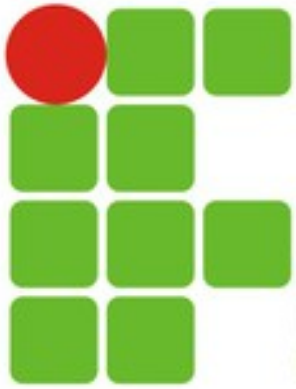


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Efetuando Login
(Tarefa I)

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

Feedback do Cliente

- Maravilha! Temos um cliente feliz.

- Mas... Eu gostaria de poder “logar” no sistema para realizar as tarefas administrativas
  - Não precisa ser nada sofisticado demais! Autenticação de login e senha já seria suficiente

Iteração I1:

ADICIONANDO USUÁRIOS

Criando a Classe de Modelo

- Inicialmente, precisamos criar a classe de modelo que representará a tabela de banco
  - Responsável pelos atributos login e senha
  - Ao invés das senhas será armazenado o resultado de uma função *hash* (*digest*) da mesma

```
depot> rails generate scaffold User name:string password:digest
```

- Logo após, redamos a “migração”

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

Criando a Classe de Modelo

- Temos assim a classe de modelo criada

Download rails40/depot_r/app/models/user.rb

```
class User < ActiveRecord::Base
  validates :name, presence: true, uniqueness: true
  has_secure_password
end
```

- Dentre as checagens necessárias:
  - Se o nome está presente e é único
  - Se o usuário “possui uma senha segura”

Criptografando Senhas

- Para usar *digest* é necessário remover o comentário de uma dada linha do “Gemfile”

Download rails40/depot_r/Gemfile

```
# Use ActiveModel has_secure_password
gem 'bcrypt-ruby', '~> 3.0.0'
```

- Depois, instalar a “gema” equivalente

```
depot> bundle install
```

Aplicando os Ajustes Necessários

- O controlador dos usuários (*scafold*)
  - Métodos padrão: **index()**, **show()**, **new()**, **edit()**, **update()** e **delete()**
  - Como a senha será omitida por segurança, não há a necessidade do **show** (só o nome para mostrar)

Aplicando os Ajustes Necessários

Download rails40/depot_r/app/controllers/users_controller.rb

```
def create
  @user = User.new(user_params)

  respond_to do |format|
    if @user.save
      format.html { redirect_to users_url,
        notice: "User #{@user.name} was successfully created." }
      format.json { render action: 'show',
        status: :created, location: @user }
    else
      format.html { render action: 'new' }
      format.json { render json: @user.errors,
        status: :unprocessable_entity }
    end
  end
end
```

Aplicando os Ajustes Necessários

- O mesmo para o **update**

```
def update
  respond_to do |format|
    if @user.update(user_params)
      format.html { redirect_to users_url,
        notice: "User #{@user.name} was successfully updated." }
      format.json { head :no_content }
    else
      format.html { render action: 'edit' }
      format.json { render json: @user.errors,
        status: :unprocessable_entity }
    end
  end
end
```

Aplicando os Ajustes Necessários

- Na listagem dos usuários, os mesmos podem ser ordenados pelo nome

```
def index
  @users = User.order(:name)
end
```

- Depois das mudanças no controlador, é hora de ajustar os elementos de visão
  - Pata exibir os avisos da criação e atualização

Aplicando os Ajustes Necessários

Download rails40/depot_r/app/views/users/index.html.erb

```
<h1>Listing users</h1>
<% if notice %>
<p id="notice"><%= notice %></p>
<% end %>

<table>
  <thead>
    <tr>
      <th>Name</th>
      <th></th>
      <th></th>
      <th></th>
    </tr>
  </thead>
```

Aplicando os Ajustes Necessários

```
  <tbody>
    <% @users.each do |user| %>
      <tr>
        <td><%= user.name %></td>
        <td><%= link_to 'Show', user %></td>
        <td><%= link_to 'Edit', edit_user_path(user) %></td>
        <td><%= link_to 'Destroy', user, method: :delete,
        data: { confirm: 'Are you sure?' } %></td>
      </tr>
    <% end %>
  </tbody>
</table>

<br>

<%= link_to 'New User', new_user_path %>
```

Aplicando os Ajustes Necessários

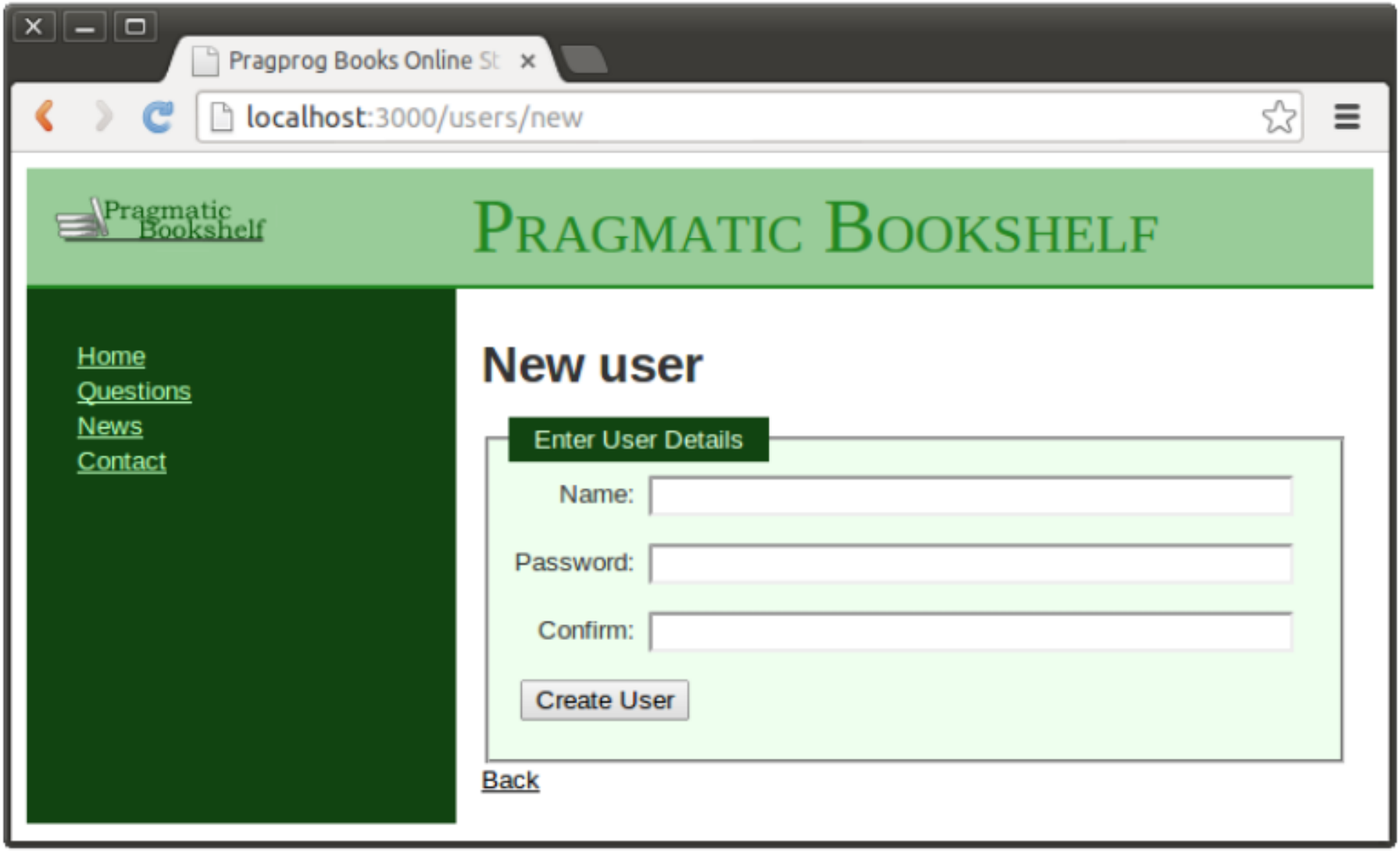
- Por fim, ajustando o formulário de criação/ atualização (***legend*** e ***fieldset***)

Download rails40/depot_r/app/views/users/_form.html.erb

```
<div class="depot_form">

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

```
  <fieldset>
  <legend>Enter User Details</legend>

  <div class="field">
    <%= f.label :name, 'Name:' %>
    <%= f.text_field :name, size: 40 %>
  </div>

  <div class="field">
    <%= f.label :password, 'Password:' %>
    <%= f.password_field :password, size: 40 %>
  </div>

  <div class="field">
    <%= f.label :password_confirmation, 'Confirm:' %>
    <%= f.password_field :password_confirmation, size: 40 %>
  </div>

  <div class="actions">
    <%= f.submit %>
  </div>

  </fieldset>
<% end %>

</div>
```

Resultado


Pragprog Books Online St
localhost:3000/users/new
Pragmatic Bookshelf
PRAGMATIC BOOKSHELF
Home
Questions
News
Contact
New user
Enter User Details
Name:
Password:
Confirm:
Create User
Back

Resultado

- No banco de dados:

```
depot> sqlite3 -line db/development.sqlite3 "select * from users"
             id = 1
           name = dave
password_digest = $2a$10$lki6/oAcOW4AWg4A0e0T8uxtri2Zx5g9taBXrd4mDSDVl3rQRWRNi
     created_at = 2013-01-29 14:40:06.230622
     updated_at = 2013-01-29 14:40:06.230622
```

Criação dos Testes

- Teste funcional #1

Download rails40/depot_r/test/controllers/users_controller_test.rb

```
test "should create user" do
  assert_difference('User.count') do
    post :create, user: { name: 'sam', password: 'secret',
      password_confirmation: 'secret' }
  end

  assert_redirected_to users_path
end
```

Criação dos Testes

- Teste funcional #2

```
test "should update user" do
  patch :update, id: @user, user: { name: @user.name, password: 'secret',
      password_confirmation: 'secret' }
  assert_redirected_to users_path
end
```

Criação dos Testes

- Fixtures:

Download rails40/depot_r/test/fixtures/users.yml

```
# Read about fixtures at
# http://api.rubyonrails.org/classes/ActiveRecord/Fixtures.html

one:
  name: dave
  password_digest: <%= BCrypt::Password.create('secret') %>

two:
  name: susannah
  password_digest: <%= BCrypt::Password.create('secret') %>
```

Iteração I2:

AUTENTICANDO USUÁRIOS

Autenticação

- O que significa adicionar suporte ao login de usuários para a aplicação da "loja"?
  - Criar um formulário para preenchimento das informações de login e senha
  - Uma vez "logado", uma forma de identificar o usuário deve ser salva na sessão do mesmo
  - Necessitamos restringir o acesso "administrativo" a certas partes da aplicação

Autenticação

- Dois controladores
  - Controlador de sessão para dar suporte ao login
  - Controlador para acompanhar os administradores

```
depot> rails generate controller Sessions new create destroy
depot> rails generate controller Admin index
```

- A ação **create** será responsável por armazenar uma informação na sessão que informe que um administrador está “logado”

Autenticação

- Será armazenado o id do usuário

Download rails40/depot_r/app/controllers/sessions_controller.rb

```
def create
  user = User.find_by(name: params[:name])
  if user and user.authenticate(params[:password])
    session[:user_id] = user.id
    redirect_to admin_url
  else
    redirect_to login_url, alert: "Invalid user/password combination"
  end
end
```

Formulário de Autenticação

Download rails40/depot_r/app/views/sessions/new.html.erb

```
<div class="depot_form">
  <% if flash[:alert] %>
    <p id="notice"><%= flash[:alert] %></p>
  <% end %>
  <%= form_tag do %>
    <fieldset>
      <legend>Please Log In</legend>
      <div>
        <%= label_tag :name, 'Name:' %>
        <%= text_field_tag :name, params[:name] %>
      </div>
      <div>
        <%= label_tag :password, 'Password:' %>
        <%= password_field_tag :password, params[:password] %>
      </div>
      <div>
        <%= submit_tag "Login" %>
      </div>
    </fieldset>
  <% end %>
</div>
```

Formulário de Autenticação

- Nota-se:
  - O formulário em questão não está associado e um elemento de modelo
  - Utilizado o **form_tag** → cria um <form> simples
  - Utilizados os seguintes campos: **text_field_tag** e **password_field_tag**
  - Os valores digitados são associados à estrutura que transporta os parâmetros → **params**

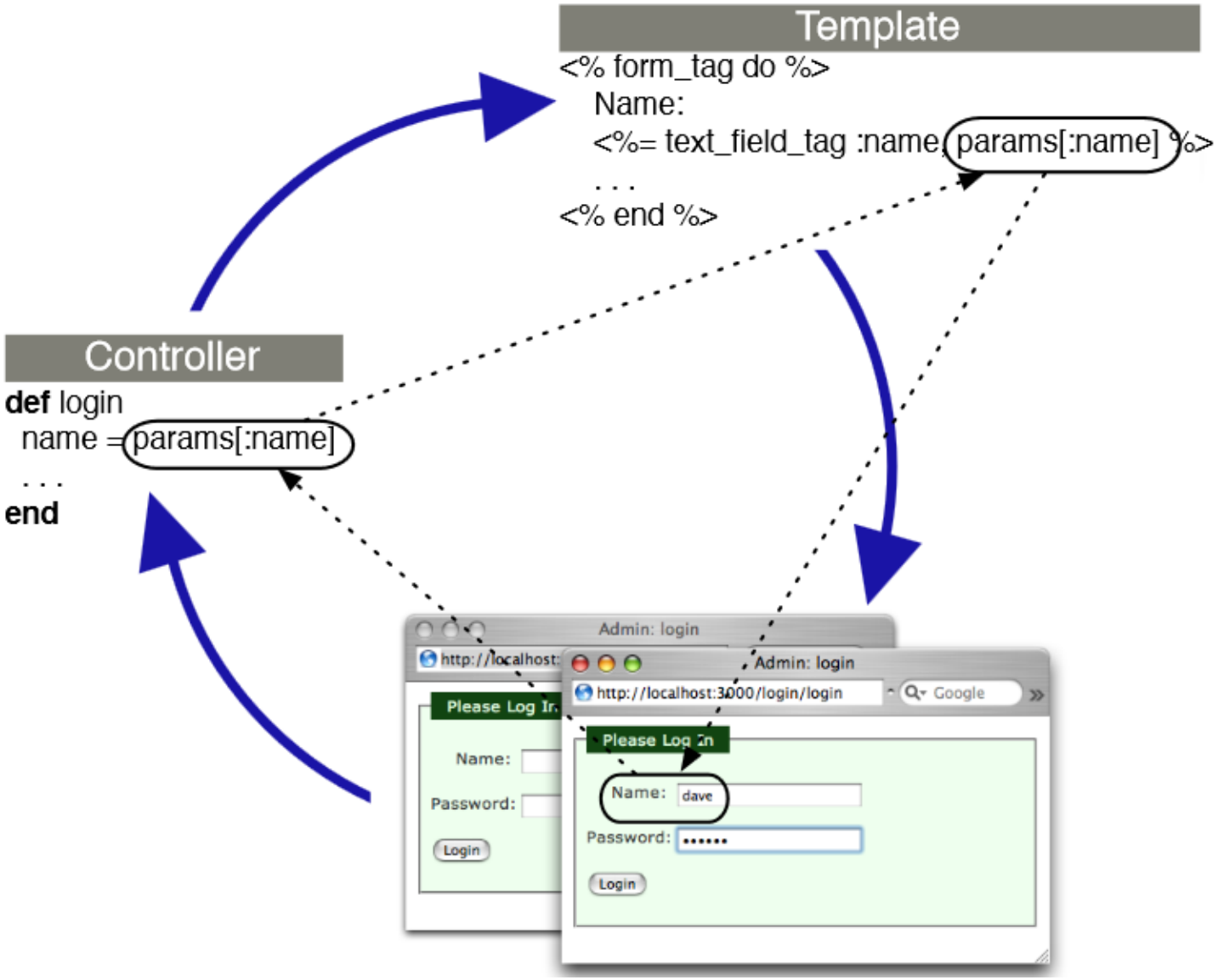
Logout

- Será utilizado o método **destroy**

Download rails40/depot_r/app/controllers/sessions_controller.rb

```
def destroy
  session[:user_id] = nil
  redirect_to store_url, notice: "Logged out"
end
```

Fluxo dos Parâmetros



Tela Inicial do Administrador

Download rails40/depot_r/app/views/admin/index.html.erb

```
<h1>Welcome</h1>

It's <%= Time.now %>
We have <%= pluralize(@total_orders, "order") %>.
```

- Utilização do *helper* **pluralize** – gera o *string* “order” ou “orders” dependendo da cardinalidade do primeiro parâmetro

Tela Inicial do Administrador

- O controlador se encarrega de definir a variável de instância – utilizada na visão

Download rails40/depot_r/app/controllers/admin_controller.rb

```
class AdminController < ApplicationController
  def index
    @total_orders = Order.count
  end
end
```

Tela Inicial do Administrador

- Precisamos realizar alguns ajustes
  - Preparar as rotas para o controlador que auxiliará o administrador
  - Remover as rotas **sessions/new**, **sessions/create**, e **sessions/destroy**

Download rails40/depot_r/config/routes.rb

```
Depot::Application.routes.draw do
  get 'admin' => 'admin#index'
  controller :sessions do
    get  'login' => :new
    post 'login' => :create
    delete 'logout' => :destroy
  end

  get "sessions/create"
  get "sessions/destroy"
  resources :users
  resources :orders
  resources :line_items
  resources :carts
```

Tela Inicial do Administrador

```
  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
```

Tela Inicial do Administrador

Ajustando os Testes Funcionais

Download rails40/depot_r/test/controllers/sessions_controller_test.rb

```
require 'test_helper'

class SessionsControllerTest < ActionController::TestCase
  test "should get new" do
    get :new
    assert_response :success
  end

  test "should login" do
    dave = users(:one)
    post :create, name: dave.name, password: 'secret'
    assert_redirected_to admin_url
    assert_equal dave.id, session[:user_id]
  end
```

Ajustando os Testes Funcionais

```
➤   test "should fail login" do
➤     dave = users(:one)
➤     post :create, name: dave.name, password: 'wrong'
➤     assert_redirected_to login_url
➤   end

➤   test "should logout" do
➤     delete :destroy
➤     assert_redirected_to store_url
➤   end
  end
```