INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Finalizando o Pedido
(Tarefa G)

Prof. Fellipe Aleixo (*fellipe.aleixo@ifrn.edu.br*)

Feedback do Cliente

- Com os itens no carrinho – é hora de permitir a finalização da venda
  - Operação conhecida como "Check Out"

- Para tal, serão necessários os dados do comprador (cliente)

Iteração G1:

CAPTURANDO O PEDIDO

Pedido

- Uma ordem de pedido, ou simplesmente pedido, é
  - uma coleção de "linha de item", mais
  - alguns detalhes da transação de compra e do comprador

- O carrinho de compras armazena os item selecionados pelo comprador

Pedido

- Alterações necessárias:
  - Um identificador do pedido – **order_id** – na tabela linha de item
  - Criar uma tabela pedido – **Order** – com alguns dados do cliente

```
depot> rails generate scaffold Order name address:text email pay_type
depot> rails generate migration add_order_to_line_item order:references
```

- Obs.: o tipo padrão dos atributos é **String**

Pedido

```
depot> rake db:migrate
==  CreateOrders: migrating ====================================
-- create_table(:orders)
   -> 0.0014s
==  CreateOrders: migrated (0.0015s) ===========================

==  AddOrderIdToLineItem: migrating ============================
-- add_column(:line_items, :order_id, :integer)
   -> 0.0008s
==  AddOrderIdToLineItem: migrated (0.0009s) ===================
```

- As respectivas “migrações” poderiam ter sido feitas em duas etapas

Criação do Pedido

- É necessária a criação de um formulário para capturar os detalhes do pedido
  - Inicialmente, será criado um botão "**Check Out**" no carrinho de compras
  - Tal botão irá disparar o método **new** do controlador de Pedido

Criação do Pedido

Download rails40/depot_o/app/views/carts/_cart.html.erb

```
<h2>Your Cart</h2>
<table>
  <%= render(cart.line_items) %>

  <tr class="total_line">
    <td colspan="2">Total</td>
    <td class="total_cell"><%= number_to_currency(cart.total_price) %></td>
  </tr>

</table>

<%= button_to "Checkout", new_order_path, method: :get %>
<%= button_to 'Empty cart', cart, method: :delete,
    data: { confirm: 'Are you sure?' } %>
```

Criação do Pedido

- O que precisa ser checado?
  - Primeiro: garantir que (i) há um carrinho de compras e que (ii) este não está vazio – se estiver vazio, voltamos a página principal
  - Em paralelo, adicionamos o teste que garanta que haja itens no carrinho de compras
  - Se o carrinho contém itens... continua

Criação do Pedido

Download rails40/depot_o/app/controllers/orders_controller.rb

```
class OrdersController < ApplicationController
  include CurrentCart
  before_action :set_cart, only: [:new, :create]
  before_action :set_order, only: [:show, :edit, :update, :destroy]

  # GET /orders
  #...
end
```

- Garantindo (i)

Criação do Pedido

Download rails40/depot_o/app/controllers/orders_controller.rb

```
def new
  if @cart.line_items.empty?
    redirect_to store_url, notice: "Your cart is empty"
    return
  end

  @order = Order.new
end
```

- Garantindo (ii)

Criação do Pedido

- Ajustando o teste:

Download rails40/depot_o/test/controllers/orders_controller_test.rb

```
test "requires item in cart" do
  get :new
  assert_redirected_to store_path
  assert_equal flash[:notice], 'Your cart is empty'
end

test "should get new" do
  item = LineItem.new
  item.build_cart
  item.product = products(:ruby)
  item.save!
  session[:cart_id] = item.cart.id
  get :new
  assert_response :success
end
```

Criando um Formulário

- O método new será responsável por recolher as informações do cliente
  - Será definida uma variável **@order** para referenciar o pedido – os elementos da visão irão popular esse objeto
  - Quando o usuário submeter o formulário, tal instância será carregada com os valores preenchidos
  - Rails métodos auxiliares (*helpers*) para ajudar na montagem de formulários – **form_from**

Criando um Formulário

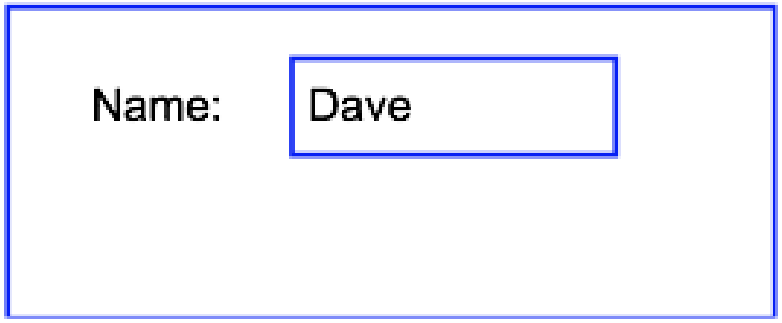
controller:

```
def edit
  @order = Order.find(...)
end

<%= form_for     @order  do |f| %>
  <p>
    <%= f.label :name, "Name:" %>
    <%= f.text_field      :name  , size: 40 %>
  </p>
<% end %>
```

model object:

@order.name → "Dave"



Criando um Formulário

Download rails40/depot_o/app/views/orders/new.html.erb

```
<div class="depot_form">
  <fieldset>
    <legend>Please Enter Your Details</legend>
    <%= render 'form' %>
  </fieldset>
</div>
```

Criando um Formulário

Download rails40/depot_o/app/views/orders/_form.html.erb

```
<%= form_for(@order) do |f| %>
  <% if @order.errors.any? %>
    <div id="error_explanation">
      <h2><%= pluralize(@order.errors.count, "error") %>
      prohibited this order from being saved:</h2>
      <ul>
      <% @order.errors.full_messages.each do |msg| %>
        <li><%= msg %></li>
      <% end %>
      </ul>
    </div>
  <% end %>
  ...
```

Criando um Formulário

```
    <div class="field">
      <%= f.label :name %><br>
➤     <%= f.text_field :name, size: 40 %>
    </div>
    <div class="field">
      <%= f.label :address %><br>
➤     <%= f.text_area :address, rows: 3, cols: 40 %>
    </div>
    <div class="field">
      <%= f.label :email %><br>
➤     <%= f.email_field :email, size: 40 %>
    </div>
    <div class="field">
      <%= f.label :pay_type %><br>
➤     <%= f.select :pay_type, Order::PAYMENT_TYPES,
➤                  prompt: 'Select a payment method' %>
    </div>
    <div class="actions">
➤     <%= f.submit 'Place Order' %>
    </div>
  <% end %>
```

Criando um Formulário

- Perceba o detalhe do código associado a uma lista de seleção

Download rails40/depot_o/app/models/order.rb

```
class Order < ActiveRecord::Base
  PAYMENT_TYPES = [ "Check", "Credit card", "Purchase order" ]
end
```

Ajustes no CSS

Download rails40/depot_o/app/assets/stylesheets/application.css.scss

```
.depot_form {
  fieldset {
    background: #efe;

    legend {
      color: #dfd;
      background: #141;
      font-family: sans-serif;
      padding: 0.2em 1em;
    }
  }

  form {
    label {
      width: 5em;
      float: left;
      text-align: right;
      padding-top: 0.2em;
      margin-right: 0.1em;
      display: block;
    }

    select, textarea, input {
      margin-left: 0.5em;
    }

    .submit {
      margin-left: 4em;
    }

    br {
      display: none
    }
  }
}
```

Ajustes no Modelo

Download rails40/depot_o/app/models/order.rb

```
class Order < ActiveRecord::Base
  # ...
  validates :name, :address, :email, presence: true
  validates :pay_type, inclusion: PAYMENT_TYPES
end
```

- Também são necessários ajustes nos *fixtures* de teste (dados de um pedido + itens)

Resultado


Pragprog Books Online St
localhost:3000/orders/new?
Pragmatic Bookshelf
PRAGMATIC BOOKSHELF
Your Cart
2× CoffeeScript $72.00
2× Programming Ruby 1.9 & 2.0 $99.90
Total $171.90
Checkout
Empty cart
Home
Questions
News
Contact
Please Enter Your Details
Name
Address
Email
Pay type
Select a payment method
Place Order

Próximos Passos

- A ação **create** do controlador, deverá:
  - Popular o novo objeto pedido
  - Adicionar as linhas de item do carrinho, destruindo o mesmo
  - Validar e salvar o pedido
  - Exibir novamente o catálogo e confirma o pedido
  - Atualizar o teste para verificar o sucesso

Definindo Relacionamentos

- Entre Linha de item e Pedido

Download rails40/depot_o/app/models/line_item.rb

```
class LineItem < ActiveRecord::Base
  belongs_to :order
  belongs_to :product
  belongs_to :cart
  def total_price
    product.price * quantity
  end
end
```

Definindo Relacionamentos

- Entre Pedido e Linha de item

Download rails40/depot_o/app/models/order.rb

```
class Order < ActiveRecord::Base
  has_many :line_items, dependent: :destroy
  # ...
end
```

Criando um Pedido

Download rails40/depot_o/app/controllers/orders_controller.rb

```
def create
  @order = Order.new(order_params)
  @order.add_line_items_from_cart(@cart)

  respond_to do |format|
    if @order.save
      Cart.destroy(session[:cart_id])
      session[:cart_id] = nil

      format.html { redirect_to store_url, notice:
        'Thank you for your order.' }
      format.json { render action: 'show', status: :created,
        location: @order }

    else
      format.html { render action: 'new' }
      format.json { render json: @order.errors,
        status: :unprocessable_entity }
    end
  end
end
```

Movendo os Itens do Carrinho

Download rails40/depot_p/app/models/order.rb

```
class Order < ActiveRecord::Base
  # ...
  def add_line_items_from_cart(cart)
    cart.line_items.each do |item|
      item.cart_id = nil
      line_items << item
    end
  end
end
```

Ajustando os Testes

Download rails40/depot_p/test/controllers/orders_controller_test.rb

```
test "should create order" do
  assert_difference('Order.count') do
    post :create, order: { address: @order.address, email: @order.email,
      name: @order.name, pay_type: @order.pay_type }
  end

  assert_redirected_to store_path
end
```

- Verificando se acontece o redirecionamento

Resultados no Bando de Dados

```
depot> sqlite3 -line db/development.sqlite3
SQLite version 3.7.4
Enter ".help" for instructions
sqlite> select * from orders;
            id = 1
          name = Dave Thomas
       address = 123 Main St
         email = customer@example.com
      pay_type = Check
    created_at = 2013-01-29 02:31:04.964785
    updated_at = 2013-01-29 02:31:04.964785
sqlite> select * from line_items;
          id = 10
  product_id = 2
     cart_id =
  created_at = 2013-01-29 02:30:26.188914
  updated_at = 2013-01-29 02:31:04.966057
    quantity = 1
       price = 36
    order_id = 1
sqlite> .quit
```

Ajustes no AJAX

- Após aceitar um pedido, e redirecionar para a página index, exibindo uma mensagem agradecendo pelo pedido – ao continuar a comprar, a mensagem permanece lá

- Podemos apaga-la ao adicionar um item no carrinho de compras

Ajustes no AJAX

Download rails40/depot_p/app/views/line_items/create.js.erb

```
$('#notice').hide();

if ($('#cart tr').length == 1) { $('#cart').show('blind', 1000); }

$('#cart').html("<%= escape_javascript render(@cart) %>");

$('#current_item').css({'background-color':'#88ff88'}).
  animate({'background-color':'#114411'}, 1000);
```

Iteração G2:

ATOM FEEDS

ATOM

- Protocolo ao nível da aplicação para publicar e editar Fontes web que são periodicamente atualizadas, como por exemplo Blogs

- Os **feeds** devem ser formados em formato XML e são identificados como
  - **application/atom+xml media type**

Geração de Informações de Uso

- Pode ser utilizada o ATOM para gerar informações sobre a utilização do sistema

- Iniciamos gerando um novo método no controlador de Produto
  - Quem comprou um determinado produto

Geração de Informações de Uso

Download rails40/depot_p/app/controllers/products_controller.rb

```
def who_bought
  @product = Product.find(params[:id])
  @latest_order = @product.orders.order(:updated_at).last
  if stale?(@latest_order)
    respond_to do |format|
      format.atom
    end
  end
end
```

- **stale** – checa se a informação está “obsoleta”

Geração de Informações de Uso

- Cada resposta contém metadados que identificam a última modificação das informações e um *hash* **ETag**

- Em requisições subsequentes esses metadados retornam, e permitem saber que não existem novas modificações a enviar

Gerador de Informações ATOM

- Ao adicionar uma resposta no modelo **format.atom**, Rails irá procurar por um *template* **who_bougth.atom.builder**

- Podem ser utilizados recursos para
  - A geração no formato XML
  - A geração no formato ATOM (**atom_feed**)

Gerador de Informações ATOM

Download rails40/depot_p/app/views/products/who_bought.atom.builder

```
atom_feed do |feed|
  feed.title "Who bought #{@product.title}"

  feed.updated @latest_order.try(:updated_at)

  @product.orders.each do |order|
    feed.entry(order) do |entry|
      entry.title "Order #{order.id}"
      entry.summary type: 'xhtml' do |xhtml|
        xhtml.p "Shipped to #{order.address}"

        ...
```

Gerador de Informações ATOM

```
xhtml.table do
  xhtml.tr do
    xhtml.th 'Product'
    xhtml.th 'Quantity'
    xhtml.th 'Total Price'
  end
  order.line_items.each do |item|
    xhtml.tr do
      xhtml.td item.product.title
      xhtml.td item.quantity
      xhtml.td number_to_currency item.total_price
    end
  end
  xhtml.tr do
    xhtml.th 'total', colspan: 2
    xhtml.th number_to_currency \
      order.line_items.map(&:total_price).sum
  end
end
```

Gerador de Informações ATOM

```
        xhtml.p "Paid by #{order.pay_type}"
      end
      entry.author do |author|
        author.name order.name
        author.email order.email
      end
    end
  end
end
```

Gerador de Informações ATOM

- Para possibilitar recolher informações sobre os produtos que foram comprados
  - Há de ser criado o relacionamento entre o pedido (através das linha de item) e um produto

Download rails40/depot_p/app/models/product.rb

```
class Product < ActiveRecord::Base
  has_many :line_items
➤ has_many :orders, through: :line_items
  #...
end
```

Gerador de Informações ATOM

- Para funcionar, é necessário definir uma rota
  - Que deverá responder a requisições HTTP GET

Download rails40/depot_p/config/routes.rb

```
Depot::Application.routes.draw do
  resources :orders
  resources :line_items
  resources :carts

  get "store/index"
  resources :products do
    get :who_bought, on: :member
  end

  # The priority is based upon order of creation:
  # first created -> highest priority.
  # See how all your routes lay out with "rake routes".
  # You can have the root of your site routed with "root"
  root 'store#index', as: 'store'
  # ...
end
```

Resultado

- Testando a partir da linha de comando...

```
depot> curl --silent http://localhost:3000/products/3/who_bought.atom
<?xml version="1.0" encoding="UTF-8"?>
<feed xml:lang="en-US" xmlns="http://www.w3.org/2005/Atom">
  <id>tag:localhost,2005:/products/3/who_bought</id>
  <link type="text/html" href="http://localhost:3000" rel="alternate"/>
  <link type="application/atom+xml"
        href="http://localhost:3000/info/who_bought/3.atom" rel="self"/>
  <title>Who bought Programming Ruby 1.9</title>
  <updated>2013-01-29T02:31:04Z</updated>
  <entry>
    <id>tag:localhost,2005:Order/1</id>
    <published>2013-01-29T02:31:04Z</published>
    <updated>2013-01-29T02:31:04Z</updated>
    <link rel="alternate" type="text/html" href="http://localhost:3000/orders/1"/>
    <title>Order 1</title>
    <summary type="xhtml">
      <div xmlns="http://www.w3.org/1999/xhtml">
        <p>Shipped to 123 Main St</p>
    ...
```

EXTRAS a Serem Tentados

1. Gerar visões alternativas para a funcionalidade “quem comprou”
   - (i) HTML, (ii) XML e (iii) JSON
   - Exemplo: **@product.to_xml(include: :orders)**
2. Com o carrinho da lateral – após acionar o “Checkout” uma vez, desabilitar o botão
3. Mover as opções de pagamento de constantes na classe pedido para o banco de dados. Mantendo a validação para este campo