PHP - Módulo 3: Funções, Strings e Bancos de Dados

Por Valdir Dias

Depois do script que envia e-mail, vamos fazer um outro que guarde as informações de um formulário HTML em um banco de dados.

Vamos usar o MySQL como banco de dados. Sugiro que você dê uma olhada no site http://www.tcx.se e leia um pouco sobre este servidor SQL. Vale a pena, pois ele é muito bom.

Funções
As funções no PHP não diferem muito das outras linguagens. Algumas características das funções:
• Devem ser declaradas antes de serem usadas.
• Podem receber parâmetros por valor ou por referência.
• Podem ter quantidade variável de parâmetros (Apenas a partir da versão 4).
• Os parâmetros podem ser declarados com um valor default.
• Uma vez definida, uma função não poderá ser "redefinida".

Alguns exemplos de funções:

```
/*
Esta função retorna TRUE ou FALSE, dependendo da validade ou não do e-mail informado.
*/
function verifica_email($email){
    if strpos ($email, "@") = 0 {
      return false;
    } else {
      return true;
    }
}

/*
Neste exemplo calculamos o valor líquido, tendo o valor bruto e o desconto a ser aplicado. Se o desconto não for informado, utilizaremos
10% como padrão.
*/
function valor_liquido($valor_bruto, $desconto = 10) {
      return ($valor_bruto - ($valor_bruto * $desconto/100));
}
```

Os dois exemplos acima receberam seus parâmetros por valor. Isso significa que as alterações de variáveis realizadas dentro da função só terão efeito no contexto da função, e estas mudanças não refletirão no resto do script. Em alguns casos pode ser interessante que os valores dos parâmetros sejam alterados pela função, e que seus novos valores reflitam no script como um todo. Para conseguir isto, usamos a técnica de passagem de parâmetro por referência. Vamos ver um exemplo:

```
function completaURL(&$v_URL) {
    $v_URL .= "http://".$v_URL;
}

$URL = "www.ibestmasters.com.br";
completaURL($URL);

echo "A URL completa fica assim: $URL\n";
```

Daqui a pouco vamos ver um pouco mais sobre funções.

Strings
No módulo 2, foram vistas algumas operações com string. Vamos completar com estas, que são muito úteis no uso com banco de dados e para formatar a saída HTML:

• ***addslashes(str):*** Esta função insere uma barra antes dos caracteres ', ", \ e NULL. Deve ser usada para formatar valores que serão inseridos em um banco de dados, pois estes caracteres, em especial o primeiro, podem causar erros no momento da inserção

Exemplo:

```
$sql = "insert into dados (produto, descricao) values ('".addslashes($prod) . ", '".addslashes($descr). "')";
```

Usando a função addslashes teremos certeza que a variável $sql estará formatada de acordo com a especificação da cláusula INSERT

INTO. Se alguma das variáveis ($prod ou $descr) possuir acima, a função cuida de inserir uma barra (\) antes deles, para que a sentença seja entendida pelo banco.

• ***htmlspecialchars(string):*** Função muito útil, pois "traduz" alguns caracteres nos seus equivalentes em HTML, conforme a tabela abaixo

Caracter	htmlspecialchars()
&	& amp;
"	& quot;
<	& lt;
>	& gt;

• ***nl2br(string):*** Converte os caracteres CRLF, que indicam quebra de linha, em
, que também indica quebra de linha, na linguagem HMTL.

• ***rawurlencode(string):*** Formata uma string de acordo com a especificação RFC1738, que padroniza as URLs. Se quiser saber mais sobre as RFCs, veja em http://www.rfc.org. Basicamente, o que esta função faz é substituir os caracteres não alfanuméricos em seus correspondentes hexadecimais, visando o uso em URLs.

Exemplo:

```
$palavra = "açafrão ou tâmara";
$url = "http://server/pesquisa.php3?p=".rawurlencode($palavra);
```

Esta função é muito útil quando queremos passar parâmetros via URL, como mostra o exemplo acima. No script do próximo módulo voltaremos a usá-la.

• ***ucwords (string):*** Converte os primeiros caracteres de strings em maiúsculo.

Exemplo:

```
$nome = ucwords("valdir henrique dias leite");
echo($nome); //Esta linha exibirá Valdir Henrique Dias Leite
```

Acesso à banco de dados

Como foi dito na apresentação do PHP, o acesso à banco de dados é um dos pontos fortes desta linguagem. Ele possui acesso nativo a ADABAS, ORACLE, SYBASE, SQL SERVER, DBASE, INFORMIX, mSQL, MySQL, POSTGRESQL, além de suportar ODBC, fazendo com que o PHP possa trabalhar praticamente com todos os bancos de dados existentes.

Neste módulo vamos ver apenas as apenas as funções relativas ao banco MySQL, pois esta dupla PHP/MySQL está sendo preferida por uma boa parte dos desenvolvedores, particularmente no ambiente Linux/Apache.

O MySQL é um servidor SQL e portanto devemos seguir alguns procedimentos e regras para acesso aos seus dados. Se você está acostumado com o Oracle ou SQL Server não terá dificuldades, mas se você usa somente bancos de dados do tipo Access ou DBF, poderá ter dificuldades em entender o mecanismo usado pelo MySQL, mas vou tentar ser o mais didático possível. E aguarde, pois estou preparando um tutorial apenas sobre servidores SQL...

A primeira regra é ter um banco de dados cadastrado e um usuário com acesso à este banco de dados. Vale lembrar que o MySQL não é um banco de dados, e sim um servidor de dados. Tenha isto em mente para entender o exemplo.

Digamos que temos um banco de dados Clientes com o usuário admin e senha admin. O primeiro passo é "logar" ao servidor. Para isso usamos o comando mysql_connect e informamos o servidor, login (usuário) e senha. Veja abaixo:

```
$conn = mysql_connect ("localhost" , "admin" , "admin");
```

Este comando abrirá uma conexão com o MySQL da máquina local (localhost), usando o usuário admin cuja senha também é admin. Uma refrência a esta conexão será gravada na variável $conn.

Depois de conectados ao servidor, devemos conectar ao banco de dados propriamente dito, usando o comando mysql_select_db, que precisa de dois parâmetros: O nome do banco de dados e a conexão. Caso a conexão não seja informada, ele tentará usar a última criada. Em nossos exemplos, iremos sempre informar os dois parâmetros.

```
$db = mysql_select_db("clientes", $conn);
```

Neste ponto já temos uma conexão com o servidor e já criamos um link com o banco de dados. Agora podemos enviar os comandos SQL

de desejarmos. Se você não souber SQL, volte a este site em algumas semanas que você vai encontrar um tutorial de SQL.

Agora segue nosso exemplo prático: Vamos usar o script do módulo passado e alterá-lo de modo que os dados digitados no formulário sejam gravados no banco de dados clientes antes de enviar o e-mail.

```
<?php

$erro = "";

# Verificar se o campo NOME está vazio.
if ($nome == "") {
      erro .= "Digite seu Nome\n";
}

# Verificar a quantidade de caracteres no campo TELEFONE.
if ((strlen($telefone) > 8) or (strlen($telefone) < 7)) {
      $erro .= "O número do telefone deve ter 7 ou 8 caracteres\n";
}

# Testar vamor do campo E-mail, verificando o caracter "@"
if strpos ($email, "@") = 0 {
      $erro .= "O e-mail digitado não é válido\n";
}

# Cabeçalho de resposta.
echo("\n");
echo("<center>\n");

if ($erro == "") {
      ;$conn = mysql_connect("localhost" , "admin" , "admin");
      $db = mysql_select_db("clientes", $conn);
      $sql = mysql_query("insert into dados (nome, email, telefone)
values ('".addslashes($nome). "','".addslashes($email)."',
'".addslashes($email)."')" or die("Não foi possível atualizar a tabela");
      mysql_connect($conn);
      mail("valdirleite@ig.com.br", "Dados do Formulário"," Nome: $nome\n
E-mail: $email\n Telefone: $telefone\n","From: $email\nDate: $date\n");
      echo("Obrigado por enviar este formulário!\n");
} else {
      echo("Não foi possível enviar o formulário!
Verifique as mensagens abaixo:

\n");
      echo("<b>$erro </b>\n");
      echo("<br><br><a href=form.htm>Voltar\n");
}
      echo("</center>");
```

As linhas em negrito foram acrescentadas para que o script possa gravar os dados do formulário na tabela DADOS do banco de dados CLIENTES, que está no mesmo servidor onde está sendo rodado o script (LOCALHOST).

Este é o procedimento padrão para usar servidores de banco de dados com o PHP:

- Conectar ao servidor
- Abrir o banco de dados (um servidor SQL pode ter mais de um banco de dados)
- Enviar os comandos SQL
- Fechar o banco de dados
- Desconectar ao servidor

A novidade deste exemplo fica por conta do comando die que finaliza o script caso a função que o precede não possa ser executada.

Veja no próximo módulo

No próximo módulo, vamos discutir um pouco mais sobre banco de dados e ver funções avançadas do PHP, como tratamento de erros, upload de arquivos, autenticação de usuários além de funções de FTP e HTTP.