão) funciona de forma semelhante, apenas tem-se um bloco de comandos de "escape" caso a condição não seja satisfeita. Seguindo o exemplo anterior:

```
<?
$valor=10;
if($valor==11){
  echo "Valor=".$valor;
}else{
  echo "Valor!=11";
}
?>
```

Neste exemplo é impresso a string "Valor!=11" pois a expressão $valor==10 retorna o valor lógico false. Lembre-se que será executado apenas um dos dois blocos de comandos.
A sintaxe básica do if else é:

```
if(condição){
   bloco_de_comandos_1
}else{
  bloco_de_comandos_2
}
```

Encadeando if else if

É muito freqüente o uso de if's encadeados durante a confecção de um programa. Vamos dar um exemplo de como seria o seu uso.

```php
<?php
/* Ler 2 números inteiros do teclado (A e B), verificar e imprimir qual deles é o maior, ou a
   mensagem "A=B" caso sejam iguais. */

$a = 5;
$b = 8;

if ( $a>$b ){
  echo $a;
}elseif( $b>$a ){
  echo $b;
}else{
  echo "A é igual a B";
}
?>
```

Switch case

O Switch case é uma outra estrutura condicional, mas diferente de outras linguagens, no PHP o switch case pode ser usado para testar condições sobre strings além de números inteiros.

A instrução switch executa case a case, quando uma instrução case é encontrada com um valor igual ao valor dentro do switch, o PHP executa as instruções seguintes.

O PHP executa as instruções até o fim do bloco switch ou na primeira vez que encontrar uma instrução break. Se você não escrever uma instrução break no fim das instruções case, o PHP continuará executando os cases seguintes. Vamos ver alguns exemplos:

```php
<?
/* imprime "i igual a 0" e continua executando os cases */
$i=0;
switch ($i) {
    case 0:
        echo "i igual a 0";
    case 1:
        echo "i igual a 1";
    case 2:
        echo "i igual a 2";
}
?>
```

Exemplo com strings:

```php
<?
switch ($i) {
    case "apple":
        echo "i is apple";
        break;
    case "bar":
        echo "i is bar";
        break;
    case "cake":
        echo "i is cake";
        break;
    }
?>
```

No exemplo acima quando o case correto é encontrado o programa sai do switch devido ao break após imprimir a frase correspondente.

O case default:

```php
<?
switch ($i) {
    case 0:
        echo "i igual a 0";
        break;
    case 1:
        echo "i igual a 1";
        break;
    case 2:
        echo "i igual a 2";
        break;
    default:
        echo "i não é igual a 0, 1 ou 2";
}
?>
```

Este case é executado quando nenhum outro é executado.

Estruturas de Repetição (Loops)

As estruturas de repetição, ou laços de repetição, são utilizadas quando um bloco de comandos precisa ser executado um determinado número de vezes.

For

O for é uma estrutura de repetição que herdou uma sintaxe bem compacta da linguagem C. Sua sintaxe básica é:

for(*inicialização*; *condição de parada*; *incremento/decremento*){
    bloco_de_comandos
}

Veja um exemplo comentado do uso do for:

```
<?
// $i foi inicializado com 0
for($i=0; $i<10; $i++){ // enquanto $i for menor do que 10 a mensagem "Programando em PHP " será impressa
  echo "Programando em PHP ";
}// $i é incrementado de 1 a cada execução do bloco de comandos
?>
```

Neste exemplo a variável $i serviu como um contador de passos. Você pode omitir a inicialização da variável, no exemplo acima não faria nenhuma diferença já que o php inicializa a variável com 0.

Se a instrução de incremento fosse omitida teríamos o que chamamos de loop infinito, ou seja, ele iria imprimir “Programando em PHP ” indefinidamente, provavelmente até o seu computador travar por falta de memória. Esse seria o exemplo, mas não vale a pena rodá-lo, é apenas didático:

```
<?
/*Exemplo de Loop Infinito*/
/*$i inicializado pelo PHP com o valor 0*/
for( ;$i<10; ){ // condição sempre satisfeita
  echo "Programando em PHP ";
}
?>
```

Note que como $i++ foi omitido ele verifica a condição (também chamada teste de parada) após executar o bloco de comandos, como o valor de $i não foi alterado, a condição é sempre verdadeira e o loop não pára.

While

Daremos os mesmos exemplos acima, mas substituindo o for pelo while com as devidas modificações. Quando avançarmos um pouco mais nas peculiaridades da linguagem teremos exemplos mais interessantes para olhar.

Sintaxe básica do while:

while(*condição*){
bloco_de_comandos
}

```php
<?
$i=0; //inicializa $i com o valor 0 (zero), note que essa linha poderia ter sido omitida
while($i<10){ // enquanto $i menor do que 10
  echo "Programando em PHP " //imprime "Programando em PHP "
  $i++; // incrementa 1 ao valor anterior de $i
}
?>
```

```php
<?
while(){ // sem um teste de parada
  echo "Programando em PHP " //imprime "Programando em PHP " até possivelmente seu navegador travar
}
?>
```

Outra forma de criar um loop infinito é testando uma condição que sempre é verdadeira. Veja o exemplo abaixo:

```php
<?
while($i<10){ // $i inicializado com 0 (zero) pelo PHP
  echo "Programando em PHP " //imprime "Programando em PHP " até possivelmente seu navegador travar
} // note que o loop é infinito, pois $i é sempre 0 (zero) menor do que 10
?>
```

Do While

Não é nada mais do que o while que permite que o bloco de comandos seja executado uma vez antes que a condição de parada seja testada.
Sintaxe do Do While:
do{
bloco_de_comandos
}while(*condição*);

Veja este simples exemplo para ilustrar o seu uso:

```php
<?
$i=9; // $i inicializado com 9
do{ // início do bloco de comandos
$i++; // $i está com 10
echo "Programando em PHP " // imprime "Programando em PHP "
}while($i<10); // condição retorna false e o loop termina
?>
```

Funções

Até agora nós usamos somente uma função, a função “echo” usada para imprimir caracteres no document do nosso browser.

Uma função é um bloco de comandos que pode ser invocado sempre quando necessário sem a necessidade de repetir todo o bloco, apenas fazendo referenciando o seu nome.

Geralmente quando se chama uma função, um conjunto de argumentos é passado e é retornado um valor.

A seguir vamos ver como declarar uma função, como passar argumentos, como retornar valores gerados dentro dela e também veremos algumas funções interessantes que usaremos no decorrer deste guia.

Construindo Suas Funções

Para declarar uma função começamos com a palavra *function* seguida do nome que queremos dá-la. Dentro de parêntesis, uma lista de argumentos separados por vírgula. Abrimos e fechamos o bloco de comandos da função respectivamente com { e }.

function nome_da_funcao(*lista_de_argumentos*){
    bloco_de_comandos
return valor
}

Veja o exemplo abaixo:

```
<?
//declaração da função soma
function soma(float $a, float $b){
return ($a+$b); // retorna a soma de $a e $b
}

echo soma(1.34, 5.02); // imprime a soma dos dois valores passados como parâmetros
echo soma(3000, 1932); // imprime a soma dos dois valores passados como parâmetros
?>
```

É claro que o exemplo acima é apenas ilustrativo. Não precisamos criar uma função para somar dois números, podemos usar o operador “+” de uma vez.

O próximo exemplo é mais interessante, pois mostra como seria incômodo repetir o bloco de comandos da função toda vez que precisássemos dela.

A partir dele você pode inferir como é útil construir funções para resolver problemas maiores.

```php
<?
function fatorial(int $n){
  $fat=1;
  for($i=$n; $i>=1; $i--){
      $fat=$fat*$i;
  }
return $fat; // retorna o valor do fatorial de $n calculado
}

echo fatorial(5); // imprime 5! que é 120
?>
```

Veremos nos exemplos a seguir que uma função pode não precisar de uma lista de argumentos e/ou retornar algum valor.

```php
<?
//exemplo de uma função sem passagem de parâmetros
function mensagem(){
return "Programando em PHP "; //retornando uma string
}

echo mensagem()."- Exemplo.";
?>
```

O exemplo anterior poderia ser assim:

```php
<?
//exemplo de uma função sem passagem de parâmetros e sem retorno
function mensagem(){
echo "Programando em PHP ";
}

mensagem();
echo "- Exemplo.";
?>
```

Algumas funções do PHP

Segue uma lista de funções do PHP que utilizaremos algumas vezes. Parte da lista foi retirada da internet e parte retirada do manual do PHP. Você pode adquirir a prática de consultá-lo sempre que necessitar.

Manipulação de Strings

1) Funções relacionadas à HTML

. htmlspecialchars

string htmlspecialchars(string str);

Retorna a string fornecida, substituindo os seguintes caracteres:

& para '&'
" para '"'
< para '<'
> para ‘>'

. htmlentities

string htmlentities(string str);

Funciona de maneira semelhante ao comando anterior, mas de maneira mais completa, pois converte todos os caracteres da string que possuem uma representação especial em html, como por exemplo:

º para 'º'
ª para 'ª'
á para 'á'
ç para ‘ç'

. nl2br

string nl2br(string str);

Retorna a string fornecida substituindo todas as quebras de linha ("\n") por quebras de linhas em html ("
").

Exemplo:

```php
<?
echo nl2br("Mauricio\nVivas\n");
/*Imprime:
Mauricio<br>Vivas<br>*/
?>
```

. *get_meta_tags*

array get_meta_tags(string arquivo);

Abre um arquivo HTML e percorre o cabeçalho em busca de "meta" tags, retornando num array todos os valores encontrados.

Exemplo:

No arquivo teste.html temos:

...
<head>
<meta name="author" content="jose">
<meta name="tags" content="php3 documentation">
...
</head><!-- busca encerra aqui -->
...

A execução da função:

```php
<?
get_meta_tags("teste.html");
//Retorna o array:
array("author"=>"jose","tags"=>"php3 documentation");
?>
```

. *strip_tags*

string strip_tags(string str);

Retorna a string fornecida, retirando todas as tags html e/ou PHP encontradas.

Exemplo:

. *urlencode*

string urlencode(string str);

Retorna a string fornecida, convertida para o formato urlencode. Esta função é útil para passar variáveis para uma próxima página.

. urldecode

string urldecode(string str);

Funciona de maneira inversa a urlencode, desta vez decodificando a string fornecida do formato urlencode para texto normal.

2) Funções relacionadas à arrays

. implode e join

string implode(string separador, array partes);
string join(string separador, array partes);

As duas funções são idênticas. Retornam uma string contendo todos os elementos do array fornecido separados pela string também fornecida.

Exemplo:

```
<?
$partes = array("a", "casa número", 13, "é azul");
$inteiro = join(" ",$partes);
/*$inteiro passa a conter a string:
"a casa número 13 é azul"*/
?>
```

. split

array split(string padrao, string str, int [limite]);

Retorna um array contendo partes da string fornecida separadas pelo padrão fornecido, podendo limitar o número de elementos do array.

Exemplo:

```
<?
$data = "11/14/1975";
$data_array = split("/",$data);
/*
O código acima faz com que a variável $data_array receba o valor: array(11,14,1975);*/
?>
```

. explode

array explode(string padrao, string str);

Funciona de maneira bastante semelhante à função split, com a diferença que não é possível estabelecer

um limite para o número de elementos do array.

Comparações entre strings

. similar_text

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

Compara as duas strings fornecidas e retorna o número de caracteres coincidentes. Opcionalmente pode ser fornecida uma variável passada por referência (ver tópico sobre funções), que receberá o valor percentual de igualdade entre as strings. Esta função é case sensitive, ou seja, maiúsculas e minúsculas são tratadas como diferentes.

Exemplo:

```
<?
$num = similar_text("teste", "testando",&$porc);
/*
As variáveis passam a ter os seguintes valores:
$num == 4; $porc == 61.538461538462
*/
?>
```

. strcasecmp

int strcasecmp(string str1, string str2);

Compara as duas strings e retorna 0 (zero) se forem iguais, um valor maior que zero se str1 > str2, e um valor menor que zero se str1 < str2. Esta função é case insensitive, ou seja, maiúsculas e minúsculas são tratadas como iguais.

. strcmp

int strcasecmp(string str1, string str2);

Funciona de maneira semelhante à função strcasecmp, com a diferença que esta é case sensitive, ou seja, maiúsculas e minúsculas são tratadas como diferentes.

. strstr

string strstr(string str1, string str2);
string strchr(string str1, string str2);

As duas funções são idênticas. Procura a primeira ocorrência de str2 em str1. Se não encontrar, retorna uma string vazia, e se encontrar retorna todos os caracteres de str1 a partir desse ponto.

Exemplo:

```
<?
strstr("Mauricio Vivas", "Viv"); // retorna "Vivas"
?>
```

. stristr

string stristr(string str1, string str2);

Funciona de maneira semelhante à função strstr, com a diferença que esta é case insensitive, ou seja, maiúsculas e minúsculas são tratadas como iguais.

. strpos

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

Retorna a posição da primeira ocorrência de str2 em str1, ou zero se não houver. O parâmetro opcional offset determina a partir de qual caractere de str1 será efetuada a busca. Mesmo utilizando o offset, o valor de retorno é referente ao início de str1.

. strrpos

int strrpos(string haystack, char needle);

Retorna a posição da última ocorrência de str2 em str1, ou zero se não houver.

3) Funções para edição de strings

. chop

string chop(string str);

Retira espaços e linhas em branco do final da string fornecida.

Exemplo:

```
<?
chop(" Teste \n \n "); //retorna " Teste"
?>
```

. ltrim

string ltrim(string str);

Retira espaços e linhas em branco do final da string fornecida.
Exemplo:
. trim

```
<?
ltrim(" Teste \n \n "); //retorna "Teste \n \n"
?>
```

string trim(string str);

Retira espaços e linhas em branco do início e do final da string fornecida.

Exemplo:

```
<?
trim(" Teste \n \n "); //retorna "Teste"
?>
```

. strrev

string strrev(string str);

Retorna a string fornecida invertida.

Exemplo:

```
<?
strrev("Teste"); // "etseT"
?>
```

. strtolower

string strtolower(string str);

Retorna a string fornecida com todas as letras minúsculas.

Exemplo:

```
<?
strtolower("Teste"); // retorna "teste"
?>
```

. strtoupper

string strtoupper(string str);

Retorna a string fornecida com todas as letras maiúsculas.

Exemplo:

```php
<?
strtoupper("Teste"); // retorna "TESTE"
?>
```

. ucfirst

string ucfirst(string str);

Retorna a string fornecida com o primeiro caracter convertido para letra maiúscula.

Exemplo:

```php
<?
ucfirst("teste de funcao"); // retorna "Teste de funcao"
?>
```

. ucwords

string ucwords(string str);

Retorna a string fornecida com todas as palavras iniciadas por letras maiúsculas.

Exemplo:

```php
<?
ucwords("teste de funcao"); // retorna "Teste De Funcao"
?>
```

. str_replace

string str_replace(string str1, string str2, string str3);

Altera todas as ocorrências de str1 em str3 pela string str2.

4) Funções diversas

. chr

string chr(int ascii);

Retorna o caractere correspondente ao código ASCII fornecido.

. ord

int ord(string string);

Retorna o código ASCII correspondente ao caractere fornecido.
. echo

echo(string arg1, string [argn]...);

Imprime os argumentos fornecidos.

. print

print(string arg);

Imprime o argumento fornecido.

. strlen

int strlen(string str);

Retorna o tamanho da string fornecida.

Manipulação de Arrays

array

array array(...);

É a função que cria um array a partir dos parâmetros fornecidos. É possível fornecer o índice de cada elemento. Esse índice pode ser um valor de qualquer tipo, e não apenas de inteiro. Se o índice não for fornecido o PHP atribui um valor inteiro seqüencial, a partir do 0 ou do último índice inteiro explicitado. Vejamos alguns exemplos:

Exemplo 1:

```
<?
$teste = array("um", "dois","tr"=>"tres",5=>"quatro","cinco");
?>
```

Temos o seguinte mapeamento:

0 => "um" (0 é o primeiro índice, se não houver um explicito)
1 => "dois" (o inteiro seguinte)
"tr" => "tres"
5 => "quatro" (valor explicitado)
6 => "cinco" (o inteiro seguinte ao último atribuído, e não o próximo valor, que seria 2)

```php
<?
$teste = array("um", 6=>"dois","tr"=>"tres",5=>"quatro","cinco");
?>
```

Exemplo 2:

Temos o seguinte mapeamento:

0 => "um”
6 => "dois”
“tr" => tres
5 => "quatro" (seria 7, se não fosse explicitado)
7 => "cinco" (seria 6, se não estivesse ocupado)

Em geral, não é recomendável utilizar arrays com vários tipos de índices, já que isso pode confundir o programador. No caso de realmente haver a necessidade de utilizar esse recurso, deve-se ter bastante atenção ao manipular os índices do array.

. range

array range(int minimo, int maximo);

A função range cria um array cujos elementos são os inteiros pertencentes ao intervalo fornecido, inclusive. Se o valor do primeiro parâmetro for maior do que o do segundo, a função retorna false (valor vazio).

. shuffle

void shuffle(array &arr);

Esta função "embaralha" o array, ou seja, troca as posições dos elementos aleatoriamente e não retorna valor algum.

. sizeof

int sizeof(array arr);

Retorna um valor inteiro contendo o número de elementos de um array. Se for utilizada com uma variável cujo valor não é do tipo array, retorna 1. Se a variável não estiver setada ou for um array vazio, retorna 0.

1) Funções de "navegação”

Toda variável do tipo array possui um ponteiro interno indicando o próximo elemento a ser acessado no

caso de não ser especificado um índice. As funções seguintes servem para modificar esse ponteiro, permitindo assim percorrer um array para verificar seu conteúdo (chaves e elementos).
. reset

mixed reset(array arr);

Seta o ponteiro interno para o primeiro elemento do array, e retorna o conteúdo desse elemento.

. end

mixed end(array arr);

Seta o ponteiro interno para o último elemento do array, e retorna o conteúdo desse elemento.

. next

mixed next(array arr);

Seta o ponteiro interno para o próximo elemento do array, e retorna o conteúdo desse elemento.

Obs.: Esta não é uma boa função para determinar se um elemento é o último do array, pois pode retornar false tanto no final do array como no caso de haver um elemento vazio.

. prev

mixed prev(array arr);

Seta o ponteiro interno para o elemento anterior do array, e retorna o conteúdo desse elemento. Funciona de maneira inversa a next.

. pos

mixed pos(array arr);

Retorna o conteúdo do elemento atual do array, indicado pelo ponteiro interno.

. key

mixed key(array arr);

Funciona de maneira bastante semelhante a pos, mas ao invés de retornar o elemento atual indicado pelo ponteiro interno do array, retorna seu índice.

. each

array each(array arr);

Retorna um array contendo o índice e o elemento atual indicado pelo ponteiro interno do array. O valor de retorno é um array de quatro elementos, cujos índices são 0, 1, "key" e "value". Os elementos de índices 0 e "key" armazenam o índice do valor atual, e os elementos de índices 1 e "value" contém o valor do elemento atual indicado pelo ponteiro.

Esta função pode ser utilizada para percorrer todos os elementos de um array e determinar se já foi encontrado o último elemento, pois no caso de haver um elemento vazio, a função não retornará o valor false. A função each só retorna false depois q o último elemento do array foi encontrado.

Exemplo:

```php
<?
/*função que percorre todos os elementos de um array e imprime seus indices e valores */
function imprime_array($arr) {
reset($arr);
while (list($chave,$valor) = each($arr))
echo "Chave: $chave. Valor: $valor";
}
?>
```

2) Funções de ordenação

São funções que servem para arrumar os elementos de um array de acordo com determinados critérios. Estes critérios são: manutenção ou não da associação entre índices e elementos; ordenação por elementos ou por índices; função de comparação entre dois elementos.

. sort

void sort(array &arr);

A função mais simples de ordenação de arrays. Ordena os elementos de um array em ordem crescente, sem manter os relacionamentos com os índices.

. rsort

void rsort(array &arr);

Funciona de maneira inversa à função sort. Ordena os elementos de um array em ordem decrescente, sem manter os relacionamentos com os índices.

. asort

void asort(array &arr);

Tem o funcionamento bastante semelhante à função sort. Ordena os elementos de um array em ordem crescente, porém mantém os relacionamentos com os índices.

. arsort

void arsort(array &arr);

Funciona de maneira inversa à função asort. Ordena os elementos de um array em ordem decrescente e mantém os relacionamentos dos elementos com os índices.

. ksort

void ksort(array &arr);

Função de ordenação baseada nos índices. Ordena os elementos de um array de acordo com seus índices, em ordem crescente, mantendo os relacionamentos.

. usort

void usort(array &arr, function compara);

Esta é uma função que utiliza outra função como parâmetro. Ordena os elementos de um array sem manter os relacionamentos com os índices, e utiliza para efeito de comparação uma função definida pelo usuário, que deve comparar dois elementos do array e retornar 0, 1 ou –1, de acordo com qualquer critério estabelecido pelo usuário.

. uasort

void uasort(array &arr, function compara);

Esta função também utiliza outra função como parâmetro. Ordena os elementos de um array e mantém os relacionamentos com os índices, utilizando para efeito de comparação uma função definida pelo usuário, que deve comparar dois elementos do array e retornar 0, 1 ou –1, de acordo com qualquer critério estabelecido pelo usuário.

. uksort

void uksort(array &arr, function compara);

Esta função ordena o array através dos índices, mantendo os relacionamentos com os elementos., e utiliza para efeito de comparação uma função definida pelo usuário, que deve comparar dois índices do array e retornar 0, 1 ou –1, de acordo com qualquer critério estabelecido pelo usuário.

Funções Matemáticas

abs()

Retorna o valor absoluto de um número.

acos()

Retorna o arco co-seno de um número.

acosh()
Retorna o arco co-seno hiperbólico de um número.

asin()
Retorna o arco seno de um número.

asinh()
Retorna o arco seno hiperbólico de um número.

atan()
Retorna o arco tangente de um número como um valor numérico compreendido entre -PI/2 e PI/2 radianos.

atan2()
Retorna o ângulo teta de um ponto (x, y) como um valor numérico entre -PI e PI radianos.

atanh()
Retorna a tangente hiperbólica inversa de um número.

base_convert()
Converte um número de uma base para outra.

bindec()
Converte um número binário para um número decimal.

ceil()
Retorna o próximo maior valor inteiro arredondando para cima do valor, se fracionário.

cos()
Retorna o co-seno de um número.

cosh()
Retorna o co-seno hiperbólico de um número.

decbin()
Converte um número decimal para um número binário.

dechex()
Converte um número decimal para um número hexadecimal.

decoct()
Converte um número decimal para um número octal.

deg2rad()
Converte grau para radiano.

exp()
Retorna o valor da exponencial de um número.

expm1()
Retorna o valor da exponencial de um número e subtrai uma unidade.

floor()
Retorna o próximo menor valor inteiro ao se arredondar para baixo do valor, se necessário.

fmod()
Retorna o módulo da divisão dos argumentos.

getrandmax()
Retorna o valor randômico máximo que pode ser retornado pela função *rand()*.

hexdec()
Converte um número hexadecimal em um número decimal.

hypot()
Retorna o comprimento da hipotenusa de um triângulo retângulo.

is_finite()
Retorna *true* se um valor é um número finito.

is_infinite()
Retorna *true* se um valor é um "número" infinito.

is_nan()
Retorna *true* se um valor não é um número.

lcg_value()
Retorna um pseudo número randômico entre 0 e 1.

log()
Retorna o logaritmo natural de um número (base E).

log10()
Retorna o logaritmo na base 10 de um número.

log1p()
Retorna o logaritmo de 1 + um número.

max()
Retorna o número com o valor mais alto dentre dois especificados.

min()
Retorna o número com o valor mais baixo dentre dois especificados.

mt_getrandmax()
Retorna o valor mais comprido que pode ser retornado por *mt_rand()*.

mt_rand()
Retorna um inteiro randômico usando o algoritmo de Mersenne Twister.

octdec()
Converte um número octal em um número decimal.

pi()
Retorna o valor de PI.

pow()
Retorna a base elevada ao expoente exp.

rad2deg()
Converte um número em radiano para graus.

rand()
Retorna um inteiro randomicamente.

round()
Arredonda um número.

sin()
Retorna o seno de um número.

sinh()
Retorna o seno hiperbólico de um número.
sqrt()
Retorna a raiz quadrada de um número.

tan()
Retorna a tangente de um ângulo.

tanh()
Retorna a tangente hiperbólica de um ângulo.

PHP – Constantes Matemáticas

Na última coluna está a versão do PHP que suporta a constante especificada.

Constante	Descrição	PHP
M_E	Retorna e (aprox. 2.718)	4
M_EULER	Retorna a constante de Euler (aprox. 0.577)	4
M_LNPI	Retorna o logaritmo natural de PI (aprox. 1.144)	4
M_LN2	Retorna o logaritmo natural de 2 (aprox. 0.693)	4
M_LN10	Retorna o logaritmo natural de 10 (aprox. 2.302)	4
M_LOG2E	Retorna o logaritmo de E na base 2 (aprox. 1.442)	4
M_LOG10E	Retorna o logaritmo de E na base 10(aprox. 0.434)	4
M_PI	Retorna PI (aprox. 3.14159)	3
M_PI_2	Retorna PI/2 (aprox. 1.570)	4
M_PI_4	Retorna PI/4 (aprox. 0.785)	4
M_1_PI	Retorna 1/PI (aprox. 0.318)	4
M_2_PI	Retorna 2/PI (aprox. 0.636)	4
M_SQRTPI	Retorna a raiz quadrada de PI (aprox. 1.772)	4
M_2_SQRTPI	Retorna 2/raiz quadrada de PI (aprox. 1.128)	4
M_SQRT1_2	Retorna raiz quadrada de 1/2 (aprox. 0.707)	4
M_SQRT2	Retorna a raiz quadrada de 2 (aprox. 1.414)	4
M_SQRT3	Retorna a raiz quadrada de 3 (aprox. 1.732)	4

Exercícios Propostos

1. Dado um vetor qualquer com 8 números reais, faça um algoritmo que informa se há ou não números repetidos nesse vetor.

2. Crie um módulo que imprime na tela os elementos de um vetor de 5 números inteiros na ordem inversa.

3. Calcular a quantidade de dinheiro gasta por um fumante. Dados: o número de anos que ele fuma; o número de cigarros fumados por dia e o preço de uma carteira.

4. Ler dois números inteiros, x e y, e imprimir o quociente e o resto da divisão inteira entre eles.

5. Dados dois números inteiros (A e B), verificar e imprimir qual deles é o maior, ou a mensagem "A=B" caso sejam iguais.

6. Dados três valores, dizer se eles formam um triângulo. Caso afirmativo, dizer seu tipo (eqüilátero, isósceles ou escaleno).

7. Ler um número e verificar se ele é par ou ímpar.

8. Imprimir a série de Fibonacci.

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89,...

9. Calcular a soma dos primeiros termos da série:

1 + 1/2 + 1/3 + 1/4 + 1/5 + 1/6 + 1/7 + 1/8 + 1/9 + 1/10 + …

10. Faça um algoritmo para indicar o maior entre 3 números.

11. Faça um algoritmo para escrever por extenso qualquer número de 1 algarismo digitado pelo usuário.

12. Elabore um algoritmo para informar qual a idade do usuário (quantos anos tem, apenas), sendo informado a data atual e sua data de nascimento.

13. Faça um algoritmo para calcular o fatorial de qualquer número inteiro.

14. Faça um algoritmo para ordenar um vetor de números inteiros.

15. Construa um algoritmo que calcule o valor de um número n elevado a qualquer número x.

16. Calcule o dígito verificador de uma conta corrente segundo os seguintes critérios:
- Some cada algarismo da conta
- Se o resultado dessa soma for composto de apenas um algarismo, esse é o dígito verificador
- Se no resultado dessa soma houver mais de um algarismo, some-os até encontrar apenas um

algarismo que será, então, o dígito verificador.
- O algoritmo deve mostrar na tela a conta corrente completa, com o dígito verificador.

Por exemplo:

Conta Corrente: 46793
Soma dos algarismos: 4+6+7+9+3= 29
Como 29 possui dois algarismos, deve-se somar novamente: 2+9 = 11
Como 11 também possui dois algarismos, deve-se somar novamente: 1+1 = 2
Como 2 possui apenas um algarismo, esse é o dígito verificador procurado.
A resposta na tela deveria ser: 46793-2

17. Faça um algoritmo para calcular todos os números primos entre 2 e 1000.

18. Uma empresa decidiu fazer um levantamento sobre o perfil dos candidatos que se apresentarem para o preenchimento de vagas de seu quadro de funcionários. Para cada candidato, será solicitado o sexo, a idade e se tem ou não experiência no serviço. Esses dados serão guardados em vetores.
Escrever um algoritmo para ler os dados dos candidatos e, a seguir, calcular e imprimir as seguintes informações:
a) O número de candidatos do sexo feminino e masculino
b) A idade média das mulheres que já têm experiência no serviço
c) A menor idade entre as mulheres que já têm experiência no serviço
d) A porcentagem dos homens com mais de 45 anos, com relação ao total de homens.
e) O número de homens com idade inferior a 30 anos sem experiência no serviço.

19. Escreva funções de soma, subtração, multiplicação e divisão, passados dois valores por parâmetro e implemente-as.

20. Escreva uma única função que faça as quatro operações matemáticas elementares.

Formulários e Interação com HTML

Começaremos agora a interagir com o HTML, HyperText Marker Language, ou ainda, Linguagem de Marcação de Hipertexto.

Nosso primeiro passo nesta segunda etapa do guia é receber valores passados através de formulários HTML, trabalharmos com eles e exibir o resultado no document.

Por fim, teremos scripts muito mais flexíveis no sentido de interação com o usuário. Nos exemplos e exercícios acima não vimos nenhuma interação entre o usuário e o script.

Veja a resolução do exercício 5 com os dados passados pelo usuário através do teclado:

Arquivo html (num.html):

```
<html>
<head><title>Maior de Dois</title></head>
<body>

<form name="valores" action="maior.php" method="POST">
  Valor 1: <input type="text" name="valor1"><br>
  Valor 2: <input type="text" name="valor2"><br>
  <input type="submit" name="ok" value="Ok">
</form>

</body>
</html>
```

No arquivo acima temos um formulário com dois campos de texto onde serão digitados os valores e um botão submit para enviar os dados. Veja que na linha form temos um action com o nome do arquivo que receberá os dados digitados (no nosso caso maior.php) e qual será o method (no nosso caso POST).

É necessário que todos os campos que terão seus valores utilizados no script PHP tenham o parâmetro name.

O método indica como os dados serão passados. No nosso caso, POST significa que os dados serão passados sem serem exibidos na URL e possibilita o envio de longas informações, ao contrário do método GET.

Arquivo php (maior.php):

```php
<?
$valor1=$_POST['valor1']; //captura valor1 através do método POST
$valor2=$_POST['valor2'];//captura valor2 através do método POST

//verifica e imprime o maior dos dois valores
if($valor1>$valor2){
  echo $valor1.">".$valor2;
}else if($valor1<$valor2){
    echo $valor1."<".$valor2;
}else
    echo $valor1."=".$valor2;
?>
```

No exemplo acima poderíamos ter usado simplesmente $valor1 e $valor2 ao invés de $_POST, que o PHP entenderia que seriam os campos de formulário com mesmo nome. Mas desde que a diretiva *register_globals* no seu php.ini esteja ativada, caso contrário, $valor1=$valor2=0.

Também existe uma outra solução com o uso de *import_request_variables* que veremos mais adiante.

Ainda existem alguns problemas que precisamos tratar. Não sabemos se o usuário digitou dois números de fato, ele pode ter digito um texto ou até mesmo não ter digitado nada. Precisamos validar os nossos dados.

Veja como ficaria com as alterações de validação:

```php
<?
if (isset($_POST['valor1'], $_POST['valor2'])){ // verifica se as variáveis foram definidas

$valor1=$_POST['valor1']; //captura valor1 através do método POST
$valor2=$_POST['valor2']; //captura valor2 através do método POST

  if(!is_nan($valor1) && !is_nan($valor2)){ // verifica se valor1 e valor2 são ambos números

    //verifica e imprime o maior dos dois valores
    if($valor>$valor2){
        echo $valor1.">".$valor2;
    }else if($valor1<$valor2){
        echo $valor1."<".$valor2;
    }else{
    echo $valor1."=".$valor2;
    }
  }
}
?>
```

A função *isset* serve para verificar se uma variável foi definida ou não. A função *is_nan* verifica se o parâmetro passado é de fato um número.

Pronto, nosso exercício está resolvido.

Um pouco sobre Sessões

Basicamente, as sessões são métodos que preservam determinados dados ativos enquanto o navegador do cliente estiver aberto, ou enquanto a sessão não expirar.

Quando acessa um web site, um usuário recebe um identificador chamado id de sessão. Este é salvo em um cookie do lado do cliente ou propagado via URL. Você pode armazenar várias informações em uma sessão.

No site oficial do manual do PHP você encontrará diversas funções para manipular as sessões: http://www.php.net/manual/pt_BR/ref.session.php

Você já deve ter tido contado com sessões ao comprar em uma loja virtual e ter adicionado seus produtos ao carrinho de compras, ao fazer login de usuário em algum site, fórum e etc.

Em nossa aplicação desenvolveremos uma área de login para ilustrar o uso de sessões.

Veja este exemplo do uso de sessão:

```php
<?
ob_start();
session_start();
$nome="Daniel";
$idade="20 anos";
//gravando as informações das variáveis dentro das sessões
$_SESSION[nome] = $nome;
$_SESSION[idade] = $idade;
$_SESSION[user_agente] = $_SERVER['HTTP_USER_AGENT'];
?>
```

No exemplo acima utilizamos a função *ob_start()* que inicializa o buffer e impede qualquer saída para o navegador até que você o encerre. Em seguida, abrimos a sessão com a função *session_start()* e gravamos um nome e uma string com uma idade no array de sessão $_SESSION.

Também gravamos na sessão o *user agent,* isto é, a identificação do navegador.

MySQL

O MySQL é um Sistema de Gerenciamento de Banco de Dados (SGBD) da linguagem SQL – Structured Query Language, que significa linguagem de consulta estruturada.

Existem outros SGBD populares, mais o MySQL se destaca pela facilidade de integração aos scripts PHP.

Dentre suas características podemos destacar:

- Portabilidade (suporta praticamente qualquer plataforma atual);
- Compatibilidade com diversas linguagens de programação.
- Excelente desempenho e estabilidade.
- Não exige muito recurso de hardware.
- Fácil de usar.
- É um software livre com base na GPL.
- Suporta Triggers.
- Interfaces gráficas ([MySQL Toolkit]) de fácil utilização cedidos pela MySQL Inc.

Você pode adquirir mais informações sobre versões, documentação e downloads no site oficial do MySQL: http://www.mysql.com

Criando Banco de Dados e Tabelas com PHPMyAdmin

Basicamente, um banco é composto por tabelas que são formadas por campos. Cada tabela precisa ter um campo chave primária que faz referência a sua tabela em outras tabelas.

Agora utilizaremos o famoso PHPMyAdmin para criar nossos Banco de Dados e Tabelas. Se você utiliza o VertrigoServ pode acessá-lo iniciando o programa, em seguida clicando sobre o ícone na barra de tarefas. Um menu vai aparece em cima da barra, agora vá até Tools (ou ferramentas) e clique em PHPMyAdmin.

Caso esteja utilizando o Xampp, inicie o programa e em seguida abra o seu navegador. Digite http://localhost, você verá a página principal do Xampp, no menu esquerdo procure por Tools (ou ferramentas) e pelo PHPMyAdmin. Pronto.

Na página principal do PHPMyAdmin encontramos do lado esquerdo um menu superior com os botões: Home (para voltar a página principal do PHPMyAdmin), SQL - Query Window (para escrever um script SQL ou importá-lo de algum arquivo texto), PHPMyAdmin Documentation (para visualizar a

documentação do seu PHPMyAdmin) e Documentation (para acessar a documentação do SQL).

Logo embaixo você pode visualizar todos os bancos de dados criados. Clicando sobre eles pode visualizar as tabelas de cada banco e clicando sobre elas pode visualizar os seus campos, com mais um clique sobre Browser, poderá visualizar os dados armazenados nela.

Agora vamos criar nosso primeiro banco de dados com PHPMyAdmin. Acesse-o pelo seu navegador e em seguida, no lado direito você verá escrito “create new database” e um campo text. Se o seu PHPMyAdmin estiver configurado em português provavelmente estará “criar novo banco de dados”.

Neste campo text digitaremos o nome do banco que queremos criar, no nosso caso, digite “infpessoais” sem as aspas. Agora clique no botão “create”.

Pronto, nosso banco de dados infpessoais está criado. Note que aparecerá no seu navegador uma coluna à esquerda com o nome do seu banco e um zero entre parêntesis indicando que ele não possue nenhuma tabela. Do lado direito do document do navegador você verá além do menu superior do PHPMyAdmin, a mensagem “No tables found in database” e logo abaixo “Create new table on database infpessoais”, isto é, crie uma nova tabela no banco de dados infpessoais.

Então no campo text indicado como “name” digite “informacoes” que é o nome da nossa tabela. Em “Number of fields” digite 3, isso significa que nossa tabela terá 3 campos. Agora clique em “go”.

Agora temos a tela de inserção dos 3 campos da nossa tabela “informacoes”. Na mesma coluna escreva no primeiro text “id_contato”, tipo INT, escolha o extra “auto_increment” e o radio button “Primary”. Com isso você está dizendo que o campo id_contato é uma chave primária, ou seja, é o índice da tabela que pode fazer referência dela em outras tabelas do mesmo banco. Nesse primeiro exemplo só temos uma tabela. O argumento auto_increment diz que ao adicionar um registro na tabela esse campo é automaticamente incrementado de 1.

Nas duas últimas colunas insira respectivamente nos campos text “nome” e “telefone” e escolha o tipo dos campos como TEXT.

Agora clique em “go” e nossa tabela informacoes está pronta.

Clicando em “informacoes” no lado esquerdo podemos ver todos os campos da nossa tabela. Veja que clicando em “Insert” no menu superior do PHPMyAdmin é possível inserir dados nos campos da nossa tabela. Mas isso não tem muita graça aqui, pois queremos fazer isso através do PHP.

Alternativamente, podemos criar um banco de dados, criar tabelas neste banco e inserir seus campos através do código SQL correspondente. Perderíamos a interface intuitiva do PHPMyAdmin, mas ganharíamos na praticidade.

Veja como ficaria o código para criar o banco infpessoais e a tabela informacoes com seus campos:

```
CREATE TABLE IF NOT EXISTS `informacoes` (
  `id_contato` int(11) NOT NULL auto_increment,
  `nome` text NOT NULL,
  `telefone` text NOT NULL,
  PRIMARY KEY  (`id_contato`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
```

O atributo NOT NULL significa que o campo não pode ser nulo. Isso é especificado pelo programador.

Inserir, deletar e selecionar com MySQL

Agora que temos nosso banco “infpessoais” criado, podemos inserir, deletar e selecionar dados nele. Afinal, é para isso que ele serve.

Nesta parte da apostila veremos como fazer isso em código SQL e posteriormente, como fazer através do PHP.

Inserir

Suponha que você queira adicionar um contato na sua tabela “informacoes”. Veja o exemplo abaixo de como fazer:

```
INSERT INTO `infpessoais`.`informacoes` (
`id_contato` ,
`nome` ,
`telefone`
)
VALUES (
'1', 'Daniel Moreira dos Santos', '9999-0000'
);
```

O código acima significa: Insira no banco 'infpessoais', na tabela 'informacoes' que possui os campos 'id_contato', 'nome' e 'telefone', os valores '1', 'Daniel Moreira dos Santos', '9999-0000' respectivamente.

Deletar

```
DELETE FROM `informacoes` WHERE `id_contato` = 1
```

Percebemos que o código é bem sugestivo. Delete de 'informacoes', onde 'id_contato' = 1. Lembre que id_contato é nossa chave primária.

Selecionar

Através do código SQL podemos selecionar um registro ou grupo de registros que satisfazem à uma condição estabelecida. Como exemplo, suponha que temos os seguintes registros na tabela ”informacoes”:

1	Daniel Moreira dos Santos	9999-0000
2	José Carlos de Oliveira Jr.	0000-9999
3	Jairo Ludtke Jr.	8888-0000

E queremos selecionar para um fim, (mostrar em uma tabela, por exemplo), os registros que possuem 8 no número de telefone. Nosso código ficaria assim:

```
SELECT * FROM `informacoes` WHERE telefone LIKE `%"8888-8888"%` ORDER BY `id_contato`
```

O asterisco (*) no código acima faz referência a todos os campos da tabela informacoes. Então temos, “Selecione todos os campos da tabela informacoes onde telefone parece com 8888-8888 e ordene por id_contato”.

Atualizar

Suponha que queremos atualizar o telefone de um dos contatos registrados. Para isso, usamos o comando UPDATE. Veja o exemplo:

```
UPDATE `informacoes` SET telefone=`9999-9999` WHERE id_contato=`2`
```

Note que só será atualizado o telefone do registro com id_contato igual a 2, que por sua vez é único.

Destruir

```
DROP TABLE `informacoes`
```

O comando DROP exclui definitivamente a nossa tabela com todos os seus registros. É bom tomar certo cuidado com este comando.

Funções do MySQL

Assim como o PHP, o MySQL possui algumas funções muito úteis para manipulação dos registros de um banco de dados. Aqui registrei algumas:

avg(coluna)
Retorna a média dos valores da coluna.

count(item)
Se item for uma coluna, será retornado o número de valores não nulos(NULL) nesta coluna.

Se a palavra-chave DISTINCT for colocada na frente do nome da coluna, será retornado o número de valores distintos nesta coluna.

Se for passado count(*), será retornado o número total de registros independente de quantos tenham valor NULL.

min(coluna)
Retorna o valor mínimo da coluna.

max(coluna)
Retorna o valor máximo da coluna.

sdt(coluna)
Retorna o desvio padrão dos valores da coluna.

sdtdev(coluna)
O mesmo que SDT(coluna).

sum(coluna)
Retorna a soma dos valores da coluna.

PHP interagindo com o Banco de Dados MySQL

Chegou a hora de interagir com o banco de dados e preparar o terreno para a construção da nossa aplicação PHP/MySQL.

Refaça a sua tabela 'informacoes' no banco de dados infpessoais. Iremos utilizar o PHP para inserir e manipular alguns registros.

mysql_affected_rows()
Retorna o número de linhas atingidas na operação anterior do MySQL. Obtém o número de linhas atingidas pela consulta INSERT, UPDATE, REPLACE ou DELETE mais recente associada ao *link_identifier.*

mysql_close()
Fecha a conexão MySQL não persistente ao servidor MySQL que esta associado ao identificador de conexão dado. Se *link_identifier* não for especificado, a ultima conexão aberta é usada. Retorna true em caso de êxito e false caso contrário.

mysql_connect()
Abre uma conexão com um servidor MySQL. Os parâmetros são: Server, usuário e senha do banco de dados.

mysql_fetch_array()
Obtém uma linha como uma matriz associativa, uma matriz numérica, ou ambas. Retorna uma matriz que corresponde a linha obtida e move o ponteiro interno dos dados adiante.

mysql_fetch_row()
Retorna o resultado de uma linha numa matriz numérica. Retorna uma matriz numérica que corresponde à linha, ou false se não houver mais linhas.

mysql_field_len()
Retorna o tamanho do campo especificado.

mysql_field_name()
Retorna o nome do campo especificado no resultado de uma query.

mysql_field_table()
Retorna o nome da tabela onde esta o campo especificado.

mysql_field_type()
Retorna o tipo do campo especificado em um resultado de query.

mysql_get_client_info()
Retorna informação da versão do cliente MySQL, uma string que representa a versão da biblioteca do cliente.

mysql_get_server_info()
Retorna informação do servidor MySQL.

mysql_num_rows()
Retorna o número de linhas em um resultado. Este comando é valido apenas para obter o número de linhas afetadas por um SELECT.

mysql_query()
Realiza uma query MySQL. Envia uma query para o banco de dados ativo no servidor da conexão informada através do identificador.

Se o parâmetro *identificador* não é especificado, a ultima conexão aberta é usada. Se nenhuma conexão esta aberta, a função tenta estabelecer uma conexão como *mysql_connect()* seja chamada sem argumentos e usa-a. O resultado é guardado em buffer.

Uma observação é que a string da query não deve terminar com ponto e vírgula (;).

mysql_select_db()
Seleciona um banco de dados MySQL. Veremos como isso é feito mais adiante.

Construindo uma Aplicação PHP+MySQL

Agora chegamos à parte mais esperada, onde aplicaremos tudo que aprendemos nas páginas anteriores. Vamos construir como exemplo de aplicação, uma agenda de contatos com: login, cadastro, exclusão e edição de contatos.

Iremos utilizar o banco de dados infpessoais criado anteriormente e a tabela informacoes para armazenar os dados dos nossos contatos.

Em primeiro lugar vamos construir a tela de login de usuários. Como você já está familiarizado com HTML não terá dificuldades e poderá fazer como quiser, modificando e melhorando este simples exemplo.

```
<!-- Script da Tela de Login da Agenda de Contatos  -->
<html>
<head>
<title>Agenda de Contatos</title>
</head>
<body>
<br><br><br><br><br>
<br><br><br><br><br>
<center>
<table align="center" cellpadding="5" cellspacing="5" border="1">
<form name="logar" action="logar.php" method="POST">
        <tr>
                <td>User:</td><td><input type="text" name="login"></td>
        </tr>
        <tr>
                <td>Pass:</td><td><input type="password" name="senha"></td>
        </tr>
        <tr>
                <td colspan="2" align="right"><input type="submit" name="entrar" value="Entrar"></td>
        </tr>
</form>
</table>
</center>
</body>
</html>
```

Observe que os nomes dos campos são login e senha e não User e Pass. Então, nossa tela inicial ficou assim:

Agora, precisamos verificar se o usuário e a senha foram digitados corretamente. Essa verificação nós faremos no arquivo apontado pelo “action” que está no cabeçalho do nosso formulário, que neste caso é o arquivo logar.php. Veja:

```
<?
ob_start();

$login = $_POST['login'];
$senha = $_POST['senha'];

if ($login=="danielms" && $senha=="cX%u@lka!_0"){

$dia = date('d');
$mes = date('m');
$ano = date('Y');
$data_login=$dia."/".$mes."/".$ano;

session_start();

//gravo as informações das variáveis dentro da sessão
$_SESSION[login] = $login;
$_SESSION[data_login] = $data_login;

header ("Location: agenda.php"); // entra na agenda de contatos
}
else{

header ("Location: logar.htm"); //volta para a tela de login
}
?>
```

Note que nosso exemplo de login ainda é muito simples, sem interação com banco de dados e possui apenas um usuário e senha.

Abrimos e gravamos em seção as variáveis $login (que no nosso caso é a string “danielms”) e $data_login que é a data em que foi acessada a agenda de contatos.

Logando com sucesso, veremos a tela principal da nossa aplicação. Esta é bem simples e possui um visual feito com HTML bem básico.

Este é o arquivo agenda.php:

```
<?
ob_start();
session_start();
if($_SESSION[login] == "danielms"){
?>

<html>
<head>
<title>Agenda de Contatos</title>
</head>
<body>

<table align="center" cellpadding="5" cellspacing="5" border="1" width="800" height="460">
        <tr><td><b>Bem-vindo <? echo $_SESSION[login]; ?></b> Data: <? echo $_SESSION[data_login]; ?></td><td height="10"><a href="form_inserir.php"
target="principal">Inserir</a> | <a href="deslogar.php">SAIR</a> </td></tr><tr>
            <td colspan="2">
                <iframe src="listar.php" width="800" height="450" name="principal"></iframe>
            </td>

        </tr>
</table>

</body>
</html>

<?
}else{
echo "Sem permissões para acessar a página!";
}
?>
```

A tela principal está dividida na parte superior com as boas-vindas e os links “Inserir” e “SAIR” e na parte inferior onde os contatos são listados e podem ser editados ou excluídos.

Abaixo segue o arquivo listar.php:

```
<?
ob_start();
session_start();
if($_SESSION[login] == "danielms"){
?>
<?
$conexao = mysql_connect("localhost", "usuario", "senha") or die("Nao pude conectar: " . mysql_error());
$base=mysql_select_db("infpessoais", $conexao) or die("Nao pude selecionar o banco de dados");
?>

<table align="center" border="1" cellspacing="0" cellpadding="0" bordercolor="#000000">
<tr align="center" bgcolor="#778899"><td><center><b><font color="#FFFFFF" size="-1">Id</font></b></center></td><td><center><b><font color="#FFFFFF" size="-1">Nome</
font></b></center></td><td><center><b><font color="#FFFFFF" size="-1">Telefone</font></b></center></td><td><center><b><font color="#FFFFFF" size="-1">Editar</font></
b></center></td><td><center><b><font color="#FFFFFF" size="-1">Excluir</font></b></center></td></tr>
<?
$query="SELECT * FROM informacoes";
$resultado=mysql_query($query, $conexao);
while($linha=mysql_fetch_row($resultado)){
?>
<tr bgcolor="#D3D3D3">
<td><font color="#000000" size="-1"><? echo $linha[0]; ?></font></td>
<td><font color="#000000" size="-1"><? echo $linha[1]; ?></font></td>
<td><font color="#000000" size="-1"><? echo $linha[2]; ?></font></td>
<td><center><a href="form_editar.php?id=<? echo $linha[0]; ?>">+++</a></center></td>
<td><center><a href="excluir.php?id=<? echo $linha[0]; ?>">XX</a></center></td>
</tr>
<?}?>
</table>
<?
}else{
echo "Sem permissões para acessar a página!";
}
?>
```

O próximo passo é a inserção de contatos no sistema. Segue os arquivos form_inserir.php e inserir.php respectivamente. Quando clicamos em “Inserir” no menu superior da agenda, o arquivo form_inserir.php é chamado no frame principal, onde encontramos um formulário a ser preenchido.

```
<?
ob_start();
session_start();
if($_SESSION[login] == "danielms"){
?>
<table align="center" border="1" cellspacing="0" cellpadding="0">
<form name="email" method="POST" action="inserir.php">
<tr><td>Nome:</td><td><input type="text" name="nome"></td></tr>
<tr><td>Tel.:</td><td><input type="text" name="telefone"></td></tr>
<tr align="right"><td colspan="2"><input type="submit" value="Inserir"></td></tr>
</form>
</table>

<?
}else{
echo "Sem permissões para acessar a página!";
}
?>
```

```
<?
ob_start();
session_start();
if($_SESSION[login] == "danielms"){
?>
<?
header("Location: listar.php");

$conexao = mysql_connect("localhost", "usuario", "senha") or die("Nao pude conectar: " . mysql_error()); // usuario e senha do MySQL
$base=mysql_select_db("infpessoais", $conexao) or die("Nao pude selecionar o banco de dados");

$nome=$_POST['nome'];
$telefone=$_POST['telefone'];

$query="INSERT INTO informacoes(id_contato, nome, telefone) VALUES(null, '$nome', '$telefone')";
mysql_query($query, $conexao);
?>

<?
}else{
echo "Sem permissões para acessar a página!";
}
?>
```

Agora vamos à edição de contatos cadastrados no sistema. Segue os arquivos form_editar.php e editar.php respectivamente. Quando clicamos em "++" ao lado do contato listado na agenda, o arquivo form_editar.php é chamado no frame principal, onde encontramos um formulário preenchido com os dados do contato e estes podem ser alterados.

```
<?
ob_start();
session_start();
if($_SESSION[login] == "danielms"){
?>
<?
$conexao = mysql_connect("localhost", "usuario", "senha") or die("Nao pude conectar: " . mysql_error());
$base=mysql_select_db("infpessoais", $conexao) or die("Nao pude selecionar o banco de dados");

$query="SELECT * FROM informacoes WHERE id_contato='$id'";
$resultado=mysql_query($query, $conexao);
$linha=mysql_fetch_row($resultado);
?>
<table align="center" border="1" cellspacing="0" cellpadding="0">
<form name="telefone" method="POST" action="editar.php">
<tr><td>Nome:</td><td><input type="text" name="nome" value="<? echo $linha[1]; ?>" ></td></tr>
<tr><td>Tel.:</td><td><input type="text" name="telefone" value="<? echo $linha[2]; ?>" ></td></tr>
<tr align="right"><td colspan="2"><input type="hidden" name="id" value="<? echo $id; ?>"><input type="submit" value="Editar"></td></tr>
</form>
</table>
<?
}else{
echo "Sem permissões para acessar a página!";
}
?>
```

```
<?
ob_start();
session_start();
if($_SESSION[login] == "danielms"){
?>
<?
$conexao = mysql_connect("localhost", "usuario", "senha") or die("Nao pude conectar: " . mysql_error());
$base=mysql_select_db("infpessoais", $conexao) or die("Nao pude selecionar o banco de dados");

$id=$_POST['id'];
$nome=$_POST['nome'];
$telefone=$_POST['telefone'];

$query="UPDATE informacoes SET nome='$nome', telefone='$telefone' WHERE id_contato='$id'";
mysql_query($query, $conexao);
header("Location: listar.php");
?>
<?
}else{
echo "Sem permissões para acessar a página!";
}
?>
```

Quando clicamos em “XX” ao lado do contato listado na agenda, o arquivo excluir.php é chamado no frame principal e o registro é então eliminado da tabela informacoes. Veja o arquivo excluir.php:

```php
<?
ob_start();
session_start();
if($_SESSION[login] == "danielms"){
?>
<?
header("Location: listar.php");

$conexao = mysql_connect("localhost", "usuario", "senha") or die("Nao pude conectar: " . mysql_error());
$base=mysql_select_db("infpessoais", $conexao) or die("Nao pude selecionar o banco de dados");

$query="DELETE FROM informacoes WHERE id_contato='$id'";
mysql_query($query, $conexao);
?>
<?
}else{
echo "Sem permissões para acessar a página!";
}
?>
```

Ao clicarmos no link “SAIR” a sessão é encerrada e voltamos a tela inicial de login da aplicação. Isso acontece porque o link chama o arquivo deslogar.php abaixo:

```php
<?
ob_start();
session_start();

//destrói a sessão
unset($_SESSION[login]);
unset($_SESSION[data_login]);
session_destroy();

header("Location: logar.htm");
?>
```

Exercícios Propostos

Faça uma agenda de contatos com login e interação ao banco de dados, cadastro, exclusão e edição de contatos, envio de e-mail com formatação de texto e imagem, aviso de compromissos, upload de foto no cadastro do contato, cadastro múltiplo de contatos através de leitura de arquivo texto.

Este exercício será resolvido na próxima versão do guia. Um abraço a todos.

Referências Bibliográficas

MANUAL do PHP. http://php.net/manual/pt_BR/index.php

PHP 5 AND MYSQL BIBLE. Tim Converse and Joyce Park with Clark Morgan. Wiley Publishing, Inc.

Beginning PHP 5, Apache, MySQL Web Development. Elizabeth Naramore, Jason Gerner, Yann Le Scouarnec, Jeremy Stolz, Michael K. Glass. Wrox.

Códigos na Web. http://www.codigosnaweb.com

HTML Staff. http://www.htmlstaff.org