

PROGRAMAÇÃO PHP

```
<?php
```

```
echo "INICIANTE";
```

```
echo "INTERMEDIÁRIO";
```

```
echo "AVANÇADO";
```

```
?>
```



Anderson Makiyama

Atenção: Este ebook é vendido com direitos de revenda inclusos. As pessoas estão autorizadas a: fazer cópias, revendê-las ou distribuí-las quantas vezes desejarem, porém **é expressamente proibido alterar o conteúdo deste material**, sob pena de ser processado pelo autor da obra. Fica eleito o foro de Joinville, Santa Catarina, para dirimir as questões decorrentes da execução destes termos!

PROGRAMAÇÃO PHP

Iniciante, Intermediário e Avançado

Anderson Makiyama

PROGRAMAÇÃO PHP

Iniciante,
Intermediário e
Avançado

Sobre o Autor

Anderson Makiyama é um programador de computador que dedica a maior parte de seu tempo criando soluções para a Internet e escrevendo sobre o assunto. É formado em Sistemas para a Internet pela faculdade Anhanguera. Vive com a esposa no Brasil na cidade de Joinville. Ele mantém alguns sites na internet nos endereços:

<http://Fazer-Site.net>

<http://GanharDinheiroBlog.net>

CAPÍTULO 1

Introdução ao PHP	11
Por que aprender PHP	12

CAPÍTULO 2

Rodando PHP Localmente	13
Instalando um servidor web localmente	14
Alô mundo no PHP.....	19

CAPÍTULO 3

Iniciando em Programação PHP	23
Entendendo as Variáveis	26
Entendendo Funções.....	28
Concatenação.....	30
Operadores Aritméticos.....	31
Operadores de Comparação.....	33
Estruturas de repetição.....	37
Comentários.....	39
If / ElseIf / Else.....	40
Switch / Case	42

CAPÍTULO 4

Avançando em Programação PHP ...	45
Trabalhando com Arrays	49
Trabalhando com Datas	56
Validando Datas	58
Enviando Emails.....	61
Verificando se uma Variável está vazia	63
Validando CPF	64
Incluindo arquivos externos	65

Trabalhando com casas decimais	66
Dominando a Função Preg_Match	68
Dominando Expressões Regulares	73
Dominando a Função STR_Replace	78
Dominando a Função PREG_Replace.....	81
Redirecionando no PHP	86
Tirando Espaços em Branco de Variáveis	88
Renomeando Arquivos e Pastas	91
Trabalhando com Permissões	93
Variáveis de Sessão	95

CAPÍTULO 5

Trabalhando com Formulários	105
Passando valores entre páginas PHP	108

CAPÍTULO 6

Trabalhando com Banco de Dados	115
Testando a conexão com o Banco de dados	
Mysql	116
Inserindo dados no banco Mysql	122
Excluindo registros do banco Mysql	124
Atualizando dados do Banco Mysql	126
Renomeando um Banco de Dados Mysql	132

CAPÍTULO 7

Erros e Soluções	135
Fatal Error: Maximum Execution Time	
Exceeded.....	136
Erro 500 – Internal Server Error	140

Fatal Error: Register Globals is disabled in php.ini	142
Eregi() is deprecated	144
Cannot Modify Header Information	145

CAPÍTULO 8

Programação Orientada a Objetos .	149
Construtores e Destrutores.....	151
Trabalhando com Herança.....	153
Entendendo os Métodos	155
Acessibilidade dos Métodos e propriedades...	156
Entendendo Constantes	159
Trabalhando com Atributos e Métodos Static	163
PDO (PHP Data Object)	166

CAPÍTULO 9

Desenvolvendo um Chat (Sistema de Bate-Papo)	171
Definindo o funcionamento do Chat	172
Estruturando o Banco de Dados	174
Codificando o Chat.....	176

CAPÍTULO 10

Breve Despedida	179
------------------------------	------------

CAPÍTULO 1

Introdução ao PHP

O PHP está entre as linguagens de programação web mais utilizadas no mundo, seja por aprendizes ou mesmo por profissionais web. Todo esse sucesso deve-se a vários fatores, dentre os quais podemos destacar:

1- sintaxe de fácil entendimento e aprendizado.

2- possui interpretadores em várias plataformas de sistemas operacionais, inclusive linux e windows.

3- possui suporte para programação orientada a objeto (POO).

Quando se está aprendendo uma nova linguagem de programação, a melhor forma de se assimilar a sintaxe é a repetição prática. Pensando nisso, veremos no decorrer deste livro, vários exemplos práticos para facilitar o entendimento do candidato a programador PHP.

Por que aprender PHP

Como já citado no começo deste livro, a linguagem de programação PHP é uma das mais utilizadas atualmente, então, caso você pretenda ser um profissional Web, certamente irá precisar dominar a linguagem de programação PHP, seja para criar soluções, seja para editar soluções web já existentes que foram criadas por outros programadores.

CAPÍTULO 2

Rodando PHP Localmente

A melhor forma para se testar scripts que você irá desenvolver na linguagem de programação PHP, é você instalar um servidor web no seu próprio computador e rodar os scripts localmente. Vamos supor que você esteja fazendo alterações em um determinado site a pedido de um determinado cliente. Não seria um bocado desagradável se os visitantes do tal site se deparassem com uma mensagem de erro ao acessar ele? Pois bem, eis aí mais uma grande razão para se ter um servidor web instalado em seu computador e testar os códigos localmente, antes de publicá-los para o mundo.

Instalando um servidor web localmente

Visto que a maioria esmagadora dos usuários de computadores, são fãs, ou ao menos usuários de algum sabor do sistema operacional Windows da Microsoft, irei abordar nesse livro a instalação nesse OS.

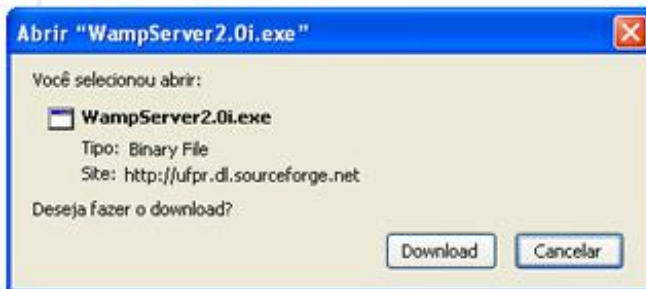
Irei utilizar o Wamp Server, que é um pacote de aplicativos que já vem com: Apache 2.2.11, PHP 5.3.0, MySQL 5.1.36 e PhpMyAdmin. Note que Apache é o nome do servidor web; PHP, aí, refere-se ao interpretador PHP que vem incluso no servidor web; MySQL é o servidor do Banco de Dados; e por fim, PhpMyAdmin é um script para gerenciamento do banco.

Veremos agora como baixar e instalar o WampServer passo a passo:

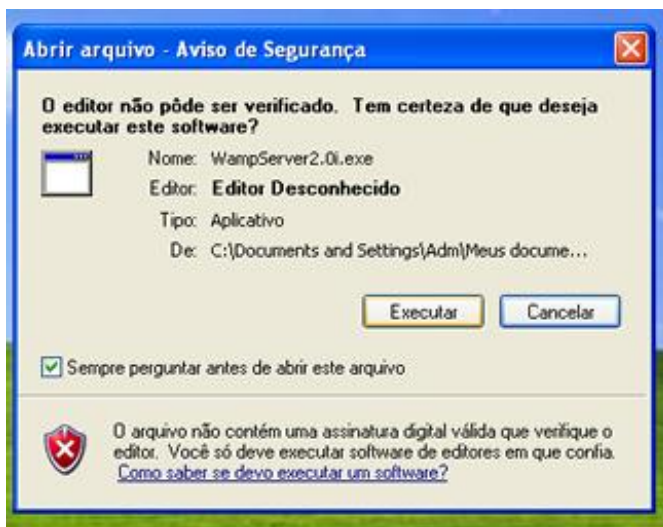
- 1- Acesse o site do programa <http://www.wampserver.com> e clique no menu DOWNLOAD.
- 2- Na página de download, haverá um formulário de inscrição que é opcional, para não perdermos muito tempo, apenas clique sobre o link **TELECHARGER WampServer 2.0i** que fica no final da página. Caso prefira, poderá acessar diretamente o seguinte url: <http://www.wampserver.com/dl.php>
- 3- Se tudo ocorrer conforme o esperado, você será redirecionado para o site sourceforge.net e uma janela de autorização do download será exibida em seu navegador.
- 4- Clique no botão Download para baixar o programa para seu computador.

Your **WampServer** download will start shortly

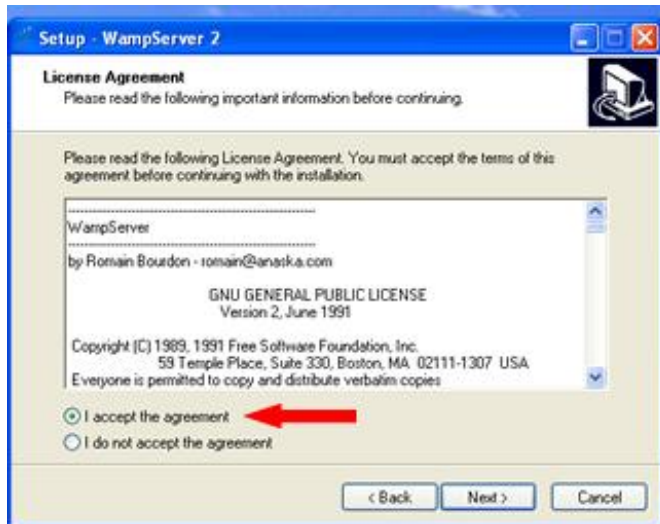
Problems with the download? Please use this [direct link](#) or try another mirror



- 5- Após concluído o download, execute o programa recém baixado. Caso o sistema exiba algum alerta de segurança com uma confirmação de execução de software, clique em executar para permitir a instalação do servidor em seu sistema local.



- 6- Agora o assistente de instalação do WampServer irá auxiliar você durante a instalação. Na primeira tela clique em Next. Na segunda tela “License Agreement” marque a opção “I accept the agreement” e clique em Next.



- 7- A próxima tela é muito importante, porque é nessa etapa que você irá definir onde o WampServer deve ser instalado. Por padrão ele é instalado em c:\wamp e eu lhe aconselho a não alterar esse caminho. Nesse livro, vou considerar que você instalou o WampServer no caminho padrão. Clique no botão Next para continuar com a instalação.
- 8- Na próxima tela marque as duas opções “Create a Quick Launch icon” e “Create a Desktop icon”. Essas opções vão criar um ícone de inicialização rápida e um ícone na sua área de trabalho para poder iniciar o Wamp mais facilmente. Clique em Next para ir para a próxima tela.
- 9- Agora é só concluir a instalação clicando no botão “Install” e aguardar a conclusão do

processo. Note que se durante o processo de instalação, for pedido para você escolher o browser padrão (Please choose your default browser), basta clicar em abrir para acelerar o processo de instalação. Ainda durante a instalação, o firewall do windows poderá pedir uma confirmação de bloqueio do Apache, nesse caso clique em Desbloquear. Por fim, você ainda poderá ser perguntado sobre parâmetros de servidor smtp, nesse caso apenas clique em Next.

- 10- Por fim, quando chegar na tela “Completing the WampServer 2 Setup Wizard”, clique em Finish e pronto, o Wamp está pronto para ser utilizado.

Alô mundo no PHP

Caso você ainda não tenha executado o WampServer após instala-lo, faça isso agora. Para isso vá em Iniciar >> Todos os Programas >> WampServer >> Iniciar WampServer. Uma vez em execução, o ícone do programa será exibido na barra de tarefas. Veja na imagem:



Agora abra o explorer e acesse o caminho c:\wamp ou se preferir vá em Iniciar >> Executar e digite c:\wamp. Note que todos os arquivos que estão dentro da pasta wamp compõem o aplicativo WampServer. Note ainda que dentro da pasta wamp existe uma outra pasta chamada www. Essa pasta é a área pública. É nela que o seu website deve estar para que esteja acessível para os visitantes.

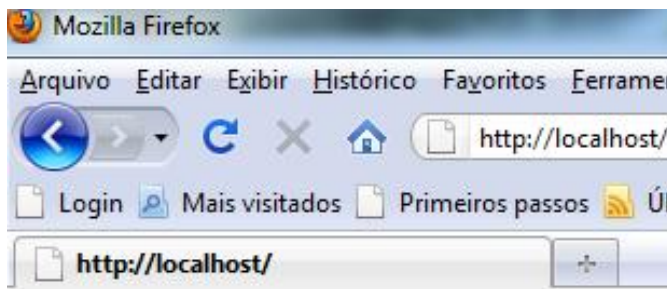
Abra a pasta www. Dentro da pasta www existe uma página chamada index.php que é criada automaticamente durante a instalação do WampServer. Pode excluir esse arquivo “index.php”. Agora crie um novo arquivo em seu editor html/php preferido – eu gosto do Dreamweaver -, e dê a ele ou salve-o com o nome de index.php. Para facilitar nossa comunicação, vou considerar que você está utilizando o Dreamweaver, visto que é um dos editores mais utilizados. Na parte de edição de código digite o seguinte:

```
<?php  
echo “Alo Mundo!”;
```

?>

Por incrível que pareça, esse é o nosso “hello world” em PHP. Viram como é simples? Agora salve as alterações que fizemos no arquivo index.php, acesse seu navegador web preferido – eu gosto do Firefox -, e digite na barra de endereço o seguinte: <http://localhost>. Se você seguiu o que descrevi acima, deveria estar vendo a mensagem “**Alo mundo**” impressa no navegador web. Como isso ocorreu? Bem, o servidor web sempre que acessado, irá procurar pela página index.php e caso encontre essa página, irá executar o código php que existe nela e imprimir para o usuário o resultado em html puro.

Os códigos php não são visíveis para os usuários que acessam seu website. O que os visitantes conseguem ver é o código html da página, mas jamais o código PHP. Esse sistema servidor/cliente permite você manter seu código PHP a salvo dos plagiadores digitais.



Alo Mundo!

Nos exemplos a seguir, sempre que for testar um exemplo ou código que está no livro, você deve abrir esse arquivo **index.php** no Dreamweaver, excluir o código que já havia nele e escrever o novo código. Não se esqueça de sempre salvar o arquivo antes de abrir o url (<http://localhost>) no navegador web.

As tags **<?php** e **?>**, indicam respectivamente o início e o fim de código php. Isso é importante para o interpretador saber o que deve ser interpretado e o que não deve. Ah, outra informação interessante para quem não sabe, é que esse url <http://localhost> é o endereço da sua própria placa de rede. Ou seja, quando você acessa esse url no navegador, ele tentará acessar algum servidor web instalado na sua máquina na porta 80. Talvez você esteja pensando: “não configuramos nenhuma porta”. Bem, de fato, nós não precisamos configurar, uma vez que o Apache, que é instalado junto no pacote WampServer, já vem configurado para ficar “esperando” conexões na porta 80.

CAPÍTULO 3

Iniciando em Programação PHP

PHP possui algumas funções responsáveis por imprimir valores e strings para os usuários. Dentre essas funções, a mais utilizada é a echo. A sintaxe da echo é a seguinte:

```
<?php  
    echo (“qualquer coisa aqui”);  
?>
```

No final de cada comando do PHP, você precisa inserir um ponto-e-vírgula. Você também pode imprimir códigos HTML, nesse caso, eles serão interpretados e não exibidos para o usuário. Por exemplo, caso você imprimir o código
, o navegador irá entender que é uma quebra de linha. Não sei se você, caro leitor, conhece bem html, mas antes de continuarmos vou colocar aqui alguns códigos html básicos e suas funções:

 - quebra de linha

 - negrito

<h1></h1> - título

<hr> - linha horizontal

<div></div> - insere um div

<input type='text' name='txt'> - insere uma entrada de texto.

Note que existem inúmeros códigos html, não vou me estender neles porque foge do foco deste livro, mas se você pretende ser um profissional web, certamente deverá buscar mais conhecimento sobre codificação html e estilos.

Vamos imprimir alguns códigos html junto com textos para você apreender bem a informação.

```
<?php  
echo "teste de quebra de linha<br>outra linha";  
?>
```

O código acima irá imprimir exatamente o seguinte:

**teste de quebra de linha
outra linha**

Vejamos mais exemplos:

```
<?php  
echo "<h1>Titulo aqui</h1>";  
echo "texto simples";  
echo "<br>";  
echo "<strong>texto em negrito</strong>";  
?>
```

Você pode fazer outras combinações com códigos html que você conhece. Não precisa ficar com receio, lembre-se que a hora de cometer os erros e aprender a acertar, é exatamente agora.

Além da função **echo** o PHP também possui uma função muito similar, que é a função **print**. Veja como é a sintaxe dessa função:

```
<?php  
print("usando o print");
```

?>

Os mesmos exemplos que fizemos para **echo** também são válidos para a função **print**

Entendendo as Variáveis

Variáveis servem para armazenar dados temporariamente. Em PHP, o sinal de cifrão antes de um nome, indica que se trata de uma variável. Uma variável pode conter vários tipos de dados. O tempo de vida de uma variável é “run time”, ou seja, o servidor armazena seus valores enquanto o script onde ela aparece estiver em execução. Note que existem ainda variáveis de sessão, essas variáveis são capazes de armazenar dados até que a sessão expire. Fique tranquilo, falarei sobre variáveis de sessão mais à frente. Exemplos com variáveis:

```
<?php
$variavel_01 = “Sou a variavel 01”;
$variavel_02 = “Sou a variavel 02”;

echo $variavel_01;
echo “<br>”;
echo $variavel_02;
?>
```

Não utilize caracteres especiais ou acentos em nome de variáveis, e sempre inicie o nome com

o símbolo cifrão \$ para que o PHP entenda que se trata de uma variável. Acima, utilizei nomes bem intuitivos, mas você pode utilizar qualquer nome que seja sugestivo para você. Veja esse exemplo:

```
<?php
$preto = "cor preta = black";
$branco = "cor branca = white";
echo $preto;
echo "<br>";
echo $branco;
?>
```

Ou ainda:

```
<?php
$meu_primeiro_carro = "Monza";
$minha_primeira_moto = "Titan";
echo $meu_primeiro_carro;
echo "<br>";
echo $minha_primeira_moto;
?>
```

Você também pode utilizar as variáveis para trabalhar com valores: somar, dividir, multiplicar, etc. Vejam os exemplos:

```
<?php
$total = 15;
$saida = 4;
echo $total - $saida;
echo "<br>";
```

```
echo $total + $saida;  
echo "<br>";  
echo $total / $saida;  
echo "<br>";  
echo $total * $saida;  
?>
```

Entendendo Funções

A utilidade de uma função é fazer uma tarefa específica sempre que invocada. Algumas funções podem retornar valores, outras apenas processar dados. No núcleo do PHP já existem várias funções prontas, as quais você precisa apenas saber o nome delas e invoca-las. Como exemplo, temos as duas funções que já vimos: `echo` e `print`. Mas o mais interessante é que você pode criar suas próprias funções. A sintaxe para criar uma função no PHP é a seguinte:

```
<?php  
function somar($x,$y){  
    return ($x + $y);  
}  
?>
```

A função acima possui dois parâmetros e quando invocada irá retornar a soma dos dois valores passados como parâmetro. Veja:

```
<?php
```

```
function somar($x,$y){
    return ($x + $y);
}

echo somar(2,2);
echo "<br>";
echo somar(1250,12511);
?>
```

O retorno da função acima não precisa, obrigatoriamente, ser impresso com a função echo. Você pode utilizar o resultado para continuar seu calculo ou fazer outras coisas. Veja:

```
<?php
function somar($x,$y){
    return ($x + $y);
}

$resultado = somar(2,2);
$resultado2 = somar($resultado,10);
echo $resultado2;
?>
```

Você também pode criar uma função para imprimir valores invés de retorná-los. Veja:

```
<?php
function echo_soma($x,$y){
    echo ($x + $y);
}

echo_somar(2,2);
```

?>

Concatenação

Para concatenar textos no PHP você deve utilizar um ponto-final entre os valores a serem concatenados. Vejam os exemplos:

```
<?php
$texto= "Meu nome é";
$nome = "Anderson";
```

```
echo $texto . $nome;
?>
```

Também é possível concatenar valores e armazenar o resultado em variáveis fazendo uso do ponto-final seguido do sinal de igualdade. Vejam:

```
<?php
$texto= "Meu nome é";
$texto .= "Anderson";
$texto .= "<br>Qual é o Seu?";
echo $texto;
?>
```

Outra forma de concatenação é inserir as variáveis dentro dos textos. Vejam:

```
<?php
$texto= "Meu nome é";
```

```
$texto= “$texto Anderson”;  
echo $texto;  
?>
```

Operadores Aritméticos

O PHP suporta alguns operadores aritméticos, vejamos:

- Subtração.

```
<?php  
$laranjas= 15;  
$vendas = 13;  
$resto = $laranjas – $vendas;  
  
echo “Existem $resto laranjas no estoque”;  
?>
```

+ Soma.

```
<?php  
$laranjas= 15;  
$compras = 7;  
$total = $laranjas + $compras;  
  
echo “Existem $total laranjas no estoque”;  
?>
```

*** Multiplicação.**


```

<?php
$meses= 12;
$dias = 30;
$dias_ano = $meses * $dias;

echo "Um ano possui aproximadamente $dias_ano
dias";
?>

```

/ Divisão

```

<?php
$laranjas= 60;
$alunos = 30;
$laranjas_alunos = $laranjas / $alunos;

echo "Cada aluno vai receber $laranjas_alunos
laranja";
?>

```

^ Módulo (Resto de uma Divisão)

```

<?php
$laranjas= 15;
$alunos = 13;
$resto = $laranjas ^ $alunos;

echo "Após a divisão sobraram $resto laranjas";
?>

```

Operadores de Comparação

Verifica se dois valores são iguais ou idênticos. Antes de vermos os exemplos com os operadores de comparação, vamos entender o IF. O if é um comando do PHP que permite o programador executar partes de código específico baseado no retorno das condições. Ou seja, se a condição for verdadeira, o código entre as chaves será executado; caso contrário, o código entre as chaves será desconsiderado. Veremos mais sobre o if mais para frente.

== Igual. A comparação será verdadeira se os valores comparados forem iguais:

```
<?php
$laranjas= 15;
$alunos = 13;
$iguais = "não";

if($laranjas == $alunos){
    $iguais = "";
}

echo "O número de laranjas $iguais é igual ao
número de alunos";
?>
```

!= Diferente. A comparação será verdadeira se os valores comparados forem diferentes.

```

<?php
$laranjas= 15;
$alunos = 13;
$iguais = "";

if($laranjas != $alunos){
    $iguais = "não";
}

echo "O número de laranjas $iguais é igual ao
número de alunos";
?>

```

<> Diferente. Igual ao operador !=.

```

<?php
$laranjas= 15;
$alunos = 13;
$iguais = "";

if($laranjas <> $alunos){
    $iguais = "não";
}

echo "O número de laranjas $iguais é igual ao
número de alunos";
?>

```

=== Idêntico. Retorna verdadeiro caso o valor e o tipo seja igual.

```

<?php
$laranjas= 15;
$alunos = "15";
$iguais = "não";

if($laranjas === $alunos){
    $iguais = "";
}

echo "Laranjas e alunos $iguais são idênticos";
?>

```

A saída para o código acima será: Laranjas e alunos não são idênticos. Isso ocorre porque \$laranjas é um número e \$alunos – já que está entre aspas – é um texto, ou seja, são de tipos diferentes.

!== Não Idêntico. Retorna verdadeiro se o valor ou o tipo seja diferente.

```

<?php
$laranjas= 15;
$alunos = "15";
$iguais = "";

if($laranjas !== $alunos){
    $iguais = "não";
}

echo "Laranjas e alunos $iguais são idênticos";
?>

```

A saída para o código cima será: Laranjas e alunos não são idênticos.

> **Maior que.** Retorna verdadeiro se o primeiro valor for maior que o segundo valor da comparação.

```
<?php
$portugues = 9;
$matematica = 8.5;
$resultado = "menor";

if($portugues > $matematica){
    $resultado = "maior";
}

echo "Minha nota em Português é $resultado que
em Matemática";
?>
```

>= **Maior ou igual.** Retorna verdadeiro se o primeiro valor for maior ou igual ao segundo valor da comparação.

```
<?php
if($valor_01 >= $valor_02){
    echo $valor_01 . " é maior ou igual a " .
$valor_02;
}
?>
```

< Menor que. Retorna verdadeiro caso o primeiro valor seja menor que o segundo valor da comparação.

```
<?php
if($valor_01 < $valor_02){
    echo $valor_01 . “ é menor que “.
    $valor_02;
}
?>
```

<= Menor ou Igual. Retorna verdadeiro caso o primeiro valor seja menor ou igual ao segundo valor da comparação.

```
<?php
if($valor_01 <= $valor_02){
    echo $valor_01 . “ é ou igual a “.
    $valor_02;
}
?>
```

Estruturas de repetição

O PHP possui algumas estruturas de repetição que são muito úteis durante a codificação. Veremos a seguir duas dessas estruturas: For e While. A sintaxe do **FOR** é a seguinte.

```
for(inicialização, condição, incremento){
    Código a ser executado;
```

```
}
```

Veja um exemplo:

```
<?php
for($i=1;$i<=10;$i++){
    echo "i = " . $i . "<br>";
}
?>
```

No exemplo acima, inicializei uma variável com o valor 1, coloquei uma condição menor ou igual a 10 e defini o incremento da variável de um em um. Note que o nome da variável não é importante, eu poderia ter escolhido qualquer outro nome. Além disso, as outras duas partes da estrutura também podem ser editadas para satisfazer sua necessidade como programador web. Vamos supor que você queira imprimir os números pares de 0 a 100. Para isso, você poderia reescrever o código acima da seguinte forma.

```
<?php
for($i=0;$i<=100;$i=$i+2){
    echo "i = " . $i . "<br>";
}
?>
```

A sintaxe do **WHILE** é a seguinte:

```
while(condição){
    Código a ser executado;
```

```
}
```

Veja um exemplo:

```
<?php
$i = 0;

while($i<=10){
    echo "i = " . $i . "<br>";
    $i++;
}
?>
```

Comentários

Toda linguagem de programação que se preze possui comandos para comentários. Em programação, comentários são trechos de textos onde você coloca informações sobre o que determinada parte do código faz. Isso ajuda a compreensão do código por você ou por outro programador que venha a editar seu código PHP.

O PHP suporta dois tipos de comandos para comentários: `/**/` e `//`. `//` é utilizado para comentários de uma única linha, já `/**/` é utilizado para comentários de múltiplas linhas. Vejam o exemplo:

```
<?php
//Abaixo atribuo o valor 0 para a variável i
$i = 0;
```


/*

*No código abaixo eu poderia ter utilizado o for,
mas utilizei o while para ilustrar sua sintaxe.
Há casos onde o while é mais aconselhável e há
casos onde o for é o preferível.
Conhecer bem o funcionamento de cada um
ajudará na hora da escolha.*

*/

```
while($i<=10){  
    echo "i = " . $i . "<br>";  
    $i++;  
}  
?>
```

If / Elseif / Else

O comando IF além de permitir a execução de determinado trecho de código baseado em determinada condição, também possui seu complementar else. Veja a sintaxe:

```
if(condição){  
    Código a ser executado se a condição é  
    verdadeira.  
}else{  
    Código a ser executado se a condição é  
    falsa.  
}
```

Vejam um exemplo prático:

```
<?php
$idade = 20;

if($idade > 19){
    echo "Você é adulto";
}else{
    echo "Você é adolescente";
}
?>
```

Você pode ir incrementando condições para satisfazer sua necessidade utilizando o elseif, veja como fica:

```
<?php
$idade = 50;

if($idade <= 19){
    echo "Você é adolescente";
}elseif($idade <= 69){
    echo "Você é adulto";
}elseif($idade <= 100){
    echo "Você é idoso";
}else{
    echo "Nossa! Você tem mais de 100 anos.
Parabéns!";
}
?>
```

Switch / Case

O Switch / Case é utilizado em casos onde você possui várias condições para serem verificadas, o que com else/elseif exigiria muito código. Vejam como é simples a sintaxe:

```
switch(variável a ser conferida){  
    case "condição":  
        código a ser executado;  
    break;  
    case "condição":  
        código a ser executado;  
    break;  
    default:  
        código a ser executado;  
}
```

A primeira condição é a primeira a ser verificada, depois a segunda, a terceira e assim sucessivamente até o final das condições. Quando uma condição verificada for verdadeira o código logo abaixo dela é executado e o comando break finaliza o switch. Caso nenhuma condição seja verdadeira o bloco de código abaixo de default será executado. Veja um exemplo prático:

```
<?php
```

```
$nota = 10;
```

```
switch($nota){
```

```

        case $nota < 7:
            echo "Nota abaixo da média. Estude
mais!";
            break;
        case $nota < 10:
            echo "Nota boa. Continue
estudando!";
            break;
        default:
            echo "Parabéns! Você é um aluno
exemplar!";
    }
?>

```

O Switch / case também pode ser utilizado sem condições no case, apenas conferindo o valor do parâmetro de switch. Vejam como fica:

```

<?php
$hoje = "domingo";
switch($hoje){
    case "domingo":
        echo "Bom domingo!";
        break;
    case "segunda":
        echo "Boa segunda";
        break;
    case "terça":
        echo "Boa terça";
        break;
    case "quarta":
        echo "Boa quarta";

```

```
break;
case "quinta":
    echo "Boa quinta";
break;
case "sexta":
    echo "Boa sexta";
break;
case "sábado";
    echo "bom sábado";
break;
default:
    echo "Dia da semana não
encontrado!";
}
?>
```

CAPÍTULO 4

Avançando em Programação PHP

Além dos operadores já vistos, PHP suporta **Operador Ternário**, mas exige certa cautela na utilização do mesmo. Mais para frente falaremos sobre isso, por hora basta saber que o operador ternário utiliza-se dos operadores de comparação e também dos sinais: **?:**

\$resultado = condição ? “valor 01” : “valor 02”;

No modelo acima se define que:

- * **\$resultado** é uma variável onde o resultado da condição será armazenado

- * **condição** é qualquer condição que pode retornar verdadeiro ou falso, por exemplo:

“\$x==\$y”, “5<3”, “\$y!==false”, etc etc...

- * **valor 01** é qualquer valor que será retornado caso a **condição** seja verdadeira

- * **valor 02** é qualquer valor que será retornado caso a **condição** seja falsa

Bom, dito isto vamos ver um exemplo funcional:

```
<?php
//Exemplo simples:
$idade = 25;
$descriativo = “”;

if($idade < 13){
    $descriativo = “Criança”;
}else{
    $descriativo = “Jovem”;
}
```

echo "Quem possui \$idade anos é \$descritivo";

//Saída = Quem possui 25 anos é Jovem;

?>

E agora o mesmo script utilizando-se de operadores ternários encadeados:

<?php

\$idade = 25;

\$descritivo =

\$idade<13?"Criança":(\$idade<20?"Adolescente":
"Adulto");

echo "Quem possui \$idade anos é \$descritivo";

//Saída = Quem possui 25 anos é adulto;

?>

O uso de parênteses é recomendado para definir prioridades e também para facilitar a compreensão do código, pois o encadeamento de operadores pode não ser de fácil entendimento. Veja o exemplo abaixo:

<?php

\$idade = 12;

\$descritivo =

\$idade<13?"Criança":\$idade<20?"Adolescente":
Adulto";

echo "Quem possui \$idade anos é \$descritivo";

//Saída = Quem possui 12 anos é Adolescente;

?>

Talvez, a uma primeira vista, a string esperada em \$descricao fosse “Criança”, mas isso não ocorre porque os operadores ternários são resolvidos da esquerda para a direita. O mesmo exemplo, só que agora com os parênteses, vai lhe ajudar a compreender o que aconteceu.

```
<?php
$idade = 12;
$descricao =
(($idade<13?"Criança":$idade<20)?"Adolescente"
:"Adulto");
echo "Quem possui $idade anos é $descricao";
//Saída = Quem possui 12 anos é Adolescente;
```

?>

Para fixar bem, vamos alterar a posição dos parênteses para obter a prioridade que almejávamos, veja:

```
<?php
$idade = 12;
$descricao =
($idade<13?"Criança":($idade<20?"Adolescente"
:"Adulto"));
echo "Quem possui $idade anos é $descricao";
//Saída = Quem possui 12 anos é criança;
```

?>

Trabalhando com Arrays

Um array é uma forma de armazenamento de variáveis que podem ser acessadas por índices, sejam eles numéricos ou não. Existem trabalhos onde os **arrays** são essenciais para facilitar nosso trabalho, reduzir o código e tornar o script mais “limpo”. No PHP é muito simples trabalhar com arrays, a seguir vamos ver como declarar arrays e como acessar seus valores. Você pode declarar um array sem dimensionar ele, ou ainda, dimensionando ele, veja:

Sem dimensionar

```
$notas_musicais = array();
```

No exemplo acima, as dimensões do array não foram definidas, esse recurso é muito útil porque muitas vezes você não sabe qual a dimensão que um array vai precisar durante a execução do script.

Dimensionando o Array

```
$notas_musicais = array(7);
```

Nesse exemplo estamos dimensionando o array em 7 itens com índices de 0 a 6.

Também é possível atribuir um valor array a uma variável não array com a função `explode()`, veja

```
$notas_musicais =  
explode(",", "do,re,mi,fa,sol,la,si");
```

Essa função aceita três parâmetros, o primeiro é a string de separação para base da divisão dos índices, o segundo é a própria string a ser dividida, o terceiro parâmetro não foi utilizado, mas é um valor de quantidade de divisões, se definido como 2 por exemplo, teríamos em \$notas_musicais um array de comprimento dois, contendo no índice 0 a string “do”, e todo o resto da string no índice 1;

Para acessar os valores de um array você pode utilizar algumas estruturas de repetição como: for, foreach, ou ainda acessar o índice diretamente, veja os exemplos:

```
<?php  
$notas_musicais =  
explode(",", "do,re,mi,fa,sol,la,si");
```

```
echo "A primeira nota = " . $notas_musicais[0] .  
"<br>";  
echo "A sexta nota = " . $notas_musicais[5] .  
"<br>";
```

```
echo "<hr>";
```

```
for($i=0;$i<count($notas_musicais);$i++){  
    echo "Loop for, nota $i = " .  
    $notas_musicais[$i] . "<br>";  
}
```

```
echo "<hr>";
```

```
foreach($notas_musicais as $key>$value){  
    echo "Loop foreach, nota $key = ". $value .  
    "<br>";  
}  
  
?>
```

Saída:

```
A primeira nota = do  
A sexta nota = la  
Loop for, nota 0 = do  
Loop for, nota 1 = re  
Loop for, nota 2 = mi  
Loop for, nota 3 = fa  
Loop for, nota 4 = sol  
Loop for, nota 5 = la  
Loop for, nota 6 = si  
Loop foreach, nota 0 = do  
Loop foreach, nota 1 = re  
Loop foreach, nota 2 = mi  
Loop foreach, nota 3 = fa  
Loop foreach, nota 4 = sol  
Loop foreach, nota 5 = la  
Loop foreach, nota 6 = si
```

As vezes é necessário sabermos se um array contém um determinado valor que procuramos,

para isso o PHP nos oferece a função `array_search()`.

A função **`array_search()`** possui três parâmetros sendo que os dois primeiros são obrigatórios e o último é opcional. Bem, vamos entender o que é cada um: o primeiro parâmetro é a string ou valor que você deseja procurar no array, o segundo é o próprio array onde será feita a busca, já o terceiro parâmetro é um booleano, que representa se é pra considerar os tipos dos dados buscados. Caso o terceiro parâmetro seja preenchido como verdadeiro e você fizer uma busca por “2” - mesmo o array contendo o número 2 – a função retornará false.

O retorno da função `array_search()` é a posição do valor buscado no array, caso ele contenha esse valor, ou false caso o valor não seja encontrado.

Veja esse exemplo:

```
<?php
$vertebrados =
array("cachorro","gato","homem");

if(array_search("homem",$vertebrados)!=false){
    echo "homens são vertebrados";
}else{
    echo "homens não são vertebrados";
}
```

?>

Reparou que utilizei o operador `!==` ? Fiz isso porque o operador `!=` não confere o tipo e vai entender 0 como false, e 0 é uma possível posição de retorno da função `array_search()`. Caso queira conferir se o valor é false utilize o operador `===`, pelo mesmo motivo.

Agora, ainda existe um porém, essa função é case sensitive, ou seja, diferencia letras maiúsculas de minúsculas. Imagine por exemplo que você passe um valor para a busca acima, como esse:

“CachOrRo”, isso retornará falso.

Para resolver essa questão, eu criei minha própria função, veja:

```
<?php
function array_searchi($needle, $haystack,
$check_type=false){
    for($i=0;$i<count($haystack);$i++){
        if($check_type){

            if(strtolower($haystack[$i])==strtolower(
$needle)) return $i;
        }else{

            if(strtolower($haystack[$i])==strtolower($
needle)) return $i;
        }
    }
}
```

```

        return false;
    }

    if(array_searchi("CachOrRo",$vertebrados)!=false){
        echo "Cães são vertebrados";
    }else{
        echo "Cães não são vertebrados";
    }
?>

```

Você utiliza essa função, do mesmo modo que a função original, tendo também o terceiro parâmetro opcional e retornando a posição do valor procurado no array ou falso em caso de valor não encontrado. O `i` no final é de insensitive, tudo tem uma lógica, não é mesmo?

PHP suporta arrays multidimensionais, o que é muito útil em alguns casos. Trabalhar com array multidimensionais é muito simples, você pode simplesmente atribuir a um array outro array e assim sucessivamente até satisfazer a cadeia de arrays que você precise. No exemplo abaixo trabalhei com arrays bidimensionais e tridimensionais para demonstrar como acessar esses arrays multidimensionais no php. Veja que, para facilitar meu trabalho, defini quatro constantes e as utilizei pra acessar os respectivos índices de três arrays. Essa é uma boa idéia, principalmente se você pretende trabalhar com muitos arrays encadeados. Isso facilita o entendimento do script, principalmente em caso de suporte, caso o

responsável não foi o mesmo que criou o script.
Bom, vamos ao código:

```
<?php
$ pessoa_data = array();
$ pessoa_data[0] =
array("Masculino","Feminino");
$ pessoa_data[1] =
array("Solteiro","Casado","Divorciado","Viúvo");
$ pessoa_data[2] = array(2);
$ pessoa_data[2][0] = array("salário menor que
R$900","salário entre R$900 e R$2000","salário
acima de R$2000");
$ pessoa_data[2][1] = "Desempregado";
$ pessoa_data[3] = "";

define("SEXO", 0,false);
define("ESTADO_CIVIL",1,false);
define("ESTADO_OCUPACIONAL",2,false);
define("NAME",3,false);

$ pessoa = array();

$ pessoa[0][0] = $ pessoa_data[SEXO][0];
$ pessoa[0][1] =
$ pessoa_data[ESTADO_CIVIL][1];
$ pessoa[0][2] =
$ pessoa_data[ESTADO_OCUPACIONAL][0][1];
$ pessoa[0][3] = $ pessoa_data[NAME]
="Pedrinho";

$ pessoa[1][0] = $ pessoa_data[SEXO][1];
```



```

$ pessoa[1][1] =
$ pessoa_data[ESTADO_CIVIL][2];
$ pessoa[1][2] =
$ pessoa_data[ESTADO_OCUPACIONAL][1];
$ pessoa[1][3] = $ pessoa_data[NAME]
= "Mariazinha";

for($i=0;$i<count($pessoa);$i++){
    echo "<p>" . $pessoa[$i][NAME] . ",
    ".$pessoa[$i][SEXO] . " , " .
    $pessoa[$i][ESTADO_CIVIL] . " , " .
    $pessoa[$i][ESTADO_OCUPACIONAL] . "</p>";
}
/*A saída vai ser
Pedrinho, Masculino, Casado, salário entre R$900
e R$2000

Mariazinha, Feminino, Divorciado, Desempregado
*/
?>

```

Trabalhando com Datas

Trabalhar com Datas no PHP é uma tarefa inevitável, quem é programador PHP que o diga. Por exemplo, para gerar uma data de vencimento daqui a um mês, subentendendo que o suposto cliente tenha assinado um serviço hoje. Para isso você vai precisar gerar a nova data e salvar no banco de dados.

Para facilitar o trabalho com datas nos meus

projetos, eu criei essa função abaixo que utiliza o mktime para retonar as datas desejadas só que de forma mais fácil, veja:

```
<?php
function make_data($data,
$anoConta,$mesConta,$diaConta){

    $ano = substr($data,0,4);

    $mes = substr($data,5,2);

    $dia = substr($data,8,2);

    return date('Y-m-d',mktime (0, 0, 0,
$mes+($mesConta), $dia+($diaConta),
$ano+($anoConta)));

}

//Por exemplo pra saber qual a data daqui a 45
dias;

$today = date('Y-m-d');

$45days_later = make_data($today,0,0,45);

//Daqui um ano, dois meses e 23 dias:

$today = date('Y-m-d');
```

```
$new_date = make_data($today,1,2,23);  
?>
```

Fácil, não é? Só passar os valores para a função que ela retorna a data para nós. O primeiro valor é a data atual, o segundo é a quantidade de anos, depois meses e por último dias.

Para saber uma data no passado, passe um valor negativo, por exemplo -5 no campo de anos, para saber a data que foi há 5 anos atrás.

Validando Datas

A utilidade de se verificar datas são diversas, como exemplo básico, podemos citar uma entrada, onde você quer certificar-se de que o usuário informou uma data válida para o respectivo campo, então, cabe a você, validar a data no php.

Por incrível que pareça, PHP não possui algo do tipo `isDate()`, uma função que é encontrada em várias de linguagens de programação, mas, em contrapartida, PHP oferece uma função com o mesmo objetivo só que com uma sintaxe diferente. Trata-se da função `checkdate()`. A função `checkdate()` confere uma data e retorna `true` caso ela seja verdadeira ou `false` caso contrário, no entanto, não basta passar a data que se deseja conferir para a função, é necessário passar ela dividida em mês, dia e ano, exatamente nessa ordem. Veja o exemplo abaixo:

```

<?php
$dia = "29";
$mes = "02";
$ano = "2009";

if(checkdate($mes, $dia, $ano)){
    echo "Data válida";
}else{
    echo "Data inválida";
}
?>

```

Talvez você esteja pensando: nossa, que chato ter que ficar dividindo a data em dia, mes e ano para passar para a função. Pois olha, vou te consolar, você não está sozinho nesse pensamento, afinal, de fato, é tedioso ter que ficar dividindo uma data sempre que quiser conferir se ela é válida ou não. Por conta disso, veja a função que criei em PHP para resolver esse problema. Advinhe o nome que eu dei pra ela, sim, `isDate()`, nada mais intuitivo não é? Principalmente para quem vem de outras linguagens onde há essa função. Essa função que desenvolvi, permite a você validar uma data no formato utilizado aqui no Brasil, ou seja, dia/mes/ano, e além disso, você também pode validar a data com esse formato dia-mes-ano.

Veja abaixo o código da função `isDate`:

```

<?php
$date = "28/02/2009";

function isDate($date){
    $char = strpos($date, "/")!==false?"":"/-";
    $date_array = explode($char,$date);
    if(count($date_array)!=3) return false;
    return
    checkdate($date_array[1],$date_array[0],$date_a
rray[2])?($date_array[2] . "-" . $date_array[1] . "-"
. $date_array[0]):false;

}

if($date_ckecked=isDate($date)){
    echo $date_ckecked;
}else{

    echo "Data inválida";

}
?>

```

Quem foi atento, percebeu que essa função possui outra utilidade além de validar uma data, essa outra utilidade que ela oferece é retornar uma data no formato suportado pelo banco de dados Mysql, que é ano-mes-dia. Então, basta você pegar o retorno da função e salvar direto no banco de dados Mysql, caso seja essa sua necessidade. Mas você também pode só utilizar ela pra verificar se a data é inválida, conforme foi feito no exemplo acima.

Para verificar a funcionalidade dessa função, experimente alterar o formato da data para `$date = "28-02-2009"`, notará que o script funcionará igualmente para esse formato. Tente também passar uma data inválida, como por exemplo, `$date = "28-02-2009"`, vai notar que o script retorna false. Bom, acho que isso está explícito na função, mas quem não entendeu ainda, a função `isDate` que criei aí acima, irá retornar a data convertida para o formato aceito pelo Mysql caso a data seja válida, e caso contrário ela irá retornar false.

Enviando Emails

Quase todos os projetos que faço em PHP requerem envio de email, ou para o administrador ou para os usuários. O PHP possui uma função para envio de emails, que, inclusive, possui um nome bem sugestivo: `"mail()"`. Para enviar email simplesmente chame essa função passando os parâmetros desejados.

Para facilitar minha vida, eu criei essa função abaixo para enviar os meus emails em PHP, visto que algumas configurações como charset e outras, eu sempre utilizo as mesmas, não preciso ficar passando esses parâmetros direto pra função mail, a minha função `send_mail` se encarrega disso pra mim, veja:

<?php

```

function send_mail($from, $to,$subject,$message){
$headers = 'MIME-Version: 1.0' . "\r\n";
$headers .= 'Content-type: text/html; charset=utf-8' . "\r\n";

// Additional headers
$headers .= 'From:'. $from . "\r\n";
if(mail($to, $subject, $message, $headers)) return
true;
return false;
}
?>

```

Nessa função, eu só preciso passar os parâmetros que sempre mudam como: \$from, \$to,\$subject,\$message. Onde:
 \$from – Email do Remetente
 \$to – Email do destinatário
 \$subject – assunto
 \$message – a mensagem em si.

Fácil enviar email no PHP, não é? Ah, atenção ao enviar vários emails simultâneos, porque hospedagem compartilhada limita o número máximo de envio de emails por hora, isso é feito pra evitar sobrecarregar o servidor web já que o mesmo é compartilhado com outros usuários

Verificando se uma Variável está vazia

No PHP é comum precisar verificar se uma variável está vazia, mas atenção, vamos desde já deixar bem claro que existe uma diferença gritante entre uma variável estar vazia e não estar definida. Se você tentar utilizar uma variável que não está definida isso vai gerar um erro no PHP, já se acessar uma variável vazia, não ocorrerá erro nenhum.

Veja bem, uma variável é declarada de duas formas, uma delas, pode-se dizer, é implicitamente, apenas atribuindo um valor para a variável, a outra é declarando ela explicitamente, por exemplo: var \$cor.

Vejamos abaixo como verificar se uma variável está vazia:

```
<?php
//pode-se utilizar a função empty()
if(empty($var_test)){
    //fazer algo
}else{
    //fazer algo
}

//outra forma é do seguinte jeito
if(str_replace(" ","",$var_test)==""){
    //fazer algo
}
```



```

}else{
    //fazer algo
}
?>

```

Repare que no segundo exemplo utilizei a função `str_replace` pra retirar todos os espaços em branco, ou spaces, da variável, antes de verificar se o valor dela é igual a nada, isso porque se você verificar uma variável com espaços em branco, mesmo que não tenha nada além de espaços, a função retornará diferente de vazio.

Validando CPF

A validação de cpf é muito útil para programadores PHP, não raro existem trabalhos em PHP onde é necessário verificar se o usuário informou um CPF válido, e caso contrário, não permitir o cadastro do mesmo.

```

<?php
function is_valid_cpf($cpf){
    for( $i = 0; $i < 10; $i++ ){
        if ( $cpf == str_repeat( $i , 11) or
!preg_match("@^[0-9]{11}$@", $cpf ) or $cpf ==
"12345678909" )return false;
        if ( $i < 9 ) $soma[] = $cpf{$i} * (
10 - $i );
        $soma2[] = $cpf{$i} * ( 11 -
$i );
    }
}

```

```

    }
    if(((array_sum($soma)% 11) < 2 ? 0 : 11 - (
array_sum($soma) % 11 )) != $cpf{9})return
false;
    return ((( array_sum($soma2)% 11 ) < 2 ?
0 : 11 - ( array_sum($soma2) % 11 )) != $cpf{10})
? false : true;
}
?>

```

Pra utilizar ela é muito fácil, ela retornará true se o cpf for válido e false se não for válido. Você pode utilizar ela num IF, veja esse exemplo:

```

<?php
if(is_valid_cpf($cpf_do_user)){

    //faz algo

}else{

    //faz algo

}
?>

```

Incluindo arquivos externos

Quando se está programando em PHP, é muito comum utilizar-se de includes e requires em nossos códigos parar reutilizar códigos, comandos,

páginas, etc. Eu aconselho que utilizem a função `require_once()` para incluir arquivos em seu código PHP. `Require_once()` verifica se o que está sendo incluído já não foi incluído antes, e caso positivo, ele não inclui novamente o mesmo script, evitando assim erros de lógica muito difíceis de se entender, por exemplo: quando não se sabe o motivo de uma variável ter um valor não esperado, etc. A sintaxe é simples, veja:

```
<?php
//para incluir um arquivo que esteja no mesmo
diretório
require_once("header.php");

//para incluir um arquivo em um subdiretório
require_once("subdir/header.php");

//para incluir um arquivo um diretório acima do
atual
require_once("../header.php");
?>
```

Você pode utilizar os dois pontos para acessar diretórios superiores, e não há limite de níveis.

Trabalhando com casas decimais

Alguns cálculos que fazemos no PHP podem nos retornar um número com uma casa decimal ou até nenhuma e isso pode fazer um

script trabalhar de forma não desejada. Por exemplo, para passar valores para o PagSeguro você deve retirar os pontos e vírguas, outras ocasiões você deve substituir o ponto por vírgula. Eu sempre utilizava esse código:

```
<?php
$valor = 10.25;
echo str_replace(",",".", $valor);
?>
```

A saída desse script será 10,25 , até ai tudo bem, mas vamos supor que esse valor sofra alguns calculos durante o script, por exemplo, seja multiplicado por 2, veja o script:

```
<?php
$valor = 10.25;
$valor = 10.25 * 2;
echo str_replace(",",".", $valor);

?>
```

A saída agora será, 20,5. A última casa, como resultou em 0, foi omitida, para o calculo, isso não terá problema, mas se você passar o valor 205 para alguns scripts, eles entenderão isso como 2,05 e não 20,50.

Bom, dito isso, vamos a solução para essa mazela. A solução que estou utilizando é usar a função `number_format()`, você deve passar quatro

parâmetros nessa função:

- 1- o primeiro é o próprio número a ser formatado
- 2- o segundo é a quantidade de casas decimais desejadas
- 3- o terceiro é a string que será utilizada pra separar as casas decimais
- 4- o quarto é a string que será utilizada pra separar as casas dos milhares.

Veja como ficaria nosso exemplo acima com o `number_format()`;

```
<?php
$valor = 10.25;
$valor = 10.25 * 2;
echo number_format($valor,2,"","");
?>
```

Agora a saída será 20,50, como o esperado.

Dominando a Função Preg_Match

A seguir veremos as principais formas de se utilizar a função `preg_match` do PHP. A função `preg_match` é muito útil quando se quer encontrar algo em alguma string (texto), como exemplos, podemos citar: encontrar emails, encontrar urls, encontrar imagens, encontrar classes específicas. Essa função trabalha com expressões regulares, caso você não entenda nada de expressões

regulares, saiba que já existem milhares de expressões regulares criadas por programadores e que estão disponíveis aí na web. Lembre-se que o Google é nosso amigo.

Existem algumas formas diferentes de se utilizar a função `preg_match` do PHP, mas note que algumas formas, embora suportadas, não é recomendado o uso delas. Vejamos algumas utilidades:

1-) É possível utilizar a função `preg_match` para saber se uma determinada sequência de caracteres existe em um texto. Vamos supor que você vai pegar algum texto de algum banco de dados, de alguma entrada do usuário, ou de qualquer outra fonte e queira saber se existe algum email nesse texto. Veja o código:

```
<?php
$subject = "Meu email é Obama@us-a.gov";

$pattern = '([-a-zA-Z]{1,30})@([-a-zA-Z]{1,30})([.]{1})([-a-zA-Z]{1,10})/';

$result = preg_match($pattern, $subject);

echo $result;
?>
```

No código acima, entenda o seguinte: `$subject` é o texto onde você fará a pesquisa, `$pattern` é uma expressão regular (ER) que criei e que casa um

email tranquilamente.

Dito isso, repare que eu atribuo o retorno da função `preg_match` para a variável `$result`, os valores possíveis para retorno são:

0 – caso a ER não case com nenhuma parte do texto

1 – caso a ER case com alguma parte do texto

Pra quem ainda não entendeu a utilidade, posso dizer que nesse exemplo, você poderia checar se um usuário digitou um email válido em seu cadastro.

Caso você não precise utilizar ER e queira apenas verificar se uma string está dentro de outra, você deveria usar outras funções, como o `strpos` ou `strstr`, visto que são mais rápidas.

2-) A segunda utilidade para a função `preg_match`, e é a que eu mais utilizo no dia a dia, é para pegar as strings que fecham com a ER. Tomando ainda o exemplo que foi dado acima, além de saber se existe algum email dentro do texto, nós iremos pegar esse email, veja como:

```
<?php
```

```
$subject = "Meu email é Obama@us-a.gov";
```

```
$pattern = '([-a-zA-Z]{1,30})@([-a-zA-Z]{1,30})([.]{1})([-a-zA-Z]{1,10})/';
```

```
$result = preg_match($pattern,  
$subject,$matches);
```

```
echo $matches[0];  
?>
```

A novidade aí é que foi passado um terceiro parâmetro para a função, quando você passa esse parâmetro (que inclusive deve ser uma variável), então a função irá atribuir a essa variável um array. Nesse array o primeiro índice conterá o email desejado, o segundo ítem será o primeiro conjunto de expressões que sua ER casou, e assim por diante. Se você ficou meio perdido com o que acabei de dizer, entenda pelo menos que o email que você deseja estará na variável `$matches` no índice 0, por isso que foi utilizado `echo $matches[0]`.

Existe ainda um quarto parâmetro para essa função que é o `PREG_OFFSET_CAPTURE`, quando você passa esse parâmetro a função `preg_match` irá retornar em `$matches` um array com duas dimensões, e com dois índices (0 e 1). No índice 0 (`$matches[0][0]`) você terá a string que casou, e no índice 1 (`$matches[0][1]`) você terá um número indicando qual a posição no texto que essa string que casou, foi encontrada. Veja como fica no código:

```
<?php  
$result = preg_match($pattern,
```



```
$subject,$matches,PREG_OFFSET_CAPTURE);  
?>
```

Por fim, temos ainda um quinto parâmetro, que é o off set, isso indica à função `preg_match` em que posição a busca deve iniciar. Esse parâmetro pode poupar bastante tempo caso você esteja trabalhando com um longo texto e saiba que a string procurada vai estar perto do final. Vamos supor que o texto tenha 5000 caracteres e que você saiba que a string desejada vai estar perto do caractere 4000, então você utilizaria assim:

```
<?php  
$result = preg_match($pattern,  
$subject,$matches,PREG_OFFSET_CAPTURE,39  
00);  
?>
```

Com isso sua busca irá iniciar a partir do caractere 3900, poupando tempo na execução de seu script. Caso você queira utilizar o off set mas não o `PREG_OFFSET_CAPTURE`, você pode passar o número 0 (false também funciona) para esse parâmetro, veja como:

```
<?php  
$result = preg_match($pattern,  
$subject,$matches,0,3900);  
?>
```

Entendendo bem de Expressões Regulares é

possível encontrar qualquer coisa em um texto com essa função `preg_match`. Não é a toa que ela é muito utilizada pelos programadores PHP.

Dominando Expressões Regulares

Expressões Regulares são muito utilizadas no PHP, em alguns casos para validar campos, em outros para procurar sequências específicas. Uma expressão regular poderia ser utilizada para criar uma espécie de coletor de emails, poderia ser utilizada para validar campos que um usuário venha a preencher, poderia ser utilizada para verificar se um determinado nome ou sequência de caracteres existe em uma determinada string, etc.

A utilidade de Expressões Regulares é tremenda, mas para utilizar ERs de forma satisfatória, você precisa entender bem como é a sintaxe de uma ER. O PHP suporta dois tipos de ER – POSIX-extended e PCRE – a PCRE (Pear Compatible Regular Expression) é a mais rápida e mais “poderosa”, portanto iremos focar nela.

O PHP possui várias funções que suportam Expressões Regulares, podemos citar aqui:

- `preg_match`
- `preg_quote`
- `preg_replace_callback`
- `preg_replace`

preg_match_all
preg_grep
preg_filter
preg_last_error
preg_split

Para ilustrar nossos exemplos, iremos utilizar a função preg_match que já foi dissecada aqui.

Como montar uma expressão regular?

A estrutura de uma ER pode ser assim:

```
'/regexarexp/'
```

Note que regexarexp é onde você deve inserir a expressão regular em si. Um detalhe importante é que caso você utilize barras dentro da ER, você vai precisar colocar uma barra invertida já que trata-se de meta caractere. Vejamos um exemplo pra melhor entendimento:

```
<?php
$pattern = '/http:\\\\livrophp\\.com/';
$result = "";
preg_match($pattern, $subject, $result);
echo "<pre>";
print_r($result);
echo "</pre>";
?>
```

No exemplo acima eu tenho uma string com o domínio http://livrophp.com, e uma string com a

ER, note que utilizei uma barra invertida sempre que uma barra ou ponto apareceu na ER, isso foi feito porque esses são meta caracteres especiais e caso eu não tivesse colocado a barra invertida, eles seriam interpretados por suas funções dentro da ER e não como strings a serem casadas.

Antes de prosseguirmos, é interessante dizer também que você pode utilizar o sinal % invés das barras // para montar a ER, ficaria algo desse tipo:

```
$pattern = '%http://livrophp\.com%';
```

A vantagem nisso é que daí você não vai precisar colocar uma barra invertida em toda barra que apareça na ER, em contrapartida você terá que colocar uma barra invertida antes de cada sinal de percentagem que aparecer na ER. Então é uma questão matemática, caso a barra apareça mais vezes que o sinal de percentagem na ER, você poderia utilizar '%regex%', já que pouparia tempo, caso contrário, utilize a outra forma '/regex/'.

Cadeia de Caracteres:

[]- Os sinais [] indicam uma cadeia de caracteres, por exemplo:

'/[1-8]/' - Compreende números de 1 à 8.

'/[a-zA-Z]/' - Compreende letras de a à z, tanto maiúsculas quanto minúsculas.

Metacaracteres de repetição:

Esses são caracteres que definem a quantia de vezes que um determinado caracter ou cadeia de caracteres devem aparecer, vejamos:

+ - casa no mínimo um caractere ou mais.

? - casa apenas um caractere.

* - casa nenhum ou qualquer número de caracteres.

{2} - limita a quantia exata de caracteres, nesse exemplo 2.

{2,5} - no mínimo dois caracteres e no máximo 5.

{5,} - limita apenas o mínimo em 5 caracteres.

{,200} - limita apenas o máximo em 200 caracteres.

Exemplos:

'/[1-8]+'/ - Casa números de 1 à 8, no mínimo um caractere deve ser encontrado.

'/.*/' - O ponto casa qualquer caractere e o asterisco casa nenhuma ocorrência ou ilimitadas ocorrências.

'/va[a-z]?/' - Casa exatamente nenhum caractere ou um caractere que deve estar entre a e z depois de va. ex: vac, val, va

'/ma[a-zA-Z]{3,5}/' - Irá casar macaco, manha, etc, mas não irá casar macaquice, macumbeiro, etc.

Isso se dá porque os caracteres subsequentes a ma... devem ser no mínimo 3 e máximo 5.

Outros Metacaracteres:

. - casa qualquer caracteres menos nova linha.

() - indica um grupo de expressões.

^ – casa caracteres no início da linha, já se usado dentro de [] e no início, nega uma cadeia de caracteres.

\$ – casa caracteres no final da string.

Exemplo:

'/{10}sa/' - casa qualquer cadeia de 10 caracteres seguida por sa.

'/([a-z]{1})([A-Z]{1})/' - casa qualquer dupla de caracteres de a à z desde que o primeiro seja minúsculo e o segundo maiúsculo.

'/^Contato:.*/' - casará uma linha que inicie com a palavra Contato seguida de dois pontos, mas se existir algum caractere antes da palavra contato, incluindo espaços em branco, o ER não irá casar.

'/^a-z]+/' - Carasá quando no mínimo um ou mais caracteres que não estejam entre a e z, forem encontrados.

'/^a-z]+\$/' - Mesmo que a ER acima, só que também exige que o caracter ou caracteres estejam no final da string.

Algumas outras dicas sobre ER:

1- Como casar nova linha (quebra de linha) em expressões regulares?

- Você pode utilizar essas duas idéias:

1) '\n/' - casa uma quebra de linha

2) '.*s/' - Colocar o s depois da última barra fará com que o ponto (.) case qualquer caractere, inclusive new line.

2- Como tornar uma Expressão Regular Case Insensitive (Não diferencia maiúsculas de minúsculas)?

- Veja como:

`/regexp/i` - Colocando o `i` após a última barra irá tornar a ER case insensitive.

3- Como casar início e final de linha em uma expressão regular?

- Veja como:

`/^regexp$/m` - Colocar o `m` após a última barra faz com que a ER considere as âncoras `^` e `$` para início e final de linhas

Além disso, temos também o modificador `/x`, quando utilizado, os espaços entre os caracteres não especiais não serão considerados. Você também pode unir modificadores para satisfazer seu anseio, desta forma:

`/regexp/simx`

Dominando a Função STR_Replace

Uma função que uso com certa frequência em minhas investidas programáticas, é a função `str_replace`. A função `str_replace` nos permite substituir caracteres ou cadeias de caracteres de forma muito simples e intuitiva. A função

`str_ireplace` funciona praticamente igual a `str_replace`, o `i` no início dessa função indica que ela é insensitiva, ou seja, não irá diferenciar maiúsculas de minúsculas, por isso, os exemplos que vou mostrar nesse artigo, podem também ser utilizados com o `str_ireplace`, caso necessitem.

A função `Str_Replace` possui três parâmetros, o primeiro é a string procurada (string para ser substituída), o segundo parâmetro é a nova string que você deseja inserir no lugar da antiga, e o último parâmetro é a variável onde a função irá procurar e fazer a substituição da string.

Vejamos alguns exemplos:

1- Aqui vamos substituir os caracteres – de uma determinada data pelos caracteres /, veja na prática como fica:

```
<?php
$hoje = "23-03-2010";
$hoje = str_replace("-", "/", $hoje);
echo $hoje;
?>
```

2- Você também pode utilizar arrays, substituindo várias strings sequencialmente, veja como ficaria:

```
<?php
$hoje = "Futebol ao vivo pela Internet";
```



```
$hoje = str_replace(array("Futebol","Internet")
, array("Copa do Mundo","Web"),$hoje);
echo $hoje;
?>
```

3- Além do terceiro parâmetro, a função `str_replace` também aceita um quarto parâmetro que é opcional. Esse quarto parâmetro, se for informado, irá guardar o número de substituições que a função fez na string. Veja o exemplo abaixo onde usamos esse recurso para contar quantos “a” tem na variável em questão:

```
<?php
$hoje = "Olimpíadas de Vancouver estão passando
agora na Record News";
$hoje = str_replace("a","")(($",$hoje, $count);
$hoje = str_replace("")(("$", "a", $hoje);
echo "A string possui $count a(s)";
?>
```

O que foi feito foi utilizar o quarto parâmetro para guardar a quantidade de “a” que existem na string, e depois devolvê-los a string original. A saída para o script acima será:

```
A string possui 8 a(s)
```

Viram como essa função pode ser utilizada de várias formas criativas para se obter soluções diversas no PHP? O que conta mais é a criatividade do programador, mas cuidado com as famosas

Gambi. Saber todas as possibilidades e recursos disponíveis em uma função, vai evitar você quebrar a cabeça tentando reinventar a roda durante sua jornada de desenvolvimento em PHP.

Dominando a Função PREG_Replace

A função `preg_replace` do PHP é uma função de substituição como a `str_replace`, mas com algumas diferenças. Ela suporta expressões regulares e outros recursos mais poderosos. `Preg_replace` pode ser utilizada para fazer substituições ou mesmo para adicionar caracteres de posições específicas em um determinado texto.

A sintaxe da função `preg_replace` é a seguinte:

```
<?php
preg_replace ( mixed $pattern , mixed
$replacement , mixed $subject [, int $limit = -1
[, int &$count ]] )
?>
```

O primeiro parâmetro é a expressão regular ou array de expressões regulares, o segundo parâmetro é o conteúdo que será usado para substituição - pode ser um array também -, o terceiro parâmetro é o texto ou string a ser editado - também pode ser um array -, o quarto parâmetro é um inteiro que indica o número máximo de substituições, esse parâmetro é opcional, o padrão é -1, ou seja, sem

limites. Há ainda um quinto parâmetro que deve ser uma variável que irá guardar a quantidade de substituições ou iterações que ocorreram durante a execução da função; esse último parâmetro também é opcional.

Achou complicado? Fique tranquilo que na prática é muito simples, veja os exemplos abaixo e entenderá como as coisas funcionam, mas antes, note que iremos fazer alguns tratamentos com emails; na prática muitos sistemas modificam os emails para evitar spammers, portanto veremos algumas possíveis idéias utilizando `preg_replace`.

Exemplo 01:

Nesse exemplo vamos substituir a segunda parte do email após o arroba `@` por três pontos (...), veja como ficaria:

```
<?php
$text = "Entre em contato comigo no email
admiyn@livrophp.com ou no email
contato@gmail.com, eu irei responder vc quando
eu puder mas ja agradeço pelo contato";
```

```
$text = preg_replace('/@([-\.\0-9a-zA-Z]+)/','@...',$text);
```

```
echo $text;
?>
```

A saída do script acima será “Entre em contato comigo no email admiyn@... ou no email contato@..., eu irei responder vc quando eu puder mas ja agradeço pelo contato”. Ou seja, substituímos o domínio por três pontos (...).

Exemplo 02:

Agora nossa missão é substituir o arroba @ por isso “(at)”, inclusive vocês já devem ter visto essa substituição em alguns sites, não é mesmo? Vejam como ficaria:

```
<?php
$text = preg_replace('/@/', '( at )', $text);

echo $text;
?>
```

Exemplo 03:

Agora vamos extrapolar um pouco, vamos supor que você quer duplicar os emails, para isso você vai precisar utilizar duas barras invertidas seguido do número do conjunto de expressões regulares. Veja o exemplo para entender:

```
<?php
$text = preg_replace('/([-\.\0-9a-zA-Z]+)@([-\.\0-9a-zA-Z]+)\/', '\\1@\\2,\\1@\\2', $text);

echo $text;
?>
```

Víram como é simples? Um conjunto de expressões é definido pelos parênteses, logo nós temos dois conjuntos, para você ter acesso aos valores que casaram em cada conjunto, basta informar o número do conjunto após duas barras invertidas. Mas você também pode utilizar invés de barras o símbolo $\{1\}$, essa notação é mais interessante porque resolve alguns problemas. Por exemplo, imagine que você queira pegar o primeiro conjunto que casou na expressão regular e colocar o número 1 logo após, se você fizer isso `\\11`, a função vai entender que você quer pegar o valor do conjunto 11, e não o valor do conjunto 1 e colocar o número um em seguida. Entenderam? Já com a outra notação nós podemos fazer isso de forma fácil. No exemplo abaixo vamos utilizar a notação $\{1\}$.

Exemplo 04:

Nesse exemplo vamos informar o limite para 1, ou seja, o segundo email do texto não será afetado.

Vejam:

```
<?php
$text = preg_replace('/([-\.\0-9a-zA-Z]+)@([-\.\0-9a-zA-Z]+)/', '{1}1@{2}', $text, 1);

echo $text;
?>
```

A saída do script acima será: “Entre em contato comigo no email `admiyn1@livrophp.com` ou no email `admiyn@gmail.com`, eu irei responder vc

quando eu puder mas ja agradeço pelo contato”.
Notaram que o segundo email ficou intocado, e foi inserido o número 1 antes do arroba no primeiro email?

Exemplo 05:

Por último, vamos fazer um exemplo que utilize o último parâmetro da função `preg_replace`, lembrando que a função dele é guardar em uma variável o número de casamentos que ocorreram. Vejam:

```
<?php
$text = preg_replace('/([-\.\0-9a-zA-Z]+)@([-\.\0-9a-zA-Z]+)\/','${1}1@${2}', $text, -1, $total);

echo "<br>". $total;
?>
```

A saída do script acima será 2, ou seja, houveram dois casamentos no texto informado. Um detalhe é que no parâmetro limite eu coloquei o valor -1, esse valor informa que não é para considerar nenhum limite, visto que a intenção é somar todos os casamentos.

Ah, não vimos um exemplo com arrays, então vamos, e dessa vez de verdade, finalizar com um exemplo usando arrays.

```
<?php
$pattern = array();
```

```
$pattern[] = '([-\.0-9a-zA-Z]+)@livrophp.com/';
```

```
$pattern[] = '([-\.0-9a-zA-Z]+)@([^\.livrophp.com])[-\.0-9a-zA-Z]+)';
```

```
$replace = array();
```

```
$replace[] = '{$1}@...da-casa...';
```

```
$replace[] = '{$1}@...outro...';
```

```
$text = preg_replace($pattern,$replace,$text);
```

```
echo $text;
```

```
?>
```

A saída do script acima será “Entre em contato comigo no email `admiyn@...da-casa...` ou no email `admiyn@...outro...`, eu irei responder vc quando eu puder mas ja agradeço pelo contato”. Ou seja, se o domínio do email for “`livrophp.com`” o script substitui por “`...da-casa...`”, caso contrário irá substituir o domínio por “`...outro...`”

Redirecionando no PHP

Redirecionar em PHP é uma função indispensável para programadores web, de vez em quando eu preciso utilizar essa função para redirecionar um usuário para uma determinada

página, baseado no que ocorreu no script ou na entrada do usuário. Então vamos lá!

A forma mais comum para redirecionar o usuário é utilizar o header, veja o exemplo abaixo:

```
<?php  
header("Location: http://profissionais.ws");  
?>
```

Você também pode imprimir um javascript que redirecione o usuário, algo desse tipo:

```
<?php  
echo  
"<script>document.location='http://profissionais.  
ws'</script>"  
?>
```

Nesse caso, PHP não irá redirecionar o usuário, mas irá imprimir um script em javascript que, ao ser interpretado pelo navegador, redirecionará para a página ou site desejado.

Alguns casos, onde o redirecionamento é mais utilizado, são em áreas restritas, para proibir que determinado usuário tenha acesso ao conteúdo de uma determinada página. Veja o exemplo abaixo onde o script verifica se o usuário 120 é o usuário corrente, caso não seja, redireciona ele para outra página.


```

<?php
$user =
isset($_SESSION["user"])?(int)$_SESSION["user"]
:0;

if($user!=120){

    header("Location: area-publica.php");

    exit();

}
?>

```

Acima, utilizei o `isset` para verificar se a variável existia, isso evita um erro no PHP. Se você tentar ler uma variável inexistente, PHP vai gerar um erro.

Note que há outras formas de se redirecionar uma página, mas a forma mais fácil e usual são essas que expus aqui.

Tirando Espaços em Branco de Variáveis

Quando se pega dados vindos de formulários, onde há interação com usuários, são comuns alguns campos acabarem vindo com espaços em branco ou ainda com caracteres indesejados. Por isso, veremos aqui como utilizar

as funções: Trim, Ltrim e Rtrim, para excluir espaços em branco ou outros caracteres.

Note que existem várias funções para se trabalhar com strings no PHP, mas o foco aqui será as funções citadas acima. Vejamos alguns exemplos de situações onde tais funções são as mais indicadas:

1- Caso você queira eliminar todos os espaços em branco no início e no final de uma determinada variável, utilize a função Trim da seguinte forma:

```
<?php
$nome = " Vanessa Gata ";
$nome = trim($nome);
echo $nome;
?>
```

2- Caso você queira eliminar somente os espaços no início da variável, você deve utilizar o LTrim, veja esse exemplo:

```
<?php
$nome = " Vanessa Gata ";
$nome = ltrim($nome);
echo $nome;
?>
```

3- Caso você queira eliminar somente os espaços no final da variável, você deve usar o Rtrim, veja o exemplo abaixo:

```
<?php
$nome = " Vanessa Gata ";
$nome = rtrim($nome);
echo $nome;
?>
```

4- Muito simples, não? Mas vamos supor que o nome, invés de " Vanessa Gata " seja "Vanessa Gata xxx". E agora, como retirar esses xxx que estão no final da string? Muito fácil, só passar o segundo parâmetro para a função rtrim, informando que você quer eliminar o caractere x, veja como fica.

```
<?php
$nome = "Vanessa Gata xxx";
$nome = rtrim($nome,"x");
echo $nome;
?>
```

Isso eliminará todos os três "x", porque essa função é recursiva, ou seja, ao tirar o ultimo x, o segundo passa a estar no final da string, e é retirado, já o primeiro x passa agora a estar no final da string e também é eliminado.

5- Além disso, você também pode informar vários caracteres para serem substituídos. No exemplo abaixo vamos eliminar os caracteres x e espaço em branco que possam estar no final ou no início da string. Veja:

```
<?php  
$nome = " Vanessa Gata xxx ";  
$nome = trim($nome,"x ");  
echo $nome;  
?>
```

Como pode ser visto, com as funções, trim, ltrim e rtrim, nós temos bons meios de se eliminar partes indesejadas de variáveis, basta chamar a função, passando os parâmetros de forma correta.

Renomeando Arquivos e Pastas

No PHP você pode renomear qualquer tipo de arquivo de forma muito fácil através da função rename. Você pode alterar apenas o nome de um arquivo ou o nome e a extensão. A função Rename do PHP também lhe permite renomear diretórios (pastas).

A sintaxe básica da função rename é rename(nome antigo, nome novo). Atente para duas coisas: a primeira é que você deve informar o nome junto com a extensão do arquivo, a segunda é que caso o arquivo que você pretende alterar o nome não esteja no mesmo diretório do script, você deverá informar o caminho completo para o arquivo, juntamente com seu nome e extensão. Vejamos alguns exemplos como amostra:

Exemplo 01:

Caso você queira renomear apenas um arquivo, o uso prático será da seguinte forma:

```
<?php
$old_name = "nome.txt";
$new_name = "novo-nome.txt";
rename($old_name,$new_name);
?>
```

Agora, vamos supor que o mesmo arquivo estivesse dentro de um subdiretório chamado tmp, então você deveria informar o caminho completo, da seguinte forma:

```
<?php
$old_name = "tmp/nome.txt";
$new_name = "tmp/novo-nome.txt";
rename($old_name,$new_name);
?>
```

Exemplo 02:

Agora, vamos supor que você tivesse 500 arquivos com o nome arquivo1.txt até arquivo500.txt, e que você queira mudar a extensão deles para .html, então você poderia utilizar um script semelhante a este:

```
<?php
for($i=1;$i<501;$i++){
```

```

    $old_name = "arquivo" . $i . ".txt";
    $new_name = "arquivo" . $i . ".html";
    rename($old_name,$new_name);
}
?>

```

Exemplo 03:

Você também pode utilizar a função rename para mudar o diretório de um determinado arquivo ou conjunto de arquivos. Veja o exemplo abaixo onde vamos mover o arquivo chamado arquivo.txt para dentro de um subdiretório chamado tmp:

```

<?php
$old_name = "arquivo.txt";
$new_name = "tmp/arquivo.txt";
rename($old_name,$new_name);
?>

```

Trabalhando com Permissões

Mudar a permissão de um arquivo ou de uma pasta (diretório) através de cliente ftp é tão simples como: clicar com o botão direito do mouse sobre o arquivo ou pasta desejado, mover o mouse até a opção “Permissões do Arquivo” e por fim, selecionar a permissão desejada.

Até aqui, nada de novo. Nada além do velho e bom feijão com arroz; mas e se você, por algum obséquio do destino, precise alterar a permissão

através do PHP? Muito bem, dado o problema, vamos à solução...

Se você mesmo vai criar uma pasta através do PHP, já pode definir a permissão no próprio comando `mkdir`. Para quem não sabe, o comando `mkdir` suporta dois parâmetros: o primeiro é o diretório a ser criado e o segundo é exatamente a permissão dele.

Vejam:

```
<?php
for($i=1;$i<100;$i++){
    mkdir($i,0777);
    chmod($i,0777);
}
?>
```

No script acima, serão criados 100 pastas com o comando `mkdir` e já será atribuído a essas pastas a permissão (0777) que significa: leitura, escrita e execução.

Mas e quanto aos arquivos? Mas e quanto as pastas já criadas? Bem, nessa hora entra em cena o comando `chmod`. Essa função do PHP permite você mudar permissões de arquivos e pastas. Vejam como é simples:

```
<?php
$my_file = "filosofo_olavo.pdf";
```

```
chmod($my_file,0777);  
?>
```

Tenha em mente que o primeiro parâmetro é o nome do arquivo ou do diretório, e o segundo parâmetro é a permissão do mesmo.

Variáveis de Sessão

PHP suporta variáveis de sessão que ficam acessíveis enquanto durar a sessão. Uma sessão inicia quando o usuário requisitar a primeira página do servidor web e permanece “viva” enquanto o usuário estiver interagindo com o servidor. Após algum tempo sem interação com o servidor web, ou seja, o usuário abandonou o site ou foi fazer outra atividade, a sessão caduca e deixa de existir. A utilidade em se utilizar variáveis de sessão é que elas permitem a passagem de valores entre todas as páginas do site. Entenda que os arquivos que representam as variáveis de sessão ficam armazenados no servidor web e estes arquivos podem ser acessados por qualquer script PHP do site. Este fato torna as variáveis de sessão ideais para manter informações de autenticação. Por exemplo, digamos que um determinado usuário efetuou login no sistema. Como o servidor saberá se esse usuário já entrou no sistema ou ainda não? Talvez você nunca tenha pensado nisso, mas é exatamente verificando as variáveis de sessão que

you will be able to know if a determined user already performed login or not.

To inform PHP that a determined variable is a session variable, you must use this syntax:

```
$_SESSION["Nome_da_variavel"]='Valor';
```

In the example above, where it says "Nome_da_variavel" you must place the name of the variable and in value, the value that this variable will contain.

Let's see some examples:

- 1) Create a file called `pagina1.php` and write the following code inside it.

```
<?php
```

```
session_start();
```

```
$_SESSION["meu_nome"] = "Anderson";
```

```
echo $_SESSION["meu_nome"];
```

```
?>
```

- 2) Create a new file called `pagina2.php` and write the following code inside it.

```
<?php
```

```
session_start();
```

```
echo $_SESSION["meu_nome"];
```

```
?>
```

Agora, abra o primeiro arquivo em seu navegador web através do servidor. Até aí nada de novidade, o navegador irá simplesmente imprimir no browser o valor que foi posto na variável de sessão `meu_nome`. Agora abra o segundo arquivo `pagina2.php`. Surpresa! O browser irá imprimir o valor da variável `meu_nome`, mesmo que nenhuma variável tenha sido criada nessa página. Um detalhe é que quando se pretende trabalhar com variáveis de sessão, deve-se sempre iniciar o script com a função `session_start()`. Como o próprio nome sugere, essa função inicia a sessão.

Para saber se uma determinada variável de sessão existe, você deve utilizar a função `isset()`. Por exemplo, caso você quisesse verificar se a variável `meu nome` existe antes de imprimir ela na tela, o código ficaria assim:

```
<?php
```

```
session_start();
```

```
if(isset($_SESSION["meu_nome"])) echo  
$_SESSION["meu_nome"];
```

```
?>
```

Utilizar a função `isset()` é importante, visto que caso você tente acessar uma variável que não existe, um erro irá ocorrer. Outra função interessante quando se trabalha com variáveis de sessão é a função `unset()`. Essa função destrói a variável de sessão passada como parâmetro. Vejam um exemplo:

```
<?php
```

```
session_start();
```

```
unset($_SESSION["meu_nome"]);
```

```
if(isset($_SESSION["meu_nome"])){
```

```
    echo $_SESSION["meu_nome"];
```

```
}else{
```

```
    echo "variável meu_nome não existe";
```

```
}
```

```
?>
```

O código acima irá imprimir a seguinte mensagem:
“variável meu_nome não existe”;

Às vezes precisamos programar áreas restritas em sites, claro que a melhor forma para isso seria desenvolver uma aplicação com banco de dados, no entanto, vamos supor que determinado servidor onde seu cliente hospeda o site não suporte banco de dados, ou ainda que o cliente pediu uma aplicação simples, que terá um número limitado de usuários restritos. Nesses casos você poderá criar uma aplicação que requer login, ofereça áreas restritas e que não necessite de banco de dados.

A idéia por trás disso é muito simples, você precisa apenas criar uma variável onde serão atribuídos valores pares contendo o nome do usuário e sua senha, sempre que a página de login for requisitada. Bom, veja o código e continuamos depois:

```
<?php

session_start();

//----- Logins

$login_dados = array();

function add_login($user,$pass){

    global $login_dados;

    if(!is_single($user)) throw new Exception("Usuário já existe");

    $login_dados[] = array($user,$pass);
```

```

}

function is_single($user){

    global $login_datas;

    if(count($login_datas) < 1) return true;

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

        if($login_datas[$i][0] == $user) return false;

    }

    return true;

}

function login($user,$pass){

    global $login_datas;

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

        if($login_datas[$i][0] == $user && $login_datas[$i][1] == $pass){

            $_SESSION["current_user"] = $user;

            return;

        }

    }

}

function logoff(){

```

```

        unset($_SESSION["current_user"]);

    }

    add_login("Cristiano Ronaldo","123456");

    add_login("Cristiano","hardpass");

    //-----

    $user = isset($_REQUEST["user"])?$_REQUEST["user":"","";

    $pass = isset($_REQUEST["pass"])?$_REQUEST["pass":"","";

    $logoff = isset($_REQUEST["logoff"])?(int)$_REQUEST["logoff"]:0;

    if(!empty($user) && !empty($pass)){

        login($user,$pass);

    }elseif($logoff===1){

        logoff();

    }

    ?>

<html>

<body>

<?php

```

```

        if(isset($_SESSION["current_user"])){

            echo "<p>Seja bem Vindo <strong>" .
htmlspecialchars($_SESSION["current_user"],ENT_QUOTES) .
"</strong></p>";

            echo "<p><a href='".
$_SERVER['PHP_SELF']."?logout=1'>Clique aqui para sair do
Sistema</a></p>";

        }else{

?>

<form>

    <p>

        <h2>Faça o Login</h2>

    </p>

    <p>

        Nome: <input type="text" name="user" /><br />

        Senha: <input type="text" name="pass" />

    </p>

    <p>

        <input type="submit" value="Logar" />

    </p>

</form>

<?php

```

```

    }

?>

</body>

</html>

```

Creio que o código está bem intuitivo, mas vamos dar umas explicações, a começar pela função `is_single`; essa função verifica se um usuário já não foi adicionado, já que o nosso controle nesse sisteminha é o nome do usuário, não é certo permitir que o mesmo se repita. Uma outra opção seria você criar um código para cada usuário e utilizar ele como identificador único para os usuários.

Para adicionar usuarios basta adicionar `add_login("nome do usuario", "senha")` no código e para eliminar um usuário basta comentar a linha onde o mesmo é adicionado ou excluir a mesma.

Agora, **como criar páginas restritas?**

Bem, isso depende, vamos supor que você queira criar áreas restritas para usuários logados no sistema, como você pode ver no nosso exemplo, basta verificar a variável de sessão `current_user`, exemplo:

```

<?php

if(isset($_SESSION["current_user"])){

```



```

        echo "logado";

    }else{

        echo "não logado";

    }

?>

```

Você deverá adicionar o código acima em toda página que você pretende restringir o acesso à usuários logados.

Já para criar áreas específicas para cada usuário, você deve verificar o nome do mesmo, veja:

```

<?php
if(isset($_SESSION["current_user"]) &&
$_SESSION["current_user"] == "Maria de
Fátima"){

    echo "logado";

}else{

    echo "não logado";

}

?>

```

Caso você não tenha entendido muito bem a parte dos formulários, talvez seja melhor ler o próximo capítulo e retornar a esse capítulo depois.

CAPÍTULO 5

Trabalhando com Formulários

É através dos formulários do site que o visitante envia dados para o servidor web. A sintaxe básica de um formulário html é:

```
<html>
<body>

<form action="pega-dados.php" method="POST">
<input type="text" value="" name="nome">
<input type="submit" value="Enviar">
</form>

</body>
</html>
```

No código acima, as informações que nos interessam são:

- na tag form, action é uma propriedade do formulário que indica para qual página os dados do formulário devem ser enviados.
- ainda na tag form, a propriedade method indica qual o método de envio dos dados. Note que existem dois métodos: Get e Post. Quando utilizado o método Get, as variáveis e seus valores irão aparecer na barra de endereço do navegador do usuário, para evitar isso, utilize o método Post.
- dentro do formulário temos duas tags input. A primeira é uma entrada de texto, e a segunda é um botão de envio de formulário. Quando o usuário clicar sobre esse botão, o navegador irá enviar o formulário para o endereço especificado na propriedade action do formulário.

Existem vários tipos de entradas de dados, caso queira obter mais conhecimentos nesse sentido, aconselho que leia um livro sobre html.

Para prosseguirmos, digite o código do formulário acima numa página chamada index.php, e abra-a em seu navegador web. Repare que ao clicar no botão enviar, o navegador irá enviar o formulário para uma possível página chamada pega-dados.php, porém essa página não existe, então uma mensagem de erro 404 ou uma mensagem do tipo “Not Found” será exibida. Isso já era esperado, já que nós ainda não criamos a página pega-dados.php. Faça isso agora, crie um novo arquivo com o nome de pega-dados.php e digite o seguinte código dentro dele:

```
<?php
$nome =
isset($_POST["nome"])?$_POST["nome"]:"";
echo $nome;
?>
```

Agora, repita o teste feito anteriormente. Notará que o código acima pega o nome digitado no formulário e imprime na tela do usuário.

Para receber os valores dos formulários, PHP disponibiliza três formas: \$_GET, \$_POST, \$_REQUEST. A saber:

- \$_GET deve ser utilizado quando o método do formulário for configurado para GET ou para acessar qualquer variável passada junto com o url. Variáveis passadas via url são separadas com o

símbolo &. Veja um exemplo:

<http://livrophp.com/index.php?nome=Anderson&estado=SC>

- \$_POST deve ser utilizado quando o método do formulário for configurado para POST.

- \$_REQUEST pode ser utilizado para acessar qualquer variável, seja GET ou POST.

Passando valores entre páginas PHP

As razões para se passar valores de uma página para outra, são inúmeras. Por exemplo, imaginem um cadastro dividido em três partes. Na primeira parte o usuário coloca seus dados pessoais, na segunda parte o usuário informa seus dados profissionais, e na terceira parte o usuário digita seus objetivos. E, imaginemos ainda, que, caso o usuário não complete os três passos, o sistema não conclua seu cadastro.

No caso exposto acima, uma boa idéia é trabalhar com variáveis e passar elas de página para página até a conclusão do cadastro. Caso ocorra de o usuário não finalizar a etapa três, você simplesmente não o insere no banco de dados.

Você pode utilizar passagem de variáveis através dos dois métodos oferecidos pelo protocolo http, um é o Get e o outro é o POST, o Get irá passar os valores junto com a url. Já viram links iguais a esse <http://site/var=2&comprar=true&cor=xyz>, então,

isso é o método Get em ação. O outro método é o POST, para usar o post você vai precisar de um formulário html e informar esse método nele, veja:

```
/*etapa 1*/
```

```
<form action="etapa2.php" method="POST">
```

```
    <input type="text" name="nome" value="" />
```

```
    <input type="submit" value="Continuar">
```

```
</form>
```

No exemplo acima, o usuário vai digitar o nome e, ao clicar no botão Continuar, o formulário será submetido para a página etapa2.php. Na página etapa2.php você deve verificar o valor nome, pegar os outros dados e seguir para a etapa 3. Veja como seria:

```
<?php
```

```
/*etapa 2*/
```

```
$nome =
```

```
isset($_POST["nome"])?$_POST["nome"]:"";
```

```
if(empty($nome)){
```

```
    echo "O nome não foi informado!";
```

```

        exit();

    }else{

        echo'

<form action="etapa3.php" method="POST">

        <input type="text" name="escolaridade"
value="" />

        <input type="hidden" name="nome" value="" .
$nome . "" />

        <input type="submit" value="Continuar">

</form>';

?>

```

Uma informação importante, no exemplo que foi exposto acima, é que o valor nome, vindo do primeiro formulário, foi inserido no segundo formulário inserindo um campo oculto input type="hidden", contendo o nome do usuário. Com isso, o formulário 02, ao ser submetido para a etapa três, estará passando também esse valor do campo nome. Note que você pode fazer isso várias vezes, passando variáveis entre várias páginas até que colete todos os dados necessários.

A etapa 3 seria algo desse tipo:

```
<?php

/*etapa 3*/

$nome =
isset($_POST["nome"])?$_POST["nome":"",";

$escolaridade =
isset($_POST["escolaridade"])?$_POST["escolari
dade":"",";

if(empty($nome) || empty($escolaridade)){

    echo "Alguns dados não foram informados!";

    exit();

}else{

    //aqui você pode colocar o script para salvar o
usuário no banco de dados

    echo "Cadastro realizado com sucesso!";
}

?>
```

Para se trabalhar com o método Get, basta alterar no formulário o method="GET". E no php, quando for pegar os valores vindos do formulário, utilize

`$_GET` invés de `$_POST`. Além de get e post, existem outras formas de se passar valores entre paginas web, como: Cookies e Session.

Variáveis Session, uma vez que são declaradas, ficam ativas para todas as páginas requisitadas pela sessão atual. Ou seja, se eu tenho 200 páginas php, e na primeira página que o usuário acessar, eu definir uma variável de sessão, essa mesma variável estará disponível para todas as outras páginas, enquanto a sessão estiver ativa. Uma sessão pode caducar por inatividade do usuário, por limite máximo configurado no servidor, ou mesmo por intermédio de código, nesse ultimo caso quando você deseja destruir uma variável de sessão.

Utilizando o mesmo exemplo que dei aqui, veja como ficaria ele com variáveis de sessão:

```
<?php
```

```
/*etapa 2*/
```

```
$_SESSION["nome"] =  
isset($_POST["nome"])?$_POST["nome"]:"";
```

```
if(empty($_SESSION["nome"])){
```

```
    echo "O nome não foi informado!";
```

```
    exit();
```

```

}else{

    echo'

    <form action="etapa3.php" method="POST">

        <input type="text" name="escolaridade"
value="" />

        <input type="submit" value="Continuar">

    </form>';

}

?>

```

Notaram que agora eu não inseri o campo oculto `input type="hidden"`? Com as variáveis de sessão, você não precisa se preocupar em repassar valores entre os formulários, visto que uma vez declarada, a variável de sessão estará ativa para todas as páginas php. A etapa 3 ficaria assim:

```

<?php

/*etapa 3*/

$nome =
isset($_SESSION["nome"])?$_SESSION["nome"]:'
';

```

```

$escolaridade =
isset($_POST["escolaridade"])?$_POST["escolari
dade"]:"";

if(empty($nome) || empty($escolaridade)){

    echo "Alguns dados não foram informados!";

    exit();

}else{

    //aqui você pode colocar o script para salvar o
usuário no banco de dados

    echo "Cadastro realizado com sucesso!";

}

?>

```

Fácil, não? Viram como é fácil passar valores entre páginas PHP, seja através dos métodos get, post ou mesmo cookies e session? Bem, não abuse muito de variáveis de sessão, porque o excesso delas pode tornar a interação com o usuário lenta, visto que consomem memória do servidor. O ideal é utilizar variáveis de sessão somente quando forem poucas variáveis, e dar preferência para cookies ou usar formulários com get e post, caso contrário.

CAPÍTULO 6

Trabalhando com Banco de Dados

A maioria das aplicações web que utilizam PHP, trabalham com o banco de dados Mysql. Já vimos como instalar o Wamp Server, que por sua vez já instala também o servidor de banco de dados Mysql, então, iremos adotar o Mysql Server como o nosso banco de dados padrão para os exemplos a seguir.

Testando a conexão com o Banco de dados Mysql

Bom, a primeira coisa a se fazer é criar um banco de dados, para isso acesse o phpmyadmin, para quem está em localhost, normalmente terá acesso ao phpmyadmin acessando o link <http://localhost/phpmyadmin>. Esse sistema é muito intuitivo, para criar um novo banco de dados procure pelo campo “Criar novo Banco de Dados”, digite aí um nome sugestivo para o banco e clique em “Criar”. Na próxima tela você terá a opção para criar as tabelas do recém criado banco de dados, o descritivo será “Criar nova tabela no Banco de Dado” e haverá dois campos: Nome e Número de arquivos respectivamente. Em nome, coloque o nome da tabela desejada e em número de arquivos, o número de colunas que esta tabela irá conter. Para nosso exemplo pode colocar “usuário” em nome e em número de arquivos coloque 2. Feito isso clique em executar.

Na próxima tela você terá acesso para criar a estrutura da tabela usuário, como nós informamos o número 2 em número de arquivos, então nós teremos dois campos para configurar. O primeiro será o Identificador do usuário, um número único, e o segundo campo será utilizado para armazenar o nome de cada usuário.

Veja a seguinte imagem:

The screenshot shows the phpMyAdmin interface for configuring a table named 'usuario' in a database named 'banco'. The interface is divided into two main sections: a left sidebar and a main configuration area.

Left Sidebar:

- Top: phpMyAdmin logo and navigation icons.
- Database: 'banco' (selected).
- Tables: 'banco (0)'.
- Message: 'Nenhuma tabela encontrada no Banco de Dados'.

Main Configuration Area:

At the top, the path is shown: 'Servidor: localhost' > 'Banco de Dados: banco' > 'Tabela: usuario'.

Campo	id_usuario	nm_usuario
Tipo	INT	VARCHAR
Tamanho/Definição		50
Padrão	None	None
Collation		
Atributos		
Nulo	<input type="checkbox"/>	<input type="checkbox"/>
Índice	PRIMARY	---
AUTO_INCREMENT	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Comentários		

Below the table configuration, there are three sections:

- Comentários da tabela:** A text area for table comments.
- Storage Engine:** A dropdown menu set to 'MyISAM'.
- Collation:** A dropdown menu.

At the bottom, there is a 'PARTITION definition' section with a text area.

At the very bottom, there are buttons: 'Salvar', 'Ou Adicionar 1 campo(s)', and 'Executar'.

Na imagem acima pode ser visto que utilizamos o nome id_usuario para o primeiro campo, repare que como ele irá conter um inteiro, em tipo, deixei configurado como INT. Repare também as propriedades contornadas com uma linha vermelha: Índice e Auto Increment. Em Índice foi configurado para Primary, isso informa ao banco que essa será a chave primária dessa tabela. Um chave primária serve para agilizar consultas e

manter a integridade da tabela. Por exemplo, uma chave primária nunca irá se repetir, será sempre um número único, e caso alguém tentar inserir um valor repetido para uma chave primária, isso irá resultar em um erro, pois o sistema não irá aceitar tal ação. Para quem ainda não entendeu a real utilidade, imagine o seguinte, digamos que você precise diferenciar cada usuário, se você utilizar o nome como base, poderá cometer equívocos, visto que é comum nomes pessoais se repetirem para várias pessoas, nesse caso poderá utilizar esse Identificador para saber quem é quem. A outra propriedade também contornada de vermelho “Auto Increment” informa ao banco que esse valor deverá ser gerado pelo sistema automaticamente. Em nosso exemplo será utilizado para que o próprio banco gere o código de cada novo usuário que for inserido. Note que esse não é um número aleatório, mas sim um número sequencial, por exemplo, se você tem dois usuários e insere mais um, provavelmente esse será o ID 3.

No outro campo, em tipo, nós definimos Varchar, isso porque trata-se de um campo que irá armazenar strings, o nome do usuário no caso. Observe que quando você define que um campo será varchar, deverá também informar o comprimento desse campo. No nosso exemplo eu coloquei o valor 50, cinquenta é um valor bom para nomes pessoais brasileiros.



Feito isso, clique no botão Salvar, agora sim, está pronta sua base de dados para teste, mas antes de testar a conexão com o banco de dados, vamos inserir alguns dados nessa tabela. Para inserir dados de forma fácil, clique no link SQL que se encontra no menu superior. Essa opção permite a você digitar comandos sql diretamente para serem executados no banco corrente. Simplesmente digite essas três linhas:

```
insert into usuario(nm_usuario) values("Elvis Presley");
```

```
insert into usuario(nm_usuario) values("Susan Boyle");
```

```
insert into usuario(nm_usuario) values("Chris Brown");
```

Após exescutar os comandos acima, sua tabela deveria estar como essa:

			id_usuario	nm_usuario
<input type="checkbox"/>			1	Elvis Presley
<input type="checkbox"/>			2	Susan Boyle
<input type="checkbox"/>			3	Chris Brown

Bom, repare que além de testar a conexão com o banco, iremos fazer uma consulta, por isso que criamos a tabela usuários, caso você só queria

testar uma conexão, não seria necessário criar uma tabela, mas somente o banco.

Vamos ao código e na sequência irei comentar ele para vocês entenderem o que ocorre:

```
<?php
$conn = mysql_connect("localhost", "root", "") or
die("Não pude conectar: " . mysql_error());

mysql_select_db("banco") or die("Não pude
selecionar o banco de dados");

$result = mysql_query("select * from
usuario",$conn);

while($row = mysql_fetch_array($result)){
    echo $row["id_usuario"] . " - " .
    $row["nm_usuario"] . "<br>";
}
?>
```

A primeira linha usa a função `mysql_connect` para se conectar a um banco de dados, essa função requer três parâmetros, sendo que o primeiro é o url do servidor mysql - pode ser o ip também -, o segundo parâmetro é o usuário do banco, e o terceiro a senha do usuário do banco. No nosso caso, como estou com a aplicação em localhost, utilizei localhost como servidor do banco, em usuário coloquei root que é o usuário padrão do

mysql, e em senha deixei em branco, já que não configurei uma senha para o banco. Repare ainda nessa linha a função `die`, ela será invocada quando algum erro ocorrer durante o processo de conexão, e caso isso ocorra aquela mensagem que colocamos ali será impressa na tela do usuário.

Na próxima linha temos a seleção do banco de dados, no meu caso específico, eu criei um banco chamado `banco`, nomezinho sugestivo né? Bom, vamos em frente que nessa linha não há mistério algum.

Então, na próxima linha nós chamamos a função `mysql_query`, essa função irá executar uma query no banco e retornar o resultado na variável que nós atribuímos, no nosso caso na variável `$result`. No exemplo acima, nós passamos dois parâmetros, o primeiro é exatamente a query que será executada, e o segundo é o link da conexão com o banco de dados que criamos momentos antes.

Encerrando, nós fizemos uso da função `mysql_fetch_array`, essa função irá converter o resultado de uma query para um array, onde poderemos acessar os valores exatamente como acessamos valores de arrays. O parâmetro para essa função é o resultado de uma query. Repare ainda que utilizamos o `while`, com ele nós iremos percorrer todos os valores contidos no resultado do query, ou seja, enquanto houver resultados imprima o `id` e o nome de cada usuário.

Inserindo dados no banco Mysql

Inserir dados e selecionar dados, insert e select, são as rotinas mais comuns para usuários mysql, então veremos nesse artigo como inserir dados em um banco de dados mysql. A sintaxe para inserir dados no mysql é a seguinte:

```
<?php
Insert Into Banco_de_dados values
(valores_dos_campos);
?>
```

Para exemplificar, vamos supor que temos uma tabela chamada usuarios com as seguintes colunas: id_usuario, nm_usuario, senha_usuario. Agora vamos inserir três usuários para dentro dessa tabela, veja como ficaria:

```
<?php
insert into usuarios(nm_usuario,senha_usuario)
values("Marcos","senha123");
insert into usuarios(nm_usuario, senha_usuario)
values("Tadeu","senha456");
insert into usuarios(nm_usuario, senha_usuario)
values("Vitor","senha789");
?>
```

Talvez você esteja perguntando: e o campo id_usuario? Pois bem, esse campo você deveria ter

configurado como `auto_incremente` e com salto de 1 em 1, isso é necessário para que o próprio mysql gere o id de cada novo usuário que você insira no banco. Por exemplo, o primeiro usuário terá o id = 1, o segundo terá o id = 2 e assim sucessivamente.

Caso você esteja trabalhando com uma tabela sem campos auto-incremente, e caso você vai informar todos os campos no insert, você não precisa informar os campos que serão afetados – como fizemos no exemplo anterior -, para fixar, vamos supor que o `id_usuario` não seja auto-incremente, então a sintaxe do insert no mysql poderia ser da seguinte forma:

```
<?php
insert into usuarios values("Marcos","senha123");
?>
```

Caso você vá informar todos os campos, você não precisa indicar eles, mas caso vá informar somente alguns campos no insert, então você precisa informar ao mysql quais campos estão sendo passados. Mais um exemplo pra fixar. Nesse exemplo vamos supor que você só irá informar o nome do usuário. Ficaria dessa forma:

```
<?php
insert into usuarios(nm_usuario)
values("Marcos");
?>
```

Uma coisa importante pra ter em mente é que os campos que você pretende não informar durante um insert, precisam permitir o valor null, ou então você deve configurar um valor padrão para eles.

Excluindo registros do banco Mysql

A exclusão de registros no banco de dados Mysql é uma atividade rotineira para quem trabalha com aplicações dinâmicas, e saber como montar a query é um fator essencial para se dar bem nessas horas.

No mysql a sintaxe para excluir registros é a seguinte:

```
<?php  
Delete From Nome_da_tabela where Condição;  
?>
```

Vejamos alguns exemplos pra você fixar bem como utilizar esse comando de exclusão de registros do mysql. Vamos supor que tenhamos uma tabela chamada usuarios e as colunas dessa tabela sejam: id_usuario, nm_usuario e senha_usuario. Dentro dessa tabela temos três usuarios cadastrados:

```
1 | marcos | senha123  
2 | tadeu  | senha456  
3 | vitor  | senha789
```

Vejamos uma bateria de exemplos para você fixar bem.

Excluir todos os registros da tabela:

```
“delete from usuarios”
```

Excluir somente o usuário com id igual a 2:

```
”delete from usuarios where id_usuario=2”
```

Excluir todos os registros menos o usuário com id igual a 3:

```
“delete from usuarios where id_usuario <> 3”
```

Excluir somente o usuário marcos:

```
“delete from usuarios where nm_usuario =  
'marcos”
```

Excluir os usuários com id igual a 1 ou 3:

```
“delete from usuarios where id_usuario in (1,3)”
```

Excluir usuários que estão sem senha:

```
“delete from usuarios where senha_usuario = ””
```

Excluir usuários ou com id =2 ou com o nome vitor:

```
“delete from usuarios where id_usuario =2 or  
nm_usuario = 'vitor”
```

Excluir somente o usuários com nome marcos e sem senha:

```
“delete from usuarios where nm_usuario = 'marcos'  
and senha_usuario=””
```

Caso preciso verificar mais dados, você pode ir incluindo and ou or e as devidas condições até satisfazer seu anseio.

Atualizando dados do Banco Mysql

A sintaxe para atualizar dados do banco é muito simples. Veja;

```
“update tabela set coluna = novo_valor”
```

Para melhor compreensão, vamos supor que estamos atualizando uma tabela chamada usuarios contendo os seguintes dados:

1	marcos	senha123
2	tadeu	senha456
3	vitor	senha789

Sendo que a primeira coluna é o id, a segunda é o nome e a terceira é a senha, temos:

Mudar a senha de todos os usuários:

```
“update usuarios set senha = ‘nova_senha’”
```

Mudar somente a senha do usuário com id igual a 2:

```
“update usuários set senha=’nova_senha’ where id=2”
```

Mudar a senha dos usuários com id igual a 1 ou 3:

```
“update usuários set senha=’nova_senha’ where id in(1,3)”
```

Mudar a senha do usuário chamado tadeu:

```
“update usuários set senha=’nova_senha’ where nome=’tadeu’”
```

Mudar a senha do usuário que possui senha igual a ‘senha123’ ou nome igual a vitor:

```
“update usuários set senha=’nova_senha’ where senha=’senha123’ or nome=’vitor’”
```

Mudar a senha do usuário com id igual a 1, mas somente se a senha for igual a ‘senha123’:


```
“update usuários set senha='nova_senha' where  
id=1 and senha='senha123’”
```

Caso seja preciso verificar mais dados, você pode ir incluindo and ou or e as devidas condições até satisfazer seu anseio.

Às vezes precisamos fazer um update que envolve várias tabelas, ou porque você quer executar essa query diretamente de um aplicativo que entende somente a sintaxe sql ou porque você quer ganhar velocidade na execução do update, etc.

As razões podem ser várias, mas antes de vermos como fazer um update envolvendo mais de uma tabela, vamos criar uma situação onde precisaremos desse update e também veremos como fazer o serviço via PHP. Isso tudo pra facilitar o entendimento de quem está iniciando com mysql.

Problema:

Você recebe um banco de dados de um cliente contendo duas tabelas (autor, artigo):

A tabela autor tem os seguintes campos:

- ID
- user_nicename
- display_name

A tabela artigo tem os seguintes campos:

- ID

- post_autor
- post_title
- post_content

Acontece que, invés do usuário cliente ter utilizado o ID da tabela autor como referência na tabela artigo no campo post_autor, ele utilizou ali o user_nicename. Como se trata de um campo não inteiro, essa consulta pode levar muito mais tempo que levaria. Vem-lhe a idéia de arrumar isso – eis que nossa situação está criada -, então vejamos o passo a passo para a solução:

- 1- Primeiro você precisará criar outro campo inteiro na tabela artigo, coloque um nome sugestivo como autor_id
- 2- Crie um arquivo php para fazer o trabalho para você, veja como ficaria o script:

```
<?php
$conn = mysql_connect("localhost", "root", "") or
die("erro na conexão");

mysql_select_db("banco",$conn);

$result = mysql_query("select ID, user_nicename
from autor",$conn);

while($row = mysql_fetch_array($result)){

    update_artigo($row["ID"],$row["user_nice
name"]);
```

```

}

function update_artigo($id_author, $nm_author){

    global $conn;

    $result = mysql_query("update artigo set
autor_id = $id_author where
post_autor='".$nm_author.'"', $conn);

}
?>

```

No exemplo acima foi feito basicamente o seguinte, você se conectou ao servidor, selecionou o banco de dados, executou uma consulta que retornou todos os ID e user_nicename de todos os autores, fez um while nos autores e executou a função update_artigo passando sempre dois parâmetros, o ID e o user_nicename; A função update_artigo atualiza o campo autor_id da tabela artigo conferindo antes se o campo post_autor corresponde ao campo user_nicename da tabela autor.

Esse exemplo em PHP funciona perfeitamente, mas como você notou nós utilizamos duas queries, e o intuito aqui é utilizar apenas um, vamos ver como fazer isso com um update de duas tabelas

3- O grande segredo está na montagem da query, veja como você deve montar a query usando o inner join:

```
"update artigo ar inner join autor au on  
au.user_nicename = ar.post_autor set  
ar.autor_id=au.ID"
```

Fácil né, resolvemos aqui nosso problema, mas vejamos outros exemplos com mais condições:

```
'update artigo ar inner join autor au on  
au.user_nicename = ar.post_autor set  
ar.autor_id=au.ID WHERE au.ID > 50 AND  
ar.post_title = "update com duas tabelas com Inner  
join"
```

Caso você precise conferir mais condições, basta ir inserindo AND e a condição em sí, mas e se o update envolvesse mais tabelas? Bem, é simples, bastaria unir elas com o inner join, veja esse exemplo com quatro tabelas:

```
'update artigo ar inner join autor au on  
au.user_nicename = ar.post_autor inner join  
comentario c on c.id_artigo = ar.ID inner join tag t  
on t.id_artigo = ar.ID set ar.autor_id=au.ID  
WHERE au.ID > 50 AND ar.post_title = "update  
com duas tabelas com Inner join" AND  
c.user_name<>"spamviagra" AND  
t.ds_tag<>"teste"
```

Claro que quanto mais condições você colocar nas cláusulas, mais demorado será para executar o comando.

Renomeando um Banco de Dados Mysql

Não raro, nós que somos desenvolvedores web, precisamos renomear um bando de dados mysql. Para quem já trabalha com o mysql há algum tempo, deve conhecer o comando **RENAME DATABASE nome_do_banco TO novo_nome_do_banco;**, essa linha de comando poderia ser executada em algum programa de gerenciamento remoto ou de gerenciamento web como o caso do phpmyadmin, até ai nada de novo, mas acontece que esse recurso foi removido do Mysql desde a versão MySQL 5.1.23.

E agora, como renomear um bando de dados mysql sem o rename database? Bem, eu conheço duas formas para resolver essa necessidade, vamos a elas:

Renomear banco Mysql localmente ou em servidores vps ou dedicados:

Quando você tem acesso aos arquivos de configuração do mysql, você pode editar o nome da pasta que representa o banco em questão e pronto, fácil assim. Para quem usa o WampServer o

caminho onde ficam os bancos de dados é C:\wamp\bin\mysql\mysql5.1.36\data, nesse caso pra quem estiver com a versão 5.1.36, é óbvio. Aí dentro dessa pasta “data” estarão os seus bancos de dados em forma de subpastas, procure o banco que você quer renomear e renomeie-o, mas atenção, você deve fechar o mysqlserver antes de renomear o banco, e executar ele novamente somente após ter renomeado o banco em questão, se estiver usando wampserver basta clicar com o botão direito sobre o ícone da barra de tarefas e fechar o aplicativo, renomear o banco e abrir o wampserver novamente.

Renomear banco de dados mysql em uma hospedagem compartilhada:

Em hospedagem compartilhada você não terá acesso aos arquivos de configuração de bancos do mysql, nesse caso não vejo uma forma de – literalmente – renomear o banco de dados, mas a solução nesse caso é você exportar o banco existente (somente as tabelas), criar um novo banco com o nome desejado e importar as tabelas exportadas anteriormente, após isso você pode excluir o banco antigo.

CAPÍTULO 7

Erros e Soluções

Quando se está iniciando em PHP é comum se deparar com erros. Alguns erros podem ser causados por erro de programação, já outros podem aparecer devido a configurações errôneas. Há ainda erros que ocorrerem por falta de recursos do servidor onde o determinado script foi instalado.

Veremos a seguir os erros mais rotineiros na vida dos programadores PHP e os possíveis passos para solucioná-los.

Fatal Error: Maximum Execution Time Exceeded

Um erro comum no PHP é o **Fatal error: Maximum execution time exceeded**, esse erro ocorre quando o teu script demorou muito tempo para concluir a execução do mesmo, excedendo o limite que está configurado em seu sistema. Veremos a seguir algumas soluções para resolver esse problema.

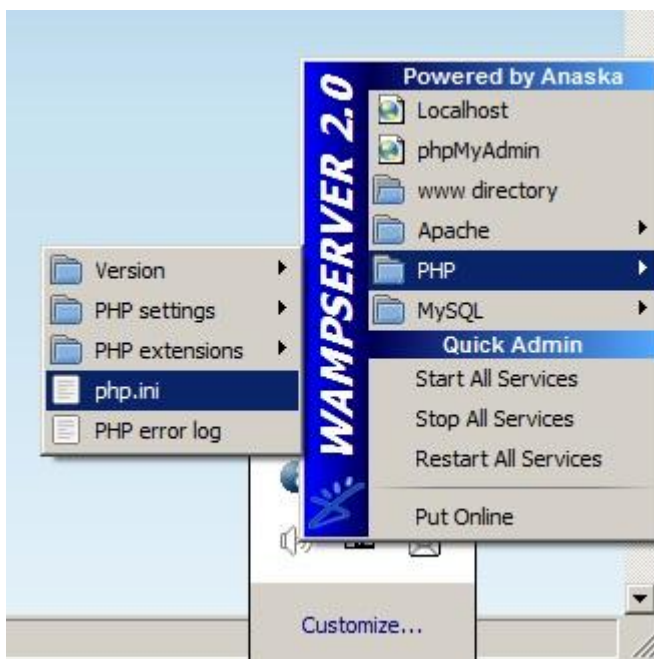
Para quem está testando o script localmente ou em um servidor remoto com acesso root

Você pode editar o php.ini manualmente, para usuários do Wamp Server, isso é muito simples, siga esses passos:

- 1- Após abrir o WampServer, no ícone que fica na barra de tarefas clique com o botão esquerdo
- 2- Arraste o mouse até a pasta PHP, e verá um

ícone do PHP.INI

3- Clique sobre o ícone do php.ini para abri-lo no bloco de notas automaticamente. Veja uma imagem:



4- Agora, com o arquivo aberto e com o foco dentro dele, precione as teclas Control + F, isso é um atalho para fazer uma busca

5- No campo da busca digite max_execution_time e clique em find next.

6- Certamente o notepad vai exibir a linha com essa configuração, agora basta aumentar o valor limite para a execução de script (Lembrando que o valor

para o tempo limite de execução deve ser informado em segundos)

7- Repita o processo 5 e 6 só que agora para a variável `max_input_time`, pode colocar o mesmo valor que utilizou em `max_execution_time`

8- Caso você esteja enviando muitos dados para o servidor é interessante aumentar também o valor para `post_max_size`, para isso repita o processo 5 e 6 procurando por `post_max_size`, só que agora utilize um valor em Megabytes, dessa forma `post_max_size = 50M` (nesse caso estou limitando o post para no máximo 50Megabytes)

9- Para quem vai fazer uploads de arquivos grandes, repita o processo 5 e 6 procurando agora por `upload_max_filesize`, coloque aí um valor em Megabytes, dessa forma: `upload_max_filesize = 100M` (nesse caso estou limitando o tamanho do arquivo para upload em 100Megabytes)

10- Feitas as alterações necessárias salve e feche o arquivo.

11- Clique agora com o botão direito do mouse sobre o ícone do Wamp na barra de tarefas e depois clique em Refresh.

Para quem está enfrentando o problema em um servidor remoto compartilhado (hospedagem convencional):

A hospedagem compartilhada não permite ao usuário editar o arquivo `php.ini`, mas em alguns casos você pode criar um arquivo `php.ini` localmente para resolver seu problema, veja como:

- 1- Crie um arquivo chamado php.ini
- 2- Abra-o com o bloco de notas e digite o seguinte

```
max_execution_time = 3600  
max_input_time = 3600  
post_max_size = 50M  
upload_max_filesize = 100M
```

Nesse caso eu coloquei 3600 segundos, o que equivale a 1 hora, além disso configurei as outras variáveis com valores que achei convenientes para a minha aplicação web, mas você deve colocar valores que correspondam à seu anseio. Feito isso salve o arquivo e envie-o para o servidor remoto. Atenção, você deverá enviar esse arquivo para dentro do mesmo diretório onde o seu script que excedeu o limite de execução se encontra.

Outra forma possível de resolver esse erro é através de um arquivo .htaccess:

Crie um arquivo chamado .htaccess e abra-o com o bloco de notas, dentro dele digite o seguinte:

```
php_value max_input_time 3600  
php_value post_max_size 50M  
php_value max_execution_time 3600  
php_value upload_max_filesize 500M
```

Feito isso salve as alterações e envie esse arquivo para a pasta principal de sua hospedagem web, geralmente é www ou public_html, se houverem os

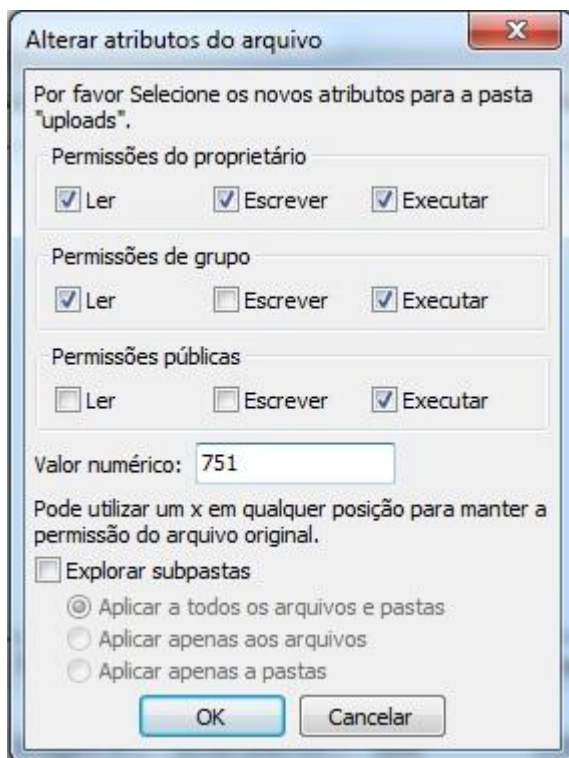
dois diretórios, tanto faz, pois na verdade, www é como um atalho para o diretório public_html.

Erro 500 – Internal Server Error

Esse é um erro de configuração do servidor, que normalmente é acompanhado da seguinte mensagem: “The server encountered an internal error or misconfiguration and was unable to complete your request”

Quem trabalha com web, host, criação de websites, ou supote nesse meio, já deve ter se deparado com esse erro, eu sou um que já se deparou e não foram poucas as vezes, então vejamos algumas possíveis soluções para o erro **500 – Internal Server Error**

A primeira coisa é verificar as permissões da pasta, geralmente esse erro ocorre quando a permissão está com 777, mude para 775 e caso não resolva mude para 751. Para alterar as permissões, basta acessar o ftp da hospedagem web com seu programa favorito (indico o Filezilla), navegar até a pasta em questão, clicar com o botão direito do mouse e no menu que é exibido clique em atributos do arquivo, na nova janela coloque as permissões que você quer.



Para os arquivos dentro da pasta em questão, talvez você precise configurar também as permissões, nesse caso a permissão vai depender do que você quer fazer com determinado arquivo.

Caso tenha observado que as idéias acima não tenham surtido efeito positivo para você, verifique se há um arquivo chamado “.htaccess” dentro da pasta em questão. Caso houver, tente renomear ele para outro nome do tipo .htaccess_old, feito isso

tente acessar novamente a pagina ou arquivo o qual exibia o erro 500 internal server error.

Creio que com o exposto acima você conseguirá resolver seus sinistros com permissões no seu servidor de hospedagem web.

Fatal Error: Register Globals is disabled in php.ini

Esse erro ***FATAL ERROR: register_globals is disabled in php.ini***, ocorre quando a diretiva `register_globals` está desativada. Alguns defendem que deixar o `register_globals` ativo é uma falha de segurança, já outros não sabem trabalhar sem ele. Bom, não tenho o objetivo de dizer o que é certo ou errado, mas sim mostrar como resolver esse problema.

A solução, que sempre funcionou para mim, é criar um arquivo `.htaccess` na pasta principal, `public_html`. Então abra o bloco de notas e insira o seguinte:

Para desativar o register globals:

```
#desativando o register globals  
php_flag register_globals off
```

Order allow,deny
Allow from all

Para ativar o register_globals:

```
#ativando o register_globals  
php_flag register_globals on  
Order allow,deny  
Allow from all
```

Após ativar ou desativar, salve o arquivo **.htaccess** e envie para o public_html.

Outro forma possível é através de um php.ini, mas a opção de utilizar php.ini pra alterar configurações do servidor precisa estar ativo no servidor em questão, isso é feito unicamente pelo administrador do servidor. Vamos supor que este recurso esteja liberado para você, veja então como proceder:

- Crie um arquivo chamado php.ini
- Abra-o com o bloco de notas e digite o seguinte:

Para desativar o register_globals:

```
register_globals = off
```

Para ativar o register_globals:

```
register_globals = on
```


Um detalhe importante é que você precisa enviar esse arquivo `php.ini` para dentro de cada pasta onde o recurso é exigido. Vamos supor que você tenha a seguinte cadeia de diretórios `www/scripts/site01`, e também que há um script que precisa do register globals ativo dentro da pasta `scripts` e outro dentro da pasta `site01`, nesse caso você precisará enviar o arquivo que criamos – `php.ini` –, para dentro dessas duas pastas: `scripts` e `site01`.

Eregi() is deprecated

Para quem é developer em PHP, é possível que tenha encontrado em alguns de seus scripts a mensagem “`eregi() is deprecated`”. Isso ocorre porque essa função tornou-se obsoleta no PHP 5.3, e é desencorajada já que é muito lenta.

A função que você deve utilizar para substituir a `eregi` é a `preg_match()` que também trabalha com expressões regulares.

Bom, vamos ver um exemplo onde `eregi` era utilizado pra verificar se um arquivo era do tipo `jpg` e seu equivalente com o `preg_match`:

```
<?php
//Deprecated
if(!eregi("^image/(jpeg/jpg/jpeg)$",
$arquivo["type"])){
```

```

    //do something;
}
//

if(!preg_match("/^(pjpeg/jpg/jpeg){1}$/i",
$arquivo["type"], $matches)){
    //do something;
}
?>

```

Um detalhe interessante é o /i no final da expressão regular, ele significa “insensitive”, ou seja, não é para diferenciar maiúsculas e minúsculas, agora, caso você estiver substituindo ereg invés de eregi, pode suprimir o /i no final da expressão regular.

Cannot Modify Header Information

Um problema que ocorre frequentemente, mas muito frequentemente mesmo, na vida dos webdevelopers, webmasters, programadores web, webdesigners (metidos, vão mecher no código, rs), etc, é o erro “Cannot modify header information – headers already sent by”

Esse é um erro que pode tirar-nos várias horas de sono, quem já se deparou com ele que o diga. O problema se dá porque a priori não se sabe qual a razão desse problema já que olhando no código não há nada de errado. Bom, veja bem, a pista para a

solução desse problema está exatamente no enunciado, ou seja, você está tentando mudar ou configurar uma informação de header quando algo já foi impresso para o usuário.

Veja isso pra entender melhor:

```
<?php  
echo "I love the days that errors don't happen";  
header("Location: algumapagina.php");  
?>
```

O código acima, provavelmente vai gerar o erro “Cannot modify header information – headers already sent by”, isso porque você já imprimiu informações com o comando echo e depois tentou mudar as informações de header. É até sem sentido você querer imprimir algo na tela e redirecionar a página no comando seguinte.

O erro “Cannot modify header information – headers already sent by” também pode ocorrer se, por exemplo, você fizer isso:

```
<html>  
<header><title>Titulo do site</title></header>  
  
<body>  
  
<?php
```

```

if(condição){

    echo "Bem vindo";

}else{

    header("Location: algumapagina.php");

}

?>

</body>
</html>

```

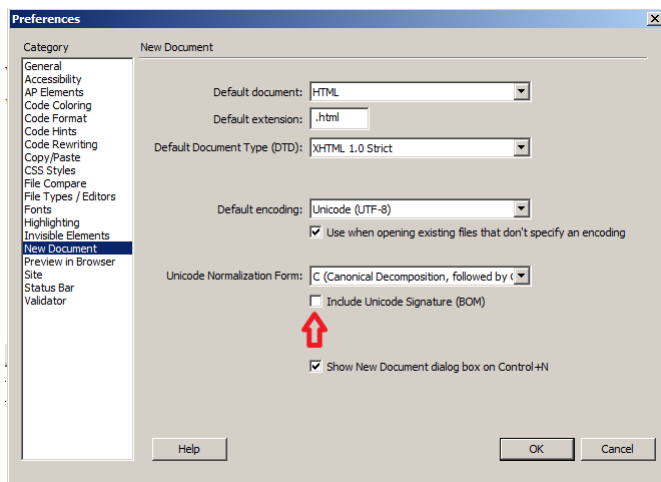
No exemplo acima, caso a condição seja falsa, gerará o erro, isso porque o header também já foi definido pelo html e você está tentando mudar ele com o comando header() do PHP.

Outra coisa que também pode gerar esse erro são espaços em branco antes das tags < ?php ?>, esses espaços podem gerar o erro. Isso ocorre porque nenhuma informação pode ser impressa se você quer mudar o header, portanto exclua todo e qualquer espaço em branco antes ou depois das tags do PHP.

Agora, caso já tenha verificado todos os detalhes acima e o erro continua, seu problema pode ser com o Unicode BOM, esse é um dos mais difíceis de adivinhar, o código BOM, é uma informação

oculta que é posta no cabeçalho do documento, e você não consegue apagar ela na interface visual.

Para evitar esse problema, no DreamWeaver, clique em Edit >> Preferences ou Pressione Control + U (atalho), feito isso será aberta uma janela de preferências, em categorias selecione New Document, e desmarque a opção “include unicode signature (BOM)”, caso esteja marcada, é claro. Veja a imagem abaixo;



É isso, pra finalizar, só um alerta, fique atento aos includes e requires, por exemplo, caso você incluía algum arquivo que imprima alguma coisa ou que contenha BOM unicode signature, o erro vai aparecer até que você limpe todos os arquivos que são incluídos antes do comando header ser invocado.

CAPÍTULO 8

Programação Orientada a Objetos

A partir do PHP 5, o suporte a POO (Programação Orientada a Objeto) foi melhorado consideravelmente, por isso iremos abordar POO tendo como base PHP 5 ou superior. As vantagens de se utilizar POO no PHP são: você pode reutilizar códigos complexos em vários scripts, apenas importando a classe desejada; uma parte do código pode ser testada separadamente; é possível restringir acesso a métodos e propriedades; o código torna-se mais intuitivo, facilitando a manutenção do mesmo; etc.

Um objeto, por assim dizer, é uma classe instanciada. Note que vários objetos podem ser instancias da mesma classe. Veja o exemplo abaixo:

```
<?php
class carro{
    public $rodas = 4;
    public $portas = 4;
}

$gol = new carro();
echo "gol=" . $gol->rodas . "rodas";

echo "<br>";

$picasso = new carro();
echo "picasso=" . $picasso->rodas . "rodas";
?>
```

No exemplo acima, tanto o objeto gol como o objeto picasso, são instâncias da mesma classe carro, por isso, ambos os objetos possuem as mesmas propriedades: rodas e portas.

Construtores e Destrutores

Quando você instancia um objeto, o método `__construct` é invocado e quando o objeto finaliza suas funções, o método `__destruct` é invocado.

Veja:

```
<?php
class carro{
    function __construct(){
        echo "inicio";
    }

    function faz_algo(){
        echo "faz algo";
    }

    function __destruct(){
        echo "fim";
    }
}

$gol = new carro();
$gol->faz_algo();
?>
```


Mas qual é a utilidade desses métodos? Bem, quando você não define um construtor e destrutor em uma determinada classe, o PHP cria construtores e destrutores vazios, que nada fazem. Porém, como você pode definir eles, você também pode iniciar valores, chamar métodos, etc, já na instanciação de um objeto ou logo antes da finalização da instância. Por exemplo, no nosso objeto gol, vamos iniciar ele configurando a propriedade portas para o valor 2. Veja como ficaria na prática.

```
<?php
class carro{
    public $portas;
    public $rodas;

    function __construct($portas = 4){
        $this->portas = $portas;
    }

    function __destruct(){
    }
}

$picasso = new carro();
echo "picasso= " . $picasso->portas . "
portas<br>";
$gol = new carro(2);
echo "gol= " . $gol->portas . " portas";
?>
```

No constructor, coloquei um valor padrão para o argumento \$portas, isso permite-nos invocar o método passando o parâmetro porta ou omitindo ele, nesse ultimo caso o valor padrão 4 será utilizado pelo script. Outro detalhe deste script é que utilizei o \$this. O \$this é uma referência para o objeto que instanciou a classe, ou seja, para a própria classe instanciada. O \$this foi utilizado para alterar o valor da propriedade porta do próprio objeto que foi instanciado, no nosso caso, o objeto gol.

Trabalhando com Herança

Para entender herança em PHP é aconselhável ter como base uma relação entre pai e filho, sendo que a classe principal é a classe pai e a classe que importa ou recebe as características da classe pai, é a classe filho. Vejam esse exemplo:

```
<?php
class carro{
    public $portas;
    public $rodas = 4;
    public $funcao = "transporte";
}

class onibus extends carro{
    public $rodas = 8;
    public $horario_centro = "2,4,6,8";
    public $horario_bairro = "3,5,7";
}
```

```
}
```

```
$onibus = new onibus();  
echo "onibus = ". $onibus->rodas . " rodas";  
echo "<br>onibus função = ". $onibus->funcao;  
echo "<br>onibus horario centro = ". $onibus->horario_centro;  
?>
```

A classe ônibus herdou da classe carro todas as suas características, porém ela também tem suas próprias características. A classe ônibus não possui uma propriedade função, porém ela herdou essa propriedade da classe carro. Além disso, a classe ônibus sobrescreveu a propriedade rodas. Quando uma classe filho declara propriedades e métodos com o mesmo nome da classe pai, esses métodos e propriedades são sobrescritos.

Uma classe pode ter herdado características de várias outras classes, sempre obedecendo a uma determinada cadeia hereditária. Veja o código abaixo:

```
<?php  
class carro{  
    public $portas;  
    public $rodas = 4;  
    public $funcao = "transporte";  
}
```

```
class onibus extends carro{  
    public $rodas = 8;
```

```

}

class meu_carro extends onibus{
    public $ano = "2010";
}

$meu_carro = new meu_carro();
echo "Meu Carro = " . $meu_carro->rodas . "
rodas";
echo "<br>Meu Carro função = " . $meu_carro-
>funcao;
echo "<br>Meu Carro ano = " . $meu_carro-
>ano;
?>

```

Entendendo os Métodos

Métodos são funções dos objetos, ou seja, funções que estão embutidas em uma determinada classe. Um método pode fazer inúmeras coisas: imprimir algo na tela; alterar valores de propriedades do objeto, salvar dados em um banco de dados, etc. Os métodos também são transmitidos para as classes filhas assim como as propriedades, vejamos:

```

<?php
class soma{
    function somar($x,$y){
        return ($x+$y);
    }
}

```

```

}

class subtrai extends soma{
    function subtrair($x,$y){
        return ($x-$y);
    }
}

class calc extends subtrai{
    function dividir($x,$y){
        return ($x/$y);
    }
    function multiplicar($x,$y){
        return ($x*$y);
    }
}

$calc = new calc();

echo $calc->somar(2,2);
echo "<br>" . $calc->subtrair(2,2);
echo "<br>" . $calc->multiplicar(2,2);
echo "<br>" . $calc->dividir(2,2);

?>

```

Acessibilidade dos Métodos e propriedades

Você pode declarar um método ou propriedade como:

- **public** – define um método ou propriedade como público, ou seja, acessível dentro da própria classe, para outras classes ou diretamente via invocação a partir do objeto
- **protected** – define um método ou propriedade apenas acessível à outras classes e a própria classe..
- **private** - define um método ou propriedade como acessível apenas dentro da própria classe.

```
<?php
class Matematica{
    protected $valor_de_pi = "3,1415926";

    private function somar($x,$y){
        return ($x+$y);
    }

    public function imprime_soma($y,$z){
        return $this->somar($y,$z);
    }
}

class Pi extends Matematica{
    public function pegar_pi(){
        return $this->valor_de_pi;
    }

    public function pegar_soma($x,$y){
        return $this->somar($x,y);
    }
}
```

```
$calc = new Pi();  
?>
```

Considerando o código acima, note o seguinte:

```
echo $calc->pegar_pi();  
echo $calc->valor_de_pi;  
?>
```

O primeiro echo vai imprimir o valor de PI normalmente, já o segundo vai gerar um erro. Isso ocorre porque a propriedade `valor_de_pi` está configurada como `protected`, ou seja, não pode ser acessada diretamente pelo objeto. No primeiro echo, essa mesma propriedade foi acessada através de um método chamado `pegar_pi` e por isso não ocorreu nenhum erro. Note ainda que:

```
echo $calc->imprime_soma(2,2);  
echo $calc->pegar_soma(2,2);
```

O primeiro echo vai imprimir a soma de 2 mais 2, já o segundo echo vai gerar um erro. Embora exista um método chamado `pegar_soma`, o método `somar` está definido como `private`, ou seja, acessível apenas dentro da própria classe. Isso explica porque o primeiro echo imprimiu a soma sem erros, já que foi utilizado um método chamado `imprime_soma` que está declarado dentro da própria classe do método `somar`.

Entendendo Constantes

Quando você vai criar um script e já de antemão lhe é informado valores que serão fixos e que serão utilizados em seu script, armazenar esses valores em constantes é uma boa idéia. As constantes diferem das variáveis porque, como o nome diz, são constantes, ou seja, não podem ser alteradas depois da declaração, já as variáveis podem variar de valor conforme o script necessite, com isso, trabalhar com constantes é muito seguro e evita o programador cometer equívocos como alterar um valor que não deveria ser alterado durante a programação.

Vejamos algumas formas de se declarar Constantes e pegar os valores delas no PHP:

Declarando e definindo constantes no php com define:

```
<?php
function echo_br($x){
    echo $x . "<br>";
}

define("PRETO", "000", true);
define("BRANCO", "fff");

echo_br(PRETO);
```



```
echo_br(BRANCO);  
?>
```

Aí acima, a função `echo_br` é apenas auxiliar e vai inserir uma quebra de linha após imprimir a constante, mas não dê atenção para essa função, o foco importante são as declarações utilizando `define`. `Define` possui três parâmetros, o primeiro é o nome da constante, o segundo é o valor – arrays e objetos não são aceitos -, e o terceiro é opcional e trata-se de um valor booleano (`true` ou `false`), esse terceiro parâmetro define se a constante irá diferenciar minúsculas de maiúsculas quando chamada, o valor padrão é `true`. Aqui também é importante salientar que se convém utilizar um nome todo em caixa-alta para constantes, isso vai facilitar a compreensão do código por outro programador e mesmo por você.

Declarando e definindo constantes no php com `const`:

Além do `define`, você pode definir constantes no PHP utilizando o `const`, o `const` trabalha como uma declaração de tipo que você coloca antes do nome da constante, veja o código como fica na prática:

```
<?php  
const BLACK = "000";  
const WHITE = "fff";  
  
echo_br(BLACK);
```

```
echo_br(WHITE);  
?>
```

Declarando e definindo constantes no php dentro de Funções:

Para definir constantes dentro de funções você deve utilizar o define, veja como fica:

```
<?php  
function cores(){  
    define("AMARELO", "YELLOW");  
    echo_br(AMARELO);  
}  
  
cores();  
?>
```

Note que a constante não fica restrita para a função, e uma vez declarada não poderá ser declarada novamente porque isso vai gerar um erro. Nesse caso, essa função do jeito que está aí só poderá ser chamada uma única vez. Declarar constantes dentro de funções, salvo em casos específicos, pode ser um desserviço, portanto só utilize esse recurso se souber bem o que está fazendo.

Declarando e definindo constantes no php dentro de Classes:

Para declarar constantes dentro de classes você deve utilizar o const, veja como fica na prática:

```

<?php
class Cores{
    const AZUL = "BLUE";

    public function get_azul(){
        return self::AZUL;
    }
}
?>

```

Acessando o valor de Constantes no php:

Veja como acessar o valor da constante AZUL sem instanciar a classe em um objeto:

```

echo_br(Cores::AZUL);
echo_br(Cores::get_azul());

```

Veja agora como acessar o valor da constante AZUL através de um objeto instanciado:

```

$cores = new Cores();
echo_br($cores->get_azul());

```

Veja como acessar o valor de uma constante com um nome desconhecido de antemão:

```

$azul = "AZUL";
echo_br(constant("Cores::". $azul));

```

Veja como acessar o valor de uma constante quando o nome da classe é desconhecido de antemão:

```
$cores_class = "Cores";  
echo_br($cores_class::AZUL);
```

Como saber se uma constante já foi declarada (definida) no php:

O php possui uma função para conferir se uma constante já foi definida, é a função `defined()`, essa função irá retornar `true` em caso positivo e `false` em caso negativo, veja como fica na prática:

```
const AZUL = "BLUE";  
if(defined("AZUL")) echo AZUL;
```

E para verificar se uma constante existe dentro de uma classe você pode fazer da seguinte forma:

```
if(defined("Cores::AZUL")) echo "AZUL possui  
valor= " . Cores::AZUL;
```

Trabalhando com Atributos e Métodos Static

Para acessar atributos e métodos static você utiliza a mesma sintaxe das constantes. Veja o exemplo:

```

<?php
class Matematica{
    public static $valor_de_pi = "3,1415926";

    public static function somar($x,$y){
        self::$valor_de_pi = "123";
        return ($x+$y);
    }

}

/*class Pi extends Matematica(){
    public static funcao altera_pi(){
        parent::$valor_de_pi = "123";
    }
}*/

echo Matematica::$valor_de_pi;
echo "<br>";
echo Matematica::somar(2,2);
echo "<br>";
echo Matematica::$valor_de_pi;
?>

```

Utilizar métodos e atributos estáticos torna possível acessar propriedades e invocar métodos das classes sem precisar instanciar um objeto. Notaram que no exemplo acima o valor de Pi foi pego da classe Matemática sem a necessidade de se instanciar nenhum objeto? A sintaxe para acessar um atributo static é a mesma que nós utilizamos para acessar uma contante de classe. Ou seja,

NomeDaClasse::atributo. Porém não confunda atributos constantes com atributos static. Tenha em mente que constantes são valores que não sofrem alterações, já atributos static podem sofrer alteração de valor, assim como fizemos no exemplo acima, alterando o valor de pi. Quando você está dentro de uma classe, deve utilizar self para fazer referência à própria classe e poder acessar atributos e métodos static ou constantes. Já quando está dentro de uma classe filha, deverá utilizar parent para fazer referência à classe pai.

Trabalhando com atributos static, mesmo que você instancie dois objetos, quando você alterar o valor de algum atributo static em qualquer um desses objetos, ele será alterado para todos os objetos. Veja que interessante:

```
<?php
class Matematica{
    public static $valor_de_pi = "3,1415926";

    public static function somar($x,$y){
        self::$valor_de_pi = "123";
        return ($x+$y);
    }

    public static function altera_pi(){
        self::$valor_de_pi = "456";
    }
}
```

```
$calc = new Matematica();  
echo $calc::$valor_de_pi;  
echo "<br>";  
echo $calc->somar(2,2);  
echo "<br>";  
echo $calc::$valor_de_pi;  
  
echo "<hr>";  
$calc2 = new Matematica();  
echo $calc2::$valor_de_pi;  
echo "<br>";  
$calc2::altera_pi();  
echo $calc::$valor_de_pi;  
?>
```

PDO (PHP Data Object)

PDO é uma classe para trabalhar com banco de dados que já vem no núcleo do PHP 5. Algumas das vantagens são: ela é mais rápida que outras classes do tipo que existem por aí, visto que é compilada e não interpretada, protege contra sql injection, é de fácil entendimento, etc.

Para o exemplo abaixo, crie um banco de dados com o nome usuarios, dentro desse banco crie uma tabela chamada usuario contendo três colunas:

- 1- id (inteiro)
- 2- nome (varchar 30)
- 3- senha (varchar 15)

Dentro da tabela usuario, insira os seguintes usuários e suas senhas:

Edemar - Edi2010

Cristiano - mudar123

Maria – escola

Agora, vamos ver um script que irá fazer conexão com o banco de dados recém criado e listar todos os usuários. Veja:

```
<?php
$servidor = "localhost";
$tipo_servidor = "mysql";
$nome_do_banco = "usuarios";
$usuario = "root";
$senha = "";

$conn = new
PDO("$tipo_servidor:host=$servidor;dbname=$n
ome_do_banco",$usuario,$senha);

$tb_usuarios = $conn->prepare("select * from
usuario");
$tb_usuarios->execute();

while($linha = $tb_usuarios-
>fetch(PDO::FETCH_ASSOC)){
    echo $linha["nome"] . " - " .
$linha["senha"] . "<br>";
}
```


?>

Talvez alguns leitores estejam se perguntando onde está a classe PDO. Como já foi dito aqui no livro, a classe PDO vem no núcleo do PHP 5, por isso você não precisa incluir ela de nenhum arquivo externo, basta apenas instanciar ela em algum objeto e pronto. No nosso caso, eu a instanciei em um objeto chamado \$conn. Mas como inserir parâmetros na string sql de forma segura? Muito simples, veja o exemplo abaixo:

```
<?php
$servidor = "localhost";
$tipo_servidor = "mysql";
$nome_do_banco = "usuarios";
$usuario = "root";
$senha = "";

$conn = new
PDO("$tipo_servidor:host=$servidor;dbname=$n
ome_do_banco",$usuario,$senha);

$id = "2";

$tb_usuarios = $conn->prepare("select * from
usuario where id=:id");
$tb_usuarios->bindParam(":id",$id,
PDO::PARAM_INT);
$tb_usuarios->execute();
```

```

while($linha = $tb_usuarios-
>fetch(PDO::FETCH_ASSOC)){
    echo $linha["nome"] . " - " .
$linha["senha"] . "<br>";
}
?>

```

Como visto no exemplo acima, para inserir parâmetros numa string sql, PDO possui um método chamado bindParam(), este método possui três parâmetros: o primeiro parâmetro é o nome que está na string sql - note que na string sql você deve colocar dois pontos antes do nome a ser substituído, no nosso caso foi utilizado “:id” -, o segundo parâmetro é a variável que contém o valor a ser substituído na string sql, por último temos o terceiro parâmetro que é o tipo de dado do segundo parâmetro. Como se tratava de um inteiro, utilizei “PDO::PARAM_INT”. Caso fosse um texto, eu iria utilizar “PDO::PARAM_STR”.

Se necessário, você pode ir inserindo mais parâmetros, conforme o exemplo abaixo:

```

<?php
$servidor = "localhost";
$tipo_servidor = "mysql";
$nome_do_banco = "usuarios";
$usuario = "root";
$senha = "";

```

```
$conn = new  
PDO("$tipo_servidor:host=$servidor;dbname=$nome_do_banco", $usuario, $senha);
```

```
$id = "10";  
$senha = "mudar123";  
$nome = "maria";
```

```
$tb_usuarios = $conn->prepare("select * from  
usuario where id<>:id and senha <> :senha and  
nome <> :nome");  
$tb_usuarios->bindParam(":id", $id,  
PDO::PARAM_INT);  
$tb_usuarios->bindParam(":senha", $senha,  
PDO::PARAM_STR);  
$tb_usuarios->bindParam(":nome", $nome,  
PDO::PARAM_INT);  
$tb_usuarios->execute();
```

```
while($linha = $tb_usuarios-  
>fetch(PDO::FETCH_ASSOC)){  
    echo $linha["nome"] . " - " .  
$linha["senha"] . "<br>";  
}  
?>
```

Senhores, para quem quer aprender mais sobre PDO, peço que seja um visitante assíduo de meu website <http://Fazer-Site.Net> onde estarei sempre criando novos artigos sobre PDO e sobre PHP em geral. Dúvidas serão bem vindas.

CAPÍTULO 9

Desenvolvendo um Chat (Sistema de Bate-Papo)

Para fixar o nosso aprendizado vamos aprender como criar um webchat (Sistema de Bate-papo) em PHP. Nesse projeto iremos focar na parte funcional do sistema, então não me venham dizer que o layout ou design do chat está feio. Como esse é um livro sobre PHP, vou dar prioridade para o ensino da parte programável do sistema, quanto ao layout e visual, após aprender a fazer o sistema funcionar, você poderá editar a seu gosto ou a gosto de um possível cliente.

Definindo o funcionamento do Chat

Antes de tudo, é necessário definir como o nosso sistema irá funcionar, sem isso, é praticamente impossível desenvolver qualquer coisa que seja. Então vamos lá:

- 1- O Sistema terá uma página inicial onde o visitante escolherá um nome e selecionará a sala que deseja entrar.
- 2- Após clicar no botão entrar, o usuário será redirecionado para a página de conversas. Nessa página serão listados todos os usuários que estão online.
- 3- Ainda na página de conversas haverá um campo para o usuário digitar o texto juntamente com um botão para envio do texto para a sala. Também será possível selecionar um usuário para que a mensagem seja direcionada para ele, porém a

mensagem será pública e todos os outros usuários poderão ver ela.

4- As conversas deverão ser atualizadas a cada 6 segundos.

5- Deverá ter um link para o usuário poder sair da sala.

6- Quando o usuário sair da sala, uma mensagem deverá ser enviada para a respectiva sala avisando que o usuário saiu da sala.

Esse seria o funcionamento básico do chat, porém há outros detalhes a serem levados em consideração. Vejamos:

- 1- Não poderão conter dois usuários com o mesmo nome na sala. Isso iria gerar confusão.
- 2- O sistema deve enviar uma mensagem dizendo que determinado usuário saiu da sala, mesmo quando ele não clique no botão sair e feche o navegador ou a página do chat.
- 3- Embora as conversas devam ser atualizadas a cada 6 segundos, isso não pode ocorrer na página inteira. Isso iria irritar o usuário e tornar a conversa praticamente impossível, visto que ele teria seu texto interrompido a cada 6 segundos.
- 4- O usuário só poderá visualizar as conversas que foram enviadas depois que ele entrou na sala.

- 5- As interações mais antigas que 10 horas deverão ser excluídas para poupar espaço em banco de dados.

Estruturando o Banco de Dados

O primeiro passo é criar um banco de dados com algum nome sugestivo. Sugiro nomeá-lo como chat, nada mais sugestivo, não é mesmo? O segundo passo é criar as tabelas que irão armazenar os dados. Vejam as tabelas que criei, com explicações:

1- **salas** – a tabela salas conterá duas colunas:

- a) id_sala (int) – identificador da sala
- b) nm_sala (varchar 20) – nome da sala

A função da tabela salas é armazenar o identificador e o nome das salas. Nessa tabela você pode inserir quantas salas desejar.

2- **usuarios** – a tabela usuários conterá 4 colunas:

- a) id_usuario (int) – identificador do usuario
- b) nm_usuario (varchar 20) – nome do usuario
- c) id_sala (int) – chave estrangeira provinda da coluna id_sala da tabela salas.
- d) dt_refresh (datetime) – guarda a última data e hora que houve interação entre o chat e o respectivo usuário.

A função da tabela usuários é servir como controle para sabermos quais usuários estão online, em quais salas eles entraram, quais são os nomes desses usuários e qual foi a última data e hora que determinado usuário interagiu com o sistema. Note que, nesse caso, interagir não significa que o usuário enviou mensagens no chat, mas simplesmente que a página está aberta e se atualizando (conectando ao servidor web e buscando as últimas conversas).

3- interacoes – a tabela interações conterá 5 colunas:

a) nm_usuario (varchar 20) – nome do usuário que originou a interação. Note que interação, nesse caso, significa alguma ação em relação ao chat. Por exemplo: entrou na sala, saiu da sala, fala com alguém. Talvez você esteja se perguntando qual a razão de ter esse nome de usuário visto que a tabela usuários já possui esse dado. Bem, não podemos esquecer que a tabela usuário é uma tabela dinâmica, ou seja, os usuários de lá serão excluídos quando saírem da sala.

b) id_sala (int) – chave estrangeira provinda da coluna id_sala da tabela salas. Servirá para sabermos em qual sala, dada interação ocorreu.

c) dt_interacao (datetime) – data e hora que a interação ocorreu.

d) ds_interacao (varchar 500) – campo que irá armazenar o descritivo da interação, ou seja, a

mensagem que os usuários enviaram e suas interações com o sistema.

e) nm_destinatario (varchar 200) – caso uma mensagem tenha sido enviada para um usuário específico, o nome desse usuário será armazenado nesse campo.

È isso senhores. Por incrível que pareça, nosso banco de dados está pronto. Agora só precisamos fazer a parte mais legal do sistema, codificar ele.

Você pode baixar o sql do banco no url:

<http://livrophp.com/downloads/chat/chat.sql.zip>

Codificando o Chat

O sistema será dividido em 8 arquivos:

- 1) index.php – página inicial onde o usuário deverá escolher um nome e uma sala para iniciar o chat.
- 2) Chat.php – página principal do chat. Será onde o usuário digitará e visualizará as mensagens. Nessa página também serão listados os usuários que estão dentro daquela sala. As mensagens e interações serão exibidas dentro de um iframe que abre o arquivo chamado interacao.php. Isso é necessário para que a página desse iframe fique se atualizando a cada 6 segundos sem

que isso interfira na digitação das mensagens. Chat.php irá incluir ainda, dois arquivos: writing.php e users-online.php. Esses dois arquivos só foram criados separadamente para que o código ficasse mais “limpo” e intuitivo.

- 3) Interacao.php – página que irá buscar as mensagens e interações do banco e exibirem na tela.
- 4) Writing.php – página onde estão dois elementos importantes: campo texto de digitação de mensagem e botão de envio de mensagens. Esse arquivo será incluído automaticamente na página chat.php.
- 5) Users-online.php – página que lista todos os usuários que estão online na respectiva sala. Clicando sobre um determinado usuário, será possível enviar uma mensagem destinada a esse usuário. Este arquivo será incluído automaticamente em chat.php.
- 6) Functions.php – contém todas as funções utilizadas no sistema de bate-papo.
- 7) Config.php – arquivo de configuração que instancia uma conexão ao banco de dados. É onde você deve colocar os dados para acesso ao banco de dados.

- 8) Sair.php – destrói a sessão e redireciona o usuário para a página index.php. Será útil para tirar o usuário de uma determinada sala quando ele clicar no link “Sair da Sala”.

Baixe o chat completo com páginas comentadas e mais um brinde surpresa no url:

<http://melhor.ws/go/chat>

Caso tenha alguma dúvida sobre o chat, acesse o site <http://fazer-site.net> e deixe um comentário ou entre em contato.

CAPÍTULO 10

Breve Despedida

Quero, antes de tudo, agradecer a você que adquiriu esse livro, seja no formato digital ou impresso. Espero que ele tenha sido útil para seu aprendizado sobre a linguagem PHP. Como já disse aqui no livro, caso ficou alguma dúvida, não hesite em contactar-me. Agora, pra quem quer ir mais longe, sugiro adquirir meu novo Livro intitulado “Criar Site com PHP” onde vou te mostrar de forma prática como criar vários tipos de sites utilizando a nossa linguagem de programação preferida. Veja mais detalhes do novo Livro no url:

<http://comprar-livro.net/php>

Conheça outros sites que mantenho:

<http://Fazer-Site.net>

<http://GanharDinheiroBlog.net>

Fique com Deus e sucesso na sua vida!

Att: **Anderson Makiyama**

Copyright 2013-2014, Anderson Makiyama – Direitos Reservados

Atenção: Este ebook é vendido com direitos de revenda inclusos. As pessoas estão autorizadas a: fazer cópias, revendê-las ou distribuí-las quantas vezes desejarem, porém **é expressamente proibido alterar o conteúdo deste material**, sob pena de ser processado pelo autor da obra. Fica eleito o foro de Joinville, Santa Catarina, para dirimir as questões decorrentes da execução destes termos!