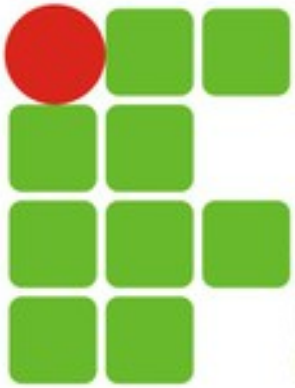

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Adicionando AJAX (Tarefa F)

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

Feedback do Cliente

- Ok! A aplicação que você está desenvolvendo já está tomando forma
- Mas, … gostaria que fosse incluído o suporte a AJAX no site da “loja”

- Ok! Mas, o que é AJAX, mesmo?!?

Características das Aplicações Web

- Inicialmente as aplicações precisaram se adequar aos protocolos base – HTTP e HTML
  - Interação limitada
  - Requisição + Resposta + Renderização

- Os navegadores "modernos" suprem tais limitações e permitem a execução de código
  - Linguagem Javascript

Javascript

- Javascript permite a interação, em segundo plano, com aplicação no servidor
  - E consequentemente a atualização de partes específicas da página exibida pelo navegador

- Esse estilo de interação foi denominado de AJAX – *Asynchronous JavaScript and XML*

Aplicando AJAX ao “Armazém TADS”

- Na aplicação em desenvolvimento, serão realizadas as seguintes modificações:
  - O carrinho de compras será colocado em uma das laterais do catálogo
  - Com AJAX, as atualizações no carrinho de compras serão realizadas sem a necessidade de haver uma requisição da página completa

Iteração F1:

MOVENDO O CARRINHO PARA A LATERAL DO CATÁLOGO

Movendo o Carrinho de Compras

- O carrinho é renderizado pela ação **show** do **CartController**
  - E o template correspondente (.html.erb)
  - A renderização dessa parte específica ocorrerá na exibição do catálogo
  - Utilização da funcionalidade de ***partial templates***

Partial Templates

- Estratégia similar a “métodos para visões”
  - Arquivo separado contendo uma “parcial”
  - Tem a sua renderização invocada por outra visão
  - Podem ser passados parâmetros para uma “parcial”

- Como é feita a visualização do carrinho?

Download rails40/depot_i/app/views/carts/show.html.erb

```
<% if notice %>
<p id="notice"><%= notice %></p>
<% end %>

<h2>Your Cart</h2>
<table>
  <% @cart.line_items.each do |item| %>
    <tr>
      <td><%= item.quantity %>&times;</td>
      <td><%= item.product.title %></td>
      <td class="item_price"><%= number_to_currency(item.total_price) %></td>
    </tr>
  <% end %>

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

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

Aplicação de Parciais

- Em um primeiro nível a estratégia de parciais pode ser utilizada para fatorar a complexidade
  - Podemos utilizar uma parcial específica para mostrar uma linha de item
  - Ao chamar a parcial definida para tal, passamos a coleção de linhas de item do carrinho

Aplicação de Parciais

Download rails40/depot_j/app/views/carts/show.html.erb

```
<% if notice %>
<p id="notice"><%= notice %></p>
<% end %>

<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 'Empty cart', @cart, method: :delete,
    data: { confirm: 'Are you sure?' } %>
```

Aplicação de Parciais

- O método **render** iterará na coleção recebida
- O arquivo da parcial fica no mesmo diretório que o objeto sendo renderizado, por padrão
  - Rails anexará um "sublinha" ao nome do arquivo
  - É utilizada uma variável com o nome do *template*

Download rails40/depot_j/app/views/line_items/_line_item.html.erb

```
<tr>
  <td><%= line_item.quantity %>&times;</td>
  <td><%= line_item.product.title %></td>
  <td class="item_price"><%= number_to_currency(line_item.total_price) %></td>
</tr>
```

Transformando o Carrinho em Parcial

- Como trata-se de uma parcial, as variáveis utilizadas são passadas por parâmetro

Download rails40/depot_j/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 'Empty cart', cart, method: :delete,
    data: { confirm: 'Are you sure?' } %>
```

Acionando a Parcial

- A invocação da parcial é responsável por fornecer (por meio de parâmetro) as informações necessitadas por ela

Download rails40/depot_k/app/views/carts/show.html.erb

```
<% if notice %>
<p id="notice"><%= notice %></p>
<% end %>

<%= render @cart %>
```

Incluindo o Carrinho da Lateral

Download rails40/depot_k/app/views/layouts/application.html.erb

```
<!DOCTYPE html>
<html>
<head>
  <title>Pragprog Books Online Store</title>
  <%= stylesheet_link_tag    "application", media: "all",
    "data-turbolinks-track" => true %>
  <%= javascript_include_tag "application", "data-turbolinks-track" => true %>
  <%= csrf_meta_tags %>
</head>
<body class="<%= controller.controller_name %>">
  <div id="banner">
    <%= image_tag("logo.png") %>
    <%= @page_title || "Pragmatic Bookshelf" %>
  </div>
  <div id="columns">
    <div id="side">
      <div id="cart">
        <%= render @cart %>
      </div>
```

Incluindo o Carrinho da Lateral

```
...
    <ul>
      <li><a href="http://www....">Home</a></li>
      <li><a href="http://www..../faq">Questions</a></li>
      <li><a href="http://www..../news">News</a></li>
      <li><a href="http://www..../contact">Contact</a></li>
    </ul>
  </div>
  <div id="main">
    <%= yield %>
  </div>
 </div>
</body>
</html>
```

Incluindo o Carrinho da Lateral

- Para funcionar, o **store_controller** precisa definir o atributo **@cart**

Download rails40/depot_k/app/controllers/store_controller.rb

```
class StoreController < ApplicationController
  include CurrentCart
  before_action :set_cart
  def index
    @products = Product.order(:title)
  end
end
```

Pequenas Alterações no Estilo

Download rails40/depot_k/app/assets/stylesheets/carts.css.scss

```
// Place all the styles related to the Carts controller here.
// They will automatically be included in application.css.
// You can use Sass (SCSS) here: http://sass-lang.com/

.carts, #side #cart {
  .item_price, .total_line {
    text-align: right;
  }

  .total_line .total_cell {
    font-weight: bold;
    border-top: 1px solid #595;
  }
}
```

Pequenas Alterações no Estilo

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

```
#side {
  float: left;
  padding: 1em 2em;
  width: 13em;
  background: #141;

  form, div {
    display: inline;
  }

  input {
    font-size: small;
  }

  #cart {
    font-size: smaller;
    color:     white;

    table {
      border-top:    1px dotted #595;
      border-bottom: 1px dotted #595;
      margin-bottom: 10px;
```

Continuação...

```
    }
  }

  ul {
    padding: 0;

    li {
      list-style: none;

      a {
        color: #bfb;
        font-size: small;
      }
    }
  }
}
```

Resultado Obtido

- O clique em “Add to Cart” reexibe o catálogo

Iteração F2:

APLICANDO AJAX

AJAX

- AJAX permite que seja inserido código que interage com a aplicação no servidor
  - Gostaríamos que o botão “Add to Cart” invocasse a ação **create** do controlador de linha de item
    - Invocação em segundo plano
    - Inclusão do parâmetro **remote: true**
  - Nesse caso o servidor retornaria apenas o HTML referente ao carrinho

Download rails40/depot_l/app/views/store/index.html.erb

```
<% if notice %>
<p id="notice"><%= notice %></p>
<% end %>

<h1>Your Pragmatic Catalog</h1>

<% cache ['store', Product.latest] do %>
  <% @products.each do |product| %>
    <% cache ['entry', product] do %>
      <div class="entry">
        <%= image_tag(product.image_url) %>
        <h3><%= product.title %></h3>
        <%= sanitize(product.description) %>
        <div class="price_line">
          <span class="price"><%= number_to_currency(product.price) %></span>
          <%= button_to 'Add to Cart', line_items_path(product_id: product),
            remote: true %>
        </div>
      </div>
    <% end %>
  <% end %>
<% end %>
```

Preparando a Resposta

- Idéia:
  - Criar um fragmento de HTML com as informações atualizadas do carrinho e retornar ao navegador
  - Substituir o fragmento de HTML correspondente
  - Utilizar o DOM (*Document Object Model*) para referenciar os elementos específicos a alterar

Preparando a Resposta

- Na execução do método **create** do controlador de linha de item
  - Após a conclusão do mesmo, não redirecionar mais para o **index** do controlador do carrinho (se a requisição vier do Javascript)

Preparando a Resposta

Download rails40/depot_l/app/controllers/line_items_controller.rb

```
def create
  product = Product.find(params[:product_id])
  @line_item = @cart.add_product(product.id)

  respond_to do |format|
    if @line_item.save
      format.html { redirect_to store_url }
      format.js
      format.json { render action: 'show',
        status: :created, location: @line_item }
    else
      format.html { render action: 'new' }
      format.json { render json: @line_item.errors,
        status: :unprocessable_entity }
    end
  end
end
```

Preparando a Resposta

- Devido a mudança anteriormente ilustrada, ao finaliza a ação **create**, Rails irá procurar pelo *template* **create** para renderizar
- Rails suporta templates que geram Javascript

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

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

  - Substituir o elemento de “id = cart”
  - Aplicação do JQuery
  - **escape_javascript** converte string Rails ao JS