Changelog for Ruby Module

sanemat {AT} tachikoma.io

changelog 読んでる?

- gem を使うときに readme 見る人
- gem を使いはじめるときに changelog 見る人
  - `changelog.md`
  - `github.com/{:repos}/releases`
  - `github.com/{:repos}/compare/{:base}...{:head}`
- gem のバージョンアップするときに changelog 見る人
- changelog のない gem のバージョンアップしなくちゃで、バカかな? とおもうけど、自分の書いたモジュールにはチェンジログない人?
- semvarって聞いたことある人?
- gem の pre-release のフォーマットが semvar じゃなくって、あれ。。?ってなったことある人
  - semvar: 2.3.4-alpha1
  - gem pre-release: 2.3.4.alpha1

結論として、ユーザーは見てるし、changelog 補助のツールもあるのでいろいろ使いましょう

changelog 補助のツール, gem とは?

日本語、gem, changelog, の組み合わせだとこれが参考になる。社内 gem と OSS の gem のメンテについて- くりにっき

github-changelog-generator に wiki ページがあり、そこに比較がある。 Alternatives · skywinder/github-changelog-generator Wiki 比較自体はちょっと古い気もする (conventional-changelog が github integration なしになってる) けど。

ブラウザから読みやすい VS gem package に含まれている

ブラウザから読みやすい VS gem package に含まれている

確かに一番読みやすいのはキチンと書かれた GitHub releases

しかしやっぱり gem package に changelog.md 入れたい気持ちがある

GitHub 落ちてたらどうするの、とか、政治的に中立なところにあったほうが良い気がする、とか。

GitHub は Progressive Enhancement 的な

パッケージング順番

となると必然、

バージョンアップの準備が整う

-> バージョン番号インクリメント, changelog 書く (順不同)

-> bundle exec rake release

(rake release 内部で build, git tag, push to github, push to rubygems)

重要なのは、package module する前に、changelog を書きたいということ。そして、package module と git tag はほぼ同時 (になってしまう) こと。

Changelog toolchain

各言語で changelog ツールチェーンの取り組みはいくつもあって選びにくいが E.g 社内 gem と OSS の gem のメンテについて- くりにっき

手を入れやすくてよく出来ているもの、私見では `conventional-changelog`。

conventional-changelog

TDD の t_wada さんおすすめ。 OSS についてあれこれ jser.info の azu さんおすすめ。われわれは、いかにして変更点を追うか

```
                    component        commit title
commit type        /                /
          \       |                |
```

```
                feat(ngInclude): add template url parameter to events

        body ->  The 'src` (i.e. the url of the template to load) is now provided to the
                 `$includeContentRequested`, `$includeContentLoaded` and `$includeConten
                 events.

 referenced  ->  Closes #8453
 issues          Closes #8454
```

規約に従ってコミットログを書くと、そこから changelog を生成する。規約は自分で決められるが、preset として angularjs や jquery のものがある。おすすめは angularjs。

これいいじゃん!

でも nodejs なんだよねー

私見　こういう細かいツールチェーンは nodejs に乗ればよくない？

ちょっと前は、同じ機能のもの後から車輪の再発明しおって!(rake とか sass とか) と若干思ってたけど最近は golang や nodejs で書いて、shell や cmd.exe からどう使うか考えればいいのでは、という方にマインドが傾いている。

私見終わり

changelog range　ココカラ、ココマデ、のうち、ココカラ、は git tag から取るので問題ない。よくあるツールで、ココマデ、を git tag から取ってしまうのが多い。でも、tag 打つ前に changelog 書きたいので、ココマデを git tag から取ってしまうのは、使いたい条件を満たさない。引数なり設定なりで渡せればいいかも。 `conventional-changelog` はココマデをデフォルトでは package.json から取得してしまう。どうしても言語依存になってしまう?

しかし、設定が js でかけるので、問題ない。

問題ない？

Ruby module から conventional-changelog を使う part1

```
$ echo '{}' > package.json
$ npm i --save-dev conventional-changelog
```

バージョンのファイルを require して、printすればいいね。 host, owner, repository もいったん手書きすればいいね。

```
# .conventional-changelog.context.js
'use strict';
var execSync = require('child_process').execSync;
var version = "" + execSync('ruby -e \'require "./lib/saddler/reporter/github/version"; p
var host = 'https://github.com';
var owner = 'packsaddle';
var repository = 'ruby-saddler-reporter-github';

module.exports = {
  version: gemspec.version,
  host: host,
  owner: owner,
  repository: repository
};

$ node_modules/.bin/conventional-changelog -i changelog.md --overwrite --preset angular
```

こういうのが生成できる

```
<a name="0.2.0"></a>
# [0.2.0](https://github.com/packsaddle/ruby-saddler-reporter-github/compare/v0.1.6...v0


### Features

* **patch:** use inherited patch, patches ([05e2306](https://github.com/packsaddle/ruby-s
* **repository:** use inherited repository ([3834b1f](https://github.com/packsaddle/ruby-



<a name="0.1.6"></a>
## [0.1.6](https://github.com/packsaddle/ruby-saddler-reporter-github/compare/v0.1.5...v0

* Improve document.


<a name="0.1.5"></a>
## [0.1.5](https://github.com/packsaddle/ruby-saddler-reporter-github/compare/v0.1.4...v0

#### Features
```

```
* **client:** Compatibility to Jenkins Pull Request Builder ([56fa18d](https://github.co
```

はじめだけちょっと頑張ると、あとはツールの流れに乗れるのでよいですね。

Ruby module から conventional-changelog を使う part2

とはいえ、module ごとに違う場所にある version のファイル探して、require して printするのツライ。 homepage も gemspec と二重管理になるのでツライ。 version 同様定数に持たせてもいいけど、あんまり。

そこで、.gemspec をparse していい感じにする parse_gemspec と、そのcli のparse_gemspec-cli を使う。

```
$ parse-gemspec-cli checkstyle_filter-git.gemspec | jq .
{
  "name": "checkstyle_filter-git",
  "version": "1.0.3.pre.beta",
  "homepage": "https://github.com/packsaddle/ruby-checkstyle_filter-git"
}
```

JSON として output するので、あとは言語中立。

```
'use strict';
var execSync = require('child_process').execSync;
var gemspec = JSON.parse(execSync('bundle exec parse-gemspec-cli parse_gemspec-cli.gemsp

module.exports = {
  version: gemspec.version
};
```

こう使える。

conventional-changelog(npm) を Ruby pruduct から使う | 實松アウトプット

最終的にこうなって

```
# .conventional-changelog.context.js
'use strict';
var execSync = require('child_process').execSync;
var URI = require('urijs');
```

```
var gemspec = JSON.parse(execSync('bundle exec parse-gemspec-cli saddler-reporter-github
var homepageUrl = gemspec.homepage;
var url = new URI(homepageUrl);
var host = url.protocol() + '://' + url.authority();
var owner = url.pathname().split('/')[1];
var repository = url.pathname().split('/')[2];

module.exports = {
  version: gemspec.version,
  host: host,
  owner: owner,
  repository: repository
};

# package.json
{
  "devDependencies": {
    "conventional-changelog": "0.4.3",
    "urijs": "^1.16.1"
  },
  "scripts": {
    "changelog": "conventional-changelog -i CHANGELOG.md --overwrite --preset angular --
  }
}
```

となり、

バージョンアップの準備が整う

-> バージョン番号インクリメント, changelog 書く (`$ npm run changelog`)
(順不同)

-> bundle exec rake release

これが実現できる。

まとめ

changelog 半自動生成のツールを使って、楽に changelog を書こう。おすすめは `conventional-changelog` です。