

My way with Ruby

Kouhei Sutou

ClearCode Inc.

RubyKaigi 2018
2018-06-01

Ad: Silver sponsor

Silver Sponsors



ClearCode
Inc.

<http://www.clear-code.com/>

Free software is
important in
ClearCode. We
develop/support

Acknowledgment

感謝

@drbrain
Eric Hodel

He fixed English in this slide
英語をチェックしてくれたよ！ありがとう！

Keynote-ish topic

キーノートっぽい話題

Did you think
about it?

考えたことある？

Keynote-ish topic1

キーノートっぽい話題1

Future

未来のこと

Keynote-ish topic2

キーノートっぽい話題2

Focus on
one thing
deeply
なにかを深掘り

Keynote-ish topic3

キーノートっぽい話題3

Overview

俯瞰した話

My activities as a Rubyist

私のRubyist活動

- ✓ Increase what Ruby can do
with free software
フリーソフトウェアを使ってRubyでできることを増やす
- ✓ Maintain libraries
ライブラリーのメンテナンス

of libraries maintained

メンテナンスしているライブラリー数

About 130

130くらい

Today's topic

今日の話題

Overview

what we can do

with Ruby

Rubyでできるようになったことをたくさん紹介

How to find targets?

そんなにネタがあるの？

Just I needed

単に自分が必要だったから

Opening1

きっかけ1

Web feed

Webフィード

RSS Parser

RSS/Atom parser with **validation**

バリデーション機能付きのRSS/Atomパーサー

RSS Parser - History

歴史

✓ 2003-05: The first release

最初のリリース

✓ No other RSS/Atom parser that supports validation even now
今でもバリデーション付きのパarserは他にない

✓ 2004-01: Ruby bundles this

RubyがRSS Parserをバンドル

✓ I became a Ruby committer
Rubyコミッターになる

Validate RSS/Atom

RSS/Atomのバリデーション

✓ Important for me
私にとっては大事

✓ Most wild RSS/Atom feeds are
invalid
野生のRSS/Atomの多くは不正

✓ Validation helps to find problems
バリデーションがあると問題を見つけやすくなる

RSS::Parser.parse

```
# Validates by default  
# デフォルトでバリデーション  
RSS::Parser.parse(rss)  
# Validation can be disabled  
# 無効にできる  
RSS::Parser.parse(rss, false)
```

Since Ruby 2.6

Ruby 2.6以降

```
# Supports keyword argument  
# キーワード引数対応  
parse(rss, validate: false)
```

REXML

XML parser written in pure Ruby

Ruby実装のXMLパーサー

REXML - History

歴史

✓ 2001: Started by Sean Russell

Seanさんが開発を開始

✓ Based on Electric XML (Java)

Java実装のElectric XMLを参考の開発

✓ REXML is "Ruby Electric XML"

REXMLは「Ruby Electric XML」

✓ 2003-01: Ruby bundles this

RubyがREXMLをバンドル

REXML - Side story

おまけ話

✓ Sean was "書運" in Kanji

Seanさんには「書運」という漢字表記があった

✓ He was interested in Japan

日本好きだった

✓ "How to write your name in Kanji?"

「君の名前は漢字でどう書くの？」

✓ We can connect with Ruby!

RubyistはRubyをきっかけにつながれる！

Ad: Code Party

宣伝：コード懇親会

RubyKaigi 2018 Code Party

🕒 19:15 - 21:30

¥ free

👥 about 40 people

📍 Rakuten Sendai branch office cafeteria
楽天仙台支社 カフェテリア

Details

Sponsored by Speee, Inc.

Ad: Code Party

宣伝：コード懇親会

This is a challenge
実験的な企画

- ✓ Ruby focus: to have fun
Rubyは楽しさを大事にしている
- ✓ We have fun writing Ruby
Rubyistは楽しくRubyを書いている
- ✓ We have fun together with writing Ruby at after party!?
だったら懇親会で一緒にRubyを書くと楽しそう！？

Ad: Code Party

宣伝：コード懇親会

- ✓ Matz attends Code Party
まつもとさんもコード懇親会に参加
- ✓ Sponsored by Speee, Inc.
Speeeさんがスポンサー

REXML - Recent1

最近1

- ✓ 2010-08: RubyKaigi 2010
 - ✓ I became the maintainer
私がメンテナーになった
 - ✓ Because RSS Parser uses it
RSS Parserが使っているから

REXML – Recent2

最近2

✓ 2016: `element[attribute_name]`

✓ Ruby 2.5 ships it
Ruby 2.5以降で使える

REXML - Recent3

最近3

✓ 2018: Fix XPath related bugs

XPath関連のバグ修正

✓ Ruby 2.6 ships it

Ruby 2.6以降で使える

REXML - Future?

未来はあるの？

✓ Pure Ruby is valuable

Rubyだけで書かれていることには価値がある

✓ Easy to install

インストールが簡単

✓ JIT may improve performance

NOTE: We should improve general logic before we expect JIT to improve performance 😊

JITで速くなるかもしれない

JITの前に普通にロジックを改良するのが先だけどね 😊

Recent my works

最近の仕事

✓ XML/HTML libraries for LuaJIT

LuaJIT用のXML/HTMLライブラリー

✓ XMLua: <https://clear-code.github.io/xmlua/>
libxml2 based XML/HTML parser

✓ LuaCS: <https://clear-code.github.io/luacs/>
CSS Selectors→XPath converter

✓ Found what is lacking in REXML API

REXMLのAPIに足りないものはなにか考えた

REXML – Future1

未来1

Introduce NodeSet

NodeSetが足りないんじゃないか

REXML - NodeSet

```
doc.  
  search("//list"). # => NodeSet  
  search("item"). # => All <item> in <list>  
  text # All texts in <item> in <list>
```

REXML – Future2 未来2

Support CSS Selectors

CSSセレクターが足りないんじゃないか

REXML – CSS Selectors

CSSセレクター

```
doc.css_select("ul li, dl dt")
```

REXML – Future3

未来3

Support HTML5 support

HTML5対応が足りないんじゃないか

REXML - HTML5

```
doc = REXML::HTML5Document.new(html5)
doc.search("//li")
doc.css_select("ul li")
```

REXML – Future

未来

- ✓ Low priority in my activities
優先度は高くない
- ✓ Do you want to work with me?
一緒にやりたい人はいる？

Opening2

きっかけ1

Presentation

プレゼンテーション

Rabbit

Presentation tool for **Rubyist**

Rubyist用のプレゼンツール

Rabbit - History

歴史

✓ 2004-07: The first release

最初のリリース

✓ No other presentation tool for a Rubyist even now

今でもRubyist用のプレゼンツールは他にない

✓ 2010: Matz migrated to Rabbit

まつもとさんがRabbitに乗り換えた

✓ Since RubyKaigi 2010?

RubyKaigi 2010から？

For Rubyist?

Rubyist向けに必要なもの

RD support

RDサポート

✓ Ruby Document

✓ Designed by Matz (Right?)
まつもとさんがデザインしたはず

✓ A text based markup language
テキストベースのマークアップ言語

✓ Version controllable
バージョン管理できる

For Rubyist?

Rubyist向けに必要なもの

Publish
our slides
as usual

いつも通りスライドを公開できる

Publish as usual

いつも通り公開

```
% gem push your-slide-1.0.gem
```

Published!

公開完了！



Rabbit Slide Show

It's showtime! Love it?



New!

MariaDBとMroongaで作る
全書対応
超高速全文検索システム
須藤功平 クリアコード
第一回 伊藤氏 公開発表会

ブログを Octopress 2 +
GitHub Pages から
Jekyll 3 + AMP +
Netlify に移行した話

Red Data Tools
美しく実装すればいいじゃんねー
須藤功平
株式会社クリアコード

<https://slide.rabbit-shocker.org/>

What's needed for presentation tool?

プレゼンツールに必要なもの

GUI

Ruby/GTK3

Multi-platform GUI toolkit

複数プラットフォーム対応GUIツールキット

Ruby/GTK3 - History

歴史

- ✓ 1998-01: 1st release by Matz
[\[ruby-list:5877\]](#)
まつもとさんが最初のリリース
- ✓ 2004-05: I joined development
私が開発に参加

Example - Window

```
require "gtk3"
app = Gtk::Application.new
app.signal_connect(:activate) do
  window = Gtk::ApplicationWindow.new(app)
  window.show_all
end
app.run
```

Approaches on missing libraries(1)

ライブラリーがない時のやり方 (1)

1. Implement only the needed features
必要な機能だけ実装
2. then back to Rabbit
必要な機能ができればRabbitに戻る

Approaches on missing libraries(2)

ライブラリーがない時のやり方 (2)

1. Implement not only the needed features
必要な機能だけじゃなく
2. but also almost all features
ほぼすべての機能を実装
3. then back to Rabbit
終わったらRabbitに戻る

My approach

私のやり方

Implement all features

testing with Rabbit
すべての機能を実装

My priority

私の優先度

✓ Rabbit is important

Rabbitは大事だけど

✓ Increasing what Ruby can do
is important too

Rubyでできることを増やすのも大事

GTK+ 3 - Size

サイズ

3000 over APIs

3000以上のAPI

How to implement 実装方法

Handwriting

手書き



Auto generation

自動生成

Ruby/GI

Generate bindings
automatically
at run-time

実行時に自動でバインディングを生成

GI: GObject Introspection

Ruby/GI - History

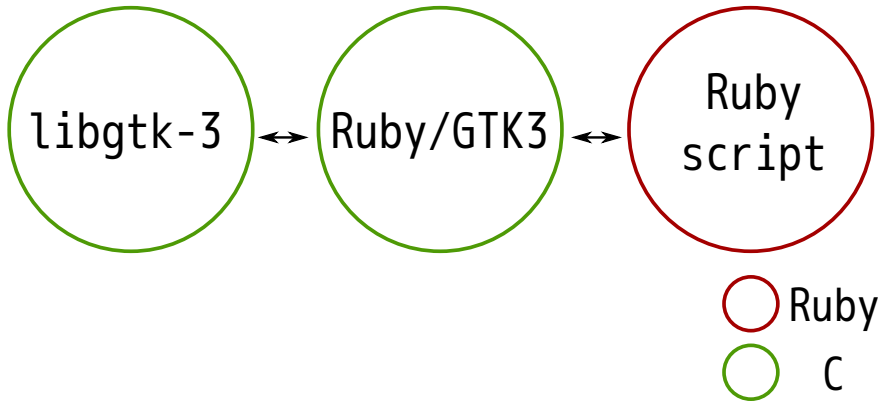
歴史

- ✓ 2012: The first commit by me
最初のコミット
- ✓ 2014: Ruby/GTK3 used Ruby/GI
Ruby/GTK3をRuby/GIベースに移行

Handwriting

手書き

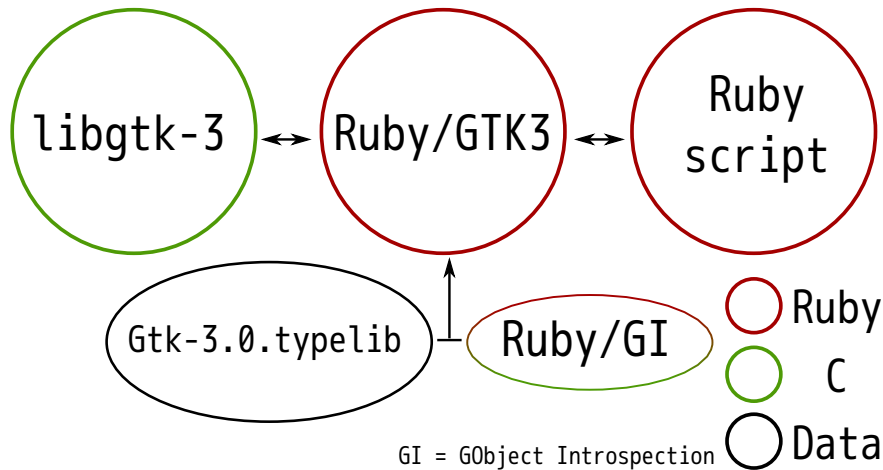
Handwritten bindings



Auto generation

自動生成

Generated at run-time



Performance (性能)

- ✓ Slower than handwriting
手書きより遅い
- ✓ Overhead
オーバーヘッド
- ✓ Dynamic arguments conversion
動的な引数の変換
- ✓ libffi based function call
libffiを使った関数呼び出し

Improve idea (改善案)

JIT compiling

JITコンパイル

JIT compiling (JITコンパイル)

```
VALUE rb_method_generic() {  
    func = dlsym(name); ...  
    ffi_call(func, ..., &result);  
    return C2RB(result);  
} ↓  
// Build rb_method() at run-time and call it.  
VALUE rb_method() {return C2RB(name(...));}
```


Ruby/GI - See also

参考情報

- ✓ "How to create bindings 2016" at RubyKaigi 2016

- ✓ <http://rubykaigi.org/2016/presentations/ktou.html>

- ✓ "GI Introduction" (in Japanese)
「GObject Introspection入門」

- ✓ <https://github.com/RubyData/workshop-materials/blob/master/gobject-introspection/introduction.md>

- ✓ Build system: Meson + Ninja
ビルドシステム: Meson + Ninja

Ruby/GI based bindings

Ruby/GIベースのバインディング

Ruby/Pango

Ruby/Pango

Text layout engine with i18n support

国際化対応のテキストレイアウトエンジン

i18n: Internationalization

Prohibition processing

禁則処理

```
widget.signal_connect(:draw) do |_, context|  
  layout = context.create_pango_layout  
  layout.text = "Helloこんにちは。🍣"  
  context.show_pango_layout(layout)  
  GLib::Source::CONTINUE  
end
```

Bidirectional text

双方向テキスト

Hello

مرحبا

こんにちは

Ruby/GI based bindings

Ruby/GIベースのバインディング

Ruby/GdkPixbuf2

Ruby/GdkPixbuf2

Image manipulation

画像操作

Half image

画像を半分に

```
require "gdk_pixbuf2"
# Load an image: Format is auto detected
pixbuf = GdkPixbuf::Pixbuf.new(file: "x.png")
# Scale to half size
half = pixbuf.scale(pixbuf.width / 2,
                    pixbuf.height / 2,
                    :bilinear)
# Save as different format
half.save("half.jpg")
```


Animated GIF

アニメーションGIF

Too fast

33

Interface



66

Powered by Rabbit 2.2.2

Ruby/GI based bindings

Ruby/GIベースのバインディング

Ruby/Poppler

Ruby/Poppler

PDF parser/renderer

PDFパーサー・レンダラー

Text extraction

テキスト抽出

```
require "poppler"  
doc = Poppler::Document.new("x.pdf")  
doc.each do |page|  
  puts(page.text) # Extract all texts  
end
```

Embed PDF

PDFの埋め込み



Improve extension API

C++ as better language for extension

Kouhei Sutou

ClearCode Inc.

RubyKaigi 2017

2017-09-18

Improve extension API - C++ as better language for extension

Powered by Rabbit 2.2.2

<http://rubykaigi.org/2017/presentations/ktou.html>

Ruby/GI based bindings

Ruby/GIベースのバインディング

Ruby/GStreamer

Ruby/GStreamer

Audio/Video player

音声・動画プレイヤー

(*) Streaming media framework

本当はストリーミングメディアフレームワーク

Camera (カメラ)

```
require "gst"
description = [
  "autovideosrc", # Camera
  "videoconvert", # Filter
  "autovideosink", # Window
].join(" ! ")
pipeline = Gst.parse_launch(description)
pipeline.play
until pipeline.bus.poll.type.eos? do
end # Main loop
pipeline.stop
```


Face detection (顔認識)

```
description = [  
  "autovideosrc",  # Camera  
  "videoconvert",  # Filter  
  "facedetect",    # Face detection (!)  
  "videoconvert",  # Filter          (!)  
  "autovideosink", # Window  
].join(" ! ")  
pipeline = Gst.parse_launch(description)  
pipeline.play
```

What's needed for presentation tool?

プレゼンツールに必要なもの

PDF output

PDF出力

rcairo

2D graphics renderer

2次元画像レンダラー

rcairo - Outputs

出力

✓ PNG • SVG

✓ PDF

✓ Display (X/macOS/Windows)

✓ ...

rcairo - History

歴史

- ✓ 2003-10: The initial commit
最初のコミット
- ✓ 2005-09: I started developing
私が開発に参加

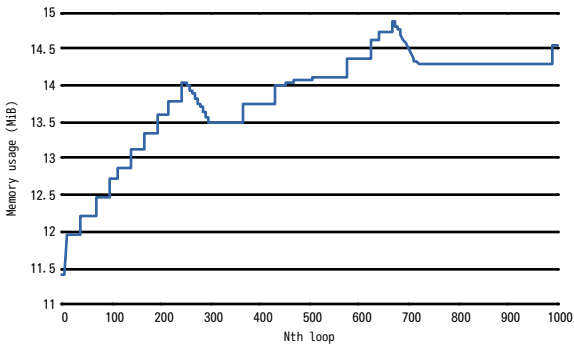
Red A4 PDF

赤いA4のPDFを出力

```
require "cairo"  
include Cairo  
PDFSurface.create("x.pdf", "A4") do |surface|  
  Context.create(surface) do |context|  
    context.set_source_color(:red)  
    context.paint  
  end  
end
```

rcairo - GC

```
1000.times do  
  Cairo::ImageSurface.new(:argb32, 6000, 6000)  
end
```



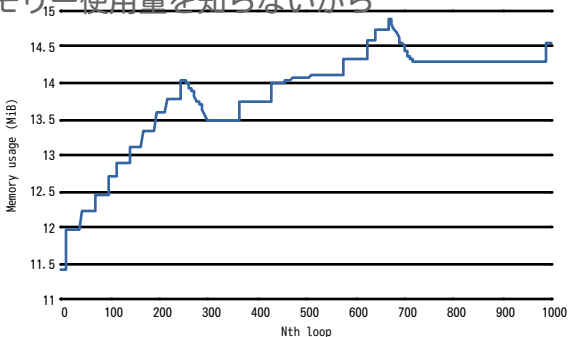
GC - Cause (原因)

✓ GC isn't run often enough

GCの実行頻度が十分じゃなかった

✓ Because Ruby doesn't know how much memory used by cairo

Rubyはcairoのメモリー使用量を知らないから

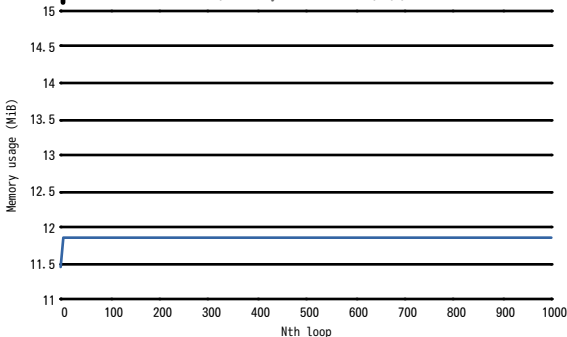


GC - Fix (修正)

✓ `rb_gc_adjust_memory_usage()`

✓ Improve GC frequency (GC実行頻度を改善)

✓ Ruby 2.4 ships it (Ruby 2.4以降)



What's needed for presentation tool?

プレゼンツールに必要なもの

Easy to install

簡単インストール

native-package-installer

Install
system packages
on `gem install`

`gem install`時にシステムのパッケージをインストール

extconf.rb/Rakefile

```
require "pkg-config"
require "native-package-installer"
unless PKGConfig.check_version?("gdk-3.0")
  packages = {
    altlinux: "libgtk+3-devel",
    debian: "libgtk-3-dev",
    redhat: "pkgconfig(gdk-3.0)",
    homebrew: "gtk+3",
    macports: "gtk3",
    msys2: "gtk3",
  }
  unless NativePackageInstaller.install(packages)
    exit(false)
  end
end
```

rake-compiler

Build fat gem by cross compile

クロスコンパイルでfat gemをビルド

rake-compiler - History

歴史

- ✓ 2008-11: The first commit
最初のコミット
- ✓ 2014-12: Orphan
だれかメンテナー変わってー
- ✓ 2014-12:
I became the maintainer
私がメンテナーになった

Opening3

きっかけ3

Test

テスト

test-unit

Testing framework to write tests in Ruby

Rubyでテストを書けるテストフレームワーク

test-unit - History

歴史

✓ 2003-02: Import to Ruby

Rubyに取り込まれる

✓ 2008-05: I became the maintainer

私がメンテナーになった

✓ 2008-10: Removed from Ruby

Rubyから削除

See also: "The history of testing framework in Ruby"

<http://rubykaigi.org/2015/presentations/kou>

test-unit – New feature

新機能

Grouping

グループ化

Grouping

グループ化

- ✓ The most important feature
一番大事な機能
- ✓ Keep tests maintainable
メンテナンスできるテストを維持できる

Example (例)

```
class StackTest < Test::Unit::TestCase
  class PushTest < self
    def test_XXX; end
  end
  class PopTest < self
    def test_XXX; end
  end
end
```

Method style (メソッド形式)

```
class StackTest < Test::Unit::TestCase
  sub_test_case("#push") do
    def test_XXX; end
  end
  sub_test_case("#pop") do
    def test_XXX; end
  end
end
```

test-unit - New feature

新機能

Data driven test

データ駆動テスト

Data driven test

データ駆動テスト

```
data("positive", [3, 1, 2])
data("negative", [-4, 1, -5])
def test_add(data)
  expected, augend, addend = data
  assert_equal(expected,
               add(augend, addend))
end
```

test-unit - New feature

新機能

Reverse backtrace

逆順のバックトレース

Reverse backtrace

逆順のバックトレース

- ✓ Reverse backtrace only for terminal output
ターミナル出力のときだけ逆順
- ✓ The same change as Ruby 2.5.0
Ruby 2.5.0と同じ変更

test-unit - New feature

新機能

Test double

テストダブル

test-unit-rr

RR integration

RRとの統合

RR - History

歴史

- ✓ 2007-06: The initial commit
最初のコミット
- ✓ 2014-12: Orphan
だれかメンテナー変わってー
- ✓ 2015-05:
I became the maintainer
私がメンテナーになった

Stub (スタブ)

```
adder = Object.new  
adder.add(1, 2) # => Error  
stub(adder).add(1, 2) {3}  
adder.add(1, 2) # => 3
```

Opening4

きっかけ4

Full text search

全文検索

Rroonga

Full text search library

全文検索ライブラリー

Library vs Client

ライブラリー対クライアント

✓ No server process

サーバープロセスがない

✓ Easy to start

簡単に使い始められる

✓ Write in Ruby

Rubyで書ける

Create DB (データベース作成)

```
require "groonga"
```

```
Groonga::Database.create(path: "/tmp/db")
```

Define schema (スキーマ定義)

```
Groonga::Schema.define do |schema|
  schema.create_table("docs") do |table|
    # The column to store text
    table.text("content")
  end
  # The index for full text search
  schema.create_lexicon("terms") do |table|
    table.index("docs.content")
  end
end
```

Add records (レコード追加)

```
docs = Groonga["docs"]  
docs.add(content: "String#<< concatenates ...")  
docs.add(content: "String#dup duplicates ...")
```

Search (検索)

```
matches = docs.select do |record|
  record.content.match("concat")
end
p matches.size # => 1
matches.each do |record|
  p record.content # => "String#<< concat..."
end
```

User - Rabbit Slide Show



<https://slide.rabbit-shocker.org/>

User - Rurema Search



るりまサーチ

[すべて](#)

139213件

[1.8.7](#)

17619件

[1.9.3](#)

18361件

[2.0.0](#)

18446件

[2.1.0](#)

18170件

[2.2.0](#)

18161件

[2.3.0](#)

18217件

[2.4.0](#)

15088件

[2.5.0](#)

15151件

最速Rubyリファレンスマニュアル検索！

検索

組み込みクラス一覧

A～E

[ARGF.class](#) (518)[Array](#) (1139)[BasicObject](#) (98)[Bignum](#) (190)

<https://docs.ruby-lang.org/ja/search/>

Rurema Search

るりまサーチ

✓ Super fast!
すごく速い!

✓ Tuned for Ruby documents
Rubyのドキュメント用にチューニング

User - RDoc Search

✓ Planning

考えてはいるけど。。。。

✓ Do you want to work with me?

一緒にやりたい人はいる？

Rurema and RDoc

Project	Language	Target
Rurema	Japanese	Japanese Rubyists
RDoc	English	All Rubyists

Source ソース

✓ Shared nothing

共有していない

✓ Copy based share

共有するときはコピー

✓ e. g. :

Description,
Sample codes,

...

例：説明やサンプルコードなどをコピー

From my point of view

私思うこと

✓ Can we share documents?

ドキュメントを共有できないかな

✓ How to work together deeply?

もっと協力してできないかな

I 18n 国際化

✓ Source: RDoc

ソースはRDoc

✓ For all Rubyists

これは全Rubyist向け

✓ Translate to Japanese

RDocのドキュメントを日本語に翻訳

✓ For Japanese Rubyists

これは日本人Rubyist向け

Add i18n support

国際化サポートを追加

✓ YARD

✓ Since 0.8.0 at 2012-04

✓ RDoc

✓ Since 4.2.0 at 2014-12

YARD - i18n

```
# Generates po/yard.pot  
# po/yard.potを生成  
% yard i18n
```

YARD - i18n

```
# Create po/ja.po from po/yard.pot
# po/yard.potからpo/ja.poを作成
% msginit \
    --locale=ja_JP.UTF-8 \
    --input=po/yard.pot \
    --output-file=po/ja.po
```

YARD - i18n

```
# Translate messages in po/ja.po  
# po/ja.po内のメッセージを翻訳  
% editor po/ja.po
```


YARD - i18n

```
# Generate documents with  
# translated messages  
# 翻訳したメッセージを使って  
# ドキュメント生成  
% yard --locale ja
```

Packnga

Rake task for YARD i18n

YARDの国際化機能向けのRakeタスク

Setting 設定

```
# Rakefile  
require "packnga"  
Packnga::DocumentTask.new(spec) do |task|  
  task.original_language = "en"  
  task.translate_languages = ["ja"]  
end
```

Workflow

ワークフロー

```
% rake reference:translate  
% editor doc/po/ja/x.edit.po  
% rake reference:translate  
% editor lib/x.rb  
% rake reference:translate  
...  

```

Users

ユーザー

✓ test-unit

✓ Rroonga

✓ ...

RDoc - i18n

```
# Generates doc/rdoc.pot  
# doc/rdoc.potを生成  
% rdoc --format=pot
```

RDoc - i18n

```
# Create locale/ja.po
# from doc/rdoc.pot
# doc/rdoc.potからlocale/ja.poを作成
% mkdir -p locale
% msginit \
    --locale=ja_JP.UTF-8 \
    --input=doc/rdoc.pot \
    --output-file=locale/ja.po
```

RDoc - i18n

```
# Translate messages in locale/ja.po  
# locale/ja.po内のメッセージを翻訳  
% editor locale/ja.po
```


RDoc - i18n

```
# Generate documents with  
# translated messages  
# 翻訳したメッセージを使って  
# ドキュメント生成  
% rdoc --locale ja
```

RDoc, Rurema and i18n

✓ No progress...

ツールの整備まででそれ以降は進んでいない。。。。

✓ Do you want to work with me?

一緒にやりたい人はいる？

jeekyll-task-i18n

Jekyll + i18n

Features

機能

✓ Support all markups!

すべてのマークアップ対応！

✓ GitHub Pages ready!

GitHub Pagesでも使える！

Setting (設定)

```
# Rakefile
require "jekyll/task/i18n"
Jekyll::Task::I18n.define do |task|
  task.locales = ["ja"]
  task.files = Rake::FileList["**/*.md"]
  task.files -= Rake::FileList["_*/**/*.md"]
  task.locales.each do |locale|
    task.files -= Rake::FileList["#{locale}/**/*.md"]
  end
end
task default: ["jekyll:i18n:translate"]
```

Workflow (ワークフロー)

```
% editor index.md  
% rake  
% editor _po/ja/index.edit.po  
% rake  
% git commit -a
```

User – Red Data Tools



<https://red-data-tools.github.io/>

groonga-client

Full text
search
client
全文検索クライアント

Library vs Client

ライブラリー対クライアント

✓ Less dependencies

依存関係が少ない

✓ Less resources needed

必要なリソースが少ない

Search (検索)

```
require "groonga/client"
url = "http://localhost:10041"
Groonga::Client.open(url: url) do |client|
  response =
    client.select(table: "docs",
                  match_columns: "content",
                  query: "concat")
  p response.n_hits # => 1
end
```

Asynchronous (非同期)

```
# Call with block
client.select(table: "docs",
              match_columns: "content",
              query: "concat") do |response|
  p response.n_hits # => 1
end
p :here # => :here then ↑
sleep(0.1)
```

Asynchronous - wait

```
request =  
  client.select(table: "docs",  
                match_columns: "content",  
                query: "concat") do |response|  
    p response.n_hits # => 1  
  end  
  p :here # => :here then ↑  
  request.wait
```

groonga-client-rails

Ruby on Rails integration for groonga-client

Ruby on Railsで使う

Architecture

アーキテクチャー

✓ Data: RDBMS
データはRDBMSに格納

✓ Full text search: Groonga
全文検索はGroongaで処理

Define app searcher

アプリケーションサーチャーを定義

```
# app/searchers/application_searcher.rb  
class ApplicationSearcher <  
  Groonga::Client::Searcher  
end
```

Define searcher

サーチャーを定義

```
# app/searchers/document_searcher.rb
class DocumentsSearcher < ApplicationSearcher
  # Define a full text search index as "content"
  # 全文検索用のインデックスを定義
  schema.column :content, {
    type: "Text",
    index: true,
    index_type: :full_text_search,
  }
end
```


Bind to model

モデルと結びつける

```
# app/models/document.rb  
class Document < ApplicationRecord  
  # DocumentsSearcher searches Document model  
  source = DocumentsSearcher.source(self)  
  # Bind Document's "content" column to  
  # DocumentsSearcher's "content" index  
  source.content = :content  
end
```

Search (検索)

```
# app/controllers/documents_controller.rb
class DocumentsController < ApplicationController
  def index
    @query = params[:query]
    searcher = DocumentSearcher.new
    @result_set = searcher.search.
      query(@query).
      result_set
  end
end
```

See also

参考情報

✓ Tutorial in Japanese

日本語のチュートリアル

✓ <http://www.clear-code.com/blog/2016/12/22.html>

Ranguba (WIP) (開発中)

Full text search system

全文検索システム

Use cases

利用例

✓ File server search
ファイルサーバー検索

✓ E-mail search
メール検索

✓ Web site search
Webサイト検索

Features

機能

✓ Crawlers
クローラー

✓ Web UI

✓ Command line interface

✓ Update documents
更新

✓ Search documents
検索

Text extractor

テキスト抽出

Supported formats

対応フォーマット

- ✓ PDF
- ✓ Office documents (オフィス文書)
 - ✓ OpenDocument, Word, Excel, ...
- ✓ E-mail (メール)
- ✓ ...

Interface

インターフェイス

- ✓ HTTP
- ✓ Web UI
- ✓ Command line interface
- ✓ API (Library)

Install - Docker

```
% GITHUB=https://github.com
% git clone \
    ${GITHUB}/ranguba/chupa-text-docker.git
% cd chupa-text-docker
% docker-compose up --build
```

How to use 使い方

```
% curl \  
  --form data=@XXX.pdf \  
  http://localhost:20080/extraction.json
```

Use cases

利用例

✓ Ranguba

✓ Full text search system

全文検索システム

✓ Commit e-mail

コミットメール

git-commit-mailer

Commit e-mail for Git

Git用のコミットメール

Features

機能

✓ HTML mail

HTMLメール

✓ Highlighted diff

diffをハイライト

✓ GitLab/GitHub Web hook

GitLab/GitHubのWebフック対応

✓ By [GitHub:clear-code/github-web-hooks-receiver](https://github.com/clear-code/github-web-hooks-receiver)

Users 利用者

✓ tDiary

✓ My products

commit-email.info

Commit e-mail as a Service

コミットメールのクラウドサービス

How to use 使い方

✓ Send a pull request to
[GitHub:kou/commit-email.info](https://github.com/kou/commit-email.info)
pull requestを送る

✓ Register a Web hook
Webフックを登録

✓ Subscribe your mailing list
メーリングリストを購読

See also <http://www.commit-email.info/>

Opening5

きっかけ5

Data processing

データ処理

CSV

CSV parser

CSVパーサー

csv - History

歴史

- ✓ 2003: Import
Rubyに取り込み
- ✓ 2007: Replaced with FasterCSV
FasterCSVで置き換え
- ✓ 2018: I became a co-maintainer with mrkn
mrknと一緒にメンテナーになった

Why? なんで？

- ✓ There are many data sources
in CSV

CSVのデータはたくさんある

- ✓ Important to process data

データを処理するためにCSVパーサーは重要

CSV format problems

CSVフォーマットの問題

✓ Slow to parse
パースが遅い

✓ Too wild
なんでもあり

Red Arrow

Apache Arrow Ruby

Red Arrow - History

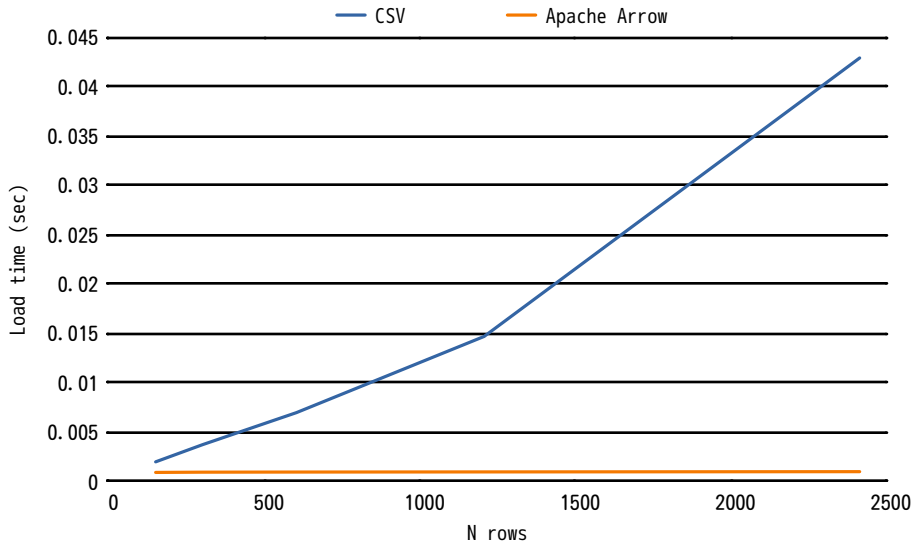
歴史

- ✓ 2017-02: The first commit
最初のコミット
- ✓ 2018-05: Became the
"official" Ruby bindings of
Apache Arrow
Apache Arrowの公式Rubyバインディングになった

Apache Arrow

- ✓ Super fast data format
すごく速いデータフォーマット
- ✓ For in-memory data
インメモリーデータ用
- ✓ Cross-language support
いろんな言語がサポート
- ✓ Easy to share data with Python,
Java, ...
PythonやJavaなどとデータ交換がしやすい

Performance (性能)



Apache Arrow – Position

立ち位置

- ✓ A very important piece in recent data processing

最近のデータ処理界隈ではすごく大事な1ピース

- ✓ Like JIT for Ruby 3

Ruby 3で例えるとJITみたいな感じ

Red Arrow - Impl.

実装

- ✓ Based on Ruby/GI
Ruby/GIを使っている
- ✓ Auto generated bindings
バインディングを自動生成

Extendable load API

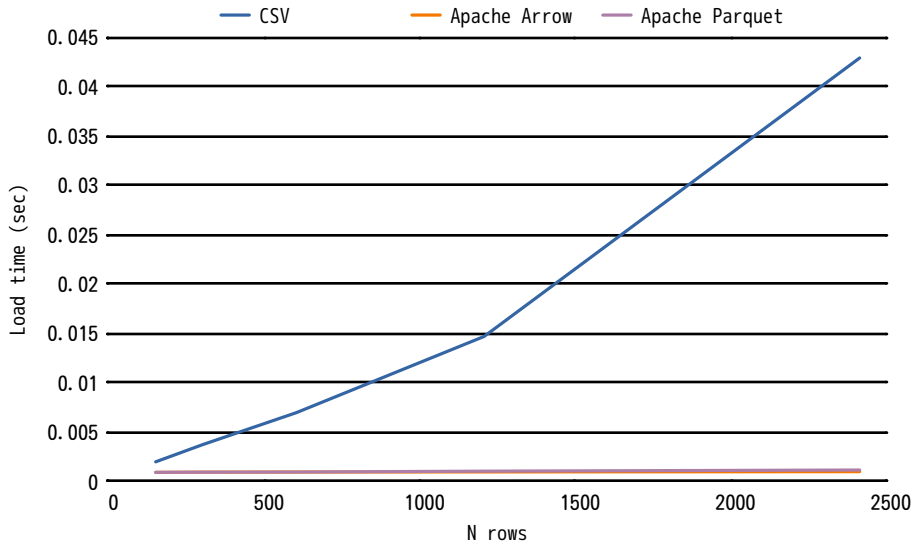
拡張可能なロードAPI

```
# Load Apache Arrow data  
Arrow::Table.load("iris.arrow")  
# Load CSV data  
Arrow::Table.load("iris.csv")  
# Load Apache Parquet data  
Arrow::Table.load("iris.parquet")
```

Apache Parquet

- ✓ **Super fast data format**
すごく速いデータフォーマット
- ✓ **For storing analysis target data**
解析対象のデータを保存する用
- ✓ **Widely used**
広く使われている

Performance (性能)



Red Parquet

Apache Parquet

Red Data Tools

A project to
make Ruby data
processable

Rubyでデータ処理できるようにするためのプロジェクト

Red Data Tools - History

歴史

- ✓ 2017-02: Start (開始)
- ✓ 2017-11-: Develop events per month at Tokyo
東京で毎月開発イベントを開催

The number of products

プロダクト数

About 20

including Red Arrow and Red Parquet
20くらい

Red ArrowやRed ParquetもRed Data Toolsプロダクツ

Red Datasets

Dataset fetcher

データセット取得

Supported datasets

対応データセット

- ✓ Iris
- ✓ CIFAR
- ✓ Wikipedia

Wikipedia

```
require "datasets"

wikipedia = Datasets::Wikipedia.new
wikipedia.each do |page|
  p page.title
end
```

Wikipedia search

```
pages = Groonga["pages"]
wikipedia = Datasets::Wikipedia.new
wikipedia.each do |page|
  pages.add(title: page.title,
            content: page.revision.text)
end
ruby_pages = pages.select do |record|
  record.match("Ruby OR Rails") do |target|
    (target.title * 10) | target.content
  end
end
p ruby_pages.size
```

jeekyll-jupyter-notebook

Jekyll
+
Jupyter
Notebook

Usage

使い方

```
{% jupyter_notebook sample.ipynb %}
```

Computer vision

コンピュータービジョン

Camera (カメラ)

```
require "cv"  
camera = CV::Camera.new  
image = camera.read  
image.write("capture.jpg")
```

Face detect (顔認識)

```
image_gray = image.convert_color(:bgr2gray)
classifier = # Face detector
  CV::CascadeClassifier.new("frontalface_alt")
objects = classifier.detect(image_gray)
color = CV::Color.new(0, 0, 255)
objects.each do |object|
  # Draw detected area
  image.draw_rectangle(object, color)
end
image.write("detect.jpg")
```

Red OpenCV - Impl.

実装

- ✓ Based on Ruby/GI
Ruby/GIを使っている
- ✓ Auto generated bindings
バインディングを自動生成

Ad: RubyData Workshop

✓ 2018-06-01 15:50/17:20

✓ Contents: (内容)

✓ Workshop by mrkn
mrknによるワークショップ

✓ Presentations from
Red Data Tools members
Red Data ToolsメンバーによるRed Data Toolsでやって
きたことの紹介

Process data with Ruby

Rubyでデータ処理

✓ We're working on it

Red Data Toolsは継続して取り組んでいる

✓ Do you want to work with us?

一緒にやりたい人はいる？

How to join1 (参加方法1)

✓ Join our chat rooms:

チャットルームに参加

✓ en: [Gitter:red-data-tools/en](https://gitter.im/red-data-tools/en)

✓ ja: [Gitter:red-data-tools/ja](https://gitter.im/red-data-tools/ja)

✓ Join monthly events at Tokyo

東京での毎月の開発イベントに参加

✓ <https://speee.connpass.com/>

How to join2 (参加方法2)

- ✓ Hire developers to work on it
仕事として開発する開発者を雇う
- ✓ e. g. : mrkn by Speee, Inc.
例 : Speeeの村田さん

How to join3 (参加方法3)

- ✓ Order ClearCode to work on it
クリアコードに開発の仕事を発注
- ✓ Join ClearCode to work on it
クリアコードに入って仕事として開発を進める

Wrap up

まとめ

- ✓ I'm working on the following as a Rubyist
Rubyistとしての私の活動
- ✓ Increase what Ruby can do with free software
フリーソフトウェアを使ってRubyでできることを増やす
- ✓ Maintain libraries
ライブラリーのメンテナンス
- ✓ Do you want to work with me?
一緒にやりたい人はいる？