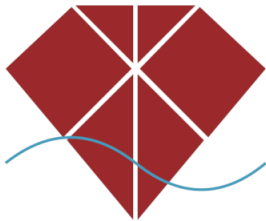


dRuby 入門者による あなたの身近にある dRuby 入門

@makicamel
大阪 Ruby 会議 04
2024.08.24



Osaka RubyKaigi 04

自己紹介

- @makicamel / 川原万季
- Ruby 💎 とビール 🍺 とお酒が好き
- 好きな VR ゲームは BeatSaber 🗡️
- (株)アンドパッド 🧑‍🏭





dRuby

- distributed ruby
 - 分散 Ruby
- 分散オブジェクトシステムを実現するライブラリ
- 1999 年初出
- 関将俊さん作



dRuby

- 1999 年初出
- <https://gist.github.com/seki/5713863>

1st druby. [ruby-list:15406]

drb.rb


Raw

```
1 drb.rb
2 #!/usr/local/bin/ruby
3 =begin
4   Tiny distributed Ruby --- dRuby
5   DRb --- dRuby module.
6   DRbProtocol --- Mixin class.
7   DRbObject --- dRuby remote object.
8   DRbConn ---
9   DRbServer --- dRuby message handler.
10 =end
11
12 require 'socket'
13 require 'marshal'
14
15 module DRb
16   def start_service(uri, front=nil)
17     @uri = uri.to_s
18     @front = front
19     @server = DRbServer.new(@uri)
20     @thread = @server.run
21   end
22   module function :start_service
```



dRubyによる分散・Webプログラミング / 関将俊
<https://www.ohmsha.co.jp/book/9784274066092/>

dRuby

- distributed ruby
 - 分散 Ruby
 - 分散オブジェクトシステムを実現するライブラリ
 - 1999 年初出
 - 関将俊さん作
- 

分散オブジェクトシステム

- とは？
- プロセスやネットワークを超えてメッセージを送る
 - 他のプロセスのオブジェクトのメソッドを呼び出せる
- とは？

分散オブジェクトシステム

- `DRb.start_service(uri=nil, front=nil, config_or_acl=nil)`
 - dRuby のサービス(サーバ)を起動するメソッド
 - uri: 起動したサービスを紐づける uri
 - front: uri に紐づけるオブジェクト

DRb.#start_service (Ruby 3.3 リファレンスマニュアル)

https://docs.ruby-lang.org/ja/3.3/method/DRb/m/start_service.html



分散オブジェクトシステム

- DRb::DRbObject
 - リモートの dRuby オブジェクトを表すオブジェクト
- DRb::DRbObject.new_with_uri(uri)
 - URI から新しい DRbObject を生成する
 - 別プロセスの DRb.#start_service で指定したフロントオブジェクトを指すリモートオブジェクトを取り出す

class DRb::DRbObject (Ruby 3.3 リファレンスマニュアル)

<https://docs.ruby-lang.org/ja/3.3/class/DRb=3a=3aDRbObject.html>

DRb::DRbObject.new_with_uri (Ruby 3.3 リファレンスマニュアル)

https://docs.ruby-lang.org/ja/latest/method/DRb=3a=3aDRbObject/s/new_with_uri.html

分散オブジェクトシステム

```
# terminal1
$ irb -r drb
front_object = { greeting: 'Hello, Osaka!' }
DRb.start_service 'druby://127.0.0.1:10001', front_object

# terminal2
$ irb -r drb
remote_object = DRbObject.new_with_uri 'druby://127.0.0.1:10001'
remote_object[:greeting]
# => "Hello, Osaka!"
remote_object[:greeting] = 'よろしゅう大阪'

# terminal1
front_object
# => { greeting: "よろしゅう大阪" }
```

分散オブジェクトシステム

```
# terminal3
$ irb -r drb
class Puts
  def self.puts(str)
    $stdout.puts(str)
  end
end
DRb.start_service('druby://127.0.0.1:10002', Puts)

# terminal4
$ irb -r drb
puts = DRbObject.new_with_uri 'druby://127.0.0.1:10002'
puts.puts 'おおきに'

#
# => ?
```

分散オブジェクトシステム

```
# terminal3
$ irb -r drb
class Puts
  def self.puts(str)
    $stdout.puts(str)
  end
end
DRb.start_service('druby://127.0.0.1:10002', Puts)

# terminal4
$ irb -r drb
server = DRbObject.new_with_uri 'druby://127.0.0.1:10002'
server.puts 'おおきに'

# terminal3
# => おおきに
```

分散オブジェクトシステム

- プロセスやネットワークを超えてメッセージを送る
 - 他のプロセスのオブジェクトのメソッドを呼び出せる
- かんたんにサーバが作れる

分散オブジェクトシステム

- プロセスやネットワークを超えてメッセージを送る
 - 他のプロセスのオブジェクトのメソッドを呼び出せる
- かんたんにサーバが作れる

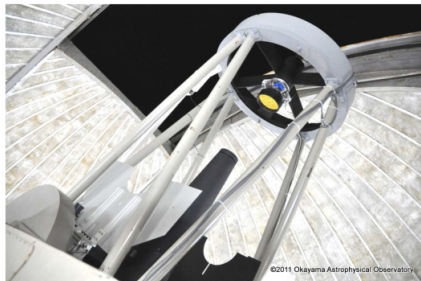
夢が広がる！！



分散オブジェクトシステム

- リアルワールド dRuby 活用例

分散オブジェクトシステム



岡山天体物理観測所広視野赤外カメラの製作 / 柳澤 顕史

https://www.astr.tohoku.ac.jp/~akiyama/astroinst2015/20151207_yanagisawa.pdf

Rinda in the real-world embedded systems. / 関将俊

https://rubykaigi.org/2020-takeout/presentations/m_seki.html

分散オブジェクトシステム

● ● ● Point of Care Testing

- Rapid tests for the Respiratory infection diseases such as SARS-CoV-2, Influenza, RS virus
- Next Generation
- +UI, Parallel processing



dRuby in the real-world embedded systems. / 関将俊 園川龍也

<https://www.druby.org/seki-RK2021.pdf>

https://rubykaigi.org/2021-takeout/presentations/m_seki.html

分散オブジェクトシステム



Scaling Twitter / Blaine Cook

<https://www.slideshare.net/slideshow/scaling-twitter/41197>

分散オブジェクトシステム

■ すごすぎる

もうちょっと
身近な例ありませんか



dRuby 入門者によるあなたの身近にある dRuby 入門



わたしたちの身近にある **dRuby**

- RSpec
- ActiveSupport
- Rabbit
- るりま

わたしたちの身近にある **dRuby**

- **RSpec** 
- ActiveSupport
- Rabbit
- るりま

RSpec

■ drb オプション

```
$ rspec --help  
-X, --[no-]drb  
    --drb-port PORT
```

Run examples via DRb.
Port to connect to the DRb server.

RSpec

- アプリケーションを dRuby プロセスに読み込んでテストサーバにする
- rspec 実行時にテストサーバでテスト実行する



RSpec



① dRuby サーバ起動
→ テストサーバ



RSpec



② `rspec -drb`



RSpec



③ テストサーバにテスト依頼



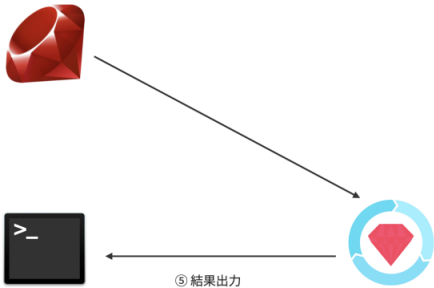
RSpec



④ テスト実行



RSpec



RSpec

- テストサーバ
- プリローダ
 - アプリケーション起動時間を省略
- e.g. spork
 - A DRb server for testing frameworks
 - ※ 最終コミットは 2014 年
 - 2024 年現在はまだあまり使われていないかも

spork
<https://github.com/sporkrb/spork>

わたしたちの身近にある dRuby

- RSpec
- ActiveSupport 
- Rabbit
- るりま

ActiveSupport

- ActiveSupport::TestCase#parallelize
- テストの並列化
- 使用マシンのコア数分プロセスをフォーク

ActiveSupport::TestCase

<https://api.rubyonrails.org/classes/ActiveSupport/TestCase.html#method-c-parallelize>



ActiveSupport

- rails new で test_helper.rb を生成
 - デフォルトで parallelize が記述されている
- 何も意識しなくてもテストが並列実行される

```
# test/test_helper.rb
module ActiveSupport
  class TestCase
    parallelize(workers: :number_of_processors)

    fixtures :all
  end
end
```

ActiveSupport

- ActiveSupport::TestCase#parallelize の並列実行のしくみ

ActiveSupport



```
Queue.new  
DRb.start_service(  
  "drbunix:",  
  @queue_server  
)
```

- ① キューを作る
このキューが dRuby サーバの
フロントオブジェクト



ActiveSupport



② フォークしてワーカを作る
これらは①で作った
dRuby サーバと通信するクライアント

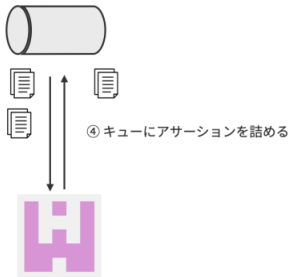
ActiveSupport



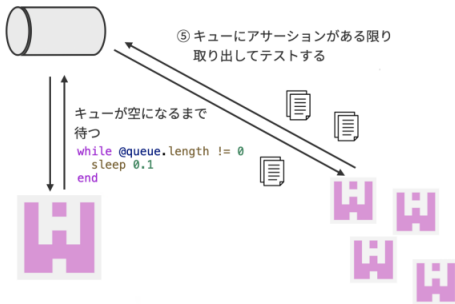
③ キューにワーカを登録する



ActiveSupport



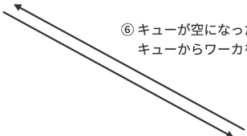
ActiveSupport



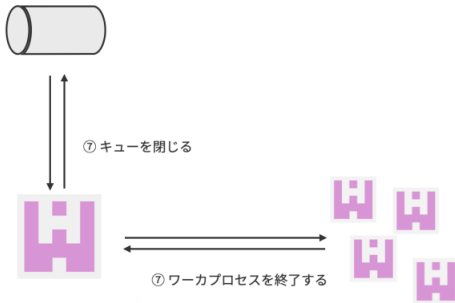
ActiveSupport



⑥ キューが空になったら
キューからワーカを削除する



ActiveSupport



ActiveSupport




⑧ テスト結果を返す

ActiveSupport

- 並列実行のしくみ
 - アサーションの管理を dRuby に任せる
 - フォークしたプロセスが各個アサーションを引き出しテスト実行

テスト結果の管理は親プロセスの仕事



わたしたちの身近にある **dRuby**

- RSpec
- ActiveSupport
- **Rabbit** 
- るりま

Joy

24



89

Powered by Rabbit 3.0.0

Contribute to Ruby - Yukihiro "Matz" Matsumoto

<https://speakerdeck.com/matz/contribute-to-ruby-rubykaigi-2022?slide=25>

https://rubykaigi.org/2022/presentations/yukihiro_matz.html

Rabbit

- Rubyist 御用達プレゼンテーションツール
 - うさぎとかめでお馴染み
- 須藤功平さん作

Rabbit

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

<https://github.com/rabbit-shocker/rabbit>



Rabbit

■ デフォルトで dRuby サーバを立ち上げる

```
class Rabbit::Command::Rabbit
  def do_display
    # ...
    front = make_front(canvas)
    setup_druby(front) if @options.use_druby
    # ...
  end

  def setup_druby(front)
    require "drb/drb"
    begin
      DRb.start_service(@options.druby_uri, front)
    # ...
    end
  end
end
```

Rabbit

```
DRb.start_service(@options.druby_uri, front)
```

フロントオブジェクトを触ることができる

Rabbit

```
# terminal5
$ irb -r drb
rabbit = DRbObject.new_with_uri 'druby://127.0.0.1:10101' # rabbitのdRubyサーバのuri
rabbit.title
# => "dRuby 入門者による\nあなたの身近にある\nndRuby 入門"
%w[Rabbit最高 🐇🐇🐇🐇🐇🐇🐇🐇 dRubyすごい 大阪たのしい🍷].each { rabbit.append_comment _1 }
rabbit.available_interfaces
# => [...]
rabbit.toggle_fullscreen
```

フロントオブジェクトにメッセージを送ることで Rabbit を操作できる

Rabbit

- API 提供

Rabbit API

■ RabbiRack

- Web ブラウザから Rabbit を操作するツール

```
get "/pages/next" do
  @rabbit.move_to_next_if_can
  haml :index
end
```

RabbiRack

<https://rabbit-shocker.org/ja/rabbirack>

<https://github.com/rabbit-shocker/rabbirack>

※ macOS では <https://github.com/rabbit-shocker/rabbit/commit/2b2ce805d98fd972d0a288ac7df944e53d48bbbf> の適用が必要

Rabbit API

■ Rabbiter

- Twitter からツイートを収集し、コメントとして Rabbit に流しこむツール

```
rabbit = DRbObject.new_with_uri(options.rabbit_uri)
client = Rabbiter::Client.new(logger)
client.start(*options.filters) do |status|
  # ...
  comment = "@#{status.user.screen_name}: #{clean_text(status, options)}"
  rabbit.append_comment(comment)
end
```

Rabbiter

<https://rabbit-shocker.org/ja/rabbiter>

<https://github.com/rabbit-shocker/rabbiter>

※ Twitter (現 X) が壊れているので 2024 年 8 月現在は動かさない

わたしたちの身近にある **dRuby**

- RSpec
- ActiveSupport
- Rabbit
- るりま 

オブジェクト指向スクリプト言語 Ruby リファレンスマニュアル

- Ruby オフィシャルサイト <https://www.ruby-lang.org/ja/> 
- version 3.3 対応リファレンス
- 原著: まつもとゆきひろ
- 最新版URL: <https://www.ruby-lang.org/ja/documentation/> 

[\[edit\]](#)

使用上の注意

組み込みクラスのリファレンスはほぼ揃っています。標準添付ライブラリのリファレンスは一部未完成です。それ以外のドキュメントについては、まだまだ書き直しが必要です。

目次

- はじめに
- コマンド
- Rubyの起動
- 環境変数

Ruby 言語仕様

Ruby でのオブジェクト:

- オブジェクト
- クラス

プロセスの実行:

- Ruby プログラムの実行
- 終了処理
- スレッド

オブジェクト指向スクリプト言語 Ruby リファレンスマニュアル

<https://docs.ruby-lang.org/ja/latest/doc/index.html>

るりま

- doctree

- ドキュメント
- <https://github.com/rurema/doctree>

- BitClust

- ドキュメントシステム
- リファレンスデータベースの更新、表示、検索、html 生成など
- <https://github.com/rurema/bitclust>

るりま

■ Ruby リファレンスマニュアル改善計画進行中

■ るりま大刷新計画2022最終報告

- <https://gist.github.com/znz/c854585d93fb2c4b6b181870c5a509a2>
- <https://www.ruby.or.jp/ja/news/20230808>

■ History of Japanese Ruby reference manual, and future - Kazuhiro NISHIYAMA

- <https://slide.rabbit-shocker.org/authors/znz/rubykaigi2022-rurema/>
- <https://rubykaigi.org/2022/presentations/znz.html>

■ docs.ruby-lang.org/ja/ の生成方法を変えた - Kazuhiro NISHIYAMA

- <https://speakerdeck.com/znz/nosheng-cheng-fang-fa-wobian-eta>

BitClust

- BitClust データベース
 - rd をコンパイル
 - データをテキストファイルで持つ

```
# BitClust データベースの初期化・生成
```

```
bitclust setup
```

```
# BitClust データベースから指定した種類(ここではライブラリ)のエントリをリストする
```

```
bitclust list --library
```

```
# BitClust データベースから静的 html を生成
```

```
bitclust statichtml -o ~/bitclust-html
```

```
# リファレンスの 1 ファイルから html を生成
```

```
bitclust htmlfile refm/api/src/_builtin/Range > Range.html
```

BitClust

■ refe

- Ruby リファレンスマニュアルのエントリを引くためのコマンドライン用ツール
- BitClust データベースを利用
- デーモン化
 - データベースをフロントオブジェクトとして dRuby サーバを起動

```
refe --server=druby://127.0.0.1:12345
```

- データベースとしての Ruby
 - PORO でデータ操作
 - ファイルに永続化
 - SQL とか O/R マッピングとか要らない
 - ふつうの Ruby プログラミング

わたしたちの身近にある **dRuby**

- プリローダ
- 並列
- API
- データベース

夢が広がる！！



わたしたちの身近にある dRuby

- プリローダ
- 並列
- API
- データベース

分散オブジェクトシステムに
入門したくなったら





dRubyによる分散・Webプログラミング / 関将俊
<https://www.ohmsha.co.jp/book/9784274066092/>

ご清聴

ありがとう

ございました