Guia de Atualização

MySQL Para MySQLi

Aprenda Passo-a-Passo
Como Atualizar Seus Scripts
de **MySQL** para **MySQLi**

ULTIMATE PHP

Seja Muito Bem-Vindo!

Antes de qualquer coisa, seja muito bem-vindo ao **Guia ULTIMATE PHP de Migração do MySQL Para o MySQLi**!

Neste guia eu vou te mostrar passo-a-passo como migrar do MySQL para o MySQLi.

Mas Por Quê?

É sempre importante perguntarmos para nós mesmos por que estamos fazendo algo.

Eu sempre faço isso, para ter certeza de que estou fazendo algo que é realmente útil e importante.

Então aqui vale a pena perguntar:

Por Que Migrar do MySQL Para o MySQLi?

Bom... eu expliquei isso com vários detalhes neste meu artigo, mas vou dar uma ideia geral aqui também.

A partir do PHP 5.5, a biblioteca MySQL é considerada obsoleta, mas ainda funciona. Porém, é recomendado não utilizá-la mais. Ela, inclusive, já foi removida do PHP na versão 7, que ainda está na versão RC, mas deve ser lançada oficialmente até o final de 2015.

A equipe do PHP sugere migrar para MySQLi ou PDO.

Particularmente, recomendo e prefiro PDO. Mas há um pequeno problema...

PDO requer um pouco de conhecimento sobre Orientação a Objetos (OOP). E eu sei que muitos iniciantes têm bastante dificuldade com isso.

Por isso resolvi montar este pequeno guia, mostrando como migrar para o MySQLi, sem usar Orientação a Objetos.

Mas saiba que o MySQLi também permite usar o padrão OOP.

MySQLi Com e Sem Orientação a Objetos

A extensão MySQLi pode ser usada da forma tradicional, com chamada de funções (estilo procedural), mas também pode ser usada seguindo o padrão de Orientação a Objetos.

Vou focar mais na forma procedural, pois talvez você tenha dificuldade com OOP.

Porém, no final vou mostrar como fazer usando OOP, para você ver a diferença.

Você verá que com OOP é bem mais simples e prático, além de não ser difícil.

Mas calma...

Um passo de cada vez...

Por Que Não É Tão Simples Como Dizem

Tem muita gente por aí dizendo que é só colocar o "i" nas chamadas das funções e pronto.

Não é bem assim.

Várias funções do MySQLi possuem parâmetros a mais, ordem diferente etc.

As funções `mysql_*` não exigia o parâmetro de identificação da conexão (valor retornado por `mysql_connect`). Elas sempre buscavam uma conexão ativa.

Com mysqli a coisa é diferente...

É **fundamental** informar o identificador da conexão.

Dessa forma, podemos ter diversas conexões no mesmo script, sem risco de conflitos ou funcionamento inesperado.

Veremos isso com mais detalhes, na prática, logo logo.

Banco de Dados de Exemplo

Vamos criar um simples banco de dados para nossos testes.

Como nosso foco aqui é apenas MySQL/MySQLi, uma simples tabela já resolve, sem qualquer relacionamento, nem chave estrangeira.

Vamos usar uma tabela de funcionários com esta estrutura:

```
CREATE TABLE funcionarios(
    id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    nome VARCHAR(50) NOT NULL,
    email VARCHAR(80) NOT NULL,
    salario DECIMAL(10,2) NOT NULL,
    nascimento DATE NOT NULL,
    PRIMARY KEY (id)
) COLLATE=utf8_unicode_ci;
```

Vamos inserir alguns dados:

```
INSERT INTO funcionarios(nome, email, salario, nascimento) VALUES
('Ana', 'ana@ana.com', 1400.00, '1993-10-16'),
('Daniela', 'dani@dani.com', 1380.50, '1996-05-15'),
('João', 'joao@joao.com', 1400.00, '1995-11-10'),
('Julia', 'julia@julia.com', 1200.95, '1997-03-22'),
('Sabrina', 'sabrina@sabrina.com', 1800.57, '1990-12-03'),
('Paulo', 'paulo@paulo.com', 1750.43, '1991-06-25');
```

Coisa simples. Sem segredo.

Exibindo os Dados Usando MySQL

Vamos criar um simples script para exibir todos os dados, usando funções `mysql_*`.

```
// dados de conexão
$host = 'localhost';
$user = 'root';
$pass = '';
$db   = 'guia_mysql';

// conecta com o banco de dados
$con = mysql_connect($host, $user, $pass);

// seleciona o banco de dados
mysql_select_db($db);

// efetua a consulta
$sql = "SELECT * FROM funcionarios ORDER BY nome";
$res = mysql_query($sql);

// total de resultados
$total = mysql_num_rows($res);

echo "<p>Total de Resultados: " . $total . "</p>";

// loop pelos resultados
while ($f = mysql_fetch_array($res))
{
    echo "<p>" . $f['nome'] . " | " .
         $f['email'] . " | " .
         $f['salario'] . " | " .
         date('d/m/Y', strtotime($f['nascimento'])) . "</p>";
}
```

```
// fecha a conexão
mysql_close();
```

O script é super simples. Apenas um `SELECT` que lista os registros da tabela, junto ao número total de registros.

Exibindo os Dados Usando MySQLi Procedural

Vamos ver agora como rescrever o script anterior usando MySQLi procedural, ou seja, usando funções.

Vou colocar o código aqui e falo sobre os principais detalhes logo em seguida.

```
$host = 'localhost';
$user = 'root';
$pass = '';
$db   = 'guia_mysql';

// conexão e seleção do banco de dados
$con = mysqlI_connect($host, $user, $pass, $db);

// executa a consulta
$sql = "SELECT * FROM funcionarios ORDER BY nome";
$res = mysqli_query($con, $sql);

// conta o número de registros
$total = mysqli_num_rows($res);

echo "<p>Total de Resultados: " . $total . "</p>";

// loop pelos registros
while ($f = mysqli_fetch_array($res))
{
    echo "<p>" . $f['nome'] . " | " .
         $f['email'] . " | " .
         $f['salario'] . " | " .
         date('d/m/Y', strtotime($f['nascimento'])) . "</p>";
}

// fecha a conexão
mysqli_close($con);
```

Podemos notar três diferenças principais:

1. `mysqli_connect` permite conectar ao MySQL e já selecionar o banco de dados. Até existe a função `mysqli_select_db`, mas você nem precisa usá-la, caso utilize apenas um banco de dados. Você só precisará dela se estiver conectado a um banco e precisar mudar para outro;
2. O primeiro parâmetro de `mysqli_query` **não é** a consulta SQL, como era com `mysql_query`. Agora é necessário passar o identificador da conexão (retornado por `mysqli_connect`) e, depois, a string com a consulta SQL;
3. `mysqli_close` exige o parâmetro que identifica a conexão. Por outro lado, para `mysql_close` ele era opcional.

Como você pode ver, só colocar o "i" não resolve 100%. Algumas pequenas modificações ainda são necessárias.

Mas percebe como é simples? Não é nada de outro mundo, super complexo ou assustador. :)

Vou colocar aqui uma tabela comparando as principais funções do `mysql` com as do `mysqli`.

Perceba as diferenças dos parâmetros.

Muitas funções do `mysql` possuem o parâmetro `$link_identifier` opcional (entre colchetes). Porém, suas correspondentes no `mysqli` **exigem** o identificador de conexão.

Só pra você entender melhor, `resource` é o tipo do retorno de `mysql_connect` e `mysql_query`. `mysqli` é o identificador da conexão, retornado por `mysqli_connect` e `mysqli_result` é o retorno de `mysqli_query`. Com isso você entende melhor quais são os parâmetros que cada função exige.

MySQL	MySQLi
`mysql_affected_rows([resource $link_identifier])`	`mysqli_affected_rows(mysqli $link)`
`mysql_close([resource $link_identifier])`	`mysqli_close(mysqli $link)`
`mysql_connect(string $server, string $username, string $password)`	`mysqli_connect(string $host, string $username, string $passwd, string $dbname)`
`mysql_db_query`	Não há correspondente. Use `mysqli_select_db` seguido de `mysqli_query`
`mysql_errno([resource $link_identifier])`	`mysqli_errno(mysqli $link)`
`mysql_error([resource $link_identifier])`	`mysqli_error(mysqli $link)`

MySQL	MySQLi
`mysql_fetch_array(resource $result)`	`mysqli_fetch_array(mysqli_result $result)`
`mysql_fetch_assoc(resource $result)`	`mysqli_fetch_assoc(mysqli_result $result)`
`mysql_fetch_object(resource $result)`	`mysqli_fetch_object(mysqli_result $result)`
`mysql_fetch_row(resource $result)`	`mysqli_fetch_row(mysqli_result $result)`
`mysql_insert_id([resource $link_identifier])`	`mysqli_insert_id(mysqli $link)`
`mysql_num_rows(resource $result)`	`mysqli_num_rows(mysqli_result $result)`
`mysql_query(string $query [, resource $link_identifier ] )`	`mysqli_query(mysqli $link, string $query)`
`mysql_real_escape_string(string $unescaped_string [, resource $link_identifier ] )`	`mysqli_real_escape_string(mysqli $link, string $escapestr )`
`mysql_select_db(string $database_name [, resource $link_identifier ] )`	`mysqli_select_db(mysqli $link, string $dbname)`

Caso tenha alguma dúvida ou queira ver as documentações completas dessas funções, vou deixar aqui os links para as listas de funções do MySQL e do MySQLi.

- Lista de Funções MySQL
- Lista de Funções MySQLi

Exibindo os Dados Usando MySQLi Orientado a Objetos

Como você deve ter percebido, é um tanto chato ter que ficar passando todos esses parâmetros.

E é aí que o MySQLi Orientado a Objeto mostra suas vantagens.

Veja o código anterior escrito no padrão Orientado a Objeto:

```
$host = 'localhost';
$user = 'root';
$pass = '';
```

```php
$db   = 'guia_mysql';

// conexão e seleção do banco de dados
$mysqli = new mysqli($host, $user, $pass, $db);

// executa a consulta
$sql = "SELECT * FROM funcionarios ORDER BY nome";
$res = $mysqli->query($sql);

// total de registros
$total = $res->num_rows;

echo "<p>Total de Resultados: " . $total . "</p>";

// loop pelos registros
while ($f = $res->fetch_array())
{
    echo "<p>" . $f['nome'] . " | " .
         $f['email'] . " | " .
         $f['salario'] . " | " .
         date('d/m/Y', strtotime($f['nascimento'])) . "</p>";
}

// fecha a conexão
$mysqli->close();
```

Vamos aos pontos importantes:

1. Em vez de `mysqli_connect`, instanciei a classe `mysqli`. Dessa forma, `$mysqli` agora é um objeto;
2. Em vez de usar `mysqli_query`, chamei o método `query` no objeto `$mysqli`. Aqui já economizei a passagem de um parâmetro;
3. Em vez de usar `mysqli_num_rows`, acessei o valor da propriedade `$num_rows` do objeto `$res`, que é do tipo `mysqli_result`, retornado pelo método `query`. Note que `$num_rows` é uma propriedade e não um método. Por isso não há parênteses;
4. Em vez de usar `mysqli_fetch_array`, usei o método `fetch_array` do objeto `$res`;
5. Para fechar a conexão, usei o método `close` do objeto `$mysqli` em vez da função `mysqli_close`.

Percebe como o código fica menor, mais limpo e mais fácil de entender?

Se você ainda não estudou Orientação a Objetos, não se preocupe. Continue usando a forma procedural, mas não deixe de estudar OOP. Vai facilitar muito a sua vida.

Vale a Pena Usar PDO?

Muita gente me pergunta se deve migrar para MySQLi ou PDO.

Como eu disse no começo, sempre recomendo PDO.

Mas tem muita gente que não conhece muito bem classes e objetos, então prefere ficar com MySQLi Procedural.

Porém eu realmente prefiro que você use PDO, por diversas razões. Falei sobre diversas delas neste artigo.

Mas, como sempre, vá com calma. Um passo por vez.

Vamos Melhorar o Mundo Juntos?

Tem muita gente ainda usando funções `mysql_*` por achar que é difícil atualizar para `mysqli_*`.

E você acabou de ver como é simples!

O que acha de me ajudar a espalhar a notícia?

Vamos melhorar o mundo, com scripts atualizados e confiáveis.

Imagine se um dia um cliente pede para você dar manutenção em um sistema... aí você descobre que, dentre vários problemas, o sistema também usa funções `mysql_*`...

Mais coisa pra você fazer e, muitas vezes, sem poder cobrar a mais.

Então evite que isso aconteça!

Compartilhe este guia com seus amigos e colegas.

Assim eles também vão ver como é simples e fácil atualizar para mysqli.

Para Onde Ir Agora

Sei que parece bastante coisa, mas, como eu disse, se você olhar com calma e tentar entender, verá que não é difícil.

Eu entendo bem sua dificuldade. Também fui iniciante, tive bastante dificuldade.

Hoje ajudo muita gente em fóruns, principalmente o iMasters, e sei exatamente quais são as principais dúvidas dos iniciantes em PHP.

E foi por isso que criei o **Curso ULTIMATE PHP**.

Nesse curso, eu explico tudo o que você precisa saber para conhecer bem o PHP, desde conceitos básicos até intermediários. E até alguns um pouquinho avançados, mas sem complicar sua vida.

O principal objetivo é ensinar a você **os fundamentos** sobre PHP.

Tem muitos cursos de PHP por aí, mas eu vejo um **Enorme Problema** na maioria deles:

Eles querem ensinar para você a melhor ferramenta pra isso, a ferramenta ideal para aquilo...

Eles ensinam o framework X, que usa padrão MVC... aí você mal sabe o que é MVC, como a estrutura funciona... você fica imaginando que tem um mago ali dentro fazendo tudo funcionar magicamente.

No final das contas, você não sabe o que está fazendo. Só sabe que a ferramenta resolve o seu problema, mesmo que não da melhor forma possível.

E o dia em que você não tiver essa ferramenta em mãos?

Aí é a Hora da Verdade!

Nessas horas que os programadores bons de verdade se destacam!

Quem realmente sabe como as coisas funcionam de verdade!

Por isso eu insisto em ensinar **os fundamentos**.

Lógico que vou mostrar algumas ferramentas para agilizar e evitar perda de tempo. Mas primeiro vou mostrar para você o que elas fazem e como elas fazem.

Por isso o **Curso ULTIMATE PHP** é diferente. Você aprende **de verdade**!

Você não será só um simples programador como qualquer outro.

Você Vai Ser Um Programador Destacado, Reconhecido e Valorizado!

Então não perca tempo com cursos e apostilas que só fazem você perder tempo, conheça agora o **Curso ULTIMATE PHP**!

Um Pouco Sobre Mim

Meu nome é Roberto Beraldo, mais conhecido apenas por **Beraldo**.

Eu conheci o PHP lá pelos idos de 2005 ou 2006. Eu conhecia HTML e precisava fazer um simples formulário de contato para um site que eu tinha feito. Foi aí que trombei com o PHP. E desde então não larguei mais! :)

Aprendi bastante do PHP lendo a documentação e analisando dúvidas de outras pessoas, em alguns fóruns, como o iMasters, do qual, aliás, sou moderador até hoje.

Tentei ler alguns livros sobre PHP, mas nunca gostei muito da didática deles. Outro problema era que, naquela época, o PHP 5 tinha acabado de ser lançado. Muitas práticas antigas já eram consideradas obsoletas na versão 5. E sem contar que o PHP 5 trouxe muitas novidades. Ou seja, os livros estavam **todos obsoletos**!

Mais tarde, em 2008, iniciei o curso de **Bacharelado em Ciência da Computação**, na **Universidade Federal do Paraná (UFPR)**. Foram anos bem corridos, cheios de trabalhos, noites de pouco sono e muito estudo.

Em 2012, finalmente terminei a graduação. Valeu a pena! Aprendi muita coisa!

Nesse meio tempo, resolvi criar o meu blog, o Blog do Beraldo. Lá escrevi (e ainda escrevo) diversos tutoriais, artigos e dicas sobre diversas áreas da Computação, mas principalmente sobre PHP.

Durante todo esse tempo, nunca deixei de estudar e trabalhar com PHP. Mais do que isso, estive sempre participando do Fórum iMasters, ajudando o pessoal principalmente de PHP. Ou seja, sei bem quais são as principais dificuldades dos iniciantes.

E assim o Blog do Beraldo foi crescendo. Muitos visitantes me mandavam mensagens sugerindo que eu criasse um curso completo de PHP, pois gostavam muito dos artigos que eu postava no Blog e também no iMasters.

Eu sempre agradecia o reconhecimento, mas dizia que não tinha tempo para criar um curso completo.

Depois de algum tempo, pensei:

"Se tantos me pedem para criar um curso, por que não criar?"

Eu sabia **exatamente** as principais dúvidas dos iniciantes e intermediários em PHP. Ou seja, eu poderia ensinar tudo isso de forma fácil, enfatizando os pontos mais críticos.

E assim surgiu o **Curso ULTIMATE PHP**!

Tenho certeza de que você vai aprender MUITO com o curso.

Se você quer realmente se tornar um excelente programador, pare de gastar tempo e dinheiro com cursos errados.

Aprenda PHP De Verdade E Sem Enrolação

Ah, e outra coisa, se ainda não compartilhou este guia, aproveite a chance de ensinar algo legal e fácil pros seus amigos!