Skip to content

Search or jump to...

-

  In this repository All GitHub ↵
  Jump to ↵

- No suggested jump to results

- Octocat Spinner Icon

- Pull requests
- Issues
- Marketplace
- Explore

-
- ▶ +
- ▶

Sign out

- Watch 60
  × Notifications
  ✓
  ● Not watching Be notified when participating or @mentioned. Watch
  ✓
  ○ Watching Be notified of all conversations. Unwatch
  ✓
  ○ Ignoring Never be notified. Stop ignoring
- ★ Unstar 1,431
  ★ Star 1,431
- Fork
  316

RailsApps/rails_apps_composer

<> Code Issues 39 Pull requests 9 Projects 0 Wiki Insights
A gem with recipes to create Rails application templates for Rails starter apps.
http://railsapps.github.io/rails_apps…

- 1,287 commits

- 10 branches
- 5 releases
- 69 contributors

1. Ruby 91.2%
2. HTML 8.8%

Ruby HTML

Branch: master

✕ Switch branches/tags

Filter branches/tags

- Branches
- Tags

✓ anydiver-tutorial ✓ bruzos ✓ feature_megatest ✓ gh-pages ✓ master ✓ rails-3-0-7 ✓ rails-3-0-8 ✓ v1.5.5 ✓ v2.0.0 ✓ v2.6.13

Nothing to show

✓ v2.6.13 ✓ v2.2.42 ✓ rails3.1.3 ✓ rails3.0.8 ✓ rails3.0.7

Nothing to show

New pull request

Create new file

Upload files Find file

▶ Clone or download

DanielKehoe option to remove Rails Turbolinks for Rails 5

Latest commit 5ae32b2 May 16, 2018

Permalink

⚠ Failed to load latest commit information.

bin	tweak .rvmrc	Apr 26, 2012
lib	Fix uninitialized constant error for Tempfile	Nov 20, 2017
megatest	make Puma web server default for Rails 5	Oct 29, 2016
my	see CHANGELOG, bump version to 2.2.4	Aug 30, 2012
recipes	option to remove Rails Turbolinks for Rails 5	May 16, 2018
spec	fix specs to replace deprecated stub! and mock	Jul 12, 2013
templates	update CHANGELOG	Nov 20, 2017

.gitignore	Using MG instead of Bundler GemHelper since that was broken.	Mar 23, 2011
.rspec	Remove 1.9.1 from test group.	Apr 8, 2011
.ruby-gemset	switch from rvmrc to ruby-version	Mar 30, 2013
.ruby-version	remove 'rails_stripe_membership_saas' option from example applications	Feb 13, 2018
.travis.yml	[CI] Test against Ruby 2.4 and 2.5	Feb 18, 2018
CHANGELOG.textile	option to remove Rails Turbolinks for Rails 5	May 16, 2018
CONTRIBUTING.textile	fix typos	Jul 14, 2014
Gemfile	Travis CI, Version Badge, and secure Gemfile	Apr 9, 2013
README.textile	update readme	Oct 2, 2015
Rakefile	Megatest (minimally) working in Rails 4	Apr 3, 2014
old-Gemfile.lock	remove 'rails_stripe_membership_saas' option	Feb 21, 2018
rails_apps_composer.gemspec	update gemspec	Feb 21, 2018
sample.rb	rename 'after_everything' and 'after_bundler'	Jun 21, 2014
version.rb	option to remove Rails Turbolinks for Rails 5	May 16, 2018

README.textile

Rails Apps Composer

gem version 3.1.36 build passing

The Rails Apps Composer gem installs a command line tool to assemble a Rails application from a collection of “recipes.”

What you can do with the rails_apps_composer gem:

- create a starter app from the command line
- make a reusable application template

You can use local recipes as well as recipes supplied with the gem.

The Rails Composer tool is built with the rails_apps_composer gem. The Rails Composer tool creates the example applications for the RailsApps project.

The Guide to the Rails Apps Composer Gem provides complete documentation:

- Overview
- Recipes
- Usage
- Diagnostics
- Hacking the Gem
- Anatomy of a Recipe
- Architecture
- Defaults File

Any issues? Please create a GitHub issue.

What’s New

See the CHANGELOG for details.

If your favorite recipe is missing, please fork the repo, add the recipe and test it, then submit a pull request. I’m happy to add new recipes.

Who This Is For

This gem is for experienced Rails developers who need a tool to create and maintain a Rails starter app or Rails application template. If you’re regularly building Rails applications that use certain popular gems or rely on common functionality such as authentication, you can benefit from the rails_apps_composer gem.

If you are new to Rails, try the Rails Composer tool instead. It provides an application template that is generated from the rails_apps_composer gem’s core recipes. The Rails Composer tool is easier to use. Use the rails_apps_composer gem if you want a custom application template, a custom combination of recipes, or the convenience of setting a defaults file.

See the CONTRIBUTING file for details about how you can contribute fixes or new features.

Alternatives

The rails_apps_composer gem is derived from Michael Bleigh’s RailsWizard gem. Early versions of the rails_apps_composer gem were a close fork of Michael Bleigh’s gem;

rails_apps_composer 2.0 differs substantially. Other notable alternatives are Dr. Nic Williams's App Scrolls and Daniel Davey's app_drone. See a list of Rails Application Template Projects, particularly a list of "Application template-generating gems" for similar projects.

Dependencies

Before using the rails_apps_composer gem, you will need:

- The Ruby language (version 2.2)
- Rails (versions 4.2)

The rails_apps_composer program uses the version of Rails that is currently installed when you launch the program. If you want to use the newest Rails version, upgrade before running rails_apps_composer.

See the article Installing Rails for advice about updating Rails and your development environment. You'll avoid many potential headaches if you review the article step by step before beginning development.

Database

Unless you use SQLite, which runs by default on Mac OS X, you'll need to have a database server running before you run rails_apps_composer. This is true if you want to use PostgreSQL, MySQL, or MongoDB. I recommend using SQLite if you don't want to set up a database server.

Gemsets and RVM

Ordinarily, when you create a new Rails application, you work with a single set of gems. Before you create your Rails application, the rails gem must be installed in your Ruby environment. If you wish to use rails_apps_composer commands to generate a Rails application or template, you must also install the rails_apps_composer gem.

The rails_apps_composer gem uses the `rails new` command to create a new Rails application. The rails_apps_composer gem modifies the new Gemfile and runs Bundler to install the gems you want in your new application. The gems will be installed in your current Ruby environment (and must not conflict with other gems already installed).

Some developers like to use RVM, the Ruby Version Manager, to manage gemsets. The "extras" recipe offers an option of creating or using a project-specific RVM gemset. That means rails_apps_composer will switch from an initial gemset to a new gemset during the process of creating your new application. If you choose the option of using the project-specific RVM gemset from the "extras" recipe, before you start you must be using RVM and you must install the rvm gem in addition to the rails and rails_apps_composer gems.

If you choose the option of creating or using a project-specific RVM gemset, you will use two gemsets. First, the gems installed in the current Ruby environment (including rails and rails_apps_composer and perhaps rvm). Second, after the `rails new` command creates a Gemfile, and before Bundler installs the gems you want, rails_apps_composer will use the rvm gem to use or create a new gemset that is specific to your new project. Configuration choices you make when you run rails_apps_composer often add gems to the new application Gemfile (such as html2haml if you select Haml). Rails_apps_composer will use Bundler to install these gems and run any `generate` commands within the project-specific gemset.

If you answer “yes” to creating a project-specific gemset and you don’t have the rvm gem installed, you will get an error during the generation process saying RVM cannot be loaded. It is important to recognize that the gemset you start with (when you launch rails_apps_composer) can be replaced with a new project-specific gemset by the “build and compose” script (the “extras.rb” recipe line `RVM.gemset_use!` in the generated application template).

Installation and Usage

I recommend installing and using RVM, the Ruby Version Manager, to create a new gemset for rails_apps_composer. Using an RVM gemset will make it easier to identify and isolate incompatibilities among gems. RVM will install Ruby and set up a global gemset with a minimal set of gems.

Here’s how to generate a new Rails application using rails_apps_composer and RVM:

```
$ mkdir myapp
$ cd myapp
$ rvm use ruby-2.1.0@myapp --ruby-version --create
$ gem install rails
$ gem install rails_apps_composer
$ gem install rvm # (only needed if creating a project-specific rvm gemset)
$ rails_apps_composer new . -r core
```

This approach avoids conflicts with outdated gem versions that may be active in your shell environment.

Troubleshooting

You may have trouble if you install the rails_apps_composer gem by including it in a Gemfile and running Bundler (reported in issue 114).

Usage Options

These commands are summarized here for your reference. See the Guide to the Rails Apps Composer Gem for details.

List Recipes

You can list recipes with short descriptions:

```
$ rails_apps_composer list

controllers    # Add controllers needed for starter apps.
core           # Select all core recipes.
email          # Configure email accounts.
example        # Example of a recipe.
extras         # Various extras.
frontend       # Install a front-end framework for HTML5 and CSS.
gems           # Add the gems your application needs.
git            # Initialize git for your application.
init           # Set up and initialize database.
models         # Add models needed for starter apps.
railsapps      # Install RailsApps example applications.
readme         # Build a README file for your application.
routes         # Add routes needed for starter apps.
setup          # Make choices for your application.
testing        # Add testing framework.
views          # Add views needed for starter apps.
```

The above list is current for rails_apps_composer 2.1; later releases will contain more recipes.

See the Guide for more detailed descriptions of recipes. You can find all available recipes in the repository recipes directory. Examining the recipe source code is the very best way to learn what a recipe will do.

Recommended Recipes

I recommend selecting the *core* recipe if you are just getting started. The *core* recipe installs a collection of the most useful recipes.

Recipe Order and Interdependency

The order in which you input a list or recipes determines the order of execution unless a recipe contains a `requires` or `run_after` directive.

A recipe may contain a `requires` directive which specifies other recipes which must be present before a recipe can run. The `requires` constraint will force the rails_apps_composer program to load any required recipes, even if you don’t add them explicitly.

Recipes may also contain a `run_after` directive. The rails_apps_composer program will organize the order of execution so that any recipes in the `run_after` list will execute earlier.

That is, the recipe will run after everything else in the list.

In general, it's best to add (or list) recipes in the order they should execute as some recipes may not contain a necessary `requires` or `run_after` directive.

See the Anatomy of a Recipe section in the Guide to learn about the format of a recipe.

Skipping Test::Unit or Active Record

If you plan to use RSpec instead of Test::Unit, or use an ORM such as Mongoid instead of Active Record, you must pass the `-T` or `-O` flags to the program so it will skip Test::Unit or Active Record.

The rails_apps_composer program will ask if you want to skip Test::Unit or Active Record when you generate an application interactively.

Any recipe can set the `-T` or `-O` arguments using the `args` directive.

Also, you can set `-T` or `-O` arguments in the defaults file.

Setting `-T` or `-O` arguments has no affect on application templates. An application template runs after the `rails new` command so it cannot set the `-T` or `-O` arguments. When you generate an application template, the program will not ask if you want to skip Test::Unit or Active Record.

Using Local Recipes

You can use local recipes you've created yourself by using the `-l` flag and supplying the name of a directory that contains local recipes. The `-l` flag can be combined with any other command so you can create applications or application templates, interactively or from a defaults file.

For example, generate an application interactively using local recipes:

```
$ rails_apps_composer new myapp -l ~/recipes/
```

If you create local recipes, please consider contributing them to the project.

If you want to only use your local recipes, and not include the default recipes, use the `-L` flag.

Generate an Application Interactively

You'll be prompted for recipes and gems:

```
$ rails_apps_composer new myapp
```

```
Would you like to skip Test::Unit? (yes for RSpec) (y/n)
Would you like to skip Active Record? (yes for MongoDB) (y/n)
```

```
Available Recipes:
collections: core
configuration: email, gems, git, railsapps, readme, setup
example: example
frontend: frontend
initialize: init
mvc: controllers, models, routes, views
other: extras
testing: testing

Which recipe would you like to add? (blank to finish)
What gem would you like to add? (blank to finish)

Generating basic application, using:
"rails new myapp -m <temp_file>"
```

If you want to skip the prompts asking about Test::Unit and Active Record, you can set these arguments in a defaults file.

You will be able to choose recipes that are contained in the rails_apps_composer gem plus any local recipes you've added with the `-l` argument.

Choose the *core* recipe if you are not sure which recipes to select.

You can specify any gem. Any string you enter will be added as a gem in the starter app Gemfile.

If you want to skip the prompts asking which recipe or gem you'd like to add, use the "quiet flag" parameter on the command line by adding `-q`.

Generate an Application from a List of Recipes

Provide a list of recipes using the `-r` flag. In this example, the *core* recipe selects all available core recipes:

```
$ rails_apps_composer new myapp -r core

Would you like to skip Test::Unit? (yes for RSpec) (y/n)
Would you like to skip Active Record? (yes for MongoDB) (y/n)
What gem would you like to add? (blank to finish)

Generating basic application, using:
"rails new myapp -m <temp_file>"
```

The program will prompt you for your preferences before generating an application (unless you set the "quiet flag" `-q`).

Generate an Application from Defaults

Use a defaults file for recipes, preferences, and extra gems so you don't have to provide them interactively. Then generate an application using the `-d` flag:

```
$ rails_apps_composer new myapp -d my_defaults.yaml
```

Use the `my_defaults.yaml` file to specify a list of recipes, preferences, and extra gems. You can use any name (and file extension) for the file. See the Defaults File section in the Guide concerning the format of the defaults file.

Use the “quiet flag” `-q` if you want to skip all prompts:

```
$ rails_apps_composer new myapp -q -d my_defaults.yaml
```

Generate an Application Template Interactively

You may want an application template to share with others. For an example, see the Rails Composer project which is an application template generated from the rails_apps_composer core recipes.

Specify a filename for the template:

```
$ rails_apps_composer template ~/Desktop/template.rb

Available Recipes:
collections: core
configuration: email, gems, git, railsapps, readme, setup
example: example
frontend: frontend
initialize: init
mvc: controllers, models, routes, views
other: extras
testing: testing

Which recipe would you like to add? (blank to finish)
What gem would you like to add? (blank to finish)
Generating and saving application template...
Done.
```

The command with the `template` argument followed by a filename generates an application template. You can add additional recipes or gems when prompted.

Generate an Application Template from a List of Recipes

Specify a filename for the template and provide a list of recipes:

```
$ rails_apps_composer template ~/Desktop/template.rb -r core

What gem would you like to add? (blank to finish)
Generating and saving application template...
Done.
```

The command with the `template` argument followed by a filename and a list of recipes generates an application template. Use the “quiet flag” `-q` if you want to skip all prompts.

Generate an Application Template from Defaults

Generate an application template using a defaults file and the -d flag:

```
$ rails_apps_composer template ~/Desktop/template.rb -d my_defaults.yaml
```

Use the my_defaults.yaml file to specify a list of recipes, preferences, and extra gems. You can use any name (and file extension) for the file. See the Defaults File section in the Guide concerning the format of the defaults file. Use the “quiet flag” -q if you want to skip all prompts.

Generate an Application Template With Your Own Layout

If you want to generate an application template, but use your own layout to do so, you can use the -t flag to set the directory where your templates are. Look at the templates/ directory in this gem to see how your templates should look.

```
$ rails_apps_composer template ~/Desktop/template.rb -t ~/Desktop/my_templates
```

Generate an Application from a Template

After you’ve created a template file, you can generate an application from a template at any time using the rails new command with the -m option:

```
$ rails new myapp -m ~/Desktop/template.rb
```

The application template will prompt you for any configuration preferences requested by the recipes.

The rails new command allows you to specify the -T -O flags to skip Test::Unit files and Active Record files when you use an application template.

```
$ rails new myapp -m ~/Desktop/template.rb -T -O
```

The rails new command with the -m option also allows you to specify an application template that can be downloaded via HTTP. This makes it possible to host a template on GitHub (for example) and allow anyone to generate an application from the hosted template. For example:

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

Diagnostics

The rails_apps_composer program runs diagnostics to reveal whether you’ve picked recipes and preferences that are known to work together.

Hundreds of developers are using this gem to build starter apps. As you can see from the commit log, the gem is actively maintained and the collection of recipes is growing. There is a drawback to all this activity, however. No developer who uses the rails_apps_composer gem uses all the recipes. And it would be very difficult to test a recipe in every possible combination. Consequently, combining some recipes or preferences may not work.

The rails_apps_composer gem contains an internal database of combinations of recipes and preferences that are known to work together. If you've picked recipes and preferences that are known to work together, you'll get a confirmation:

```
WOOT! The recipes you've selected are known to work together.
If they don't, open an issue for rails_apps_composer on GitHub.
WOOT! The preferences you've selected are known to work together.
If they don't, open an issue for rails_apps_composer on GitHub.
```

You'll get a warning if you've picked a combination we don't know about:

```
WARNING! The recipes you've selected might not work together.
Help us out by reporting whether this combination works or fails.
WARNING! The preferences you've selected might not work together.
Help us out by reporting whether this combination works or fails.
```

Go to the GitHub repository to create an issue and let us know if you've found a combination of recipes or preferences that work together. If you've found a combination that works together, we'll update the gem's internal database for the next public release.

It's a good idea to include the "readme" recipe when you run rails_apps_composer. If you do, the application's **README** file will contain a diagnostics report. Here's an example of the diagnostics report:

```
Recipes:
["gems", "readme", "setup"]

Preferences:
{:database=>"sqlite", :templates=>"erb", :email=>"none"}
```

Please include the diagnostics report when you create an issue on GitHub.

Troubleshooting

Problems? You should review the article Installing Rails to make sure you've updated all the components that are required to run Rails successfully.

Problems with "Could not be loaded… You have already activated…"

If you get an error like this:

```
Your bundle is complete! Use `bundle show [gemname]` to see where a bundled gem is installed.
    composer  Running 'after bundler' callbacks.
The template [...] could not be loaded.
Error: You have already activated ..., but your Gemfile requires ....
Using bundle exec may solve this.
```

It's due to conflicting gem versions. See the article Rails Error: "You have already activated (…)".

Problems with "Certificate Verify Failed"

Are you getting an error "OpenSSL certificate verify failed" when you try to generate a new Rails app from an application template? See suggestions to resolve the error Certificate Verify Failed.

Problems with "Segmentation Fault"

If you get a "segfault" when you try `rails new`, try removing and reinstalling RVM.

Application Template Default

The `rails new` command creates a new Rails application. If you want to use an application template for every Rails application you build, you can set options for the `rails new` command in a **.railsrc** file in your home directory. Here's how to set up a **.railsrc** file to use the Rails Composer template when you create a new Rails application:

```
echo '-m https://raw.github.com/RailsApps/rails-composer/master/composer.rb' > ~/.railsrc
```

Hacking the Gem

See the Guide to the Rails Apps Composer Gem to learn how to customize the gem either for your own use or to share with others.

See the CONTRIBUTING file for details about how you can contribute fixes or new features.

Issues

Any issues? Please create an issue on GitHub. Reporting issues (and patching!) helps everyone.

If you include the *readme* recipe when you generate an application, the generated **README** file will contain a diagnostics report (unless the program aborts before the *readme* recipe runs). Please include the diagnostics report when you create an issue on GitHub.

Credits

Daniel Kehoe maintains this gem as part of the RailsApps project.

This gem is derived from Michael Bleigh's RailsWizard gem. The original idea and the innovative implementation is the work of Michael Bleigh.

Please see the CHANGELOG for a list of contributors.

Is the gem useful to you? Follow the project on Twitter: @rails_apps. I'd love to know you were helped out by the gem.

MIT License

MIT License

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	

- Security
- Status
- Help

- Contact GitHub
- Pricing
- API
- Training
- Blog
- About

⚠ ✕ You can’t perform that action at this time.
⚠ You signed in with another tab or window. Reload to refresh your session. You signed out in another tab or window. Reload to refresh your session.
Press h to open a hovercard with more details.