Ruby 初級者向けレッスン第 31 回

okkez @ Ruby 関西

2009 年 09 月 19 日

今回の内容

- RubyGems 入門

今回のゴール

- RubyGems のコマンドを知る
- RubyGems のパッケージの作成方法を知る

RubyGems とは

`RubyGems is a sophisticated package manager for Ruby.`
`RubyGems` は `Ruby` のための高機能なパッケージ管理ツールです。

Ruby1.9 から標準添付されています。Ruby1.9.1p243 に添付されているのは RubyGems 1.3.1 です。

何が出来るの？

- gem パッケージのインストール
  - gem install <package name>
- gem パッケージのアンインストール
  - gem uninstall <package name>
- gem パッケージの検索
  - gem search <pattern>
  - gem query <options>
- gem パッケージの作成
  - gem build <gemspecfile>

- gem パッケージの配布
- gem server
- gem パッケージ配布元の管理
  - gem source

などなど。

gem コマンドの使い方

```
$ gem help commands        # サブコマンドの一覧
$ gem help <command name> # 指定したコマンドのヘルプ
$ gem <command> --help     # 指定したコマンドのヘルプ
```

コマンドのヘルプは長いので pager にパイプで出力を渡すなどするといいです。

gem パッケージをインストールする

```
$ gem install rails
```

- 今は -y, –include-dependencies を指定しなくても依存関係を解決してくれます。

  $ gem install foo bar baz

- 上記のように複数のパッケージを一度に指定することもできます。

  $ gem install rails –no-rdoc –no-ri

- gem のインストールを高速化したい人は上記のようなオプションを付けるといいでしょう。
  - ただしドキュメントがインストールされなくなります。
- gem install は /usr/lib/ruby/gems/1.8/ 以下などにインストールしようとします。
  - インストールしようとした場所に書き込み権限が無い場合は~/.gem/ruby/1.8 以下にインストールします。
  - 権限が無い場合は警告メッセージを出力します。

~/.gemrc

~/.gemrc を以下の内容で作成することができます。また、システム全体に影響する設定を書きたい場合は /etc/gemrc を作成してください。YAML 形式のハッシュでキーはシンボルか文字列になるように書きます。

```
---
:verbose: true
:sources:
- http://gems.rubyforge.org/
- http://gems.github.com/
:backtrace: false
:benchmark: false
:update_sources: true
:bulk_threshold: 1000
:gem: --no-rdoc --no-ri
```

最後の :gem: –no-rdoc –no-ri を付けるとデフォルトでドキュメントをインストールしなくなります。他にも各サブコマンドで共通のオプションを書くことができます。

その他のオプションについてはリファレンスマニュアルを参照してください。

gem パッケージを検索する

普段は search の部分一致検索だけで十分です。デフォルトではローカルにインストールされているものを検索対象にするのでリモートリポジトリにインストールされているものを検索したい時は -r オプションを付けるのを忘れないようにしてください。パッケージの説明を表示するには -d オプションを使用してください。

詳細な条件を指定して検索したい場合は query を使ってください。

```
$ gem query -n ^rails$ -r # rails にちょうど一致するものを検索する
$ gem query -n ^rails -r  # rails で始まるものを検索する
```

gem パッケージを配布する

LAN 内などで自分が作成した gem パッケージを配布したいときは server を使用します。

```
$ gem server
... デフォルトでは localhost:8808 で起動するので ...
$ gem search --source http://localhost:8808/ foo -r # このように検索することができる
$ gem install --source http://localhost:8808/ foo   # このようにするとインストールできる
```

色々なオプションを指定することができるのでヘルプを参照してください。

gem パッケージを作成する

基本的には以下で作成できます。

```
$ gem build gemspec
```

gemspec の書き方

まず以下のようにして git リポジトリを clone してください。

```
$ git clone git://github.com/okkez/hello.git
```

git が使えない人はそれぞれのアーカイブを github からダウンロードすることができます。このリポジトリに含まれている gemspec の名前は hello.gemspec であるべきかもしれません。

Downloads for okkez's hello - GitHub
http://github.com/okkez/hello/downloads

最小の gemspec

```
$ git co -b minimum origin/minimum # などでブランチを切り替えてください。
```

gemspec は以下の通りです。

```
01: Gem::Specification.new do |s|
02:   s.name = 'hello'
03:   s.version = '0.0.0'
04:
05:   s.summary = 'hello summary'
06: end
```

この gemspec を元にして gem パッケージを作成すると空の gem パッケージができます。またこの時、警告メッセージがいくつか出力されます。

ライブラリの gemspec

```
$ git co -b lib origin/lib_only # などでブランチを切り替えてください。
```

gemspec は以下の通りです。

```
01: Gem::Specification.new do |s|
02:   s.name = 'hello'
03:   s.version = '0.0.0'
04:
05:   s.summary = 'hello summary'
06:   s.files = ['lib/hello.rb']
07:   s.authors = ['okkez']
08:   s.email = 'okkez000@gmail.com'
09:   s.homepage = 'http://example.com/hello/'
10:   s.description = 'hello description'
11:   s.rubyforge_project = 'hello'
12: end
```

警告が出ないようにしてありますが、最小のもにに加えて必須なのは s.files の指定のみです。これだけでライブラリとして使用することができます。

コマンドの gemspec

`$ git co -b bin origin/bin #` などでブランチを切り替えてください。

gemspec は以下の通りです。

```
01: Gem::Specification.new do |s|
02:   s.name = 'hello'
03:   s.version = '0.0.0'
04:
05:   s.summary = 'hello summary'
06:   s.files = ['bin/hello', 'lib/hello.rb']
07:   s.executables = ['hello']
08:   s.authors = ['okkez']
09:   s.email = 'okkez000@gmail.com'
10:   s.homepage = 'http://example.com/hello'
11:   s.rubyforge_project = 'hello'
12:   s.description = 'hello description'
13: end
```

ライブラリの場合と同じように警告が出ないようにしてあります。ライブラリの場合と違うのは s.executables を指定していることです。

gemspec の書き方のまとめ

- 最小のものを用意するのは意外と簡単
- オレオレ gem パッケージを作って自分だけでテストするのも簡単

その他のオプションについてはリファレンスマニュアルを参照してください。

class Gem::Specification
: http://doc.okkez.net/191/view/class/Gem=Specification

gem パッケージを公開する

以下のようなサイトがあります。

RubyForge: Welcome
: http://rubyforge.org/

GitHub
: https://github.com/

gemcutter | awesome gem hosting
: http://gemcutter.org/

いろいろなブログに記事がありますので、それぞれについて簡単に説明しておきます。

RubyForge

- 一番の老舗。ここに登録した gem はデフォルトで検索対象に含まれます。
- 登録申請が必要です。

手順

1. アカウントを登録する 2. 新規プロジェクトを申請する 3. 申請が通ったら開発する 4. 公開できるレベルになったら gem パッケージをローカルで作成してアップロードする

GitHub

- gem パッケージの公開が簡単です。gemspec ファイルを見て自動生成してくれます。
- 誰でも申請無しで gem パッケージを公開する事が可能です。
  - 「誰が」作ったのかを意識しないと変なものを掴む可能性もあります。

手順

GitHub RubyGems

http://gems.github.com/

1. アカウントを登録する 2. 新規プロジェクトを作成する 3. プロジェクトの設定ページで RubyGem にチェックを入れる 4. <project_name>.gemspec という名前で gemspec ファイルをプロジェクトの root ディレクトリに追加する 5. gemspec を push すると gem パッケージがすぐに作成されます 6. gemspec ファイル内のバージョンを上げると新しい gem パッケージが作成されます

gemcutter

- 第三の gem パッケージホスティングサイト。
- 誰でも申請無しで gem パッケージを公開できるとうわけではない。
- ソースコードのホスティングはありません。

手順

1. アカウントを登録します 2. Jeweler を使用して gem パッケージを作成します 3. RubyGems 1.3.3 以降を使用する必要があります 4. gem install gemcutter で gemcutter をインストールしてください 5. gem tumble を実行して Gemcutter を gem のソースのメインになるようにしてください 6. gem build yourgem.gemspec を実行して gem パッケージを作成してください 7. gem push yourgem-0.0.1.gem を実行してプッシュしてください

gem パッケージの作成を簡単にする

これまで説明してきたように gem パッケージを作成するのは定形作業の連続です。これらの定形作業を楽にしてくれるツールがいくつも開発されています。

- jeweler
- cutagem
- newgem
- gemify
- rtask
  - RubyForge へのリリースを簡単に行うことに特化している。
- github
  - GitHub へのリリースを簡単に行うことに特化している。
- gemcutter
  - Gemcutter へのリリースを簡単に行うことに特化している。

jeweler

一番色々とやってくれます。GitHub を主に使う人にはこれがおすすめです。内部で github.gem も使っています。Rakefile を生成してくれます。GitHub に特化したプロジェクトの雛形作成ツール。

```
$ sudo gem install jeweler
$ jeweler --help
Usage: jeweler [options] reponame
e.g. jeweler the-perfect-gem
        --bacon                      generate bacon specifications
        --shoulda                    generate shoulda tests
        --testunit                   generate test/unit tests
        --minitest                   generate minitest tests
        --rspec                      generate rspec code examples
        --micronaut                  generate micronaut examples
        --cucumber                   generate cucumber stories in addition to the other tests
        --reek                       generate rake task for reek
        --roodi                      generate rake task for roodi
        --create-repo                create the repository on GitHub
        --gemcutter                  setup project for gemcutter
        --rubyforge                  setup project for rubyforge
        --summary [SUMMARY]          specify the summary of the project
        --description [DESCRIPTION]  specify a description of the project
```

```
        --directory [DIRECTORY]      specify the directory to generate into
        --yard                       use yard for documentation
        --rdoc                       use rdoc for documentation
    -h, --help                       display this help and exit
```

オプションを簡単に説明しておきます。

- –bacon, –shoulda, –testunit, –minites, –rspec, –micronaut, –cucumber はそれぞれ同名のテスティングフレームワークで使用するファイルの雛形を生成します。
- –reek, –roodi はそれぞれ同名の静的解析ツールを使用するための rake task を Rakefile に書き込んでくれます。
- –create-repo は GitHub にリポジトリを作ってくれます。git リポジトリを作って git remote add して git push origin master してくれます。
- –gemcutter, –rubyforge はそれぞれのサイトへリリースするためのタスクを定義してくれます。
- –summary, –description は gemspec に指定する設定をコマンドラインで行うためのものです。
- –direcorty は指定したディレクトリにファイルを生成します。
- –yard, –rdoc はそれぞれ同名のソースコード埋め込み型のドキュメンテーションツールを使用するためのタスクを定義します。

使い方は以下のようにするだけです。

```
$ jeweler hoge
        create  .gitignore
        create  Rakefile
        create  LICENSE
        create  README.rdoc
        create  .document
        create  lib
        create  lib/hoge.rb
        create  test
        create  test/test_helper.rb
        create  test/hoge_test.rb
Jeweler has prepared your gem in hoge
$ cd hoge
$ rake -T
(in /home/xxxxx/tmp/hoge)
rake build                           # Build gem
rake check_dependencies              # Check that runtime and development dependencies are instal
rake check_dependencies:development  # Check that development dependencies are installed
rake check_dependencies:runtime      # Check that runtime dependencies are installed
```

```
rake clobber_rcov                    # Remove rcov products for rcov
rake clobber_rdoc                    # Remove rdoc products
rake gemspec                         # Generate and validates gemspec
rake gemspec:generate                # Generates the gemspec, using version from VERSION
rake gemspec:validate                # Validates the gemspec
rake install                         # Install gem using sudo
rake rcov                            # Analyze code coverage with tests
rake rdoc                            # Build the rdoc HTML Files
rake release                         # Release the current version.
rake rerdoc                          # Force a rebuild of the RDOC files
rake test                            # Run tests
rake version                         # Displays the current version
rake version:bump:major              # Bump the gemspec by a major version.
rake version:bump:minor              # Bump the gemspec by a minor version.
rake version:bump:patch              # Bump the gemspec by a patch version.
rake version:write                   # Writes out an explicit version.
```

こんな感じで以下のようなことをやってくれます。

- プロジェクトの雛形の生成
- Rakefile の作成
- git リポジトリの初期化

cutagem

使い方がシンプルです。Rakefile を書き換えるとかなり細かいところまでカスタマイズできます。

```
$ sudo gem install genki-cutagem
$ cutagem --help
Usage: /usr/bin/cutagem [options] gemname

Options:
    -s, --select                     Select template interactively.
    -d, --desc                       Describe this gem.
    -c, --config                     Configure user values. Use $EDITOR
        --copy-template NAME         Copy template to user template dir naming NAME
        --gem-class GEMCLASS         Specify your gem class name explicitly
        --version                    Show version string '0.0.8.3'
$ rake -T
(in /home/xxxxx/tmp/c)
rake clean            # Remove any temporary products.
rake clobber          # Remove any generated file.
rake clobber_package  # Remove package products
```

```
rake clobber_rdoc     # Remove rdoc products
rake debug_gem        # Show information about the gem.
rake gem              # Build the gem file c-0.0.1.gem
rake gemspec          # Update gem spec
rake package          # Build all the packages
rake rdoc             # Build the rdoc HTML Files
rake release          # Package and upload the release to rubyforge.
rake repackage        # Force a rebuild of the package files
rake rerdoc           # Force a rebuild of the RDOC files
rake rubyforge        # Publish to RubyForge
rake test             # Run tests for test
```

newgem

色々と決め打ちな上に別の gem (hoe) に依存させられてしまうらしい。

```
$ sudo gem install newgem
$ newgem --help
Take any library or Rails plugin or command line application,
gemify it, and easily share it with the Ruby world.

Usage: newgem /path/to/your/app [options]

Options:
    -b=BIN_NAME[,BIN_NAME2]          Sets up executable scripts in the bin folder.
        --bin-name                   Default: none
    -e, --email=PATH                 Your email to be inserted into generated files.
                                     Default: ~/.rubyforge/user-config.yml[email]
    -i, --install=generator          Installs a generator called install_<generator>.
                                     For example, '-i cucumber' runs the install_cucumber generat
                                     Can be used multiple times for different generators.
                                     Cannot be used for generators that require argumnts.
                                     Default: none
    -j, --jruby                      Use if gem is for jruby.
    -a, --author=PATH                Your name to be inserted into generated files.
                                     Default: ~/.rubyforge/user-config.yml[user_name]
        --project=PROJECT            Rubyforge project name for the gem you are creating.
                                     Default: same as gem name
    -r, --ruby=path                  Path to the Ruby binary of your choice (otherwise scripts us
                                     Default: /usr/bin/ruby1.8
    -T, --test-with=TEST_FRAMEWORK   Select your preferred testing framework.
                                     Options: test_unit (default), rspec.
    -v, --version                    Show the newgem version number and quit.
```

```
    -V, --set-version=YOUR_VERSION   Version of the gem you are creating.
                                     Default: 0.0.1
    -w, --enable-website             Enable the generation of the website for your RubyGem.
                                     Same as '-i website'
        --simple                     Creates a simple RubyGems scaffold.
General Options:
    -h, --help                       Show this help message and quit.
    -p, --pretend                    Run but do not make any changes.
    -f, --force                      Overwrite files that already exist.
    -s, --skip                       Skip files that already exist.
    -q, --quiet                      Suppress normal output.
    -t, --backtrace                  Debugging: show backtrace on errors.
    -c, --svn                        Modify files with subversion. (Note: svn must be in path)
    -g, --git                        Modify files with git. (Note: git must be in path)
$ newgem newgem
      create
      create  lib/newgem
      create  script
      create  History.txt
      create  Rakefile
      create  README.rdoc
      create  PostInstall.txt
      create  lib/newgem.rb
  dependency  install_test_unit
      create    test
      create    test/test_helper.rb
      create    test/test_newgem.rb
  dependency  install_rubigen_scripts
      exists    script
      create    script/generate
      create    script/destroy
      create  script/console
      create  Manifest.txt
      readme  readme
Important
=========

* Open Rakefile
* Update missing details (gem description, dependent gems, etc.)
$ rake -T
(in /home/kenji/tmp/newgem)
/usr/lib/ruby/gems/1.8/gems/newgem-1.5.2/lib/hoe/../newgem.rb:5: warning: already initialized con
rake announce              # publish  # Announce your release.
```

```
rake audit                # test     # Run ZenTest against the package.
rake check_extra_deps     # deps     # Install missing dependencies.
rake check_manifest       # debug    # Verify the manifest.
rake clean                #          # Clean up all the extras.
rake clobber_docs         # publish  # Remove rdoc products
rake clobber_package      # package  # Remove package products
rake clobber_rcov         # rcov     # Remove rcov products for rcov
rake config_hoe           # debug    # Create a fresh ~/.hoerc file.
rake debug_email          # publish  # Generate email announcement file.
rake debug_gem            # debug    # Show information about the gem.
rake default              # test     # Run the default task(s).
rake deps:email           # deps     # Print a contact list for gems dependent on this gem
rake deps:fetch           # deps     # Fetch all the dependent gems of this gem into tarballs
rake deps:list            # deps     # List all the dependent gems of this gem
rake docs                 # publish  # Build the RDOC HTML Files
rake gem                  # package  # Build the gem file newgem-0.0.1.gem
rake gemspec              # newgem   # Generate a newgem.gemspec file
rake generate_key         # signing  # Generate a key for signing your gems.
rake install_gem          # package  # Install the package as a gem.
rake install_gem_no_doc   # newgem   # Install the package as a gem, without generating document
rake manifest             # manifest # Recreate Manifest.txt to include ALL files to be deployed
rake multi                # test     # Run the test suite using multiruby.
rake package              # package  # Build all the packages
rake post_blog            # publish  # Post announcement to blog.
rake post_news            # publish  # Post announcement to rubyforge.
rake publish_docs         # publish  # Publish RDoc to RubyForge.
rake rcov                 # rcov     # Analyze code coverage with tests
rake redocs               # publish  # Force a rebuild of the RDOC files
rake release              # package  # Package and upload the release.
rake release_sanity       # package  # Sanity checks for release
rake release_to_rubyforge # package  # Release to rubyforge.
rake repackage            # package  # Force a rebuild of the package files
rake ridocs               # publish  # Generate ri locally for testing.
rake test                 # test     # Run the test suite.
rake test_deps            # test     # Show which test files fail when run alone.
```

gemify

対話的に gem パッケージを作成することができる。gem パッケージをビルドすることに特化している。

```
$ sudo gem install gemify
$ gemify --help
```

```
Usage: gemify [options]

Options:
    -h, --help                       Shows this message
    -I, --no-interactive             Automaticically builds the gem
    -c, --from-vcs                   Use files from version control system
$ gemify # 以降は対話的に進める。数字を入力して指示された内容を入力するだけです。
```

参考

RubyForge: RubyGems: Project Info
http://rubyforge.org/projects/rubygems/

library rubygems
http://doc.okkez.net/191/view/library/rubygems

演習

演習 1

gem コマンドを色々使ってみましょう。

```
$ gem help commands
$ gem search ...
$ gem query ...
```

などなど。

演習 2

サンプルを使用して説明したとおりにして gem パッケージを作成してみましょう。

演習 3

いくつかの gem パッケージの gemspec を読んでみましょう。

まとめ

- そのうち 1.9.2 も出るし、標準添付ライブラリになったから RubyGems を覚えたいですね
- 今日は基本的な使い方を覚えたので、何か出来たら gem にして公開できますね！

今後の情報源

公式 Web サイト

http://www.ruby-lang.org/

Ruby リファレンスマニュアル刷新計画

http://doc.loveruby.net/

日本 Ruby の会

http://jp.rubyist.net/

Rubyist Magazine

http://jp.rubyist.net/magazine/

okkez weblog

http://typo.okkez.net/

Ruby Reference Manual (beta)

http://doc.okkez.net/