PHP - Módulo 4: Bancos de Dados e Recursos avançados

Por Valdir Dias

Este módulo 4 é a última parte da primeira série deste tutorial. Vamos falar um pouco mais sobre bancos de dados e mostrar algumas funções e recursos avançados do PHP. No script do módulo 3, fizemos a inclusão das informações em uma tabela de um banco de dados MySQL. Hoje veremos como consultar estas informações.

"SELECT"

Vale a pena repetir o método de trabalho com servidores de bancos de dados:

- Conectar ao servidor
- Abrir o banco de dados
- Enviar os comandos SQL
- Fechar o banco de dados
- Desconectar ao servidor

Sendo que as duas últimas geralmente não precisam ser feitas explicitamente, pois quando o script terminar, o banco de dados é fechado e a conexão é encerrada automaticamente.

A consulta aos dados contidos em uma tabela é feita usando o comando SELECT, da linguagem SQL. Este comando diz ao banco para separar, de todos os dados contidos em uma tabela, apenas aqueles que precisamos. Se o SQL fosse em português e se quisermos saber quais pessoas, dentre as que preencheram nosso formulário, têm o primeiro nome Maria, bastaria executar o comando: SELECIONE TUDO DE DADOS ONDE O NOME PAREÇA COM MARIA. Como o SQL é escrito em inglês, basta traduzir (com algumas pequenas adaptações) a frase acima, que ficaria assim: SELECT * (ou ALL) FROM DADOS WHERE NOME LIKE MARIA.

No padrão SQL, podemos informar como queremos a comparação do "parecido", especificando se o que queremos começa, termina ou contém o texto a ser comparado. Como queremos saber todo mundo cujo primeiro nome é Maria, quer dizer que vamos separar todos os dados que iniciam com Maria. Na linguagem SQL, a sentença ficará assim: Select * From Dados Where Nome Like "Maria%";

No MySQL:

$conn = mysql_connect ("localhost" , "admin" , "admin"); #Abre a conexão $db = mysql_select_db("clientes"); #Abre o banco de dados $result = msql_query("select * from dados where nome like 'Maria%'");

Além das funções SQL que já vimos, (SELECT e INSERT) existem várias outras disponíveis. Consulte um tutorial de SQL para conhecê-las.

Restringindo o acesso de usuários

No exemplo de hoje, vamos fazer um script para consultar as informações que foram gravadas na nossa tabela. Para isso deverá ser feita a validação de usuário, de modo que apenas as pessoas autorizadas possam acessar a página de consulta.

O PHP não consegue, sozinho, validar um usuário. Portanto devemos usar um dos métodos descritos seguintes: Se o servidor for o Apache, basta usar um arquivo com as diretivas de autenticação (Geralmente estas informações ficam em um arquivo chamado .htaccess no diretório que queremos proteger). Neste caso a todo o trabalho de validação fica por conta do servidor e, para sabermos qual usuário está "autenticado", basta verificar a variável $REMOTE_USER.

Um outro método é usar um cabeçalho de autenticação do HTTP e verificar o valor das variáveis $PHP_AUTH_USER e $PHP_AUTH_PW para saber o usuário e senha digitados, respectivamente. Com estes valores podemos consultar um banco de dados para verificar se estão corretos.

O pedaço do código para verificar a autenticação será este:

```
<?php
if ($PHP_AUTH_USER = "") { #Se for nulo, forçar o pedido de autenticação
Header("WWW-Authenticate: Basic realm=\"Acesso Restrito\"");
Header("HTTP/1.0 401 Unauthorized");
echo("Mensagem que será exibida se o usuário pressionar o botão cancelar ou entrar com uma senha inválida três vezes consecutivas.");
} else {
(...) #Continuação do código para validar as informações entradas.
}
?>
```

Como já foi dito, o método de autenticação mais eficaz é o do servidor web, mas neste caso o PHP apenas consegue ver o nome do usuário,

já autenticado. Não iremos tratar em detalhes o mecanismo de autenticação do Apache. Para mais informações, consulte o site www.apache.org.

Com o exemplo 1 e 2, já é possível escrever o script proposto. Tente fazê-lo e se houver qualquer dúvida, sinta-se a vontade para enviar um e-mail para valdirleite@ig.com.br que tentarei ajudá-lo da melhor maneira.

Dicas e códigos úteis

Se você seguiu este tutorial desde o início, já tem informações suficientes para escrever pequenas aplicações em PHP. Vamos ver algumas dicas e códigos que podem ser úteis em seu dia-a-dia.

a) - Como enviar um arquivo (upload) pelo browser, usando um código PHP.

Os browsers Netscape e Internet Explorer (versões 3 ou superiores) possuem um tipo de campo de formulário que permite o envio de arquivos de máquina do usuário para o servidor. Isto é muito útil para, por exemplo, receber curriculuns, fotos e qualquer outro tipo de arquivo, sem a necessidade de liberação de um acesso por FTP.

Vamos ver como fazer isto:

O FORMULÁRIO HTML:

```
<form action=upload.php3 method=post ENCTYPE="multipart/form-data">
<input type=file name=file><br><br>
<input type=submit value="Enviar">
</form>
```

Este código exibirá na página um campo com um botão que, ao ser clicado, abre o caixa de diálogo de seleção de arquivo. Escolhido o arquivo, basta clicar no botão enviar, que o resto fica por conta do script upload.php3

Upload.php3
O que este script deve fazer é copiar o arquivo temporário criado pelo navegador para a pasta do cliente.

```
<?php
if((!$file) or ($file=='none') or ($file_name=="")) {
   echo("<html><body bgcolor=white>");
   echo("Não foi possível enviar o arquivo!</p>");
      echo("<p align=center><a
href=javascript:history.back();>");
   echo("<img src=../img/volta.gif border=0></a>");
      echo("</body></html>");
      exit;
} else {
      $dest = "/clientes/".$REMOTE_USER."/".$file_name;
      if(@exec("cp $file $dest")!=0) {
   echo("<html><body bgcolor=white>");
         echo("Não foi possível copiar o arquivo!<br>\n");
   echo("</body></html>");
         exit;
      }
}
header("Location: ./sucesso.htm");
exit;
?>
```

As novidades deste script são:
Comando **EXEC**, que executa um comando do sistema operacional. Neste caso, ele tenta copiar o arquivo temporário, cuja referência está na variável $file para a pasta do cliente. O sinal arroba (@) colocado antes dele é para indicar ao interpretador que não exiba mensagens de erro relativos ao comando. Levamos em consideração que exista um diretório /clientes/nome_do_cliente, onde para pegar o nome_do_cliente usamos a variável $PHP_AUTH_USER, como vimos no início deste artigo.

b) - Trabalhando com Datas

- Função para verificar se uma data está correta:

```
checkdate($mes, $dia, $ano);
```

- Função para saber o dia da semana de uma data específica

```
date("D", mktime(0,0,0,$mes,$dia,$ano));
```

Exemplo:

date("D", mktime(0,0,0,2,12,2000)) #resultará "Sat"

c) - Consistindo um CPF

```
<?php
   function consistecpf($valor) {
$digito1 = 0;
$digito2 = 0;
$cpf = trim($valor);

if (strlen($cpf) < 11) {
return false;
   } else {
for ($i=1; $i<10; $i++) {
$digito1 += strval(substr($cpf,$i-1,1))*$i;
$digito2 += strval(substr($cpf,$i,1))*$i;
   }
return(((($digito1%11)*10)+($digito2%11))==strval(substr($cpf,-2));
}
?>
```

d) - FTP e HTTP

Algumas vezes pode ser útil que nosso script execute um outro script ou então transfira um arquivo para outro servidor. Para isso, podemos executar comandos HTTP e FTP de dentro do PHP. Veja os exemplos:

HTTP: Podemos fazer, dentro do script PHP, uma chamada a outro script ou programa CGI hospedado em outro servidor. Isto é muito útil quando queremos consultar algum dado em um servidor remoto, ou até mesmo para abrir uma página, usando o protocolo HTTP. Para isso, basta chamar a função Header("location: pagina.htm") para redirecionar para uma página específica ou então o usar o comando abaixo para executar um CGI passando parâmetros via URL:

Header("location: http://server/cgi/script.pl?p=1&t=".$param),

Onde $param é uma variável que pode vir de uma consulta a banco de dados ou mesmo de um formulário.

Outra função **HTTP** importante é o uso de "cookies" para gravar alguma informação no browser de quem estiver visitando sua página. Para gravar um "cookie", usamos a função setcookie(), como mostrado abaixo:

setcookie("Visitou","Sim",time()+3600);

O comando acima gravará um cookie chamado "Visitou" com o valor "sim", com apenas uma hora de duração. Note que o 3600 é o número de segundos além do horário atual que o cookie deve ficar ativo. Se no lugar de 3600, usássemos 36000, o cookie seria ativo por 10 horas.

FTP: A sequencia de tarefas para uso do protocolo FTP é Conectar ao Servidor, Identificar-se (Login e Senha), Enviar/Buscar arquivo(s), Desconectar. Os comandos para cada uma destas tarefas são:

```
$conn = ftp_connect();
$log = ftp_login($conn, 'login', 'pass');
ftp_put($con, 'arquivo_remoto', 'arquivo_local', FTP_ASCII/FTP_BINARY);
ftp_quit($conn);
```

e) - Tratamento de Erros

O PHP tem um esquema especial de "debugging" (tratamento e verificação de erros), que é acessado por uma porta TCP, que permite acompanhar a execução dos scripts e ver quaisquer erros que estiverem acontecendo.

Além disso o tratamento de erros pode ser feito no próprio script, conforme explicado abaixo:

O PHP possui 4 níveis de erros e avisos, que são:
1 - Erros de normais de Funções
2 - Avisos Normais
4 - Erro de interpretação
8 - Avisos que você pode ignorar, mas que podem causar danos à execução normal do script.

O padrão do PHP é o nível 7 (1 + 2 + 4), mas este nível pode ser alterado tanto no arquivo de configuração quanto em tempo de execução, chamando a função error_reporting($nivel) com o nível desejado.
Se usarmos o valor 0 (zero) nenhum aviso ou mensagem de erro será gerada em tempo de execução. Neste caso, podemos usar uma

variável especial ($php_errormsg) que conterá o último erro gerado pelo script, para possamos criar rotinas específicas para tratamento de erros. Podemos fazer uma analogia ao comando on error resume next do ASP, técnica muito útil para personalizarmos mensagens de erro para o usuário, entre outras coisas.

Com isso terminamos o Módulo 4 e a primeira série do Tutorial sobre o PHP. Estamos preparando o Módulo 5, que dará início à segunda série, com uma abordagem mais avançada e com muitas novidades. Espero ter ajudado!

Até lá.