**DAO - Data Access Object**

Quando queremos adicionar, remover, listar, ou fazer qualquer outra operação que envolva **acesso aos dados** da nossa classe Produto, vamos direto para o arquivo banco-produto.php. Isso acontece porque sabemos que essa responsabilidade está centralizada nesse arquivo.

Mesmo sem conhecer orientação a objetos, desde o início do curso escolhemos centralizar todas essas funções que faziam operações no banco de dados. Qual a vantagem? A grande ideia dessa boa prática é que facilita muito a manutenção, já que o desenvolvedor saberá onde procurar *queries* relacionadas a produtos, categorias, etc. Cada uma em um arquivo distinto.

Além disso, evitamos duplicação de código, já que todas essas funções podem ser reutilizadas em qualquer lugar do nosso projeto, sempre que necessário. Se em algum momento você decidir mudar de banco de dados, não precisa sair alterando todos os pontos do seu código que fazem acesso ao banco, basta mudar o arquivo que centraliza esse comportamento.

Legal, não é? Mas ainda não temos uma solução ideal, já que essas funções estão todas soltas em um arquivo. Agora que estamos trabalhando com orientação a objetos, nada mais justo do que criar uma classe para **encapsular** esse trabalho.

Uma instância dessa classe será um **objeto** responsável por **acessar** os **dados** de um produto, por isso é uma prática comum chamar esse tipo de classe de **DAO** (**D**ata **A**ccess **O**bject). Esse é um dos mais famosos padrões de projeto (*design patterns*) utilizados pelas linguagens orientadas a objeto, como PHP, Java e C#.

Vamos criar o nosso ProdutoDAO.php dentro da pasta **class**. Ele pode ficar assim:

class ProdutoDAO {

function listaProdutos($conexao) {

$produtos = array();

$resultado = mysqli\_query($conexao, "select p.\*,c.nome as categoria\_nome

from produtos as p join categorias as c on c.id=p.categoria\_id");

while($produto\_atual = mysqli\_fetch\_assoc($resultado)) {

$nome = $produto\_atual['nome'];

$preco = $produto\_atual['preco'];

$descricao = $produto\_atual['descricao'];

$categoria = new Categoria;

$categoria->setNome($produto\_atual['categoria\_nome']);

$usado = $produto\_atual['usado'];

$produto = new Produto($nome, $preco, $descricao, $categoria, $usado);

$produto->setId($produto\_atual['id']);

array\_push($produtos, $produto);

}

return $produtos;

}

function insereProduto($conexao, Produto $produto) {

$query = "insert into produtos (nome, preco, descricao, categoria\_id, usado)

values ('{$produto->getNome()}', {$produto->getPreco()},

'{$produto->getDescricao()}', {$produto->getCategoria()->getId()},

{$produto->isUsado()})";

return mysqli\_query($conexao, $query);

}

function alteraProduto($conexao, Produto $produto) {

$query = "update produtos set nome = '{$produto->getNome()}',

preco = {$produto->getPreco()}, descricao = '{$produto->getDescricao()}',

categoria\_id= {$produto->getCategoria()->getId()},

usado = {$produto->isUsado()} where id = '{$produto->getId()}'";

return mysqli\_query($conexao, $query);

}

function buscaProduto($conexao, $id) {

$query = "select \* from produtos where id = {$id}";

$resultado = mysqli\_query($conexao, $query);

return mysqli\_fetch\_assoc($resultado);

}

function removeProduto($conexao, $id) {

$query = "delete from produtos where id = {$id}";

return mysqli\_query($conexao, $query);

}

}

Veja que **todos** os métodos dessa classe precisam receber uma conexão para fazer o acesso ao banco. Já que nosso DAO **depende** de uma conexão para funcionar, podemos usar um pouco mais da orientação a objetos, forçar a passagem de uma conexão como argumento no momento de criarmos o DAO. Como fazer isso? É simples, basta criar um construtor, assim como fizemos para criar um produto já passando os seus atributos.

A classe ficará assim:

class ProdutoDAO {

private $conexao;

function \_\_construct($conexao) {

$this->conexao = $conexao;

}

function listaProdutos() {

$produtos = array();

$resultado = mysqli\_query($this->conexao, "select p.\*,c.nome as categoria\_nome

from produtos as p join categorias as c on c.id=p.categoria\_id");

while($produto\_atual = mysqli\_fetch\_assoc($resultado)) {

$nome = $produto\_atual['nome'];

$preco = $produto\_atual['preco'];

$descricao = $produto\_atual['descricao'];

$categoria = new Categoria;

$categoria->setNome($produto\_atual['categoria\_nome']);

$usado = $produto\_atual['usado'];

$produto = new Produto($nome, $preco, $descricao, $categoria, $usado);

$produto->setId($produto\_atual['id']);

array\_push($produtos, $produto);

}

return $produtos;

}

function insereProduto(Produto $produto) {

$query = "insert into produtos (nome, preco, descricao, categoria\_id, usado)

values ('{$produto->getNome()}', {$produto->getPreco()},

'{$produto->getDescricao()}', {$produto->getCategoria()->getId()},

{$produto->isUsado()})";

return mysqli\_query($this->conexao, $query);

}

function alteraProduto(Produto $produto) {

$query = "update produtos set nome = '{$produto->getNome()}',

preco = {$produto->getPreco()}, descricao = '{$produto->getDescricao()}',

categoria\_id= {$produto->getCategoria()->getId()},

usado = {$produto->isUsado()} where id = '{$produto->getId()}'";

return mysqli\_query($this->conexao, $query);

}

function buscaProduto($id) {

$query = "select \* from produtos where id = {$id}";

$resultado = mysqli\_query($this->conexao, $query);

return mysqli\_fetch\_assoc($resultado);

}

function removeProduto($id) {

$query = "delete from produtos where id = {$id}";

return mysqli\_query($this->conexao, $query);

}

}

Excelente. Sempre que criarmos um ProdutoDAO, passaremos uma conexão como parâmetro. Essa conexão será mantida como uma propriedade da classe, que pode ser utilizada em todos os seus outros métodos. Como o parâmetro não é opcional, ao esquecer de passar uma conexão como parâmetro, o PHP logo nos lembrará com um erro.

Vamos alterar os lugares que chamavam os métodos de acesso ao banco diretamente, para que no lugar disso passem a utilizar o DAO. O arquivo adiciona-produto.php, por exemplo, tem um if assim:

if (insereProduto($conexao, $produto)) { ?>

Agora, com o DAO, ficará dessa forma:

$produtoDao = new ProdutoDAO($conexao);

if($produtoDao->insereProduto($produto)) { ?>

Lembrando que todos os arquivos que instanciarem o ProdutoDAO precisam importar o arquivo conecta.php, pois eles precisam passar uma conexão para o mesmo.