PHP – PDO – Conexão com o BD

No exercício referente a esta aula, vocês terão todos os arquivos utilizados - de criação do banco de dados, as instruções para isso, e os arquivos de front-end. Vamos começar criando uma pasta denominada "**classes**", em que incluiremos um novo arquivo chamado Categoria.php.

Basicamente, neste arquivo será criada uma classe para representar cada entidade, cujas necessidades estarão sob responsabilidade de cada uma delas em nosso sistema de estoque.

Começaremos com "Categoria", da seguinte maneira:

<?php

class Categoria

{

}

?>

Agora, vamos dar uma olhada em nosso front-end para entendermos o que precisaremos fazer nesta parte de categorias. Neste projeto, não usaremos Apache, IIS, ou qualquer outro servidor web. O PHP possui um servidor web embutido, que poderá ser utilizado para facilitar nosso trabalho.

Abriremos o Windows PowerShell, mas este pode ser substituído por qualquer outro terminal de sua preferência. Navegaremos até a pasta de nosso projeto, digitando os seguintes comandos intercalados com "Enter":

cd ..

cd ..

cd PDO

ls

O último comando nos lista todos os arquivos existentes no diretório para verficiarmos se estamos no local correto, ou seja, na raiz do nosso projeto. Para que o servidor web funcione, usaremos o comando php -S 127.0.0.1:8888 passando como parâmetros o IP e a porta para o qual queremos que o servidor suba, no caso, a mesma máquina que estamos utilizando, ou o IP local.

Com "Enter", o prompt fica bloqueado para uso, sendo que poderemos parar o processamento a qualquer momento com "Ctrl + C", e será impresso o log de tudo que está acontecendo no servidor web.

Vamos ver se deu certo?

Abriremos o navegador e digitaremos 127.0.0.1:8888, e nosso sistema é carregado conforme esperávamos:



Clicando na seção "Categorias", há um botão que serve para criarmos novas categorias, uma já existente ("Livros") sendo listada na tabela, um botão de "Editar" e outro de "Excluir". Entendemos que precisaremos fazer a lista de categorias, a ser apresentada ao usuário nesta página. Vamos voltar ao código de Categoria.php.

Precisaremos fazer uma busca no banco de dados para obtermos as categorias do nosso projeto, portanto usaremos uma variável $query, que será uma *string*, e nela usaremos o SELECT para a consulta no banco de dados.

Para executarmos a *query*, usaremos o PDO, com o qual precisaremos criar uma conexão a partir da variável $conexao, e fazendo uma nova instância da classe PDO passando o parâmetro que é a *string*de conexão, a única informação que variará de acordo com o banco de dados em uso.

Tecnicamente, esta é a única mudança no caso de trocarmos de banco de dados, desde que o novo siga a mesma estrutura de tabelas e não esteja utilizando nenhuma função específica.

No nosso caso, utilizaremos o MySQL, então a *string* de conexão será criada com base nele. Em seguida, informaremos o *host* onde se encontra o servidor MySQL, e o *dbname* do projeto.

O segundo parâmetro será o usuário de conexão, e o terceiro, a senha:

<?php

class Categoria

{

public function listar()

{

$query = "SELECT id, nome FROM categorias";

$conexao = new PDO('mysql:host=127.0.0.1;dbname=estoque', 'root', 'alura');

}

}

?>

Legal, desta forma fazemos a conexão com o banco de dados, e agora precisaremos justamente executar a *query*, e para isso usaremos o método query() do PDO, passando como parâmetro a *string* $query. Queremos que isso seja retornado para uma variável $resultado. O trecho do código ficará assim:

$query = "SELECT id, nome FROM categorias";

$conexao = new PDO('mysql:host=127.0.0.1;dbname=estoque', 'root', 'alura');

$resultado = $conexao->query($query);

O método query() retorna um objeto chamado ***PDOStatement***, com várias informações da nossa execução, incluindo o retorno, as linhas retornadas. Para obtê-las, usaremos o método fetchAll(), que retornará um *array* com todas as linhas que queremos da consulta.

Colocaremos tudo isso em uma lista a ser retornada:

$query = "SELECT id, nome FROM categorias";

$conexao = new PDO('mysql:host=127.0.0.1;dbname=estoque', 'root', 'alura');

$resultado = $conexao->query($query);

$lista = $resultado->fetchAll();

return $lista;

Vamos conferir seu funcionamento indo à página de Categorias e instanciando a nova classe para testarmos. Abriremos o arquivo categorias.php, em que consta puro HTML, exceto pelo require\_once do cabeçalho e do rodapé.

Começaremos incluindo outro require\_once para chamarmos a classe Categoria. Em seguida, abriremos um bloco de PHP para a chamada para o método que acabamos de criar. E queremos que esta informação vá para uma variável chamada lista:

<?php require\_once 'classes/Categoria.php' ?>

<?php

$categoria = new Categoria();

$lista = $categoria->listar();

?>

Para verificar se tudo funciona como esperado, voltaremos ao navegador para atualizá-la. Receberemos uma mensagem de erro, pois o PHP, por padrão, precisa que a extensão do PDO esteja ativada no servidor, o que poderemos fazer no arquivo de configuração do PHP, o php.ini.

Para abri-lo e habilitarmos sua extensão de PDO, acessaremos "C: > php > php.ini" (Windows), e abriremos o arquivo no próprio bloco de notas. Veremos que ele é cheio de configurações do PHP, e o que queremos é ativar a configuração do PDO.

Com "Ctrl + F", buscaremos "PDO", o que nos mostra diversas extensões, cada uma para um banco de dados diferente. No nosso caso, como estamos usando o MySQL, habilitaremos a extensão correspondente descomentando-a, isto é, retirando o ponto e vírgula (**;**) que a antecede, deixando-a assim:

extension=php\_pdo\_mysql.dll

Feito isso, salvaremos o arquivo e reiniciaremos o servidor que deixamos rodando no PowerShell, justamente para que se usem as configurações atualizadas.

Então, voltaremos ao PowerShell, em que usaremos "Ctrl + C" para pausar a execução. Em seguida, o executaremos novamente. Vamos ver se isto resolve nosso problema?

Sim! Agora ele está funcionando direitinho. No entanto, nossas categorias ainda não aparecem, então vamos verificar se realmente deu certo debugando a variável para ver se ela está sendo populada corretamente, por meio do método print\_r() do PHP. Em categorias.php, teremos:

<?php require\_once 'classes/Categoria.php' ?>

<?php

$categoria = new Categoria();

$lista = $categoria->listar();

print\_r($lista);

?>

O print\_r() é equivalente ao toString(), e tenta imprimir na tela qualquer tipo de variável, não importa se é um *array*, inteiro, ou booleano. Ele dará um jeito de convertê-la para uma *string* e imprimir.

Vamos executar o código e ver o que acontece. No navegador, é indicado que trata-se de um *array* e nos passa todas as informações... Mas está meio ruim de visualizar assim, não?

Para facilitar a visualização do retorno do print\_r(), utilizaremos o echo e a tag <pre> do HTML:

<?php require\_once 'classes/Categoria.php' ?>

<?php

$categoria = new Categoria();

$lista = $categoria->listar();

echo "<pre>";

print\_r($lista);

echo "</pre>";

?>

Isso faz com que se acertem as quebras de linha, ajudando na compreensão de que há quatro categorias com os respectivos *id*s e nomes. Para cada categoria, existe uma chave com o nome do campo de dados (o *id*), e uma chave para o mesmo valor, porém numérica (0), possibilitando que se use a que preferir.

Se forem usar o valor numérico, atentem-se para que não haja risco de mudar a ordem dos valores depois, pois esta informação irá variar ali.

Então, poderemos deletar o debug pois já sabemos que está tudo sendo rodado como esperado. Agora, vamos jogar estas informações à nossa lista com o foreach, navegando pelo *array* linha a linha e capturando as informações para mostrá-las na tela.

Antes da tag <tr> e após <tbody> em categorias.php, digitaremos:

<?php foreach ($lista as $linha): ?>

<tr>

<!-- ... -->

</tr>

<?php endforeach ?>

Para melhorar a organização e legibilidade do nosso código quando misturamos o PHP com um arquivo HTML, costuma-se usar o padrão de escrita do PHP em uma mesma linha, usando dois pontos (**:**), como mostrado acima.

E uma vez que abrimos esta tag com dois pontos, ela será fechada com endforeach. Com isso, simplesmente imprimiremos os valores do id e nome da categoria:

<?php foreach ($lista as $linha): ?>

<tr>

<td><a href="/categorias-detalhe.php" class="btn btn-link"><?php echo $linha['id'] ?></a><td>

<td><a href="/categorias-detalhe.php" class="btn btn-link"><?php echo $linha['nome'] ?></a><td>

<td><a href="/categorias-editar.php" class="btn btn-info">Editar</a><td>

<td><a href="#" class="btn btn-danger">Excluir</a><td>

</tr>

<?php endforeach ?>

Vamos rodar o código e verificar o que acontece.

Tudo está funcionando corretamente! Agora, temos o foreach com os nossos dados, ou seja, as categorias "Livros", "Jogos", "Filmes" e "Revistas".

Até agora, criamos a classe Categoria.php, com uma $query para fazermos o SELECT. Implementamos a primeira $conexao com o PDO, em que executamos a *query*, cujo resultado foi retornado para outra página.

Em categorias.php fizemos a instância da Categoria, a chamada no método listar(), a qual verificamos com o debug para nos averiguar sobre seu funcionamento, e fizemos um loop na tabela para imprimir todos os valores na tela!