

EPI: Interfejs Graficzny 2011/2012

Laboratorium nr 5

Uwierzytelnianie z użyciem gemu

Authlogic

Aleksander Pohl

14 grudnia 2011

Plan prezentacji

Wstęp

User

UserSession

Wprowadzenie

- ▶ uwierzytelnianie, autentykacja – zweryfikowanie, że osoba jest tym, za kogo się podaje
- ▶ autoryzacja, kontrola dostępu – zweryfikowanie, że osoba wykonująca określoną operacją, ma do tego prawo

Uwierzytelnianie:

- ▶ Authlogic
- ▶ Devise
- ▶ restful-authentication

Autoryzacja:

- ▶ cancan
- ▶ declarative_authorized
- ▶ acl9

Authlogic

- ▶ prawie najbardziej popularny
- ▶ wysoce konfigurowalny
- ▶ działa z Railsami v. 3
- ▶ spore możliwości, np.
 - ▶ trwała sesja
 - ▶ resetowanie hasła przez e-mail
 - ▶ różne algorytmy hashowania
- ▶ dodatki
 - ▶ OpenID
 - ▶ LDAP
 - ▶ Facebook
 - ▶ OAuth
 - ▶ PAM

Konfiguracja

1. Gemfile
gem 'authlogic'
2. ▷ bundle install

Modele

- ▶ `UserSession`
 - ▶ zawiera logikę związaną z obsługą sesji (np. sposób jej przechowywania)
 - ▶ dziedziczy z `Authlogic::Session::Base`
- ▶ `User`
 - ▶ jest zwykłym modelem railsowym definiowanym przez użytkownika
 - ▶ może dziedziczyć np. z `ActiveRecord::Base`
 - ▶ posiada pola takie jak: login, hasło, e-mail

Kontrolery

- ▶ `UserSessionsController`
 - ▶ zarządza sesją użytkownika
 - ▶ pozwala na zalogowanie się i wylogowanie z aplikacji
- ▶ `UsersController`
 - ▶ zarządza tworzeniem użytkowników i ich modyfikacją
- ▶ `ApplicationController`
 - ▶ posiada dodatkowe metody wykorzystywane w innych kontrolerach, które służą np. do sprawdzenia, czy użytkownik jest zalogowany

Plan prezentacji

Wstęp

User

UserSession

Model User

```
1. rails g model user login:string email:string  
   ↪ crypted_password:string admin:boolean  
   ↪ persistence_token:string
```

```
2. rake db:migrate
```

```
3. app/models/user.rb
```

```
class User < ActiveRecord::Base  
  attr_protected :admin  
  acts_as_authentic do |config|  
    config.crypted_password_field = :crypted_password  
    config.require_password_confirmation = true  
  end  
end
```

Przykładowe dane

1. db/seeds.rb

```
user = User.new(:login => 'admin', :password => 'password',  
               :password_confirmation => 'passowrd',  
               :email => 'my@email.com')  
user.save  
user.update_attribute(:admin, true)
```

2. rake db:seed

Wygenerowanie kontrolera, akcje new, create

1. rails g controller users new create edit update

2. app/controllers/users_controller.rb

```
class UsersController < ApplicationController
  def new
    @user = User.new
  end
  def create
    @user = User.new(params[:user])
    if @user.save
      flash[:notice] = "Registration successful."
      redirect_to root_url
    else
      render :action => 'new'
    end
  end
end
```

Akcje edit, update

1. app/models/users_controller.rb

```
class UsersController < ApplicationController
  def edit
    @user = current_user
  end
  def update
    @user = current_user
    if @user.update_attributes(params[:user])
      flash[:notice] = "Successfully updated profile."
      redirect_to root_url
    else
      render :action => 'edit'
    end
  end
end
```

Trasowanie

1. config/routes.rb

```
AuthlogicV3::Application.routes.draw do
  resources :users, :only => [:new, :create, :edit, :update]
  # get "users/new"
  # get "users/edit"
  # ...
end
```

2. ▷ rm app/views/users/create.html.erb

3. ▷ rm app/views/users/update.html.erb

Widok new

1. app/views/users/new.html.erb

```
<%= form_for @user do |f| %>
  <!-- kod wyświetlający błędy -->
  <p>
    <%= f.label :login %><br />
    <%= f.text_field :login %>
  </p>
  <p>
    <%= f.label :email %><br />
    <%= f.text_field :email %>
  </p>
  <p>
    <%= f.label :password %><br />
    <%= f.password_field :password %>
  </p>
  <p>
    <%= f.label :password_confirmation %><br />
    <%= f.password_field :password_confirmation %>
  </p>
  <p><%= f.submit "Submit" %></p>
<% end %>
```

2. app/views/users/edit.html.erb

– podobnie, ale np. nie powinien pozwalać na zmianę loginu



Plan prezentacji

Wstęp

User

UserSession

Utworzenie modelu i kontrolera

1. app/models/user_session.rb

```
class UserSession < Authlogic::Session::Base
  login_field :login

  def to_key
    new_record? ? nil : [ self.send(self.class.primary_key) ]
  end
end
```

2. rails g controller user_sessions new create

↔ destroy

3. config/routes.rb

```
AuthlogicV3::Application.routes.draw do
  resource :user_session, :only => [:new, :create, :destroy]
  #get "user_session/new"
  #get "user_session/create"
  #get "user_session/destroy"
  # Trzeba skonfigurowac strone startowa, np.
  root :to => "books#index"
end
```

4. rm app/views/user_sessions/create.html.erb



Kontroler UserSessionsController

```
app/controllers/user_sessions_controller.rb
```

```
class UserSessionsController < ApplicationController
  def new
    @user_session = UserSession.new
  end
  def create
    @user_session = UserSession.new(params[:user_session])
    if @user_session.save
      flash[:notice] = "Successfully logged in."
      redirect_to root_url
    else
      render :action => 'new'
    end
  end
  def destroy
    @user_session = UserSession.find
    @user_session.destroy
    flash[:notice] = "Successfully logged out."
    redirect_to root_url
  end
end
```



Widok new

app/views/user_sessions/new.html.erb

```
<%= form_for @user_session do |f| %>
  <!-- komunikaty błędów -->
  <p>
    <%= f.label :login %><br />
    <%= f.text_field :login %>
  </p>
  <p>
    <%= f.label :password %><br />
    <%= f.password_field :password %>
  </p>
  <p><%= f.submit "Submit" %></p>
<% end %>
```

Kontroler ApplicationController

```
app/controllers/application_controller.rb
class ApplicationController < ActionController::Base
  protect_from_forgery
  helper_method :current_user

  private
  def current_user_session
    return @current_user_session if defined?(@current_user_session)
    @current_user_session = UserSession.find
  end
  def current_user
    return @current_user if defined?(@current_user)
    @current_user = current_user_session && current_user_session.record
  end
  def authenticate
    if !current_user
      redirect_to new_user_session_path
    end
  end
end
```



Zabezpieczenie dostępu do akcji

- ▶ jeśli chcemy aby użytkownik mógł wykonać akcje kontrolera jeśli jest zalogowany dodajemy filtr `before_filter`, który wywołuje akcję `authenticate` z kontrolera `ApplicationController`

```
class AuthorsController < ApplicationController
  before_filter :authenticate
  #...
end
```

Jej działanie może być ograniczone do określonych akcji – możemy skorzystać z opcji `:only` (tylko te akcje będą chronione) lub `:except` (te akcje nie będą chronione):

```
class AuthorsController < ApplicationController
  before_filter :authenticate, :except => [:index, :show]
  #...
end
```

Autoryzacja akcji

- ▶ Jeśli chcemy aby określone akcje były dostępne tylko dla administratora, możemy dodać odpowiednią metodę w `application_controller.rb`

```
class ApplicationController < ActionController::Base
  #...
  private
  def admin_required
    if !current_user
      redirect_to new_user_session_path
    elsif !current_user.admin?
      flash[:notice] = "You are not allowed to do that."
      redirect_to root_url
    end
  end
end
```

- ▶ Następnie używamy jej tak samo jak metody `authenticate`, tzn. korzystając z makra `before_filter`.

Zadanie

- ▶ Dodaj do layout linki do zalogowania/wylogowania się z aplikacji
- ▶ Zabezpiecze metody modyfikujące autorów i książki przed niezalogowanymi użytkownikami
- ▶ Dodaj link dla zalogowanych użytkowników pozwalający na zmianę hasła i adresu e-mail

Materiały

- ▶ Dokumentacja
`rdoc.info/projects/binarylogic/authlogic`
- ▶ Railscast (dla Rails 2.3.x)
`railscasts.com/episodes/160-authlogic`

Dziękuję!