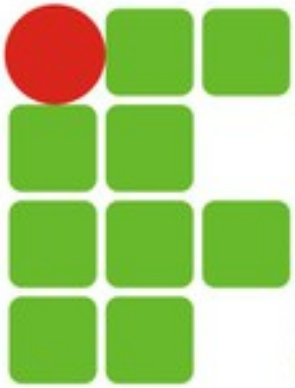


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Um Carrinho de Compras Mais Inteligente (Tarefa E)

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

Feedback do Cliente

- “A funcionalidade do carrinho de compras ainda está muito básica”

- Vejamos algumas funcionalidades que podemos adicionar:
  - Mudar quantidade de itens no carrinho
  - Manipular erros e comunicar os problemas de forma apropriada

Iteração E1:

CRIANDO UM CARRINHO INTELIGENTE

Adicionando Quantidade de Itens

- Irá requerer a alteração da tabela **line_items**
  - Necessário aplicar uma "migração"

```
depot> rails generate migration add_quantity_to_line_items quantity:integer
```

- Padrões:
  - add_XXX_to_TABLE
  - remove_XXX_from_TABLE
  - XXX é ignorado → é levado em conta o nome da coluna e o tipo informados

Adicionando Quantidade de Itens

- Será adicionada a quantidade 1 para os registros que já existirem no banco de dados

Download rails40/depot_g/db/migrate/20121130000004_add_quantity_to_line_items.rb

```
class AddQuantityToLineItems < ActiveRecord::Migration
  def change
    add_column :line_items, :quantity, :integer, default: 1
  end
end
```

  - Para completar a migração:

```
depot> rake db:migrate
```

Evoluindo a Inclusão de Produtos no Carrinho de Compras

- Necessário tornar o método **add_product()** do carrinho mais inteligente
  - Se o produto já existir, incrementar a quantidade
  - Se não existir, criar uma nova linha de item
  - Vejamos como fica tal método...

Evoluindo a Inclusão de Produtos no Carrinho de Compras

Download rails40/depot_g/app/models/cart.rb

```
def add_product(product_id)
  current_item = line_items.find_by(product_id: product_id)
  if current_item
    current_item.quantity += 1
  else
    current_item = line_items.build(product_id: product_id)
  end
  current_item
end
```

- O método **find_by()** simplificada do método **where()** → retorna um objeto ou **nil**

Evoluindo a Inclusão de Produtos no Carrinho de Compras

Download rails40/depot_g/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 @line_item.cart,
        notice: 'Line item was successfully created.' }
      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
```

Evoluindo a Inclusão de Produtos no Carrinho de Compras

- Incluindo a “quantidade” na visão

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

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

<h2>Your Pragmatic Cart</h2>
<ul>
  <% @cart.line_items.each do |item| %>
    <li><%= item.quantity %> &times; <%= item.product.title %></li>
  <% end %>
</ul>
```

Ajustando o Banco de Dados

- Se já haviam carrinhos de compra do banco de dados com vários itens relativos a um produto
  - A migração adicionou a quantidade 1
  - Para ajustar isso é necessária uma nova migração → combinar as linhas de item relativas ao mesmo produto

```
depot> rails generate migration combine_items_in_cart
```

- Estar preparado para as mudanças implica em poder se adequar às novas realidades

Ajustando o Banco de Dados

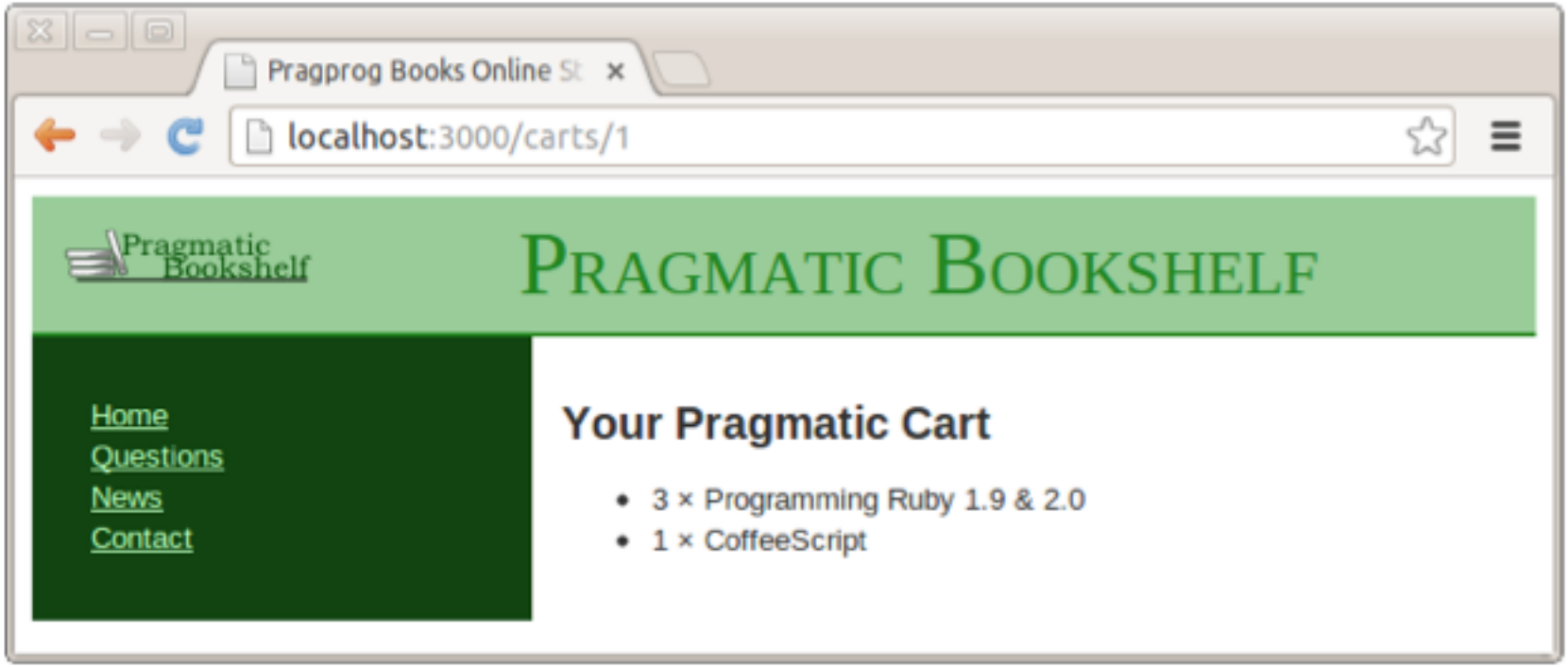
Download rails40/depot_g/db/migrate/20121130000005_combine_items_in_cart.rb

```ruby
def up
  # replace multiple items for a single product in a cart with a single item
  Cart.all.each do |cart|
    # count the number of each product in the cart
    sums = cart.line_items.group(:product_id).sum(:quantity)

    sums.each do |product_id, quantity|
      if quantity > 1
        # remove individual items
        cart.line_items.where(product_id: product_id).delete_all

        # replace with a single item
        item = cart.line_items.build(product_id: product_id)
        item.quantity = quantity
        item.save!
      end
    end
  end
end
```

Ajustando o Banco de Dados

```
depot> rake db:migrate
```

- Resultado:



Ajustando o Banco de Dados

- Todas as migrações precisam ser “reversíveis”
  - Método **down()** é responsável por reverter

Download rails40/depot_g/db/migrate/20121130000005_combine_items_in_cart.rb

```
def down
  # split items with quantity>1 into multiple items
  LineItem.where("quantity>1").each do |line_item|
    # add individual items
    line_item.quantity.times do
      LineItem.create cart_id: line_item.cart_id,
        product_id: line_item.product_id, quantity: 1
    end

    # remove original item
    line_item.destroy
  end
end
```

Ajustando o Banco de Dados

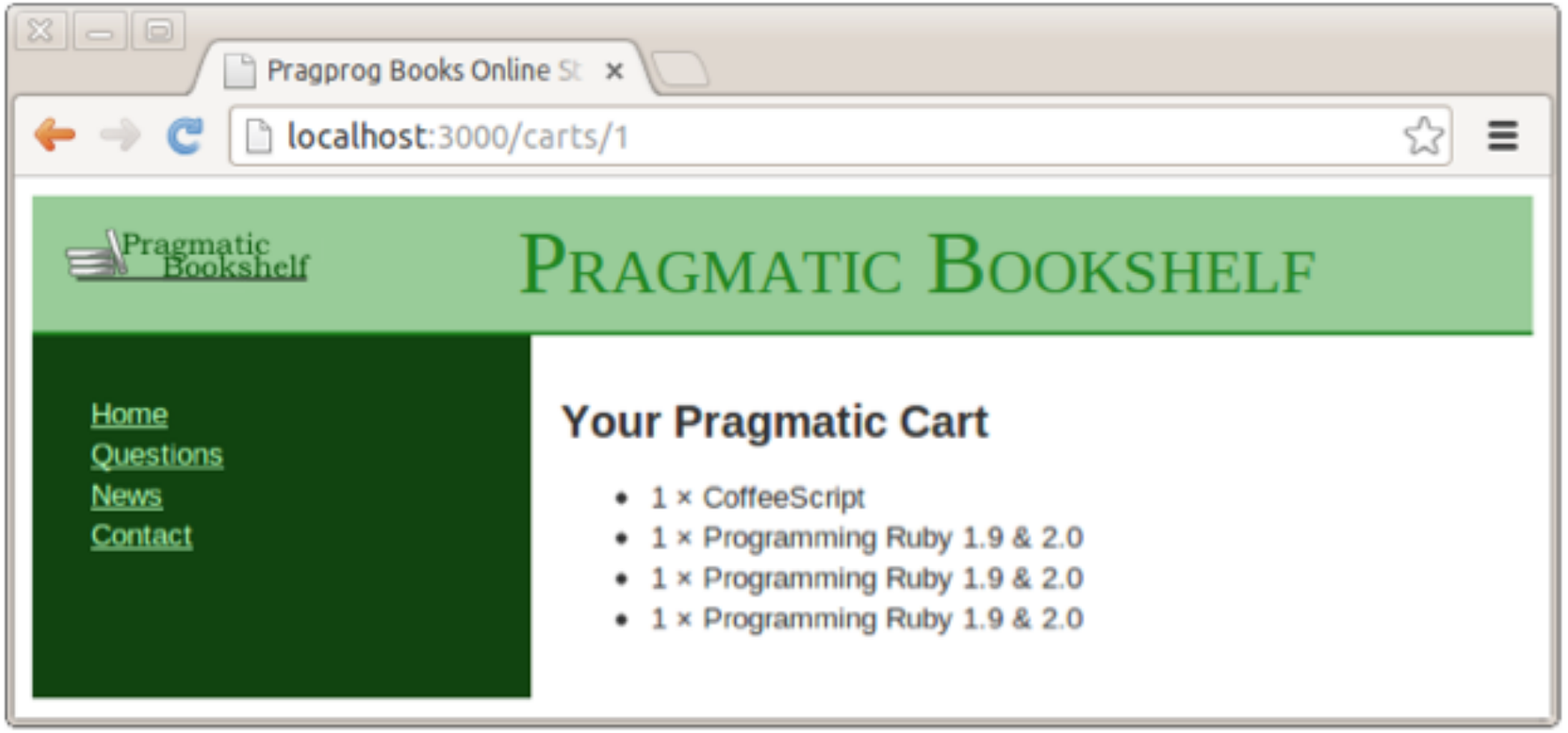
- Revertendo uma migração:

```
depot> rake db:rollback
```

- Verificando o status das migrações:

```
depot> rake db:migrate:status
database: /home/rubys/work/depot/db/development.sqlite3

  Status   Migration ID    Migration Name
--------------------------------------------------
    up     20130407000001  Create products
    up     20130407000002  Create carts
    up     20130407000003  Create line items
    up     20130407000004  Add quantity to line items
  down     20130407000005  Combine items in cart
```

Resultado


Pragprog Books Online St
localhost:3000/carts/1
Pragmatic Bookshelf
PRAGMATIC BOOKSHELF
Home
Questions
News
Contact
Your Pragmatic Cart
1 × CoffeeScript
1 × Programming Ruby 1.9 & 2.0
1 × Programming Ruby 1.9 & 2.0
1 × Programming Ruby 1.9 & 2.0

Iteração E2:

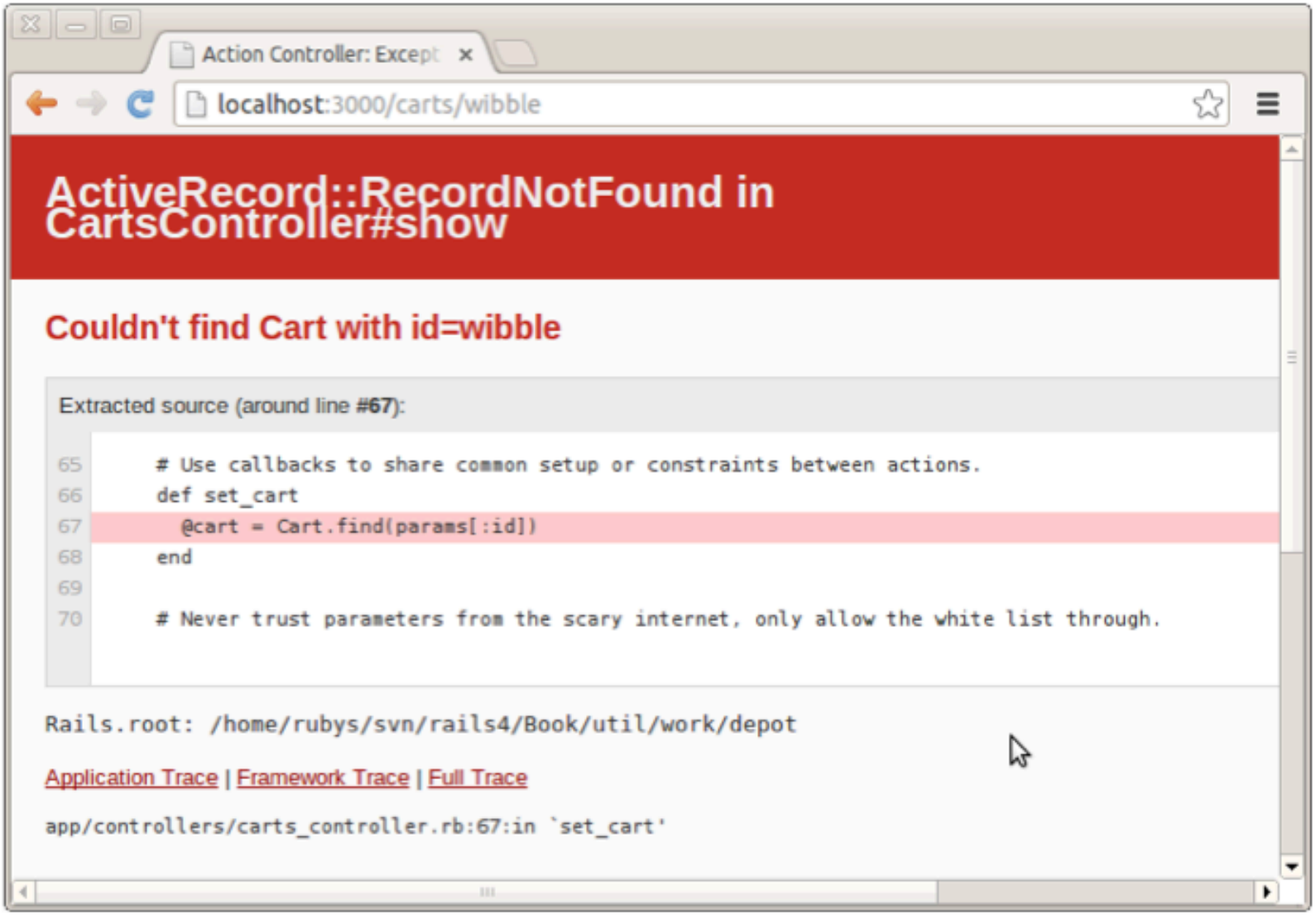
MANIPULANDO ERROS

Feedback do Cliente

- “Andei lendo alguns artigos, e estou preocupado com relação a segurança da aplicação”

- “O que acontece quando um usuário intencionalmente altera o link referenciando um carrinho de compras?”

Opsss!


Action Controller: Except
localhost:3000/carts/wibble
ActiveRecord::RecordNotFound in CartsController#show
Couldn't find Cart with id=wibble
Extracted source (around line #67):
65 # Use callbacks to share common setup or constraints between actions.
66 def set_cart
67 @cart = Cart.find(params[:id])
68 end
69
70 # Never trust parameters from the scary internet, only allow the white list through.
Rails.root: /home/rubys/svn/rails4/Book/util/work/depot
Application Trace | Framework Trace | Full Trace
app/controllers/carts_controller.rb:67:in `set_cart'

Manipulando Erros

- O que ocorreu?
  - Foi lançada um exceção, a qual não foi tratada

- Solução: capturar e tratar exceções
  - Apresentar erros de uma forma mais adequada

Manipulando Erros

- Estratégias disponibilizadas por Rails:
  - Captura e manipulação de Exceções
  - Registro de *logs* – *Logger Facility*
  - Utilização da estrutura **flash** (hash)
    - Informações para o próximo *request* da sessão
    - Usado para coletar mensagens de erro
- Solução do problema específico:
  - Armazenar no log o erro ocorrido
  - Exibir o catálogo com uma mensagem ao usuário

Manipulando Erros

- Criando o método **invalid_cart()** para reportar o problema

Download rails40/depot_h/app/controllers/carts_controller.rb

```
class CartsController < ApplicationController
  before_action :set_cart, only: [:show, :edit, :update, :destroy]
  rescue_from ActiveRecord::RecordNotFound, with: :invalid_cart
  # GET /carts
  # ...
  private
  # ...
    def invalid_cart
      logger.error "Attempt to access invalid cart #{params[:id]}"
      redirect_to store_url, notice: 'Invalid cart'
    end
end
```

Manipulando Erros

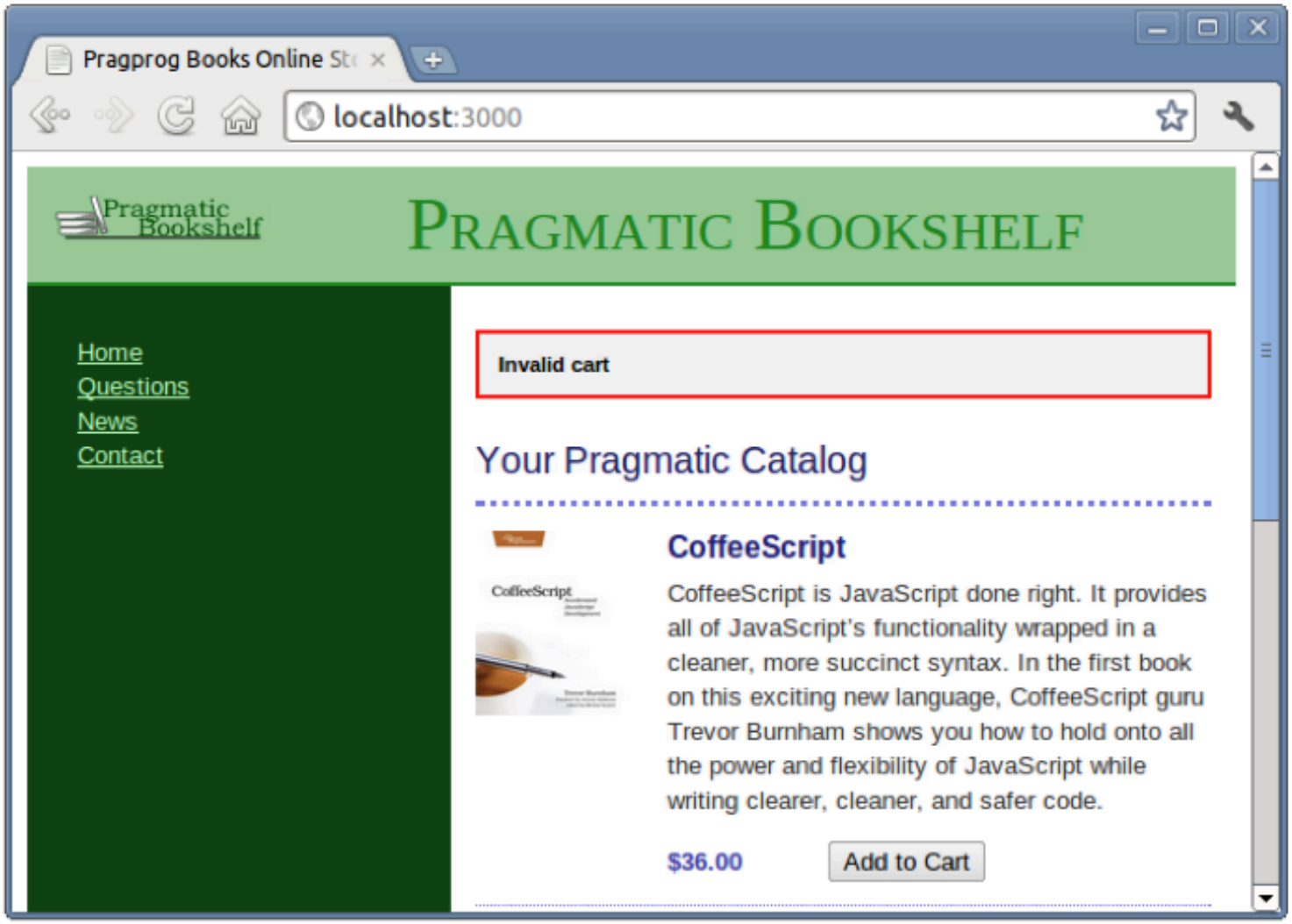
- A clausula **rescue_from** intercepta a exceção
- Todo controlador tem um atributo **logger**
- É utilizado o **redirect_to** redireciona a resposta para o catálogo
  - É adicionado um aviso (*notice*) para o usuário

Manipulando Erros

- Informações armazenadas no *log*

```
Started GET "/carts/wibble" for 127.0.0.1 at 2013-01-29 09:37:39 -0500
Processing by CartsController#show as HTML
  Parameters: {"id"=>"wibble"}
  ^[[1m^[[35mCart Load (0.1ms)^[[0m  SELECT "carts".* FROM "carts" WHERE
"carts"."id" = ? LIMIT 1  [["id", "wibble"]]
Attempt to access invalid cart wibble
Redirected to http://localhost:3000/
Completed 302 Found in 3ms (ActiveRecord: 0.4ms)
```

Resultado


Pragprog Books Online St
localhost:3000
Pragmatic Bookshelf
PRAGMATIC BOOKSHELF
Home
Questions
News
Contact
Invalid cart
Your Pragmatic Catalog
CoffeeScript
CoffeeScript is JavaScript done right. It provides all of JavaScript's functionality wrapped in a cleaner, more succinct syntax. In the first book on this exciting new language, CoffeeScript guru Trevor Burnham shows you how to hold onto all the power and flexibility of JavaScript while writing clearer, cleaner, and safer code.
$36.00
Add to Cart

Precauções com Segurança

- Definindo os parâmetros permitidos:

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

```
# Never trust parameters from the scary internet, only allow the white
# list through.
def line_item_params
  params.require(:line_item).permit(:product_id)
end
```

- Ajustando os testes:

Download rails40/depot_h/test/controllers/line_items_controller_test.rb

```
test "should update line_item" do
  patch :update, id: @line_item, line_item: { product_id: @line_item.product_id }
  assert_redirected_to line_item_path(assigns(:line_item))
end
```

Iteração E3:

FINALIZANDO O CARRINHO

Feedback do Cliente

- Algumas observações:
  - “Já estamos bem avançados e isso é bom!”
  - “Seria bom ter uma forma de remover itens e limpar o carrinho como um todo!”

Funcionalidade de Limpar Carrinho

- Para implementar a funcionalidade e limpar o carrinho de compras é necessário:
  - Adicionar um link para o carrinho
  - Modificar o método **destroy()**
  - Limpar os dados de sessão

- Vamos iniciar pela visão – alterar o *template*

Funcionalidade de Limpar Carrinho

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

```
<% if notice %>
<p id="notice"><%= notice %></p>
<% end %>
<h2>Your Pragmatic Cart</h2>
<ul>
  <% @cart.line_items.each do |item| %>
    <li><%= item.quantity %> &times; <%= item.product.title %></li>
  <% end %>
</ul>
<%= button_to 'Empty cart', @cart, method: :delete,
    data: { confirm: 'Are you sure?' } %>
```

Funcionalidade de Limpar Carrinho

- Alterando o método **destroy()** do controlador

Download rails40/depot_h/app/controllers/carts_controller.rb

```
def destroy
  @cart.destroy if @cart.id == session[:cart_id]
  session[:cart_id] = nil
  respond_to do |format|
    format.html { redirect_to store_url,
      notice: 'Your cart is currently empty' }
    format.json { head :no_content }
  end
end
```

Funcionalidade de Limpar Carrinho

- Atualizando os testes correspondentes:

Download rails40/depot_i/test/controllers/carts_controller_test.rb

```
test "should destroy cart" do
  assert_difference('Cart.count', -1) do
➤   session[:cart_id] = @cart.id
    delete :destroy, id: @cart
  end

➤ assert_redirected_to store_path
end
```

Retirando a Mensagem da Inserção de Linhas de Item

Download rails40/depot_i/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 @line_item.cart }
      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
```

Tabulando o Carrinho de Compras

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?' } %>
```

Métodos para Fazer Funcionar o Novo Carrinho de Compras

- Na linha de item:

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

```
def total_price
  product.price * quantity
end
```

- No carrinho de compras:

Download rails40/depot_i/app/models/cart.rb

```
def total_price
  line_items.to_a.sum { |item| item.total_price }
end
```

Pequenos Ajustes no Estilo

Download rails40/depot_i/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 {
  .item_price, .total_line {
    text-align: right;
  }
  .total_line .total_cell {
    font-weight: bold;
    border-top: 1px solid #595;
  }
}
```

Resultado


Pragprog Books Online St
localhost:3000/carts/2
Pragmatic Bookshelf
PRAGMATIC BOOKSHELF
Home
Questions
News
Contact
Your Cart
2× CoffeeScript $72.00
1× Programming Ruby 1.9 & 2.0 $49.95
Total $121.95
Empty cart