

Migrations

Capítulo 4 - Migrations

Na aula passada criamos uma *lista fixa* para nossos produtos de camiseta. Como os produtos variam com o tempo (compras e vendas, estoque, preço, etc), esta não é uma boa prática. Precisamos armazenar os produtos e suas informações no banco de dados.

Para tais tarefas de infraestrutura utilizaremos o *Rake*, ao invés do *Rails*. Criamos o banco de dados com o comando, no Terminal:

```
rake db:create
```

O banco de dados padrão do *Rails* é o *SQLite*, como visto no Gemfile. Dentro do banco precisa haver uma tabela com todos os produtos. Primeiramente criamos um modelo que armazena as informações de cada produto:

```
rails generate model produto nome:string descricao:text quantidade:integer preco:decimal
```

O nome do produto é do tipo *string*, a descrição do tipo *text* e assim por diante e sempre usando letras minúsculas. Serão criados tanto uma Classe de código *Ruby* que representa o produto quanto o código para gerar a tabela. Podemos visualizar este no arquivo dentro do diretório "db/migrate":

```
class CreateProdutos < ActiveRecord::Migration
  def change
    create_table :produtos do |t|
      t.string :nome
      t.text :descricao
      t.integer :quantidade
      t.decimal :preco

      t.timestamps null: false
    end
  end
end
```

Só foi gerado tal arquivo, A tabela ainda não foi criada fisicamente. Para criarmos a tabela no *SQLite* executamos uma **migração**. Todos os arquivos dentro de "db/migrate" serão rodados na ordem em que foram criados primeiramente. Então fazemos no Terminal:

```
rake db:migrate
```

Para termos certeza de que a tabela foi criada, vamos nos logar à base de dados. O comando permite isso sem precisarmos digitar nome de usuário e senha:

```
rails dbconsole
...
```

```
sqlite>
```

Para selecionar todos os produtos podemos fazer `select * from produtos`, mas ainda não adicionamos nenhum, logo não haverá retorno. Mas, para encontrarmos a tabela fazemos:

```
sqlite> .tables
produtos          schema_migrations
```

De fato, a tabela foi criada. Se quisermos ver sua estrutura:

```
sqlite>.schema produtos
CREATE TABLE "produtos" ("id" INTEGER ... NOT NULL);
```

Por padrão, para todo modelo, o *Rails* cria um *id* obrigatoriamente, assim como as datas de criação e atualização. Também podemos observar os outros campos que pedimos para serem criados (nome, descrição, etc). Para sairmos do banco de dados, teclamos "Ctrl+D" ou "Ctrl+C (x2)".

Vamos inserir agora dados de produtos. Para tal devemos entrar no console do *Ruby* com as classes do *Rails* carregadas, o *irb*:

```
rails console
Loading development environment (Rails 4.2.1)
irb(main):001:0>
```

Começemos a testar. Vamos criar um novo produto por meio de uma variável que será instanciada usando a Classe "Produto", a qual foi criada quando geramos o modelo:

```
irb(main):001:0> bigbang = Produto.new
=> #<Produto id: nil, nome: nil, descricao: nil, quantidade: nil, preco: nil, created_at: nil, updated_at: nil>
```

Precisamos preencher seus dados:

```
irb(main):002:0> bigbang.nome = "Camiseta do Big Bang Theory"

irb(main):003:0> bigbang.descricao = "Camiseta super bacana do bbt"

irb(main):004:0> bigbang.quantidade = 10

irb(main):005:0> bigbang.preco = 70.50
```

Para vermos os dados inputados:

```
irb(main):006:0> bigbang
```

O "bigbang" é uma variável que referencia um produto com todos seus dados que foram manualmente inputados. Falta salvá-los na tabela. Toda classe de modelo de produto gerada é um registro ativo no banco de dados, é uma infraestrutura já

pronta. Tal classe possui uma herança (do *ActiveRecord*) discriminada em "app/models/produto.rb":

```
class Produto < ActiveRecord::Base
end
```

Por causa dessa herança, dentro do Produto nós temos um método chamado *save*. Então, fazendo

```
irb(main):007:0> bigbang.save
(0.1ms)  begin transaction
SQL (0.4ms) ...
(1.5ms)  commit transaction
=> true
```

Salvamos o registro na base, já com as datas de criação, update e *id*.

Inputemos um novo produto. Não faz muito sentido criá-lo sem nome, preço, etc. Diferentemente do que fizemos com o "bigbang", usaremos o método *.create* para já passarmos todos os dados da camiseta de gastronomia:

```
irb(main):008:0> gastronomia = Produto.create nome: "Camiseta de gastronomia", descricao: "Camiseta
```

Dessa forma, além de já termos inserido seus dados, o produto já foi salvo na base. Para vermos todos os produtos inputados, a quantidade destes e seus dados, fazemos:

```
irb(main):009:0> todos = Produto.all    //devolve todos os produtos com todos os dados em forma de array

irb(main):010:0> todos.size    //devolve a quantidade de produtos inputados

irb(main):011:0> todos[0].nome    //devolve o campo "nome" do primeiro produto ([1] para o segundo e
```

Falta agora mostrar na página do navegador os produtos inputados no banco de dados. Para isso, o HTML deve gerar várias linhas dinamicamente na tabela. Vamos excluir os dados antigos e apenas deixar os cabeçalhos:

```
<html>
<body>
<table>
  <thead>
    <tr>
      <th>Nome</th>
      <th>Descrição</th>
      <th>Preço</th>
      <th>Quantidade</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>
      </td>
    </tr>
  </tbody>
</table>
```

```
</body>
</html>
```

Para implementar essas linhas dinamicamente fazemos um laço passando por todos os produtos. Porém estamos em um HTML querendo usar código *Ruby*. Devemos passar a informação que o que será escrito é nessa linguagem. Fazemos isso com `<% %>`. Tudo o que ficar dentro dessa *tag* é código *Ruby*.

```
<html>
<body>
<table>
  <thead>
    <tr>
      <th>Nome</th>
      <th>Descrição</th>
      <th>Preço</th>
      <th>Quantidade</th>
    </tr>
  </thead>
  <tbody>
    <% todos = Produto.all %>
    <tr>
    </tr>
  </tbody>
</table>
</body>
</html>
```

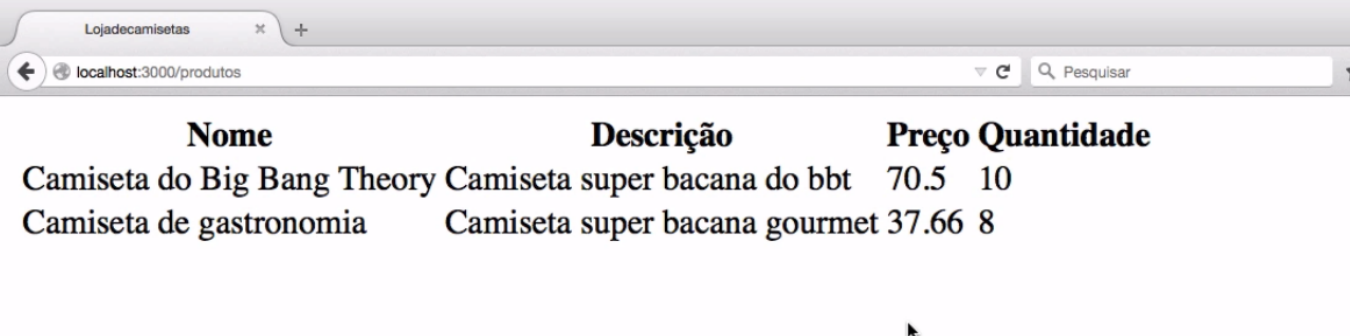
Como está, esse código será ignorado pelo navegador. Ele precisa ser interpretado e fazemos isso mudando a extensão do arquivo para ".erb": "index.html.erb". Agora podemos continuar a implementação. Vamos chamar todos os produtos, cada um em sua coluna. Como vimos, o banco retorna um *array*, esse tipo possui o método `each`. Então fazemos:

```
<html>
<body>
<table>
  <thead>
    <tr>
      <th>Nome</th>
      <th>Descrição</th>
      <th>Preço</th>
      <th>Quantidade</th>
    </tr>
  </thead>
  <tbody>
    <% todos = Produto.all %>
    <% todos.each do |produto| %>
      <tr>
      </tr>
    <% end %>
  </tbody>
</table>
</body>
</html>
```

Agora inserimos as `<td>` :

```
<html>
<body>
<table>
  <thead>
    <tr>
      <th>Nome</th>
      <th>Descrição</th>
      <th>Preço</th>
      <th>Quantidade</th>
    </tr>
  </thead>
  <tbody>
    <% todos = Produto.all %>
    <% todos.each do |produto| %>
      <tr>
        <td><%= produto.nome %></td>
        <td><%= produto.descricao %></td>
        <td><%= produto.preco %></td>
        <td><%= produto.quantidade %></td>
      </tr>
    <% end %>
  </tbody>
</table>
</body>
</html>
```

O `<%` é para que se *processe* o código *Ruby*. Para *mostrar* o código usamos `<%=` . E, rodando no navegador, de fato, funciona:

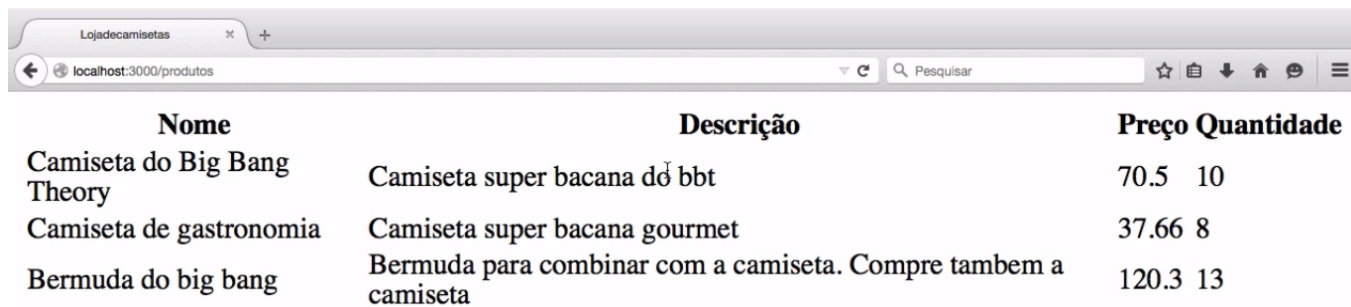


Nome	Descrição	Preço	Quantidade
Camiseta do Big Bang Theory	Camiseta super bacana do bbt	70.5	10
Camiseta de gastronomia	Camiseta super bacana gourmet	37.66	8

Vamos inserir mais uma peça de roupa para termos certeza de que está tudo funcionando:

```
irb(main):012:0> Produto.create nome: "Bermuda do big bang", descricao: "Bermuda para combinar com a"
```

E o produto foi, de fato, adicionado:



Nome	Descrição	Preço Quantidade
Camiseta do Big Bang Theory	Camiseta super bacana do bbt	70.5 10
Camiseta de gastronomia	Camiseta super bacana gourmet	37.66 8
Bermuda do big bang	Bermuda para combinar com a camiseta. Compre tambem a camiseta	120.3 13

Nesta aula vimos muitas coisas:

- Criamos o banco usando `rake db:create` ;
- Usamos `rails generate model` para criar o modelo de produto com nome, descrição, preço e quantidade;
- Para criar a tabela no banco rodamos a migração fazendo `rake db:migrate` ;
- Para acessar o console do banco de dados fizemos `rails dbconsole` ;
- Já para acessar o console do rails (*irb*) fizemos `rails console` . Aqui dentro aprendemos diversos comandos (`.save` , `.create` , etc);
- Modificamos o HTML para que ele passasse a buscar os dados dos produtos direto no banco, deixando-o dinâmico.