GitHub | This repository Search | Explore   Features   Enterprise   Blog   Sign up   Sign in

guard / **guard**   👁 Watch 113   ★ Star 4,277   ⑂ Fork 363

System notifications

irontoby edited this page on 23 Dec 2014 · 6 revisions

You can configure Guard to notify plugin results throgh one or more channels described below. If you do not specify a specific notifiation channel with the notification DSL method, then Guard auto-detects all available channels and makes use of them.

Ruby GNTP

- Runs on Mac OS X, Linux and Windows
- Supports Growl version `>= 1.3`, Growl for Linux, Growl for Windows and Snarl

The ruby_gntp gem sends system notifications over the network with the Growl Notification Transport Protocol and supports local and remote notifications. To have the images be displayed, you have to use `127.0.0.1` instead of `localhost` in your GNTP configuration.

Guard supports multiple notification channels for customizing each notification type. For Growl on Mac OS X you need to have at least version 1.3 installed.

To use `ruby_gntp` you have to add it to your `Gemfile` and run bundler:

```
group :development do
  gem 'ruby_gntp'
end
```

Growl

- Runs on Mac OS X
- Supports all Growl versions

The growl gem is compatible with all versions of Growl and uses a command line tool growlnotify that must be separately downloaded and installed. The version of the command line tool must match your Growl version. The `growl` gem does **not** support multiple notification channels.

You have to download the installer for `growlnotify` from the Growl download section.

To use `growl` you have to add it to your `Gemfile` and run bundler:

```
group :development do
  gem 'growl'
end
```

▸ **Pages 30**

Installation

- System Notifications
- Terminal Colors on Windows
- Add Readline support to Ruby on Mac OS X
- Which Growl library should I use
- Efficient Filesystem Handling

Getting Started

- List of Guard Commands
- Command Line Options for Guard
- List of available Guards
- Guardfile examples
- Run Guard within RubyMine
- Guardfile-DSL / Configuring-Guard
- Configuration Files
- Interacting with Guard
- Guard Signals

Troubleshooting

- Understanding Guard

Advanced use of Guard

- Shared configurations
- Hooks and callbacks
- Correctly using the --watchdir option

Cookbooks

- Use Guard programmatically cookbook
- Custom pry command cookbook

Development

- Create a Guard
- Programmatic use of Guard
- Upgrading to Guard 2.0
- Upgrade guide for existing guards to Guard v1.1
- Analysis of inotify events for

Libnotify

- Runs on Linux, FreeBSD, OpenBSD and Solaris
- Supports Libnotify

The libnotify gem supports the Gnome libnotify notification daemon, but it can be used on other window managers as well. You have to install the `libnotify-bin` package with your favorite package manager.

To use `libnotify` you have to add it to your `Gemfile` and run bundler:

```
group :development do
  gem 'libnotify'
end
```

If you are unable to build the `libnotify` gem on your system, Guard also has a built in notifier - `notifysend` - that shells out to the `notify-send` utility that comes with `libnotify-bin`.

Please note that NotifyOSD, which is the default notification UI on Ubuntu and possibly other systems, ignores the timeout parameter for libnotify.

Notifu

- Runs on Windows
- Supports Notifu

The rb-notifu gem supports Windows system tray notifications.

To use `rb-notifu` you have to add it to your `Gemfile` and run bundler:

```
group :development do
  gem 'rb-notifu'
end
```

Terminal Notifier

- Runs on Mac OS X 10.8 only

The terminal-notifier-guard sends notifications to the OS X Notification Center.

To use `terminal-notifier-guard` you have to add it to your `Gemfile` and run bundler:

```
group :development do
  gem 'terminal-notifier-guard'
end
```

Terminal Title

- Runs in every terminal supporting XTerm escape sequences to set the window title.

different editors

Clone this wiki locally

https://github.com/guard/gu

Emacs

- Runs on any platform with Emacs + emacsclient (http://www.emacswiki.org/emacs/EmacsClient)

TMux

- To use TMux notifications, you have to start Guard within a TMux session.

The TMux notifier will color the background of the left part of the status bar indicating the status of the notifications. Optionally you can set `:display_message => true` to display the Guard notification as 'display-message' notification.

The way these messages are formatted is configurable.

```
# Guardfile
notification :tmux,
  display_message: true,
  timeout: 5, # in seconds
  default_message_format: '%s >> %s',
  # the first %s will show the title, the second the message
  # Alternately you can also configure *success_message_format*,
  # *pending_message_format*, *failed_message_format*
  line_separator: ' > ', # since we are single line we need a separator
  color_location: 'status-left-bg', # to customize which tmux element will ch

  # Other options:
  default_message_color: 'black',
  success: 'colour150',
  failure: 'colour174',
  pending: 'colour179',

  # Notify on all tmux clients
  display_on_all_clients: false,
```

The color location option can also take an array:

```
color_location: %w[status-left-bg pane-active-border-fg pane-border-fg]
```

The result will be for RSpec using example above

```
RSpec >> 15 test, 0 failures > in 0.002 sec
```

You can use nice powerline chars here if you have that configured.

You can get the message history by using `Ctrl+b ~` (where `Ctrl+b` is your key to activate TMux).

File

- You can also have Guard write notifications to a file. Each notification will overwrite the file. This allows other commands to be run based on the status of other guard commands.

Example:

```
# Guardfile
notification :file, path: '.guard_result'

guard :shell do
  watch '.guard_result' do
    if File.read('.guard_result').lines.first.strip == 'failed'
      # ...
    end
  end
end
```

Configuration:

```
# Guardfile
notification :file,
  path: '.guard_result', # Required, no default
  format: "result: %s\ntitle: %s\nmessage: %s\n" # Default: "%s\n%s\n%s\n"
```

This wiki and the Guard README document contains a lot of information, please take your time and read these instructions carefully.

If you run into any trouble, you may start by understanding how Guard works.

We provide detailed changes for each Guard release.

Be sure to read the CONTRIBUTING guidelines before reporting a new Guard issue or open a pull request.

If you have any questions about the Guard usage or want to share some information with the Guard community, please go to one of the following places:

- Google+ community
- Google group
- StackOverflow
- IRC channel `#guard` (irc.freenode.net) for chatting