**Evoluindo a listagem de produtos**

**Evoluindo nossas listagens**

Agora que estamos trabalhando com objetos, podemos modificar um pouco nossas listagens de produto e categoria. Podemos começar pela função listaProdutos, do arquivo banco-produto.php. Note que estamos utilizando o mysqli\_fetch\_assoc para criar um ::array:: com os dados de cada produto:

* 1. function listaProdutos($conexao)
  2. {
  3. $produtos = array();
  4. $resultado = mysqli\_query($conexao, "select \* from produtos");
  5. while($produto = mysqli\_fetch\_assoc($resultado)) {
  6. array\_push($produtos, $produto);
  7. }
  8. return $produtos;
  9. }

No lugar de trabalhar com esses ::array::s diretamente na página, podemos a cada iteração criar um objetoProduto com todas as informações necessárias. Para fazer isso, basta modificar o método listaProdutos como a seguir:

* 1. function listaProdutos($conexao)
  2. {
  3. $produtos = array();
  4. $resultado = mysqli\_query($conexao, "select \* from produtos");
  5. while($array = mysqli\_fetch\_assoc($resultado)) {
  6. $produto = new Produto();
  7. $produto->id = $array['id'];
  8. $produto->nome = $array['nome'];
  9. $produto->descricao = $array['descricao'];
  10. $produto->preco = $array['preco'];
  11. $produto->usado = $array['usado'];
  12. array\_push($produtos, $produto);
  13. }
  14. return $produtos;
  15. }

Feito isso, precisaremos mudar a forma que estamos lidando com esses valores no arquivo produto-lista.php. No lugar de fazer:

* 1. <table class="table table-striped table-bordered">
  2. <?php
  3. $produtos = listaProdutos($conexao);
  4. foreach ($produtos as $produto) :
  5. ?>
  6. <tr>
  7. <td><?= $produto['nome'] ?></td>
  8. <td><?= $produto['preco'] ?></td>
  9. <!-- continuação do código omitido -->

Passaremos a utilizar a sintaxe de objetos, com o ->. Repare:

* 1. <table class="table table-striped table-bordered">
  2. <?php
  3. $produtos = listaProdutos($conexao);
  4. foreach ($produtos as $produto) :
  5. ?>
  6. <tr>
  7. <td><?= $produto->nome ?></td>
  8. <td><?= $produto->preco ?></td>
  9. <!-- continuação do código omitido -->

Excelente. Podemos fazer o mesmo com a listagem de categorias, que por enquanto ainda usa um ::array::. A alteração é bem simples, na verdade, exatamente o que fizemos agora com o produto. Basta abrir o arquivo banco-categoria.php e editar a função listaCategorias:

* 1. function listaCategorias($conexao)
  2. {
  3. $categorias = array();
  4. $query = "select \* from categorias";
  5. $resultado = mysqli\_query($conexao, $query);
  6. while($array = mysqli\_fetch\_assoc($resultado)) {
  7. $categoria = new Categoria();
  8. $categoria->id = $array['id'];
  9. $categoria->nome = $array['nome'];
  10. array\_push($categorias, $categoria);
  11. }
  12. return $categorias;
  13. }

Agora no arquivo produto-formulario.php podemos alterar os lugares que usam o ::array:: de categorias para a forma orientada a objetos:

* 1. <td>Categoria</td>
  2. <td>
  3. <select name="categoria\_id">
  4. <?php
  5. $categorias = listaCategorias($conexao);
  6. foreach($categorias as $categoria) : ?>
  7. <option value="<?=$categoria->id?>">
  8. <?=$categoria->nome?>
  9. </option>
  10. <?php endforeach ?>
  11. </select>
  12. </td>

Ótimo, mas só estamos esquecendo de uma coisa. Categoria é um elemento importante em nossa aplicação, portanto, pode (e deve) ser representado como um objeto. Vamos criar o arquivo categoria.php com a classeCategoria e suas duas propriedades.

* 1. <?php
  2. class Categoria
  3. {
  4. public $id;
  5. public $nome;
  6. }

Excelente, após essas alterações podemos executar o código e tudo continuará funcionando conforme esperado.

**Relacionamento de objetos**

Podemos notar que a categoria até então era controlada por um id referenciado no produto. Porém, queremos manter nosso sistema orientado a objetos. Dessa forma, criamos também, além da classe Produto, a classeCategoria. Essa classe vai representar uma Categoria em nosso sistema.

Agora que temos duas classes, Produto e Categoria, podemos e devemos criar um relacionamento entre elas. Sempre que criarmos um objeto Produto, poderemos colocar um outro objeto na propriedade $categoria. Nosso código de instância de Produto ficará assim:

* 1. // Propriedades comuns
  2. $produto->nome = "Livro da Casa do Código";
  3. $produto->preco = 29.90;
  4. $produto->descricao = "livro de tecnologia";
  5. $produto->usado = true;
  6. // Novo objeto Categoria
  7. $produto->categoria = new Categoria();

Assim, usaremos a propriedade de classe $categoria como uma referência a outro objeto, criando um relacionamento entre o objeto Produto e Categoria.