

Partials com locals

Capítulo 14 - *Partials* com *Locals*

Já fizemos com que o formulário de criação de produto nos permitisse selecionar o departamento ao qual ele pertence. Porém ainda não conseguimos alterar os dados dos produtos através do site, tanto na hora de criá-lo, propriamente dito, quanto na lista. Da mesma forma que incluímos um link para remover os produtos, vamos criar um para alterá-los.

O *index* chama a tabela que foi salva na *partial*. Lá vamos adicionar mais um botão:

```
<tr>
  <td><%= produto.nome %></td>
  <td><%= produto.descricao %></td>
  <td><%= produto.preco %></td>
  <td><%= button_to "Remover", produto,
    method: :delete,
    class: "btn btn-danger",
    data: { confirm: "Tem certeza que deseja remover #{produto.nome}?" } %>
  <td><%= button_to "Alterar", produto, class: "btn btn-default" %></td>
</tr>
```

Na página com a lista já observamos os botões:

Nome	Descrição	Preço	Remover	Alterar
Bermuda do big bang	Bermuda para combinar com a camiseta. Compre também a camiseta	120.3	Remover	Alterar
Blusa de la			Remover	Alterar
Blusa listrada	blusa listrada nova	3.0	Remover	Alterar
Blusa xadrez	Descrição não tão longa	33.0	Remover	Alterar
Blusa xadrez 2.0	Nova versão da blusa	30.0	Remover	Alterar

Nome	Descrição	Preço	Remover	Alterar
Blusa de la			Remover	Alterar
Blusa listrada	blusa listrada nova	3.0	Remover	Alterar

[Criar novo produto](#)

Ao clicarmos em um deles, nos é mostrada uma página de erro dizendo que o navegador tentou acessar um POST para `/produto/:id`. Primeiro trocaremos de `button_to` para `link_to`. Agora o navegador tentará um GET.

Nas rotas, devemos passar o *edit* para as *resources*:

```
Rails.application.routes.draw do
  resources :departamentos
  resources :produtos, only: [:new, :create, :destroy, :edit]
  ...
end
```

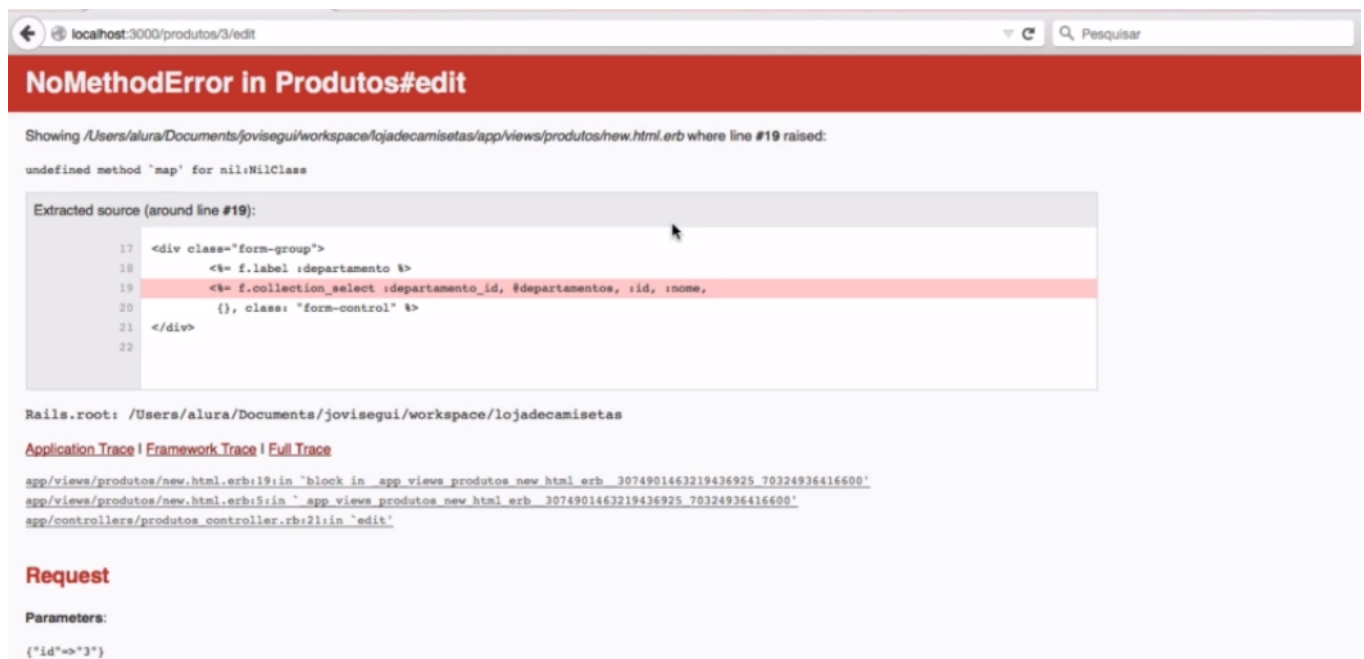
Se visualizarmos a URL criada pela *resource* veremos que será `/produto/:id/edit` e o *path* será `edit_produto`. Então chamamos o `edit_produto_path`. Ele recebe o parâmetro *id*, por isso precisamos passar qual é esse parâmetro (produto).

```
<td><%= link_to "Alterar", edit_produto_path(produto), class: "btn btn-default" %></td>
```

O link está certo. O erro que será mostrado está ligado ao *controller*, a ação "edit" não foi encontrada. Precisamos criar um método lá:

```
def edit
  id = params[:id]      #busca o parâmetro "id"
  @produto = Produto.find(id)  #procura o produto
  render :new           #queremos renderizar a view do "new", pois é no formulário que faremos as modificações
end
```

Salvamos e tentamos novamente acessar a página:



Perceba que não populamos os `@departamentos`, então vamos fazê-lo:

```
def edit
  id = params[:id]
  @produto = Produto.find(id)
  @departamentos = Departamento.all
  render :new
end
```

Agora sim, foi renderizada a página de formulário. Porém perceba que, apesar de funcionar, não é interessante que o botão siga com o nome "Criar o produto", afinal estamos modificando-o.

Método update

Também não conseguiremos salvar as alterações, pois faltará a rota para atualizar os valores do produto:

```
...
resources :produtos, only: [:new, :create, :destroy, :edit, :update]
...
```

- `:edit` mostra o formulário de edição;
- `:update` é o ato de atualizar.

Da mesma forma funcionaram o `:new`, que mostra o formulário de criação, e o `:create`, que é o ato de criar. Também será necessário criar o método no *controller*:

```
def update
  id = params[:id] #busca o parâmetro pelo id
  @produto = Produto.find(id) #busca o produto pelo id
  valores = params.require(:produto).permit(:nome, :preco, :descricao, :quantidade, :departamento)
  if @produto.update(valores) #observa cada valor e os atualiza
    flash[:notice] = "Produto atualizado com sucesso" #dando certo, aparece a mensagem
    redirect_to root_url #volta para a página inicial
  else
    render :new
  end
end
```

Atualizando a página no navegador, teremos:

Nome	Descrição	Preço		
Bermuda do big bang	Bermuda para combinar com a camiseta. Compre tambem a camiseta	120.3	Remover	Alterar
Blusa de la de festa junina	Nova blusa de lã	50.0	Remover	Alterar
Blusa listrada	blusa listrada nova	3.0	Remover	Alterar
Blusa xadrez	Descrição não tão longa	33.0	Remover	Alterar
Blusa xadrez 2.0	Nova versão da blusa	30.0	Remover	Alterar

Nome	Descrição	Preço		
Blusa listrada	blusa listrada nova	3.0	Remover	Alterar
Calça xadrez	Para combinar com a blusa xadrez	10.0	Remover	Alterar

Lembre-se que tivemos problemas quando não fizemos `@departamentos = Departamento.all` no método `edit`, então também vamos inseri-lo nos métodos `update` e `create`.

Limpendo o *controller*

Perceba que diversas linhas desse código se repetem no nosso *controller* na maioria dos métodos. Começamos isolando, em um método a parte, a busca de todos os departamentos e a chamada para a renderização da tela de criação de produto:

```
def renderiza_new
  @departamentos = Departamento.all
  render :new
end
```

Essas duas linhas podem ser substituídas agora por `renderiza_new` em todo o *controller*. Esse método que criamos é para o nosso controle e clareza do código, além de não ser chamado em nenhuma rota. Não existe a necessidade de mostrá-lo. Então o coloquemos como privado:

```
private

def renderiza_new
  @departamentos = Departamento.all
  render :new
end
```

Tudo o que vier depois do `private` será privado. A busca, ou *set* por convenção, do produto pelo *id* também se repete em vários métodos. Vamos isolá-la logo abaixo do `renderiza_new`:

```
private

def renderiza_new
  @departamentos = Departamento.all
  render :new
end

def set_produto
  id = params[:id]
  @produto = Produto.find(id)
end
```

No caso do método `destroy` faremos mais algumas mudanças:

```
def destroy
  set_produto
  @produto.destroy
  redirect_to root_url
end
```

O `set_produto` está sendo chamado em primeiro lugar em três métodos diferentes. Então, para os deixarmos mais limpos ainda, podemos usar um método, `before_action`, que chama aquele em primeiro lugar para `edit`, `update` e `destroy` (ele será inserido em cima de tudo):

```
class ProdutosController < ApplicationController

  before_action :set_produto, only: [:edit, :update, :destroy]
```

E excluimos o `set_produto` de tais métodos.

Podemos isolar o `params.require` (também privado). Por padrão o nome do método será `produto_params` :

```
def produto_params
  params.require(:produto).permit :nome, :preco, :descricao, :quantidade, :departamento_id
end
```

E nos métodos que o chamam:

```
valores = produto_params
```

Ou substituir onde houver `valores` por `produto_params` .

Mudando o botão usando *locals*

Nos resta mudar o botão da tela de alteração de produto. Ele está igual ao da tela de criação. Uma maneira de fazer isso seria criar a página "edit" e pedirmos para que ela seja renderizada onde se deve (`update` e `edit`). Apenas copiaríamos o mesmo código do "new.html.erb" e mudariamos o nome do botão. Com umas poucas mudanças no *controller* conseguimos solucionar esse problema. Porém isso acarretará em outros como, por exemplo, no caso de quisermos acrescentar um novo campo para a criação. Cada mudança terá que ser copiada para todas as páginas. Já vimos que o *Copy-Paste* é uma má prática!

Então, a melhor prática é termos um *partial* de formulário de criação, o "_form.html.erb" dentro de "/views/produtos". Agora tanto no "new.html.erb" quanto no "edit.html.erb" fazemos:

```
<%= render 'form' %>
```

E todo aquele código do formulário vai para a *partial*.

No *form*, na parte do botão, fazemos:

```
<%= f.submit texto_da_acao, class: "btn btn-primary" %>
```

A *string* do botão será a única diferença entre os dois formulários. Então para o de criação e o de edição fazemos, respectivamente:

```
<%= render 'form', locals: { texto_da_acao: 'Criar o produto' } %>
```

```
<%= render 'form', locals: { texto_da_acao: 'Atualizar o produto' } %>
```

A variável é local, então usamos o hash `locals: {}` . Além disso o formulário virou uma *partial*, então precisamos explicitar:

```
<%= render partial: 'form', locals: { texto_da_acao: 'Criar o produto' } %>
```

```
<%= render partial: 'form', locals: { texto_da_acao: 'Atualizar o produto' } %>
```

Com isso finalizamos o nosso curso! Parabéns! Com o que vimos até agora, já temos o básico para a criação de uma página de produtos com criação, alteração, exclusão, visualização e tudo que se tem direito! Vimos boas práticas de escrita de código, customizamos rotas, aprendemos sobre estruturas. Criamos um sistema automatizado, limpo e pronto para ser aplicado em qualquer negócio! Esperamos que você tenha gostado e aproveitado! Até a próxima!