

 RailsApps / rails-devise-roles

Watch 7

Star 61

Fork 27

Rails 4.2 starter app with Devise and simple role-based authorization.
<http://railsapps.github.io/rails-devise-roles>

35 commits





















2 branches


0 releases


2 contributors

 Branch: master rails-devise-roles / +



update to rails 4.2.4		
	DanielKehoe authored 10 days ago	latest commit e4b1e226ab 
	app	update to Rails 4.2.1 and current Rails Composer 4 months ago
	bin	bin a year ago
	config	update to rails 4.2.4 10 days ago
	db	before_filter becomes before_action 5 months ago
	lib	rails_apps_composer: initial commit a year ago
	public	update to rails 4.2.4 10 days ago
	spec	update to rails 4.2.4 10 days ago
	vendor/assets	rails_apps_composer: initial commit a year ago
	.gitignore	revert .gitignore a year ago
	.rspec	rails_apps_composer: testing framework a year ago
	.ruby-gemset	rails_apps_composer: create database a year ago
	.ruby-version	update ruby version 4 months ago
	Gemfile	update to rails 4.2.4 10 days ago
	Gemfile.lock	update to rails 4.2.4 10 days ago
	README	README a year ago
	README.textile	update to rails 4.2.4 10 days ago
	Rakefile	rails_apps_composer: initial commit a year ago
	config.ru	Rails 4.2 9 months ago

 README.textile



Rails Devise Roles

Rails 4.2 example application that provides user management, authentication, and simple role-based authorization. The application uses:

- Active Record Enum for roles
- Devise for user management and authentication

<> Code

Issues 1

Pull requests 0


Wiki


Pulse


Graphs

HTTPS clone URL

https://github.com



You can clone with HTTPS, SSH, or Subversion. 

 Clone in Desktop

 Download ZIP

- [Bootstrap](#) or [Foundation](#) front-end frameworks

Use this example application as a starter app for your own web applications.

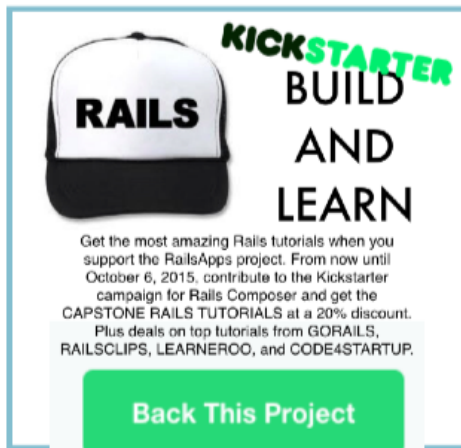
For a complete explanation of the code, see the tutorial:

- [Role-Based Authorization Tutorial](#)

Other tutorials may be helpful:

- [Rails and Devise](#)
- [RSpec Tutorial](#)

You can build this application in only a few minutes using the [Rails Composer](#) tool, choosing either a Bootstrap or Foundation front-end framework, as well as many other options, such as Haml or Slim.



What is the RailsApps Project?

The [RailsApps](#) project provides example applications that developers use as starter apps. Hundreds of developers use the apps, report problems as they arise, and propose solutions. Rails changes frequently; each application is known to work and serves as your personal “reference implementation.” Support for the project comes from subscribers. If this application is helpful to you, please [join the RailsApps project](#) to support our work.

If You Are New to Rails

If you’re new to Rails, see [What is Ruby on Rails?](#), the book [Learn Ruby on Rails](#), and recommendations for a [Rails tutorial](#).

What Is Implemented — and What Is Not

This application extends the [rails-devise](#) example application to add role-based authorization using Active Record Enum. The [rails-devise](#) example application provides:

- Home page
- Navigation bar
- Sign up (create account)
- Login
- “Forgot password?” feature
- “Remember me” (stay logged in) feature
- Edit account (edit user profile)
- List of users

This [rails-devise-roles](#) example application adds authorization, showing how to implement user roles, and limit access to pages based on user role. With this application:

- an admin can see a list of users
- an admin can change a user's role
- an ordinary user can't see a list of users
- an ordinary user can't change their role
- an ordinary user can't see (or edit) another user's profile
- an ordinary user can see (and edit) their own user profile

After examining the example application, you'll be able to implement access control in your own applications.

Database

The application requires a database. The example application uses SQLite with Rails ActiveRecord. You can easily substitute PostgreSQL, MySQL, or other databases.

Front-end Framework

The example application (here in the GitHub repository) integrates Bootstrap for a navigation bar and flash messages. The [rails_layout](#) gem is included so you can switch to the Foundation front-end framework.

Similar Examples and Tutorials

This is one in a series of Rails example apps and tutorials from the [RailsApps Project](#). See a list of additional [Rails examples](#), [tutorials](#), and [starter apps](#). Related example applications may be useful:

- [Learn Rails](#) companion to the book [Learn Ruby on Rails](#)
- [Foundation and Rails](#) shows how to integrate Foundation
- [Bootstrap and Rails](#) shows to integrate Bootstrap
- [OmniAuth and Rails](#) uses OmniAuth for authentication
- [Devise and Rails](#) uses Devise for authentication
- [Pundit and Rails](#) uses Pundit for authorization

Accounts You Will Need

Devise provides a "Forgot Password?" feature that resets a password and sends instructions to the user. You'll need an email service provider to send email from the application. You can use [Gmail](#) during development. You can get a free [Gmail](#) account if you don't already have one. For production, Gmail is not robust. Use transactional email services, such as [Mandrill](#), to send email in production. See the article [Send Email with Rails](#) for more information.

We provide instructions to deploy the application to [Heroku](#) which provides Rails application hosting. It costs nothing to set up a Heroku account and deploy as many applications as you want. To deploy an app to Heroku, you must have a Heroku account. Visit Heroku [to set up an account](#).

Dependencies

Before generating your application, you will need:

- The Ruby language – version 2.2
- The Rails gem – version 4.2

See the article [Installing Rails](#) for instructions about setting up Rails and your development environment.

Getting the Application

Local

You have several options for getting the code on your own machine. You can *fork*, *clone*, or *generate*.

Fork

If you'd like to add features (or bug fixes) to improve the example application, you can fork the GitHub repo and [make pull requests](#). Your code contributions are welcome!

Clone

If you want to copy and customize the app with changes that are only useful for your own project, you can clone the GitHub repo. You'll need to search-and-replace the project name throughout the application. You probably should generate the app instead (see below). To clone:

```
$ git clone git://github.com/RailsApps/rails-devise-roles.git
```

You'll need [git](#) on your machine. See [Rails and Git](#).

Generate

If you want to use the project as a starter application, use the [Rails Composer](#) tool to generate a new version of the example app. You'll be able to give it your own project name when you generate the app. Generating the application gives you additional options.

To build the example application, Rails 4.2 must be installed in your development environment. Run the command:

```
$ rails new rails-devise-roles -m https://raw.github.com/RailsApps/rails-composer/master
```

The `$` character indicates a shell prompt; don't include it when you run the command.

This creates a new Rails app named `rails-devise-roles` on your computer. You can use a different name if you wish.

You'll see a prompt:

```
option  Build a starter application?
        1) Build a RailsApps example application
        2) Contributed applications
        3) Custom application
```

Enter "1" to select **Build a RailsApps example application**. You'll see a prompt:

```
option  Choose a starter application.
        1) learn-rails
        2) rails-bootstrap
        3) rails-foundation
        4) rails-mailinglist-activejob
```

- 5) rails-omniauth
- 6) rails-devise
- 7) rails-devise-roles
- 8) rails-devise-pundit
- 9) rails-signup-download
- 10) rails-stripe-checkout

Choose **rails-devise-roles**. The Rails Composer tool may give you other options (other applications may have been added since these notes were written).

The application generator template will ask you for additional preferences:

```
option Web server for development?
  1) WEBrick (default)
  2) Thin
  3) Unicorn
  4) Puma
  5) Phusion Passenger (Apache/Nginx)
  6) Phusion Passenger (Standalone)
option Web server for production?
  1) Same as development
  2) Thin
  3) Unicorn
  4) Puma
  5) Phusion Passenger (Apache/Nginx)
  6) Phusion Passenger (Standalone)
option Database used in development?
  1) SQLite
  2) PostgreSQL
  3) MySQL
option Template engine?
  1) ERB
  2) Haml
  3) Slim
option Test framework?
  1) None
  2) RSpec with Capybara
option Front-end framework?
  1) None
  2) Twitter Bootstrap 3.3
  3) Twitter Bootstrap 2.3
  4) Zurb Foundation 5.5
  5) Zurb Foundation 4.0
  6) Simple CSS
setup The Devise 'forgot password' feature requires email.
option Add support for sending email?
  1) None
  2) Gmail
  3) SMTP
  4) SendGrid
  5) Mandrill
option Devise modules?
  1) Devise with default modules
  2) Devise with Confirmable module
  3) Devise with Confirmable and Invitable modules
option Use a form builder gem?
  1) None
  2) SimpleForm
option Install page-view analytics?
  1) None
  2) Google Analytics
  3) Segment.io
extras Set a robots.txt file to ban spiders? (y/n)
extras Create a GitHub repository? (y/n)
```

```
extras Use or create a project-specific rvm gemset? (y/n)
```

Web Servers

If you plan to deploy to Heroku, select Unicorn as your production webserver. Unicorn is recommended by Heroku.

Database

Use SQLite for development on Mac or Linux, unless you already have PostgreSQL installed locally. Use PostgreSQL if you plan to deploy to Heroku. You can easily change the database later if you select SQLite to start.

Template Engine

The example application uses the default “ERB” Rails template engine. Optionally, you can use another template engine, such as Haml or Slim. See instructions for [Haml and Rails](#).

Testing

The example application includes tests for RSpec with Capybara.

Front-end Framework

The example in the GitHub repository was built with the Bootstrap 3 front-end framework. Use Zurb Foundation 5 if you like. Choosing either Bootstrap or Foundation will automatically install views with attractive styling.

Email

Choose Gmail for development if you already have a Gmail account. Choose SendGrid or Mandrill for production if your site will be heavily used.

Devise Modules

The example in the GitHub repository uses Devise with default modules.

Other Choices

Set a robots.txt file to ban spiders if you want to keep your new site out of Google search results.

If you choose to create a GitHub repository, the generator will prompt you for a GitHub username and password.

It is a good idea to use [RVM](#), the Ruby Version Manager, and create a project-specific rvm gemset (not available on Windows). See [Installing Rails](#).

Troubleshooting

If you get an error “OpenSSL certificate verify failed” or “Gem::RemoteFetcher::FetchError: SSL_connect” see the article [OpenSSL errors and Rails](#).

Edit the README

If you’re storing the app in a GitHub repository, please edit the README files to add a description of the app and your contact info. If you don’t change the README, people will think I am the author of your version of the application.

Getting Started

See the article [Installing Rails](#) to make sure your development environment is prepared properly.

Use RVM

I recommend using [rvm](#), the Ruby Version Manager, to create a project-specific gemset for the application. If you generate the application with the Rails Composer tool, you can create a project-specific gemset.

Gems

Here are the gems used by the application:

- [Devise](#) for authentication and user management

These gems make development easier:

- [better_errors](#) – helps when things go wrong
- [quiet_assets](#) – suppresses distracting messages in the log
- [rails_layout](#) – generates files for an application layout

Your choice of front-end framework:

- [bootstrap-sass](#) – Bootstrap for CSS and JavaScript
- [foundation-rails](#) – Zurb Foundation for CSS and JavaScript

Install the Required Gems

If you used the [Rails Composer](#) tool to generate the example app, the application template script has already run the `bundle install` command.

If not, you should run the `bundle install` command to install the required gems on your computer:

```
$ bundle install
```

You can check which gems are installed on your computer with:

```
$ gem list
```

Keep in mind that you have installed these gems locally. When you deploy the app to another server, the same gems (and versions) must be available.

Front-end Framework

If you generate the application using the [Rails Composer](#) tool, you have the option to install either Bootstrap or Foundation. The folder **app/views/devise/** will contain attractive view files that override the views provided in the Devise gem.

Changing the Front-end Framework

The version of the application in the repository includes Bootstrap. If you wish to install Foundation instead, use the [rails_layout](#) gem to generate new files. First add a gem to the Gemfile:

```
gem 'foundation-rails'
```

Use Bundler to install the gem:

```
$ bundle install
```

To create layout files for use with Zurb Foundation 5.5:

```
$ rails generate layout:install foundation5
```

The “layout:devise” Command

Devise provides a utility command `rails generate devise:views`. The Devise command creates view files for signup, login, and related features. However, the views generated by Devise lack CSS styling.

Use the RailsLayout gem to generate Devise views with styling for Bootstrap or Foundation.

- `$ rails generate layout:devise bootstrap3`
- `$ rails generate layout:devise foundation5`

The command will create these files:

- `app/views/devise/sessions/new.html.erb`
- `app/views/devise/passwords/new.html.erb`
- `app/views/devise/registrations/edit.html.erb`
- `app/views/devise/registrations/new.html.erb`

Additionally, the command will update a file to append Sass mixins to accommodate Bootstrap or Foundation:

- `app/assets/stylesheets/framework_and_overrides.css.scss`

The Sass mixins allow any view to be used with either Bootstrap or Foundation (so we don’t have to maintain separate views for each front-end framework).

Configuration File

To consolidate configuration settings in a single location, we store credentials in the **config/secrets.yml** file. To keep your credentials private, use Unix environment variables to set your credentials. See the article [Rails Environment Variables](#) for more information.

Add your credentials to the file **config/secrets.yml**:

```
# Make sure the secrets in this file are kept private
# if you're sharing your code publicly.

development:
  admin_name: First User
  admin_email: user@example.com
  admin_password: changeme
  email_provider_username: <%= ENV["GMAIL_USERNAME"] %>
  email_provider_password: <%= ENV["GMAIL_PASSWORD"] %>
  domain_name: example.com
  secret_key_base: very_long_random_string
```



```
test:
  secret_key_base: very_long_random_string

# Do not keep production secrets in the repository,
# instead read values from the environment.
production:
  admin_name: <%= ENV["ADMIN_NAME"] %>
  admin_email: <%= ENV["ADMIN_EMAIL"] %>
  admin_password: <%= ENV["ADMIN_PASSWORD"] %>
  email_provider_username: <%= ENV["GMAIL_USERNAME"] %>
  email_provider_password: <%= ENV["GMAIL_PASSWORD"] %>
  domain_name: <%= ENV["DOMAIN_NAME"] %>
  secret_key_base: <%= ENV["SECRET_KEY_BASE"] %>
```

All configuration values in the **config/secrets.yml** file are available anywhere in the application as variables. For example, `Rails.application.secrets.email_provider_username` will return the string set in the Unix environment variable `GMAIL_USERNAME`.

For the Gmail username and password, enter the credentials you use to log in to Gmail when you check your inbox. See the article [Send Email with Rails](#) if you are using Google two factor authentication.

The values for `admin_email` and `admin_password` are used when the database is seeded. You will be able to log in to the application with these credentials. Note that it's not necessary to personalize the **config/secrets.yml** file before you deploy your app. You can deploy the app with an example user and then use the application's "Edit Account" feature to change email address and password after you log in. Use this feature to log in as an administrator and change the email and password to your own.

The variable `domain_name` is used for sending email. You can use `example.com` in development. If you already have a custom domain name you'll use when you deploy the application, you can set `domain_name`. If you deploy the application to Heroku, you'll set `domain_name` with the unique name you've given your application on Heroku. You'll have to wait until you deploy to know the name you'll use on Heroku.

If you don't want to use Unix environment variables, you can set each value directly in the **config/secrets.yml** file. The file must be in your git repository when you deploy to Heroku. However, you shouldn't save the file to a public GitHub repository where other people can see your credentials.

Roles

Roles are defined in the **app/models/User.rb** file (the `User` model).

```
class User < ActiveRecord::Base
  enum role: [:user, :vip, :admin]
  after_initialize :set_default_role, :if => :new_record?

  def set_default_role
    self.role ||= :user
  end

  # Include default devise modules. Others available are:
  # :confirmable, :lockable, :timeoutable and :omniauthable
  devise :database_authenticatable, :registerable,
         :recoverable, :rememberable, :trackable, :validatable
end
```

You can change the available roles by changing the array `[:user, :vip, :admin]`.

The application uses the ActiveRecord `enum` method to manage roles. ActiveRecord provides convenient methods to query the role attribute:

```
user.admin! # sets the role to "admin"
user.admin? # => true
user.role   # => "admin"
```

See documentation for [ActiveRecord::Enum](#) for details.

Database Seed File

The **db/seeds.rb** file initializes the database with default values.

```
# This file should contain all the record creation needed to seed the database with its
# The data can then be loaded with the rake db:seed (or created alongside the db with d
#
# Examples:
#
#   cities = City.create([ { name: 'Chicago' }, { name: 'Copenhagen' } ])
#   Mayor.create(name: 'Emanuel', city: cities.first)
user = CreateAdminService.new.call
puts 'CREATED ADMIN USER: ' << user.email
```

CreateAdminService is a service object that obtains `admin_email` and `admin_password` values from the **config/secrets.yml** file. You can examine the file **app/services/create_admin_service.rb** to see how a new user is created.

Set the Database

If you've used the Rails Composer tool to generate the application, the database is already set up with `rake db:migrate` and `rake db:seed`.

If you've cloned the repo, prepare the database and add the default user to the database by running the commands:

```
$ rake db:migrate
$ rake db:seed
```

Use `rake db:reset` if you want to empty and reseed the database.

If you're not using `rvn`, the Ruby Version Manager, you should preface each rake command with `bundle exec`. You don't need to use `bundle exec` if you are using `rvn` version 1.11.0 or newer.

Change your Application's Secret Token

If you've used the Rails Composer tool to generate the application, the application's secret token will be unique, just as with any Rails application generated with the `rails new` command.

However, if you've cloned the application directly from GitHub, it is crucial that you change the application's secret token before deploying your application in production mode. Otherwise, people could change their session information, and potentially access your site without permission. Your secret token should be at least 30 characters long and completely random.

Get a unique secret token:

```
rake secret
```

Edit the **config/secrets.yml** file to change the secret token.

Test the App

You can check that your application runs properly by entering the command:

```
$ rails server
```

To see your application in action, open a browser window and navigate to <http://localhost:3000/>.

You should see a home page with a navigation bar.

You should be able to click the navigation links for “Sign in” and “Sign up.”

Sign in with the credentials in the **config/secrets.yml** file. You’ll see a new navigation link for “Users.” Click the “Users” link and you’ll see a list of users (initially, just yourself). Log out and create a new account with the “Sign up” navigation link.

Sign in with a new account. Try clicking the “Users” link and you’ll be denied access (you are not an admin). Sign out.

Sign in with the first account (the admin account). Click the “Users” link. You’ll see both users. Change the role for the second user.

You’ve seen a simple demonstration of access control.

Stop the server with Control-C. If you test the app by starting the web server and then leave the server running while you install new gems, you’ll have to restart the server to see any changes. The same is true for changes to configuration files in the config folder. This can be confusing to new Rails developers because you can change files in the app folders without restarting the server. Stop the server each time after testing and you will avoid this issue.

RSpec Test Suite

The application contains a suite of RSpec tests. To run:

```
$ rspec
```

Deploy to Heroku

Heroku provides low cost, easily configured Rails application hosting.

From the Command Line

You can deploy from the command line.

```
$ git push origin master
```

If you’ve set configuration values in the **config/secrets.yml** file, you’ll need to set them as Heroku environment variables. You can set Heroku environment variables directly with `heroku config:add`.

For example:

```
$ heroku config:add ADMIN_NAME='First User'
$ heroku config:add ADMIN_EMAIL='user@example.com' ADMIN_PASSWORD='changeme'
$ heroku config:add GMAIL_USERNAME='myname@gmail.com' GMAIL_PASSWORD='secret'
$ heroku config:add DOMAIN_NAME='example.com'
```

See the [Tutorial for Rails on Heroku](#) for details.

Troubleshooting

Problems? Check the [issues](#).

Issues

Please create a [GitHub issue](#) if you identify any problems or have suggestions for improvements.

Where to Get Help

Your best source for help with problems is [Stack Overflow](#). Your issue may have been encountered and addressed by others.

Use the tag “railsapps” on Stack Overflow for extra attention.

Contributing

If you make improvements to this application, please share with others.

Send the author a message, create an [issue](#), or fork the project and submit a pull request.

If you add functionality to this application, create an alternative implementation, or build an application that is similar, please contact me and I'll add a note to the README so that others can find your work.

Credits

Daniel Kehoe implemented the application and wrote the tutorial.

Is the app useful to you? Follow the project on Twitter: [@rails_apps](#) and tweet some praise. I'd love to know you were helped out by what I've put together.

MIT License

[MIT License](#)

Copyright ©2014-15 Daniel Kehoe

Useful Links

Getting Started	Articles	Tutorials

Ruby on Rails	Analytics for Rails	Rails Bootstrap
What is Ruby on Rails?	Heroku and Rails	Rails Foundation
Learn Ruby on Rails	JavaScript and Rails	RSpec Tutorial
Rails Tutorial	Rails Environment Variables	Rails Devise Tutorial
Ruby on Rails Tutorial for Beginners	Git and GitHub with Rails	Devise RSpec
Install Ruby on Rails	Send Email with Rails	Devise Bootstrap
Install Ruby on Rails – Mac OS X	Haml and Rails	Rails Membership Site with Stripe
Install Ruby on Rails – Ubuntu	Rails Application Layout	Rails Subscription Site with Recurly
Ruby on Rails – Nitrous.io	HTML5 Boilerplate for Rails	Startup Prelaunch Signup Application
Update Rails	Example Gemfiles for Rails	
Rails Composer	Rails Application Templates	
Rails Examples	Rails Product Planning	
Rails Starter Apps	Rails Project Management	

