**Rails Environment Variables**

**by Taylor Mock and Daniel Kehoe**

*Last updated 15 December 2012*

Setting Rails environment variables. Using ENV variables in Rails, locally and with Heroku. A guide for developers using the example apps from the [RailsApps](http://railsapps.github.io/) project. Others may find it helpful as well.

**Environment Variables**

Many applications require configuration of settings such as email account credentials or API keys for external services. You can pass local configuration settings to an application using environment variables.

Operating systems (Linux, Mac OS X, Windows) provide mechanisms to set local [environment variables](http://en.wikipedia.org/wiki/Environment_variable), as does [Heroku](http://www.heroku.com/) and other deployment platforms. Here we show how to set local environment variables in the Unix shell. We also show two alternatives to set environment variables in your application without the Unix shell.

**Keeping Environment Variables Private**

Remote [git repositories](http://railsapps.github.io/rails-git.html) such as [GitHub](https://github.com/signup/free) are a place to store and share code. If your project is open source, any developer will have access to your code. You don’t want to share email account credentials or private API keys with the public. If you’re collaborating on a team with a private git repository, your local settings may not be suitable for all members of the team. In general, you shouldn’t save email account credentials or private API keys to a shared git repository.

**Gmail Example**

Consider an application that uses Gmail to send email messages. Access to Gmail requires a username and password for access to your Gmail account. In your Rails application, you will need to configure these credentials in the file **config/environments/production.rb**. A portion of the file might look like this:

|  |
| --- |
| config.action\_mailer.smtp\_settings = {  address: "smtp.gmail.com",  port: 587,  domain: "example.com",  authentication: "plain",  enable\_starttls\_auto: true,  user\_name: ENV["GMAIL\_USERNAME"],  password: ENV["GMAIL\_PASSWORD"]  } |

You could “hardcode” your Gmail username and password into the file but that would expose it to everyone who has access to your git repository. Instead use the Ruby variable ENV["GMAIL\_USERNAME"] to obtain an environment variable. The variable can be used anywhere in a Rails application. Ruby will replace ENV["GMAIL\_USERNAME"] with an environment variable.

Let’s consider how to set local environment variables.

**Option One: Set Unix Environment Variables**

If you’re familiar with Unix, you’ve likely had experience setting environment variables. Unix environment variables are typically set in a file that is read when starting an interactive shell (the **~/.bashrc** file for the bash shell).

Are you using a bash shell? Use echo $SHELL to find out. For a bash shell, using the Gmail example, edit the **~/.bashrc** file and add:

|  |
| --- |
| export GMAIL\_USERNAME="myname@gmail.com" |

You’ll have to open a new shell or restart your terminal application to continue.

It’s important to learn to use the Unix shell if you’re commited to improving your skills as a developer. But troubleshooting shell problems can be difficult, especially when using rvm, the [Ruby Version Manager](https://rvm.io/), or when the application is launched in nonstandard ways.

We’ll show you two alternatives to using the Unix shell that only require a few lines of code in your Rails application to set environment variables. Use one of these alternative approaches when you don’t want to set environment variables in the Unix shell.

**Option Two: Use the Figaro Gem**

The [figaro gem](https://github.com/laserlemon/figaro) offers a practical alternative to setting environment variables in the Unix shell.

The gem takes advantage of Ruby’s ability to set environment variables as well as read them. The gem reads a **config/application.yml** file and sets environment variables before anything else is configured in the Rails application.

If you want to use different credentials in development, test, or production environments, the gem will let you specify the different credentials in the **config/application.yml** file. If you’re deploying to Heroku, the gem gives you a rake task that will set all the environment variables on Heroku. The gem also gives you a syntax to access the environment variables as Figaro method calls which can be useful in setting up tests.

The figaro gem makes it so easy to set configuration variables you may find it useful to set other configuration parameters besides email account credentials or private API keys.

Here’s how to use it. In your **Gemfile**, add:

|  |
| --- |
| gem 'figaro' |

and run

|  |
| --- |
| $ bundle install |

The gem provides a generator:

|  |
| --- |
| $ rails generate figaro:install |

The generator creates a **config/application.yml** file and modifies the **.gitignore** file to prevent the file from being checked into a git repository.

You can add environment variables as key/value pairs to **config/application.yml**:

|  |
| --- |
| GMAIL\_USERNAME: Your\_Username |

The environment variables will be available anywhere in your application as ENV variables:

|  |
| --- |
| ENV["GMAIL\_USERNAME"] |

This gives you the convenience of using the same variables in code whether they are set by the Unix shell or the figaro gem’s **config/application.yml**. Variables in the **config/application.yml** file will override environment variables set in the Unix shell.

In tests or other situations where ENV variables might not be appropriate, you can access the configuration values as Figaro method calls:

|  |
| --- |
| Figaro.env.gmail\_username |

Use this syntax for setting different credentials in development, test, or production environments:

|  |
| --- |
| HELLO: world  development:  HELLO: developers  production:  HELLO: users |

In this case, ENV["HELLO"] will produce “developers” in development, “users” in production and “world” otherwise.

Figaro provides a rake task to set the environment variables in the **config/application.yml** file as Heroku environment variables:

|  |
| --- |
| rake figaro:heroku |

If you’re using one of the [RailsApps example applications](http://railsapps.github.io/) or you have generated a starter application using the [Rails Composer](http://railsapps.github.io/rails-composer/) tool, you’ll find a file named **config/application.yml**. This file was added by the figaro gem and contains key/value entries for each environment variable you may need. Each entry is commented out. Remove the comment character if you want to override environment variables from the Unix shell or set the environment variables using the **config/application.yml** file.

**Option Three: Use a *local\_env.yml* File**

Like the [figaro gem](https://github.com/laserlemon/figaro), this technique takes advantage of Ruby’s ability to set environment variables as well as read them. The figaro gem has additional features and we recommend it. However, if you want to understand how to set environment variables from within your Rails application, or you don’t want to add the figaro gem to your application, you may want to consider this implementation.

We’ll create a simple file that contains key/value pairs for each environment variable using the standard YAML file format. We’ll make sure this file is listed in the **.gitignore** file so it isn’t checked into the git repository. And we’ll add some code to the **config/application.rb** to read the file and set the environment variables before anything else is configured in the Rails application.

**The *local\_env.yml* File**

Create a file **config/local\_env.yml**:

|  |
| --- |
| # Rename this file to local\_env.yml  # Add account settings and API keys here.  # This file should be listed in .gitignore to keep your settings secret!  # Each entry gets set as a local environment variable.  # This file overrides ENV variables in the Unix shell.  # For example, setting:  # GMAIL\_USERNAME: 'Your\_Gmail\_Username'  # makes 'Your\_Gmail\_Username' available as ENV["GMAIL\_USERNAME"]  GMAIL\_USERNAME: 'Your\_Gmail\_Username' |

**Set .gitignore**

If you have created a git repository for your application, your application root directory should contain a file named **.gitignore** (it is a hidden file). Be sure your **.gitignore** file contains:

|  |
| --- |
| /config/local\_env.yml |

This prevents the **config/local\_env.yml** file from being checked into a git repository and made available for others to see.

**Rails Application Configuration File**

Rails provides the **config/application.rb** file for specifying settings for various Rails components. We want to set our environment variables before any other settings. Rails provides a config.before\_configuration method to do so. See the RailsGuide [Configuring Rails Applications](http://guides.rubyonrails.org/configuring.html) for more.

Find the following code at the end of the **config/application.rb** file:

|  |
| --- |
| # Version of your assets, change this if you want to expire all your assets  config.assets.version = '1.0' |

and add this code after it:

|  |
| --- |
| config.before\_configuration do  env\_file = File.join(Rails.root, 'config', 'local\_env.yml')  YAML.load(File.open(env\_file)).each do |key, value|  ENV[key.to\_s] = value  end if File.exists?(env\_file)  end |

The code opens the **config/local\_env.yml** file, reads each key/value pair, and sets environment variables.

The code only runs if the file exists. If the file exists, the code overrides ENV variables set in the Unix shell. If you prefer to set environment variables in the Unix shell, don’t create the **config/local\_env.yml** file.

**Using Environment Variables**

Use ENV["GMAIL\_USERNAME"] anywhere in a Rails application. Your application won’t know if it was loaded from the **config/local\_env.yml** file or from the Unix shell.

**Distinguishing Development From Test Environments**

Occasionally you’ll want to use different account credentials or API keys for test and development environments.

Give the variables different names, for example:

|  |
| --- |
| GMAIL\_USERNAME\_DEV: 'Your\_Gmail\_Username\_For\_Development'  GMAIL\_USERNAME\_TEST: 'Your\_Gmail\_Username\_For\_Tests' |

and use the variables conditionally:

|  |
| --- |
| if Rails.env.development?  config.action\_mailer.smtp\_settings = {  user\_name: ENV["GMAIL\_USERNAME\_DEV"]  }  end  if Rails.env.test?  config.action\_mailer.smtp\_settings = {  user\_name: ENV["GMAIL\_USERNAME\_TEST"]  }  end  if Rails.env.production?  config.action\_mailer.smtp\_settings = {  user\_name: ENV["GMAIL\_USERNAME"]  }  end |

This approach works with either the **config/local\_env.yml** file or environment variables obtained from the Unix shell.

**Setting Environment Variables on Heroku**

[Heroku](http://www.heroku.com/) is a popular choice for low cost, easily configured Rails application hosting. See the article [Heroku and Rails](http://railsapps.github.io/rails-heroku-tutorial.html) for details.

Heroku provides a simple mechanism for setting environment variables. After creating your Heroku app, set Heroku environment variables to provide the same data your application obtains from your local shell environment.

For example, for Gmail:

|  |
| --- |
| $ heroku config:add GMAIL\_USERNAME=myname@gmail.com |

If you have [multiple environments](https://devcenter.heroku.com/articles/multiple-environments) on Heroku:

|  |
| --- |
| $ heroku config:add GMAIL\_USERNAME=myname@gmail.com --remote staging |

You can check that everything has been added correctly by running:

|  |
| --- |
| $ heroku info --app myapp |

where “myapp” is the name of your Heroku application.

If you are using the figaro gem, just run:

|  |
| --- |
| rake figaro:heroku |

Notice that you don’t use the **config/application.yml** file. Your **.gitignore** file should prevent it from getting added to the git repository. It won’t exist when you deploy to Heroku and your application will obtain environment variables from the Heroku configuration table.

**Other Approaches**

There are other approaches to setting environment variables in Rails.

Tammer Saleh suggests a similar approach using the Rails **environment.rb** file in his blog posting, [Managing Heroku environment variables for local development](http://tammersaleh.com/posts/managing-heroku-environment-variables-for-local-development).

Brandon Keepers offers the [dotenv](https://github.com/bkeepers/dotenv) Ruby gem that handles loading environment variables in development. The dotenv gem loads environment variables from a **.env** file.

[Foreman](https://github.com/ddollar/foreman) is a tool for starting and configuring multiple processes in a complex application. Foreman will load environment variables from a **.env** file.

There are numerous gems such as [yettings](https://github.com/charlotte-ruby/yettings) that set application configuration variables or other constants from a YAML file and allow access the to values with a method such as AppConfig.gmail\_username. This approach is best used for constants that can be checked into a git repository and are not specific to a particular deployment. Local environment variables are better suited to deployment-specific configuration settings.

**Did You Like the Article?**

Was this useful to you? Follow [rails\_apps](http://twitter.com/rails_apps) on Twitter and tweet some praise. We’d love to know you were helped out by the tutorial.

Any issues? Please leave a comment below.