O Portal do conhecimento

http://apostilando.com

Trabalhando com PHP e MySQL: Uma Introdução

Table of Contents

Trabalhando com PHP e MySQL: Uma Introdução

Hugo Cisneiros,
Última atualização em 27/01/2003

1. A linguagem PHP

A linguagem PHP é uma linguagem de programação criada especialmente para o uso em páginas Web. Mas nem por isso ela não pode deixar de ser usada em ambientes desktop, aplicações servidoras, aplicações de rede, entre outros. Mas o principal escopo da linguagem é justamente trabalhar com o ambiente Web. O site do PHP é http://www.php.net/, e lá você pode encontrar os arquivos necessários para a instalação em vários sistemas, inclusive o Linux e o Windows.

Para quem não conhece linguagens de programação, PHP pode ser umpouco difícil no começo, como toda linguagem, mas quem já está habituado à programar, vai perceber que PHP é muito fácil. O PHPé uma linguagem orientada a objeto, com a sintaxe parecida com a do C, só que muito mais simples e prática. O PHP se diferencia justamente pela rapidez e agilidade que os programadores têm em fazer programas e sistemas, pois com o PHP consegue−se desenvolver muito mais rapidamente do que outras linguagens de programação para Web tais como ASP ou JSP.

E se já não bastasse, o uso do PHP já é muito difundido na Internet. Uma boa quantidade de páginas dinâmicas por aí na Internet são feitas justamente com a linguagem PHP. Então com certeza você não vai ficar isolado no meio... Com certeza você vai ter vários recursos para estudar o PHP e ver como ele é útil para a programação voltada para Web. E por último, o PHP é software livre, ou seja, seu código−fonte está disponível para todos usufruirem! Ótimas vantagens para quem quer começar a programar para Web!

2. O uso do PHP com Bancos de Dados

Uma das melhores habilidades do PHP é lidar com bancos de dados de uma forma fácil. Hoje em dia os

sistemas para Web estão cada vez mais usufruindo das capacidades magníficas que os bancos de dados podem oferecer. Neste tutorial vamos aprender a fazer algumas coisas com o banco de dados MySQL.

O MySQL é um banco de dados simples, rápido e eficiente para se trabalhar com sistemas Web. Além disso ele é bem fácil e intuitivo, então estaremos usando ele para fazer os nossos exemplos deste tutorial. A licença deste banco de dados também é GPL, então é software livre. O site do MySQL é http://www.mysql.com/.

3. Criando um livro de visitas

Antes de mais nada, este tutorial assume que você tenha pelo menos um pouco de noção sobre o que é HTML e um pouco de lógica de programação. Mas também os usuários inexperientes também podem seguir este manual sem problemas, e ir aprendendo ainda mais com ele. Primeiramente iremos criar um livro de visitas bem simples, usando como fonte de dados o banco de dados MySQL.

3.1. Conectando ao servidor MySQL e criando o banco de dados

Não vou explicar aqui como montar um servidor MySQL, porque vai fora do escopo deste tutorial. As distribuições Linux geralmente vem com o MySQL prontopara o uso, como também a instalação no Windows é bastante fácil também. Cabe à você então instalar o seu servidor MySQL. Depois que o servidor MySQL estiver instalando e funcionando, você precisará de um banco de dados para trabalhar. Para criar um, execute o comando:

```
mysqladmin −h localhost −u root −p create guestbook
```

O comando acima criará um banco de dados chamado 'guestbook'. A opção '−h localhost' diz em qual IP/HOST o servidor MySQL está localizado para efetuar a conexão, enquanto que as opções '−u root −p' significa para usar o usuário root para fazer a ação, e perguntar por uma senha. Caso o usuário não tenha senha, a opção '−p' não é requerida. Depois outro comando será necessário, neste caso para conectar ao servidor MySQL usando o nosso recém−criado banco de dados e começar a enviar Consultas para ele através do seu prompt de comando:

```
[hugo@lina hugo]$ mysql -u root -p guestbook
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5 to server version: 3.23.49

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Agora já podemos criar as nossas tabelas. Mas antes precisamos saber o que criaremos! Para o nosso livro de visitas, precisaremos de poucas coisas. Vamos pensar... Huummm... Iremos querer o seguinte do nosso usuário: Nome, Localização e Mensagem. Além disso, precisaremos também de um campo para a data e hora em que o usuário postou as informações. Agora vamos criar a tabela:

```
mysql> CREATE TABLE guestbook (
    ->    id int(5) unsigned zerofill NOT NULL DEFAULT '00000' auto_increment,
    ->    nome varchar(255),
    ->    localizacao varchar(50),
    ->    mensagem text,
    ->    data datetime,
    ->    PRIMARY KEY(id)
    -> );
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Pronto. Criamos a nossa tabela, e ela já está pronta para ser usada. O campo 'id' vai ser a identificação de cada mensagem no livro de visitas, ele terá sempre seu número incrementado (para nunca repetir), nunca será nulo e sempre vai ter '0' (zeros) antes do número. Como definimos o campo como int(5), que significa número inteiro de no máximo 5 digitos, então poderemos armazenar de uma a 99999 mensagem no livro de visitas.

O campo nome é definido com varchar(255), que significa uma string de até 255 caracteres. Então o nome da pessoa poderá ir até 255 caracteres, sendo que o nome tiver apenas 50 caracteres, o banco de dados irá usar apenas os 50 caracteres. Se fosse char(255), o banco de dados iria usar todos os 255 caracteres, desperdiçando

memória. Coloquei 255 (que é o máximo suportado pelo varchar/char) de exagero mesmo, para falar isso. Diga sério, eu nunca vi alguém com um nome tão grande que use 255 caracteres... X_x A localizacao também usa o varchar, só que comno máximo 50 caracteres.

O campo mensagem já é diferente. Ele comporta textos enormes, e não só restritos a até 255 caracteres. Esta é a característica do tipo de campo 'text'. Perfeito para o nosso uso, já que os usuários poderiam deixar mensagens grandes em nosso livro de visitas.

E por último o campo data, que é do tipo datetime. Quando adicionarmos uma entrada nesta tabela, a data que irá aparecer neste campo estará no formato Ano−Mês−Dia Hora:minuto:segundo. Vamos ver como ficou a tabela de descrição da nossa tabela. Fazemos isso da seguinte forma:

```
mysql> describe guestbook;
+-------------+--------------------------+------+-----+---------+----------------+
| Field       | Type                     | Null | Key | Default | Extra          |
+-------------+--------------------------+------+-----+---------+----------------+
| id          | int(5) unsigned zerofill |      | PRI | NULL    | auto_increment |
| nome        | varchar(255)             | YES  |     | NULL    |                |
| localizacao | varchar(50)              | YES  |     | NULL    |                |
| mensagem    | text                     | YES  |     | NULL    |                |
| data        | datetime                 | YES  |     | NULL    |                |
+-------------+--------------------------+------+-----+---------+----------------+
```

Agora vamos incluri alguns dados nesta tabela então:

```
mysql> INSERT INTO guestbook VALUES(
    ->    '00000',
    ->    'Eitch',
    ->    'Makai',
    ->    'Eu posso colocar muito texto aqui, já que este campoé do tipo text.',
    ->    NOW()
    -> );
Query OK, 1 row affected (0.00 sec)
```

```
mysql> INSERT INTO guestbook VALUES(
    ->    '00000',
    ->    'Hugo',
    ->    'Japão',
    ->    'Esta é uma outra mensagem de teste!',
    ->    NOW()
    -> );
Query OK, 1 row affected (0.00 sec)

mysql>
```

Incluímos duas linhas na nossa tabela 'guestbook'. Mas vamos supor que na segunda inclusão, eu troquei as bolas e ao invés de colocar que eu estava no Brasil, coloquei que eu estava no Japão (quem dera). Primeiro vamos visualizar os dois campos para ver suas informações, e depois atualizar os dados da segunda linha para que fique Brasil ao invés de Japão na localização:

```
mysql> SELECT * FROM guestbook;
+-------+-------+-------------+--------------------+---------------------+
| id    | nome  | localizacao | mensagem           | data                |
+-------+-------+-------------+--------------------+---------------------+
| 00001 | Eitch | Makai       | (Mensagem cortada) | 2002-09-19 17:42:53 |
| 00002 | Hugo  | Japão       | (Mensagem cortada) | 2002-09-19 17:43:13 |
+-------+-------+-------------+--------------------+---------------------+
2 rows in set (0.00 sec)

mysql> UPDATE guestbook SET localizacao='Brasil' WHERE id='00002';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM guestbook;
+-------+-------+-------------+--------------------+---------------------+
| id    | nome  | localizacao | mensagem           | data                |
+-------+-------+-------------+--------------------+---------------------+
| 00001 | Eitch | Makai       | (Mensagem cortada) | 2002-09-19 17:42:53 |
| 00002 | Hugo  | Brasil      | (Mensagem cortada) | 2002-09-19 17:43:13 |
```

```
+-------+-------+-------------+--------------------+--------------------+
2 rows in set (0.00 sec)

mysql>
```

Ok então, viu só como eu atualizei os dados? As instruções SQL são bem fáceis, principalmente para quem sabe inglês, pois você consegue 'ler' coerentemente o que está tentando dizer ao banco de dados para fazer. No caso da atualização, pedimos ao banco de dados para atualizar a tabela 'guestbook', mudando o campo localizacao para 'Brasil' nas linhas que tiverem o 'id' igual à 00002. Isso só vai atualizar a linha 2. Se não tivesse a instrução 'WHERE', todos os campos localizacao de todas as linhas seriam mudados para 'Brasil'.

E por último, percebemos que a segunda linha ficou uma porcaria, e que realmente eu não era para estar localizado no Brasil, a mensagem que deixei é ruim, e o meu nome é feio. Então para aliviar todos estes danos, vamos excluir essa linha do banco de dados. Fazemos isso assim:

```
mysql> DELETE FROM guestbook WHERE id='00002';
Query OK, 1 row affected (0.00 sec)
```

Pronto! Nos livramos da segunda linha, ficamos apenas com a primeira mesmo. Usamos o comando "SELECT" para visualizar os dados da nossa tabela. Com este comando podemos requisitar apenas determinados campos, na ordem que quisermos. Veja o exemplo a seguir:

```
mysql> select id,nome,localizacao FROM guestbook;
+-------+-------+-------------+
| id    | nome  | localizacao |
+-------+-------+-------------+
| 00001 | Eitch | Makai       |
+-------+-------+-------------+
1 row in set (0.00 sec)

mysql>
```

Agora vou adicionar uma série de linhas com vários dados diferentes, para que possamos testar mais consultas. Depois de criado mais ou menos 10 linhas diferentes, vamos testar algumas coisas legais do comando SELECT:

```
mysql> SELECT COUNT(*) FROM guestbook;
+----------+
| COUNT(*) |
+----------+
|       11 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT id,nome,localizacao FROM guestbook ORDER BY nome;
+-------+-----------------+------------------+
| id    | nome            | localizacao      |
+-------+-----------------+------------------+
| 00009 | Artemis         | Lua              |
| 00001 | Eitch           | Makai            |
| 00007 | Fulano          | Nova Zelândia    |
| 00012 | Hansi Kürch     | Middle-Earth     |
| 00006 | Joey DeMaio     | Battlefield      |
| 00010 | Kaoru           | Japão            |
| 00005 | King Diamond    | House Of God     |
| 00003 | Lina Inverse    | Mundo de Slayers |
| 00004 | Narusegawa Naru | Hinata Sou       |
| 00011 | Sakura          | Japão            |
| 00008 | Zeca Pagodinho  | Cachaçalandia    |
+-------+-----------------+------------------+
11 rows in set (0.00 sec)

mysql>
```

A função MySQL COUNT(*) conta quantas linhas tem na nossa tabela. No nosso exemplo aqui foram 11 linhas. Logo em seguida a consulta pediu os campos id,nome e localizacao da tabela guestbook, ordenando por ordem alfabética no campo nome. Agora vamos salvar todo o conteúdo de nosso banco de dados em um

arquivo texto com todos os comandos. Chamamos este arquivo de arquivo dump, e criamos ele da seguinte maneira no prompt de comando:

```
$ mysqldump −u root −p guestbook > guestbook.sql
```

O arquivo guestbook.sql será criado, e se você editá−lo com um edito de texto comum, verá que ele contém vários comandos SQL para criar as tabelas do banco de dados e colocar os dados nestas tabelas. Se por acaso você precisar importar estes dados para o seu banco de dados, você usa o seguinte comando:

```
$ mysql −u root −p guestbook < guestbook.sql
```

E já que criamos um backup de todas as nossas tabelas e todos os nossos dados, podemos deletar todos os dados da tabela guestbook com o seguinte comando:

```
mysql> DELETE FROM guestbook;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Como não temos o "WHERE", isso vai deletar todos os dados da nossa tabela guestbook :−) Mas se quisermos restaurar os dados é só utilizar o arquivo .sql que criamos acima. Outros comandos que você poderá usar no cliente MySQL:

```
mysql> SHOW TABLES;
+--------------------+
| Tables_in_guestbook |
+--------------------+
| guestbook          |
| teste              |
+--------------------+
1 row in set (0.00 sec)

mysql> DROP TABLE teste;
```

```
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Aqui respectivamente eu mostrei as tabelas que estava no banco de dados guestbook (criei uma outra tabela chamada teste só para ilustrar) e depois excluí (DROP) a tabela teste do banco de dados.

Deu para perceber como se trabalhar com MySQL? É bem fácil, mas recomendo que você olhe um pouco da documentação do programa, para obter mais dicas. Ao longo do nosso exemplo de livro de visitas, novas dicas irão surgir. Todas as instruções aprendidas neste tópico serão usadas no nosso exemplo, já que o PHP vai mandar para o servidor exatamente estas strings que colocamos no prompt do MySQL. Vamos com calma que você vai entender.

3.2. Utilizando o PHP

Antes de mais nada vamos testar o funcionamento do PHP. O PHP vem com uma função bem simples para mostrar que está funcionando. Suponhamos que o seu servidor já esteja configurado para rodar PHP em todos os arquivos com extensão .php, então vamos criar dentro do diretório de páginas do seu servidor web um arquivo chamado phpinfo.php, com o seguinte conteúdo:

```
<?
// PHP Info: Mostra informações sobre o PHP instalado na máquina
phpinfo();
?>
```

Vejamos... Note que para linhas comentadas (ignoradas pelo interpretador, servindo apenas de comentários ou explicações), o PHP utiliza duas barras (//). Bem ao estilo C, pois ele também suporte comentários de múltiplas linhas, como o a seguir:

```
<?
```

```
/*
-------------------------------------------------------------
Este é um comentário com múltiplas linhas, igual aos comentários
da linguagem C. Isto tudo será ignorado no código.
-------------------------------------------------------------
*/
phpinfo();
?>
```

Depois de salvo o phpinfo.php, coloque o endereço dele no browser e você vai ver várias informações mostradas pelo PHP. Esta função é muito útil para quem quer saber como o PHP foi compilado, com suporte a que, a versão dele, variáveis de ambiente, entre outros. O próximo passo para entender melhor como funciona o PHP, é saber que ele trabalha junto com HTML. Nos exemplos anteriores você pôde notar que começamos o código PHP com a *tag <?* e terminamos com a *tag ?>*. Isso significa que podemos colocar código HTML quando não estivermos dentro desta tag PHP. Veja o exemplo abaixo:

```
<html>
<head>
    <title>Página de Teste</title>
</head>
<body>
<?
$a = 25;
$b = 35;
$c = $a+$b;
echo "Se:<br>\n";
echo "\$a é igual à $a,<br>\n";
echo "\$b é igual à $b,<br>\n";
echo "\$c é igual à \$a+\$b,<br><br>\n";
echo "logo \$c é igual à $c! Incrível!";
```

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

Copie este exemplo para uma página PHP e aponte no browser. Veja que eu misturei código HTML com PHP, e isso é muito comum neste tipo de linguagem para Web. Isso torna o PHP muito flexível para poder ir colocando código onde quiser, misturando sempre com a saíde em HTML. É sempre bom você separar código de design HTML, mas isso se faz usando Templates, que é um passo mais complicado, que não iremos falar neste tutorial.

Vimos neste exemplo acima pouca coisa, mas vale notar que usamos a função *echo* do PHP, que imprime a string no output, ou seja, na página HTML. Mas precisamos de algo mais "complexo" para fazer o nosso guestbook funcionar. Como estamos trabalhando com MySQL, usaremos as funções MySQL incorporadas como módulo ao PHP. Pra não ficar muito chato, vamos para um exemplo:

```
<?
// Mensagens de Erro
$msg[0] = "Conexão com o banco falhou!";
$msg[1] = "Não foi possível selecionar o banco de dados!";

// Fazendo a conexão com o servidor MySQL
$conexao = mysql_pconnect("localhost","root","senha") or die($msg[0]);
mysql_select_db("guestbook",$conexao) or die($msg[1]);

// Colocando o Início da tabela
?>
<table border="1"><tr>
   <td><b>ID</b></td>
   <td><b>Nome</b></td>
   <td><b>Localização</b></td>
```

```
</tr>
<?

// Fazendo uma consulta SQL e retornando os resultados em uma tabela HTML
$query = "SELECT id,nome,localizacao FROM guestbook ORDER BY nome";
$resultado = mysql_query($query,$conexao);
while ($linha = mysql_fetch_array($resultado)) {
   ?>
   <tr>
      <td><? echo $linha['id']; ?></td>
      <td><? echo $linha['nome']; ?></td>
      <td><? echo $linha['localizacao']; ?></td>
   </tr>
   <?
}
?>
</table>
```

O resultado disso, de acordo com os dados que colocamos no nosso banco de dados, será o seguinte:

ID	Nome	Localização
00009	Artemis	Lua
00001	Eitch	Makai
00007	Fulano	Nova Zelândia
00012	Hansi Kürch	Middle−Earth
00006	Joey DeMaio	Battlefield
00010	Kaoru	Japão
00005	King Diamond	House Of God
00003	Lina Inverse	Mundo de Slayers

00004	Narusegawa Naru	Hinata Sou
00011	Sakura	Japão
00008	Zeca Pagodinho	Cachaçalandia

Viu como ficou legal? Extraímos os dados da nossa tabela *guestbook* no banco de dados *guestbook*, e colocamos numa tabela HTML, para visualização no browser. Bem fácil não? Não? Ah! Também, eu esqueci de explicar direito o que o código faz... Tá bom, vamos lá.

As primeiras linhas do código configuram uma array com as mensagens de erro que vou usar depois. Se você ainda não conhece o que é uma array, uma array é um tipo especial de variável que contém vários valores. No caso criamos a array **$msg**, que tem duas outras variáveis dentro dela. A **0** representa a mensagem de erro de que não foi possível conectar ao servidor MySQL, e a **1** representa a mensagem de que não foi possível selecionar o banco de dados. Mais para frente, usaremos array mais uma vez. Aliás, se você for programar em PHP, com certeza você ainda vai usar muita array em sua vida.

Depois é que a começa a esquentar. Começamos a usar as funções MySQL do PHP. A função **mysql_pconnect()** abre a conexão com o banco de dados, e aponta esta conexão para a variável **$conexao**. Esta variável será usada mais tarde para os outros comandos do MySQL. Note também que nesta conexão, foi preciso você informar o host/ip onde o servidor MySQL está rodando, o usuário para se conectar e a senha deste usuário. Se por alguma razão o PHP não conseguir estabelecer a conexão, ele irá chamar a função **die()**, que termina o script PHP e gera uma mensagem no browser, que no nosso caso vai ser o conteúdo da array **$msg[0]**.

Já a função **mysql_select_db()**, você seleciona o banco de dados que usará. O primeiro parâmetro indica o nome do banco de dados que você quer usar, e o segundo a variável de referência que usamos para identificar a conexão com o banco de dados MySQL, que aqui no nosso tutorial nomeamos de **$conexao**. Caso ele não consiga selecionar o banco de dados, ele chama a função **die()**, mostrando o conteúdo da array **$msg[1]**.

Agora vem a parte interessante. Lembra da flexibilidade do PHP, em que você pode sair e entrar do código PHP livremente? Então na parte que se segue você viu um exemplo disso. Saímos do "Modo PHP", e colocamos as tags HTML para iniciar uma tabela. Colocamos as colunas ID, Nome e Localização, que são os

dados que iremos pegar do banco de dados.

Depois entramos no "Modo PH" novamente, e fazemos uma consulta no banco de dados MySQL e colocamos o resultado em uma variável. Como você pode ver, a função **mysql_query()** é responsável pela consulta no banco de dados MySQL. Então você irá utilizar esta função para fazer todas as coisas, selecionar, inserir, excluir, etc, como aprendemos anteriormente no prompt do MySQL. Tudo que você digitou no prompt do MySQL também é válido como primeiro parâmetro da função myqsl_query(). No caso usamos um exemplo que já usamos anteriormente.

A seguir vem o *truque*. O laço while vai fazer o trabalho de pegar cada linha do resultado da nossa consulta MySQL e transformar em array (não falei que iríamos usar mais uma vez?). A função **mysql_fetch_array()** transforma esse resultado de consulta em array. Como temos 11 linhas, o laço vai se repetir 11 vezes. Os dados da linha no banco de dados será identificados na array **$linha**. Então identificamos o dado utilizando **$linha['nome_do_campo']**, ou seja, $linha['id'], $linha['nome'] e $linha['localizacao']. Colocamos isso em uma linha de tabela.

Depois de passado cada linha, totalizando 11 linhas, o laço while acaba e o código PHP continua, chamando a tag que finaliza a tabela, que é o

. Pronto! Extraímos os dados do banco de dados. Fácil né? Com isso deu pra entender melhor como o PHP trabalha com bancos de dados. Agora vamos ao exemplo que realmente interessa...

3.3. Criando o código do Livro de Visitas

Vamos agora criar o Livro de Visitas em si. Vamos usar dois arquivos, um para a visualização do livro de visitas, e outro para a inclusão de uma mensagem no livro de visitas. Vou já incluir todo o código pronto do programa, fragemntado em partes para a explicação. Os comentários também são esclarecedores.

Ok, vamos primeiro à Inclusão! Crie um arquivo chamado *assinar.php* e mãos à obra:

```php
<?
// Exemplo de livro de visitas para aprendizado (Assinar)
// Hugo Cisneiros, hugo_arroba_devin_ponto_com_ponto_br
//
// Eu não me responsabilizo por quaiser danos com o uso deste código.
// Se você fizer mal uso deste código, seu computador irá explodir e
// sua namorada vai te largar. Eu te avisei!
//

// Primeiro vamos verificar se o método da página é POST. Se for POST,
// quer dizer que o usuário preencheu o formulário e e apertou o botão
// Assinar.
if (getenv("REQUEST_METHOD") == "POST") {
   // Configura as variáveis do método POST para virarem variáveis
   // "normais" do PHP (Requer apenas nas versões do PHP acima da 4.1)
   $nome = $_POST['nome'];
   $localizacao = $_POST['localizacao'];
   $mensagem = $_POST['mensagem'];

   // Caso todos os campos forem preenchidos, inclui a mensagem no
   // banco de dados. Caso isso não aconteça, gera uma mensagem de
   // erro que será impressa no browser mais a frente.
   if ($nome and $localizacao and $mensagem) {
      $conexao = mysql_pconnect("localhost","root","senhasecreta");
      mysql_select_db("guestbook",$conexao);
      $query = "INSERT INTO guestbook VALUES('00000','$nome','$localizacao','$mensa
      mysql_query($query,$conexao);
      header("Location: ler.php");
   } else {
      $err = "Preencha todos os campos!";
   }
```

```
}

?>
<html>
<head>
   <title>Livro de Visitas: Assinar</title>
</head>
<body bgcolor="white">

<h1>Assine o Livro de Visitas</h1>

<?
// Se ocorreu algo de errado, então vai existir uma variável $err
// contendo a mensagem. Imprime-se então em fonte vermelha esta
// mensagem.
if ($err) {
   ?>
   <ul><font color="red"><? echo $err; ?></font></ul>
   <?
}
?>

<form method="post" action="assinar.php">
<table border="0"
<tr>
   <td>Nome: </td>
   <td><input type="text" size="15" name="nome" maxlength="250"></td>
</tr>
<tr>
   <td>Localização: </td>
   <td><input type="text" size="15" name="localizacao" maxlength="45"></td>
```

```
</tr>
<tr>
   <td colspan="2">
   <textarea cols="60" rows="10" name="mensagem">Digite aqui sua
mensagem!</textarea>
   </td>
</tr>
</table>
<input type="submit" value="Assinar">
</form>

</body>
</html>
```

Deu para perceber mais ou menos como funciona o código acima? Repare como eu faço muito o uso do controle de fluxo **if**. No começo o *if* testa se o usuário postou algum formulário. Para fazer isso, ele verifica se a variável $REQUEST_METHOD é igual à "POST". Se for "POST", quer dizer que o usuário postou alguma coisa através de um formulário, e se for "GET", quer dizer que a requisição da página foi feita pelo modo normal, sem postar nada.

Depois o código testa com o *if* novamente para ver se o usuário preencheu todos os campos. Para fazer isso ele testa se existe as variáveis correspondentes aos campos de formulário existente na nossa página. Caso aqueles campos estejam todos preenchidos, o código conecta ao servidor MySQL, seleciona o banco de dados que queremos, e depois constrói uma consulta SQL, incluindo os dados na tabela do banco de dados. A parte de inclusão do servidor MySQL não é nova para nós, que já vimos o procedimento anteriormente.

Agora algo interessante. Depois que tudo tiver feito, precisamos redirecionar o navegador do nosso visitante para outro lugar. Para fazer isso, usamos a função **header()** do PHP, que como o nove diz, manda um cabeçalho para o navegador do visitante. A função *header* só pode ser usada no começo dos scripts, sem que nada tenha sido impresso no navegador do cliente. Esta é uma restrição do próprio protocolo HTTP, e não do PHP, pois o cabeçalho sempre tem que vir antes né! :) No nosso caso, enviamos o cabeçalho "Location:", que

diz ao navegador para onde ele tem que ir. Então depois que o código inclui os dados no banco de dados MySQL, ele redireciona o navegador para a página *ler.php*. Pode−se colocar qualquer URL válida no cabeçalho "Location:". Então, continuando o código, se tudo der certo, ele vai para a página de leitura, mas se for o caso dele não ter preenchido os campos, ou se ele não enviou nada por algum campo de formulário, o resto do código é executado.

O resto do código é a página HTML em si, com os formulários para preenchimento e este tipo de coisa. Os comentários no código explicam como o fluxo do if pode ser útil para por exemplo, mostrar uma mensagem de erro, se houver um erro claro.

Agora vamos para o *ler.php*, que vai ser a página onde irá se mostrar as assinaturas do livro de visitas! Vamos ao arquivo:

```
<?
// Exemplo de livro de visitas para aprendizado (Ler)
// Hugo Cisneiros, hugo_arroba_devin_ponto_com_ponto_br
//
// Eu não me responsabilizo por quaiser danos com o uso deste código.
// Se você fizer mal uso deste código, seu computador irá explodir e
// sua namorada vai te largar. Eu te avisei!
//
?>

<html>
<head>
   <title>Livro de Visitas: Ler</title>
</head>
<body bgcolor="white">

<h1>Livro de Visitas</h1>
```

```
<?
// Verifica se existe a variável $begin, que vai indicar a número
// da mensagem que vai aparecer no começo. Se não existir, assume-se
// que vai ser o começo, ou seja, o valor 0.
$begin = $_GET['begin'];
if (!$begin) { $begin = 0; }

// Conecta ao servidor e seleciona o banco de dados
$conexao = mysql_pconnect("localhost","root","senhasecreta");
mysql_select_db("guestbook",$conexao);

// Coloca na variável $total o número total de mensagens no Guestbook
$query = "SELECT count(*) FROM guestbook";
$query = mysql_query($query,$conexao);
$query = mysql_fetch_array($query);
$total = $query[0];

?>
<p>
Total de mensagens postadas: <b><? echo $total; ?></b>
(<a href="assinar.php">Assine você também!</a>)<br>
Exibindo <b>20</b> mensagens por página, mostrando mensagens de
<b><? echo $begin+1; ?></b> a <b><? echo $begin+20; ?></b>.
</p>

<?
// Calcula os links para as próximas mensagens e as anteriores, de
// acordo com o número total de mensagens
if (($begin > 0) and ($begin <= 20)) {
   $anteriores = '<a href="ler.php?begin=0">Anteriores</a>';
} elseif (($begin > 0) and ($begin > 20)) {
```

```
   $anteriores = '<a
href="ler.php?begin=' . ($begin-20) . '">Anteriores</a>';
} else {
   $anteriores = 'Anteriores';
}

if (($begin < $total) and (($begin+20) >= $total)) {
   $proximas = 'Próximas';
} else {
   $proximas = '<a
href="ler.php?begin=' . ($begin+20) . '">Próximas</a>';
}

echo $anteriores . " | " . $proximas;

// Faz uma consulta SQL trazendo as linhas das 20 ultimas mensagens
// que foram colocadas no livro de visitas.
$query = "SELECT * FROM guestbook ORDER BY data DESC LIMIT $begin,20";
$query = mysql_query($query,$conexao);

// Gera uma tabela para cada assinatura no livro de visitas (loop)
while ($linha = mysql_fetch_array($query)) {
   // Organiza a mostragem da data, já que no campo do MySQL, a data
   // se encontra em uma forma não tão legal.
   $var = $linha['data'];
   $var = explode(" ",$var);
   $dia = $var[0];
   $hora = $var[1];
   $dia = explode("-",$dia);
   $data = "$dia[2]/$dia[1]/$dia[0] às $hora";
   ?>
```

```
   <table border="0" width="70%">
   <tr><td bgcolor="navy" colspan="2"> </td></tr>
   <tr>
      <td><b>Data:</b></td>
      <td width="100%"><? echo $data; ?></td>
   </tr>
   <tr>
      <td><b>Nome:</b></td>
      <td><? echo $linha['nome']; ?></td>
   </tr>
   <tr>
      <td><b>Localização:</b></td>
      <td><? echo $linha['localizacao']; ?></td>
   </tr>
   <tr>
      <td><b>Mensagem:</b></td>
      <td><? echo $linha['mensagem']; ?></td>
   </tr>
   </table>
   <?
}

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

Apesar de que no código já há vários comentários explicando a funcionabilidade das linhas, vamos dar uma olhada geral no que este arquivo PHP faz. Antes de mais nada, o código verifica se existe a variável $begin, obtida pelo método HTTP GET (requisição normal HTTP, enquanto que o método POST é por formulário). Esta variável será usada mais adiante para saber quais mensagens serão mostradas para o usuário. Se a

variável não existir, dá−se o valor de 0 para que ele comece da "primeira" mensagem.

Depois é hora de pegar as mensagens do servidor de banco de dados e colocá−las em variáveis para imprimirmos na página, colocando assim as assinaturas do livro de visitas. Para fazer isso, primeiro conectamos ao servidor de banco de dados, e selecionamos o banco de dados. E antes de consultar todas as mensagens, pegamos também o número total de mensagens, através da função **count()** do MySQL, como você pode ver no código. Esta função irá retornar quantas linhas tem na tabela em que se fez a consulta, ou seja, a tabela *guestbook*. Esse número total será usado junto com o $begin para poder ser usada na mostragem das mensagens para o usuário...

...Que será agora! Com um raciocínio simples, e alguns controles de fluxo com o if, podemos criar links diferentes. O que queremos fazer é que se estiver sendo mostrada desde a primeira mensagem, o link de "mensagens anteriores" não será mostrado, e sim apenas o de "próximas mensagens". A mesma coisa é feita com as próximas mensagens, só que dessa vez não aparece o link se estiver mostrando a última mensagem. Como a vida é cruel, não vou explicar detalhadamente como essa parte do código funciona, porque aprender a ler o código e entender é uma parte essencial. É mais difícil eu explicar do que você entender sem ler a explicação :)

O que vem agora é finalmente a mostragem de mensagens. Usaremos aqui dois atributos do MySQL que não conhecemos, que é o **DESC** e o **LIMIT**. Então construímos a consulta do MySQL e armazenamos seu resultado na variável *$query*. Depois vem o interessante, fazemos um laço com a função **while()** do PHP e a cada linha que se tem na nossa consulta (no nosso caso, cada mensagem), se criaria uma *array* chamada *$linha* com os valores de cada linha. No nosso caso, o while() faria com que se imprimisse no navegador uma tabela HTML para cada mensagem do nosso livro de visitas.

Repare também que como não queremos que se mostre todas as linhas da tabela MySQL na nossa consulta, usamos o atributo *LIMIT*, que como o nome diz, especifica um limite para a quantidade de linhas que será retornada. O formato deste atributo é *LIMIT inicio,quantidade*. No nosso caso, o *inicio* é a mensagem inicial, que temos através da variável $begin discutida anteriormente. E como queremos mostrar apenas 20 mensagens, colocamos o número 20, para se mostrar 20 mensagens. É como se disséssemos por exemplo: "Limite a consulta para 20 linhas, começando da linha 0."

Outra coisa que queríamos no nosso livro de visitas, é que as últimas assinaturas seriam mostradas primeiro. Então na consulta que fizemos no MySQL, usamos o atributo *DESC*, que quer dizer "decrecente". Isso já diz tudo, ele inverte a ordem das linhas e vai na ordem descrecente. E além desse, vou deixar de comentar também o uso da função **explode()**, que usei na formatação da data do MySQL, assim você poderá pesquisar melhor e entender como ela funciona. Só uma definição rápida e objetiva: ela cria uma array com a separação de uma string delimitada por um certo caracter que você mesmo especifica! Oops, eu disse tudo!

Com isso, todas as mensagens são mostradas. Experimente o programinha que você acabou de fazer e acompanhar neste tutorial, e comece a personalizá−lo, assim você vai aprender mais ainda. Uma ótima referência de PHP é o manual oficial, disponibilizado no site oficial, http://php.net/, lá você encontra todas as funções que você poderia usar no seu programa. Boa sorte e até a próxima!

1. Anexos

- Arquivo assinar.php do livro de visitas (phps)
- Arquivo ler.php do livro de visitas (phps)

Hugo Cisneiros,