Guia de Consulta Rápida

Integrando PHP 5 com MySQL

2ª edição

Juliano Niederauer

Novatec

Primeira edição: ISBN: 978-85-7522-066-7
Segunda edição: ISBN: 978-85-7522-174-7

Novatec Editora Ltda.
Rua Luis Antônio dos Santos 110
02018-012 São Paulo – Brasil
Tel.: +55 11 2959-6529
Fax: +55 11 2950-8869
E-mail: novatec@**novatec**.com.br
Site: www.**novatec**.com.br

Introdução ao Guia

Objetivos

Este guia foi criado para auxiliar os programadores que desejam integrar a linguagem PHP com o banco de dados MySQL. Para isso, serão apresentados os procedimentos de instalação, configuração e integração dessas tecnologias, incluindo a configuração do servidor web (ex: Apache).

Primeiramente, você aprenderá a instalar e configurar os softwares necessários para executar os exemplos contidos neste guia. Em seguida, veremos noções básicas de programação PHP, não com o intuito de detalhar toda a linguagem, mas de mostrar os recursos que serão úteis na integração básica com o MySQL.

Logo após, serão mostradas noções básicas do MySQL, onde você aprenderá a acessar o servidor MySQL e realizar as operações desejadas sobre um banco de dados nesse servidor.

Finalizando, será apresentada a integração completa entre o PHP e o MySQL, com exemplos simples e práticos de como gerenciar um banco de dados de produtos, com inclusões, alterações, exclusões e consultas. No final do guia, você encontrará as referências dos comandos do PHP e do MySQL.

O que é PHP?

PHP é uma das linguagens de programação mais utilizadas na web para a criação de páginas dinâmicas. Suas principais características são:

- **Gratuito e com código aberto**: o arquivo de instalação pode ser obtido gratuitamente no site `http://www.php.net`. Além disso, o PHP é um software com código-fonte aberto.
- **Embutido no HTML**: o HTML e o PHP podem ser misturados. Você pode começar a escrever em PHP, de repente escrever um trecho em HTML, depois voltar para o PHP, e assim por diante.
- **Baseado no servidor**: quando você acessa uma página PHP através do seu navegador, todo o código PHP é executado no servidor, e somente o resultado final é exibido para o usuário. Portanto, o navegador exibe a página já processada, sem consumir recursos de seu computador.
- **Bancos de dados:** diversos bancos de dados são suportados pelo PHP, ou seja, o PHP possui código que executa funções de cada um. Entre eles temos MySQL, PostgreSQL, SQLite, InterBase, Oracle, SQL Server, entre outros que oferecem suporte à linguagem SQL (Structured Query Language).
- **Portabilidade**: pode-se executar o PHP no Linux, Unix ou Windows NT.

Veja a seguir um exemplo de uma página web que contém programação PHP. Em vez de nomeá-la como `exemplo.html`, ela será nomeada como `exemplo.php`, para que o navegador possa identificar que trata-se de uma página com programação.

exemplo.php

```
<html>
<head>
<title>Exemplo</title>
</head>
<body>
<?php
    echo "Este é um script PHP!";
?>
</body>
</html>
```

O programa apresentado contém a estrutura padrão de uma página HTML, com as tags `<html>`, `<body>`, `<head>` e `<title>`. No corpo da página há um trecho de código PHP, onde foi utilizado o comando `echo` para exibir na tela o texto "Este é um script PHP!".

O que é MySQL?

MySQL é um SGBD (Sistema Gerenciador de Bancos de Dados) relacional que utiliza a linguagem padrão SQL (Structured Query Language), e é largamente utilizado em aplicações para a Internet. É o mais popular entre os bancos de dados com código-fonte aberto. Há mais de cinco milhões de instalações do MySQL no mundo todo, inclusive em sites com alto volume de dados e de tráfego, como Associated Press, Google, NASA, Sabre Holdings e Suzuki.

O MySQL á uma alternativa atrativa porque, mesmo possuindo uma tecnologia complexa de banco de dados, seu custo é bastante baixo. Tem como destaque suas características de velocidade, escalabilidade e confiabilidade, o que vem fazendo com que ele seja adotado por departamentos de TI (Tecnologia da Informação), desenvolvedores web e vendedores de pacotes de softwares.

A seguir são listadas algumas vantagens do MySQL:

- número ilimitado de utilização por usuários simultâneos;
- capacidade de manipulação de tabelas com mais de 50.000.000 de registros;
- alta velocidade de execução de comandos;
- fácil e eficiente controle de privilégios de usuários.

Portanto, o MySQL e o PHP formam uma excelente dupla para o desenvolvimento de páginas web dinâmicas, tanto para websites pequenos como para grandes portais.

Comparando MySQL e PostgreSQL

São dois excelentes SGBDs gratuitos que podem ser usados com o PHP. O MySQL está disponível sob a GPL (licença pública GNU), além de possuir uma licença convencional, para quem não quiser estar limitado aos termos da GPL. Já o PostgreSQL está disponível sob a flexível licença BSD.

O MySQL é mais utilizado no desenvolvimento de aplicações onde a velocidade é importante, enquanto que o PostgreSQL se destaca por ser mais robusto e possuir muito mais recursos. Esses recursos tornam o PostgreSQL um pouco mais qualificado do que o MySQL.

Nas últimas versões do MySQL, os desenvolvedores acrescentaram diversos recursos que já existiam no PostgreSQL, como transações (confirmação ou cancelamento de operações realizadas), triggers (gatilhos), stored procedures (procedimentos armazenados), views (visões), lock de linha (bloqueio em nível de linha) e constraints (cláusulas de integridade).

Em alguns aspectos, o PostgreSQL é um pouco mais eficiente. Por exemplo, possui um sofisticado mecanismo de bloqueio (MVCC), suporta tamanhos ilimitados de linhas, bancos de dados e tabelas (até 16TB), aceita vários tipos de subconsultas e conta com um bom mecanismo de failsafe (segurança contra falhas).

Portanto, a vantagem do MySQL é a velocidade de acesso. Para bases de dados muito grandes, o MySQL faz um acesso mais rápido que o PostgreSQL. Se seu site possuir um banco de dados muito grande, vale a pena usar o MySQL. Para base de dados menores, não há diferença na velocidade de acesso entre os dois SGBDs.

Download e instalação dos softwares

Para testar os exemplos apresentados neste guia, você deverá instalar os seguintes softwares em sua máquina:

- **PHP**: a linguagem de programação, disponível em `http://www.php.net`.
- **MySQL**: o Sistema Gerenciador de Bancos de Dados, disponível em `http://www.mysql.com`.
- **Apache**: é o servidor web, disponível em `http://httpd.apache.org`. O Apache é o servidor web mais indicado, pois o PHP roda como um módulo nativo dele.

É importante lembrar que o PHP é uma linguagem voltada para a web, portanto deve haver um servidor web, que receba as solicitações das páginas, faça o processamento pelo PHP, e retorne ao navegador (browser) um resultado.

Instalando o PHP e o Apache

Se você estiver utilizando os serviços de um provedor de hospedagem, provavelmente não precisará se preocupar com a instalação e configuração do PHP e de outros softwares. Caso contrário, faça download do PHP em `http://www.php.net/`.

- Acessando a seção "downloads", você poderá obter sempre a última versão da linguagem. Na versão para Linux, o PHP precisará ser compilado em seu sistema operacional. Para obter mais detalhes, consulte o arquivo `install.txt` que acompanha a distribuição. Na versão Windows, a distribuição está disponível em um arquivo compactado ZIP, que já contém os arquivos binários. Basta descompactá-lo em algum diretório do seu computador (ex: `C:\PHP`).
- Dependendo da versão do PHP que você instalar e das bibliotecas que você for usar, será necessário copiar alguns arquivos DLLs do diretório de instalação do PHP para o diretório de sistema do Windows (ex: `C:\Windows\System32`). Para saber quais são eles, consulte o arquivo `install.txt` que acompanha o PHP. Por exemplo, para usar o MySQL há uma biblioteca chamada `libmysql.dll`, que deve estar nesse diretório.
- No diretório do PHP, você também irá encontrar um arquivo chamado `php.ini-dist`. Trata-se do arquivo de configuração do PHP. Ele deve ser renomeado para `php.ini` e copiado para o diretório do Windows (ex: `C:\Windows`).

Conforme vimos no tópico anterior, para poder acessar seus programas pelo navegador você precisará também do servidor web (Apache), que pode ser obtido em `http://httpd.apache.org`.

Faça o download da versão mais atual do Apache para o seu sistema operacional. No caso do Linux, você pode obter o arquivo indicado como "Unix Source", que contém o código-fonte a ser compilado no Linux. No caso do Windows, você pode obter o arquivo com a indicação "Win32 Binary". Em seguida, basta executar esse arquivo e seguir as instruções para que o Apache seja instalado em sua máquina.

Configurando o Apache para scripts PHP

Após instalar o PHP e o Apache, você deve configurar o Apache para que ele aceite os programas em PHP (extensão `.php`). Essa configuração é necessária para que o servidor web reconheça quando foi feita uma chamada para um script PHP, e possa ativar o preprocessador da linguagem (ex: `php.exe`) para executá-lo. Após a execução do script, apenas o resultado final (HTML) é enviado para o navegador do usuário.

Para fazer essa configuração, você precisará acrescentar algumas linhas no arquivo `httpd.conf` (localizado no diretório `conf` do Apache). As linhas a serem acrescentadas estão especificadas no arquivo `install`.

txt, que acompanha a distribuição do PHP. Elas variam de acordo com a versão dos softwares que você está usando e com a forma que o PHP deve funcionar, mas geralmente são as seguintes:

```
LoadModule php5_module c:/php/php5apache.dll
AddModule mod_php5.c
AddType application/x-httpd-php .php
```

Se em vez de instalar o PHP como um módulo do Apache, você preferir instalá-lo como um binário (CGI), as linhas a serem incluída serão diferentes:

```
ScriptAlias /php/ "c:/php/"
AddType application/x-httpd-php .php
Action application/x-httpd-php "/php/php.exe"
```

O diretório em negrito, mostrado no exemplo anterior, é o diretório onde o PHP está instalado em sua máquina (caso você tenha escolhido outro, troque esse nome pelo nome de seu diretório).

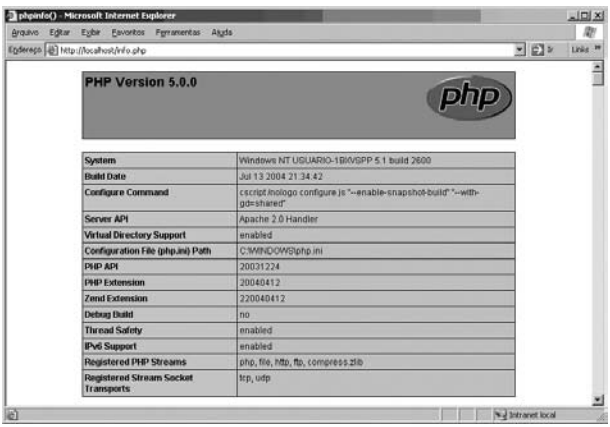
Testando o funcionamento do PHP no Apache

Para testar se o PHP está funcionando corretamente no Apache, crie um arquivo chamado info.php e coloque-o na pasta principal (document root) do seu servidor web. Neste arquivo, coloque o seguinte código PHP:

```
<?php
phpinfo();
?>
```

Para iniciar o Apache, selecione a opção "*Start Apache*", criado no menu Iniciar do Windows, no momento da instalação.

Em seguida, abra o seu navegador e digite o endereço http://localhost/info.php. Esse script listará diversas informações sobre a versão do PHP instalada em sua máquina, como mostra a figura seguir.



Download e instalação do MySQL

Para efetuar o download do MySQL, acesse o site `http://www.mysql.com`, entre na seção de produtos, escolha "*Database Server*" e faça o download do arquivo de instalação para o seu sistema operacional. Se você pretende instalar o MySQL no Linux, uma boa opção é fazer o download do pacote rpm, que é de fácil instalação.

No caso do Windows, ao terminar o download do arquivo, execute-o para inciar a instalação, e siga as instruções que irão aparecer na tela. Durante a instalação, você poderá optar se deseja que o servidor MySQL seja iniciado automaticamente na inicialização do sistema. É bom você marcar essa opção, para não ter que iniciar manualmente o servidor cada vez que a máquina for reiniciada.

Qualquer dúvida, consulte a documentação contida no subdiretório `Docs`.

Habilitando a extensão MySQL no PHP

Depois que você instalou o MySQL em sua máquina, para poder utilizá-lo com o PHP é necessário habilitar a extensão *mysql* (ou *mysqli*) no arquivo `php.ini`. Isso é feito através da seguinte linha:

```
extension=php_mysql.dll
```

Se a sua versão do MySQL for a 4.1 ou superior, você terá que usar a extensão "mysqli":

```
extension=php_mysqli.dll
```

Normalmente essas linhas já existem no seu arquivo `php.ini`, mas estão comentadas com uma vírgula na frente. Nesse caso, basta remover a vírgula. Em seguida, reinicie o servidor web para que as alterações tenham efeito.

Linguagem PHP

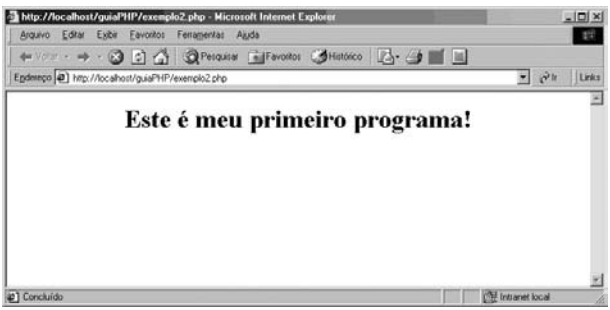
Introdução ao PHP

Um programa PHP pode ser escrito em qualquer editor de texto, como por exemplo o Bloco de Notas (Notepad) do Windows ou o VI do Linux. Um trecho de código PHP deve estar entre as tags `<?php` e `?>`, para que o servidor web reconheça que trata-se de um código de programação e possa chamar o interpretador PHP para executá-lo. Para treinar, abra o editor de texto de sua preferência e digite as quatro linhas a seguir, salvando o arquivo com o nome de `exemplo2.php`.

exemplo2.php

```
<?php
    // Que bacana, estou programando em PHP!!!
    echo  "<h1 align='center'>Este é meu primeiro programa!</h1>";
?>
```

Em seguida, envie o arquivo para o diretório raiz do Apache (document root). Para ver o resultado, basta você acessar pelo navegador o endereço `http://<seu_endereço>/exemplo2.php`, onde você deve substituir *<seu_endereço>* pelo endereço do servidor que está utilizando para executar os progrmas PHP.



A seguir é apresentado o significado de cada uma das linhas que você digitou no programa:

Elemento	Descrição
`<?php`	Indica o início de um trecho de código PHP.
`//`	Linha de comentário. Tudo que vem após estas barras na mesma linha é ignorado pelo PHP.
`echo`	É um dos comandos mais utilizados em PHP. Serve para escrever alguma coisa na tela.
`?>`	Indica o término de um trecho de código PHP.

Se você escolher a opção *Exibir-Código-fonte* em seu navegador, verá que ele não recebe nenhuma linha em PHP. Ele recebe somente código HTML puro. Isso acontece porque o código PHP é processado no servidor, que retorna somente o resultado final para o navegador.

Variáveis

Servem para armazenar dados que podem ser usados em qualquer ponto do programa. Ao contrário de linguagens tradicionais, como C, Pascal e Delphi, no PHP não é necessário fazer declaração de variáveis. Basta atribuir diretamente um valor a ela.

No PHP, as variáveis devem iniciar com o símbolo $. Após esse símbolo deve vir o identificador da variável, que não pode iniciar com um número. Exemplos de variáveis válidas:

```
$joao23
$casal20
$teste450
```

Um recurso interessante do PHP é a interpolação de variáveis, ou seja, a inclusão do valor de uma variável dentro de uma string, como mostra o exemplo a seguir.

exemplo3.php

```
<?php
    $time = "Grêmio";
    $titulo = "Campeão da América";
    echo "O $time é $titulo";
?>
```

Veja que o valor das variáveis `$time` e `$titulo` foi usado dentro da string passada para o comando `echo`. O resultado será:

```
O Grêmio é Campeão da América
```

Constantes

São valores que são predefinidos no início do programa, e que não mudam ao longo de sua execução. Você pode definir suas próprias constantes, utilizando o comando `define`, que possui a seguinte sintaxe:

```
bool define (string nome, misto valor [, bool case_insensitive])
```

O parâmetro *case_insensitive* é um valor lógico (`true` ou `false`) que indica se o PHP deve diferenciar letras maiúsculas e minúsculas. Veja o exemplo a seguir, nomeado como `exemplo4.php`, que mostra como devemos usar as constantes:

exemplo4.php

```
<?php
    define ("meunome","João");
    define ("peso",80);
    echo  "O meu nome é " . meunome;
    echo  "<br>";
    echo  "O meu peso é " . peso . " quilos";
?>
```

Executando esse programa, você terá o seguinte resultado em seu navegador:

```
O meu nome é João
O meu peso é 80 quilos
```

Arrays

As variáveis comuns (escalares) podem armazenar apenas um valor por vez. Um array (vetor) pode armazenar vários valores ao mesmo tempo. Além de possuir um identificador, um array possui índices (que podem ser números ou strings). O índice deve aparecer entre colchetes ([]) logo após o identificador do array. Veja a seguir alguns exemplos de armazenamento em arrays:

```
$vetor[0] = 30;
$vetor[1] = 40;
```

Se não colocarmos o índice do vetor entre colchetes, o PHP irá procurar o último índice utilizado e incrementá-lo, armazenando assim o valor na posição seguinte do array, conforme mostra o exemplo a seguir:

```
$vet[ ] = "Grêmio";
$vet[ ] = "Campeão";
```

Nesse exemplo teremos o valor "Grêmio" armazenado em `$vet[0]` e o valor "Campeão" armazenado em `$vet[1]`.

Até agora só vimos exemplos em que o índice do array é um valor numérico, mas o índice também pode ser um texto, e nesse casos o texto é chamado de chave associativa.

```
$vetor["time"] = "Grêmio";
$vetor["fundacao"] = 1903;
```

Existem também as matrizes, que são arrays multidimensionais. As matrizes podem possuir dois ou mais índices para referenciar uma posição de memória. Por exemplo:

```
$clube ["RS"] ["PortoAlegre"] = "Grêmio";
$clube ["MG"] ["BeloHorizonte"] = "Cruzeiro";
```

Outra forma de criar um array é por meio da função `array` do PHP. Veja o exemplo apresentado a seguir:

exemplo5.php

```
<?php
    $vetor = array (10,50,100,150,200);
    echo $vetor[2] . "<br>";
    $vet = array (1, 2, 3, "nome"=>"Juliano");
    echo $vet[0] . "<br>";
    echo $vet["nome"];
?>
```

Após a execução desse programa os resultados mostrados na tela serão os seguintes:

```
100
1
Juliano
```

Lembre-se de que o array se inicia na posição 0 (zero), por isso apesar de ser o terceiro elemento do array, o 100 foi o primeiro valor mostrado, pois seu índice é 2.

Operadores

Aritméticos

Operador	Operação
+	Adição.
-	Subtração.
*	Multiplicação.
/	Divisão.
%	Resto da divisão.

O PHP possui também outros operadores aritméticos, que atuam em apenas um operando. No PHP também é possível utilizá-lo. A tabela a seguir mostra esses operadores:

Operador	Descrição
-*oper*	Troca o sinal do operando.
++*oper*	Pré-incremento. Primeiro incrementa o valor do operando e depois realiza a operação.
--*oper*	Pré-decremento. Primeiro decrementa o valor do operando e depois realiza a operação.
oper++	Pós-incremento. Primeiro realiza a operação e depois incrementa o operando.
oper--	Pós-decremento. Primeiro realiza a operação e depois decrementa o operando.

Por exemplo, se o objetivo for somente incrementar o valor de uma variável, pode-se simplesmente digitar o nome da variável seguida do operador ++.

Exemplo:

```
$contador++;
```

Binários

Operador	Descrição
~*op1*	Inverte os bits de *op1*.
op1 & *op2*	Operação E (AND) bit a bit.
op1 | *op2*	Operação OU (OR) bit a bit.
op1 ^ *op2*	Operação OU exclusivo (XOR).
op1 >> *n*	Desloca *op1 n* bits à direita.
op1 << *n*	Desloca *op1 n* bits à esquerda.

Comparação

Operador	Descrição
op1 == *op2*	Verdadeiro se *op1* for igual a *op2*.
op1 === *op2*	Verdadeiro se *op1* for igual a *op2* e eles forem do mesmo tipo.
op1 >= *op2*	Verdadeiro se *op1* for maior ou igual a *op2*.
op1 <= *op2*	Verdadeiro se *op1* for menor ou igual a *op2*.
op1 != *op2*	Verdadeiro se *op1* for diferente de *op2*.
op1 !== *op2*	Verdadeiro se *op1* for diferente de *op2* ou eles não forem do mesmo tipo.

op1 <> *op2*	Também serve para representar diferença.
op1 > *op2*	Verdadeiro se *op1* for maior que *op2*.
op1 < *op2*	Verdadeiro se *op1* for menor que *op2*.

Atribuição

Operador	Descrição
op1 = *op2*	*op1* recebe o valor de *op2*.
op1 += *op2*	Equivale a *op1*=*op1*+*op2*.
op1 -= *op2*	Equivale a *op1*=*op1*-*op2*.
op1 *= *op2*	Equivale a *op1*=*op1***op2*.
op1 /= *op2*	Equivale a *op1*=*op1*/*op2*.
op1 .= *op2*	Concatenação: equivale a *op1*=*op1*.*op2*.
op1 %= *op2*	Equivale a *op1*=*op1*%*op2*.
op1 <<= *op2*	Equivale a *op1*=*op1*<<*op2*.
op1 >>= *op2*	Equivale a *op1*=*op1*>>*op2*.
op1 &= *op2*	Equivale a *op1*=*op1*&*op2*.
op1 |= *op2*	Equivale a *op1*=*op1*|*op2*.
op1 ^= *op2*	Equivale a *op1*=*op1*^*op2*.

Lógicos

Operador	Descrição
!*op1*	Verdadeiro se *op1* for falso.
op1 AND *op2*	Verdadeiro se *op1* E *op2* forem verdadeiros.
op1 OR *op2*	Verdadeiro se *op1* OU *op2* forem verdadeiros.
op1 XOR *op2*	Verdadeiro se só *op1* ou só *op2* for verdadeiro.
op1 && *op2*	Verdadeiro se *op1* E *op2* forem verdadeiros.
op1 || *op2*	Verdadeiro se *op1* OU *op2* forem verdadeiros.

A diferença entre os operadores `AND` e `&&`, e também entre os operadores `OR` e `||`, é a precedência dos mesmos na avaliação de expressões. A precedência mais alta é dos operadores `&&` e `||`.

Estruturas de controle em PHP

O uso dessas estruturas é fundamental para realizar decisões lógicas, testar se determinada expressão é verdadeira e repetir um bloco de comandos por um certo número de vezes. Neste guia, veremos os comandos condicionais `if` e `switch`, e os comandos de repetição `while`, `do...while`, `for` e `foreach`.

if

Comando que avalia uma expressão e, dependendo do resultado, é executado um conjunto diferente de instruções. O comando `if` pode possuir como complemento o `elseif` e/ou o `else`. Observe a sintaxe do comando `if`:

```
If ( exp1 )
    {  bloco1 }
elseif  ( exp2 )
    {  bloco2 }
else
    {  bloco3 }
```