**Inserindo produtos no banco de dados**

No capítulo anterior, vimos como criar uma página PHP, criar cabeçalhos e rodapés para evitar repetição de código e como ler parâmetros. O que falta agora é inserirmos esses parâmetros no banco de dados. Como já instalamos o MySQL (o XAMPP instalou para nós), vamos fazer uso dele.

Nosso primeiro passo é levantar o servidor do MySQL e conectar nele. Se formos na página de configuração do XAMPP (http://localhost/xampp), vemos que o MySQL está desativado. Em cada sistema operacional, isso é feito de uma maneira diferente.

Com o MySQL no ar, podemos ir no PHPMyAdmin, que é um excelente administrador de MySql, que vem no XAMPP. Basta clicar no link que existe na página de status do XAMPP, e estamos nele.

Em Create Databases, vamos criar um novo banco chamado "loja". Com o banco criado, vamos criar a tabela que armazenará a lista de produtos. Na aba SQL, digite:

* 1. create table produtos (id integer auto\_increment primary key, nome varchar(255), preco decimal(10,2))

Clicando em Go, a tabela será criada. Aqui não entraremos a fundo em SQL. Se quiser entender um pouco mais, faça o curso online dedicado ao assunto.

Vamos agora inserir alguns produtos para ver o MySql funcionando. Novamente, na aba SQL:

* 1. insert into produtos values (1, 'Carro', 20000);
  2. insert into produtos values (2, 'Motocicleta', 10000);

Podemos forçar o auto incremento, não passando o ID:

* 1. insert into produtos (nome, preco) values ('Bicicleta', 300);

Se clicarmos em Browse, vemos a lista de todos os produtos adicionados. No nosso caso, 3 produtos.

Agora que temos um banco de dados no nosso MySql, vamos fazer o PHP inserir algo nele. De volta ao adiciona-produto.php, vamos escrever a SQL nele:

* 1. <?php include("cabecalho.php"); ?>
  2. <?php
  3. $nome = $\_GET["nome"];
  4. $preco = $\_GET["preco"];
  5. $query = "insert into produtos (nome, preco) values (?????)";
  6. ?>
  7. Produto <?= $nome; ?>, <?= $preco; ?> adicionado com sucesso!
  8. <?php include("rodape.php"); ?>

Veja que guardamos a query em uma variável, mas precisamos trocar os "??" pelos valores passados pelo usuário. Geralmente, é comum que desenvolvedores concatenem esses valores na mão. Algo como:

* 1. $query = "insert into produtos (nome, preco) values ('" . $nome . "', '" . $preco . "')";

Repare que o ponto (.) soma duas strings. Mas o problema disso é que é muito trabalhoso. O PHP nos dá uma maneira mais fácil de misturar strings e variáveis. Chamamos isso de interpolação, e para isso, basta usar o símbolo { e }:

* 1. $query = "insert into produtos (nome, preco) values ('{$nome}', {$preco})";

O PHP vai automaticamente pegar o valor da varíavel nome e preco e concatenar na string. Fica mais fácil de ler. Mas repare que isso só funciona se a string for declarada com aspas duplas.

Vamos continuar, e mandar essa SQL para o banco de dados. Em primeiro lugar, conectamos no banco de dados, passando endereço, usuário e senha:

* 1. $conexao = mysqli\_connect('localhost', 'root', '', 'loja');

Em seguida, executamos a query:

* 1. mysqli\_query($conexao, $query);

Por fim, fechamos a conexão:

* 1. mysqli\_close($conexao);

Nosso código completo ficou:

* 1. $query = "insert into produtos (nome, preco) values ('{$nome}', {$preco})";
  2. $conexao = mysqli\_connect('localhost', 'root', '', 'loja');
  3. mysqli\_query($conexao, $query);
  4. mysqli\_close($conexao);

Se testarmos agora a inserção de um lápis pelo formulário, isso deve funcionar. Basta conferir no MySql para ver que a linha foi adicionada. Veja só como acessar o MySql pelo PHP é fácil. Basta apenas conhecer apenas alguns métodos da biblioteca.

Mas veja só o que acontece se não passarmos o preço no formulário. Apesar da aplicação web falar que a inserção foi feita, se formos no MySql, vemos que nenhuma linha foi adicionada. Isso acontece porque o MySql deu um erro (afinal a SQL, sem o preço, fica errada!), mas nós não tratamos esse erro.

Precisamos então tratar esse erro, e fazemos isso na função de mysqli\_query. Ela devolve true em caso de sucesso, ou false em caso de falha:

* 1. if(mysqli\_query($conexao, $query)) {
  2. // funcionou
  3. } else {
  4. // nao deu certo
  5. }

Vamos agora misturar HTML com PHP, e mostrar a mensagem de sucesso apenas se deu sucesso:

* 1. <?php
  2. if(mysqli\_query($conexao, $query)) {
  3. ?>
  4. <p class="alert-success">Produto <?= $nome; ?>, <?= $preco; ?> adicionado com sucesso!</p>
  5. <?php
  6. } else {
  7. ?>
  8. <p class="alert-danger">O produto <? = $nome; ?> não foi adicionado</p>
  9. <?php
  10. }
  11. ?>

Pronto. Agora nossa aplicação retorna um erro amigável caso a inserção falhe.

Um detalhe curioso do PHP é que ele fecha a conexão do MySql automaticamente. Ou seja, não precisamos colocar a linha mysqli\_close(). Nesse caso, quando a última linha do seu script for executada, aí sim ele a fechará.