



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

# Limitando o Acesso (Continuação da Tarefa I)

Prof. Fellipe Aleixo ([fellipe.aleixo@ifrn.edu.br](mailto:fellipe.aleixo@ifrn.edu.br))

# Feedback do Cliente

- Ótimo! O usuário já pode realizar o login na aplicação com sucesso
- Agora precisamos que apenas esse usuário logado (administrador) tenha acesso a algumas áreas da aplicação

Iteração I3:

# **LIMITANDO O ACESSO**

# Rails *Callback*

- Mecanismo que permite a “interceptação” das chamadas aos métodos de ação e realizar processamentos específicos
  - Antes de invocar o método
    - **before\_action**
  - Ao término do método
  - Ambos

# Rails *Callback*

- Na aplicação exemplo (*Depot*), o método interceptador pode checar se existe o “id do usuário” na sessão
  - Pode ser adicionado ao controlador “admin”
  - Bem como, ao controlador geral da aplicação
    - Classe base para todos os controladores da aplicação

# Rails *Callback*

Download rails40/depot\_r/app/controllers/application\_controller.rb

```
class ApplicationController < ActionController::Base
  ➤ before_action :authorize
    # ...
  ➤
  ➤ protected
  ➤
  ➤   def authorize
  ➤     unless User.find_by(id: session[:user_id])
  ➤       redirect_to login_url, notice: "Please log in"
  ➤     end
  ➤   end
end
```

# Rails *Callback*

- Temos que todas as chamadas aos controladores da aplicação irão gerar “antes” uma chamada ao método “authorize”
  - Algum problema?
- Podemos marcar apenas os métodos que requerem a autenticação (*blacklist*)
- Outra alternativa é marcar os métodos que não irão requerer autenticação (*whitelist*)

# Rails *Callback*

- Utilizando a diretiva **skip\_before\_action()**

Download rails40/depot\_r/app/controllers/store\_controller.rb

```
class StoreController < ApplicationController  
➤ skip_before_action :authorize
```

---

Download rails40/depot\_r/app/controllers/sessions\_controller.rb

```
class SessionsController < ApplicationController  
➤ skip_before_action :authorize
```



# Rails *Callback*

Download rails40/depot\_r/app/controllers/carts\_controller.rb

```
class CartsController < ApplicationController
  ➤ skip_before_action :authorize, only: [:create, :update, :destroy]
  # ...
  private
  # ...

  def invalid_cart
    logger.error "Attempt to access invalid cart #{params[:id]}"
    redirect_to store_url, notice: 'Invalid cart'
  end
end
```

# Rails *Callback*

Download rails40/depot\_r/app/controllers/line\_items\_controller.rb

```
class LineItemsController < ApplicationController
```

```
➤ skip_before_action :authorize, only: :create
```

---

Download rails40/depot\_r/app/controllers/orders\_controller.rb

```
class OrdersController < ApplicationController
```

```
➤ skip_before_action :authorize, only: [:new, :create]
```

# Ajustando os Testes

- Teste simples – acessando
  - <http://localhost:3000/products>
  - O acesso é interceptado e redirecionado para o formulário de login
- Opss! Essa alteração acaba invalidando boa parte dos nossos testes funcionais
  - Isso pode ser corrigido criando um método **setup()** no **test\_helper** – utilizando métodos **login\_as()** e **logout()** como um dado usuário

# Ajustando os Testes

Download rails40/depot\_r/test/test\_helper.rb

```
class ActiveSupport::TestCase
  # ...

  # Add more helper methods to be used by all tests here...
  def login_as(user)
    session[:user_id] = users(user).id
  end

  def logout
    session.delete :user_id
  end

  def setup
    login_as :one if defined? session
  end
end
```

Iteração 14:

# **ADICIONANDO UMA BARRA LATERAL**

# Evoluindo a Tela de Administração

- Que tal adicionar links para uma série de tarefas administrativas?
  - Definir uma barra lateral

Download rails40/depot\_r/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">
      <% if @cart %>
        <%= hidden_div_if(@cart.line_items.empty?, id: 'cart') do %>
          <%= render @cart %>
        <% end %>
      <% end %>
    <% end %>

    <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>
```

# Evoluindo a Tela de Administração

```
>
>
>     <% if session[:user_id] %>
>         <ul>
>             <li><%= link_to 'Orders',    orders_path    %></li>
>             <li><%= link_to 'Products',  products_path  %></li>
>             <li><%= link_to 'Users',     users_path     %></li>
>         </ul>
>         <%= button_to 'Logout', logout_path, method: :delete %>
>     <% end %>
> </div>
> <div id="main">
>     <%= yield %>
> </div>
> </div>
</body>
</html>
```



# Resultado

The screenshot shows a web browser window with the following elements:

- Browser Tab:** Pragprog Books Online Store
- Address Bar:** localhost:3000/users
- Header:** Pragmatic Bookshelf logo and the text "PRAGMATIC BOOKSHELF" in green.
- Left Sidebar (Dark Green):**
  - [Home](#)
  - [Questions](#)
  - [News](#)
  - [Contact](#)
  - [Orders](#)
  - [Products](#)
  - [Users](#)
  - [Logout](#) (button)
- Main Content Area:**
  - ## Listing users
  - Name**
  - dave [Show](#) [Edit](#) [Destroy](#)
  - [New User](#)

# Console Rails

- Para acessar a seção de administração é necessário que haja um usuário cadastrado
- Cadastro pode ser realizado pelo do console

```
depot> rails console
```

```
Loading development environment.
```

```
>> User.create(name: 'dave', password: 'secret', password_confirmation: 'secret')
```

```
=> #<User:0x2933060 @attributes={...} ... >
```

```
>> User.count
```

```
=> 1
```

- Para evitar que o problema se repita – impedimos que se remova o último usuário

# Evitando Remover o Último Usuário

- checagem se é foi removido o último usuário

```
Download rails40/depot_s/app/models/user.rb
```

```
after_destroy :ensure_admin_remains
```

```
private
```

```
  def ensure_admin_remains
```

```
    if User.count.zero?
```

```
      raise "Can't delete last user"
```

```
    end
```

```
  end
```

- Ao lançar uma exceção, acontece o “roll back” da “transação” do **Active Record**

# Evitando Remover o Último Usuário

- Capturando uma possível exceção

Download rails40/depot\_s/app/controllers/users\_controller.rb

```
def destroy
  ➤ begin
  ➤   @user.destroy
  ➤   flash[:notice] = "User #{@user.name} deleted"
  ➤ rescue StandardError => e
  ➤   flash[:notice] = e.message
  ➤ end
  respond_to do |format|
    format.html { redirect_to users_url }
    format.json { head :no_content }
  end
end
```