Rails Authentication with Omniauth (Document Version 1.0)

This development guide provides login functionality using Omniauth, which allows for authentication with third parties (e.g. Facebook, Google). Here the emphasis is on the Identity strategy, which stores login information (including encrypted password) in the local database. However, the Identity strategy can be swapped out to use strategies involving third parties. This guide complements the instructions in chapter 14 of Learning Rails 3 and uses code consistent with the approach presented there. Unlike the presentation in chapter 14, this version makes use of customized forms for signing in and registering. This approach is also explained in Railscast episode 304.

Installation

Installing Omniauth with the Identity strategy requires 2 gems: bcrypt-ruby and omniauth-identity. These gems are installed by editing **Gemfile** so that the bcrypt-ruby line is uncommented and the gem for omniauth-identity is added:

```
gem 'bcrypt-ruby', '~> 3.0.0'
gem 'omniauth-identity'
```

After editing this file, running **bundle install** at the command line will install the needed gems.

Finally, a new file called omniauth.rb needs to be created in the config/initializers folder so that it has the following contents:

```
Rails.application.config.middleware.use OmniAuth::Builder do
 provider :identity, on_failed_registration: lambda { |env|
    IdentitiesController.action(:new).call(env)
  }
 OmniAuth.config.on_failure = Proc.new { |env|
    OmniAuth::FailureEndpoint.new(env).redirect_to_failure
  }
end
```

When Rails starts, this line indicates that the Identity strategy should be used. It provides additional information on what to do if the registration fails and if the login fails.

This file will need to be edited if third party login is later added.

Models

Omniauth with Identity requires two models: the Identity model and the User model. The Identity model includes the user name and encrypted password. Consistent with the text, the email attribute can also be added. If a third party login is used, it will be used in place of the Identity model. The User model holds all application-dependent information about the user. Both models can be created with the generate command:

```
rails g model identity name:string email:string
password_digest:string
rails g scaffold user provider:string uid:string name:string
rake db:migrate
```

The model file app/models/identity.rb file needs to be added so that it inherits from the Identity class and so that it handles two virtual attributes: :password and :password_confirmation (note that the text has typos in this statement):

```
class Identity < OmniAuth::Identity::Models::ActiveRecord

  attr_accessible :email, :name, :password_digest,
                  :password, :password_confirmation
  validates :email, :uniqueness => true
end
```

Omniauth Identity uses the email address for login. The validation here ensures that there is only one email address. You could also add additional validations to ensure that all attributes are correctly set.

Note: the text on pp. 239 and 240 adds a create_with_omniauth method, but here we just added the needed code to the action where the session is created.

Routing

Even though Omniauth Identity has its own routes and paths for registering users and logging in, we will create our own. We create session routes for presenting the login form (new), creating the session (create), ending the session (destroy) and what to do if login fails (failure). These lines should be added to the routes.rb file:

```
match "/login" => "sessions#new", :as => :login
match "/auth/:provider/callback" => "sessions#create"
match "/logout" => "sessions#destroy", :as => :logout
match "/auth/failure" => "sessions#failure"
```

Finally, even though the Identity strategy provides routes, actions and views for creating new identities, we will want provide our own versions. For the route, we can add all needed routes with the resources statement in the routes.rb file:

```
resources :identities
```

Controllers

The controller for the User model has already been created with the scaffold generation.

Sessions Controller

Here we generate a controller for creating and ending sessions:

```
rails g controller sessions
```

The code for the Sessions Controller is at the end of this document.

Omniauth calls the create action for the Sessions controller if the credentials are correct. Omniauth also puts provides user information in the request.env table stored under "omniauth.auth". The Sessions create action then uses this information to either retrieve the corresponding User object (if it already exists) or creates a new User object. The id of the User object is then stored in the session.

The destroy action performs the logout function by setting the session to nil.

Identities Controller

The Identity strategy for Omniauth automatically supports registration, but it doesn't support cases when registration fails (e.g. the user provides an email address that isn't unique). Consequently, we need to provide our own Identities controller. Enter the following on the command line:

```
rails g controller identities
```

Fortunately, we only need to define one action in the identities controller (inside identities_controller.rb):

```
def new
  @identity = env['omniauth.identity']
end
```

Helpers

Method definitions for helpers can be placed in the file application_controller.rb (noted at the end of this document). With these helpers, a view can indicate who is logged in:

```
<%= current_user ? current_user.name : "Not logged in" %>
```

The following provide links to common authentication pages:

```
<%= link_to "Login", login_path %>

<%= link_to "Logout", logout_path %>

<%= link_to "Register", new_identity_path %>
```

Finally, the method check_login can be used as a before_filter to require that the user is logged in before the request goes any further.

Code for the Sessions Controller

```
class SessionsController < ApplicationController
  def create
    auth = request.env["omniauth.auth"]
    if (User.find_by_provider_and_uid(auth["provider"],
auth["uid"]))
     user = User.find_by_provider_and_uid(auth["provider"],
auth["uid"])
    else
      user = User.new
      user.provider = auth["provider"]
      user.uid = auth["uid"]
      user.name = auth["info"]["name"]
      user.save
    end
    session[:user_id] = user.id
    redirect_to root_url, :notice => "Welcome!"
  end

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

  def failure
    redirect_to login_path, alert: "Authentication failed, please
try again."
  end
end
```

Code for login form (views/sessions/new.html.erb)

```
<% if flash[:alert] %>
<p><%= flash[:alert] %></p>
<% end %>

<%= form_tag "/auth/identity/callback" do %>
  <div class="field">
    <%= label_tag :auth_key, "Email" %><br>
    <%= text_field_tag :auth_key %>
  </div>
  <div class="field">
    <%= label_tag :password %><br>
    <%= password_field_tag :password %>
  </div>
  <div class="actions"><%= submit_tag "Login" %></div>
<% end %>
```

Code for registration form (views/identities/new.html.erb)

```
<%= form_tag "/auth/identity/register" do %>
  <% if @identity && @identity.errors.any? %>
    <div class="error_messages">
      <h2><%= pluralize(@identity.errors.count, "error") %>
          prohibited this account from being saved:</h2>
      <ul>
      <% @identity.errors.full_messages.each do |msg| %>
        <li><%= msg %></li>
      <% end %>
      </ul>
    </div>
  <% end %>
  <div class="field">
    <%= label_tag :name %><br>
    <%= text_field_tag :name, @identity.try(:name) %>
  </div>
  <div class="field">
    <%= label_tag :email %><br>
    <%= text_field_tag :email, @identity.try(:email) %>
  </div>
  <div class="field">
    <%= label_tag :password %><br>
    <%= password_field_tag :password %>
  </div>
  <div class="field">
    <%= label_tag :password_confirmation %><br>
    <%= password_field_tag :password_confirmation %>
  </div>
  <div class="actions"><%= submit_tag "Register" %></div>
<% end %>
```

Helper definitions for application_controller.rb

```
helper_method :current_user

def current_user
   if session[:user_id]
     @current_user ||= User.find(session[:user_id])
   else
     @current_user = nil
   end
 end

def check_login
  unless logged_in?
    redirect_to login_path
  end
end

def logged_in?
  if session[:user_id]
    return true
  else
    return false
  end
end
```