

RDoc meets YARD - RDocへの YARD解析機能の組み込み

Ruby Association Activity Report

OKURA Masafumi, 2025-08-28

pritty_print(self)

- 名前：大倉雅史（おおくらまさふみ）
- 所属：フリーランス
- 活動：Kaigi on Rails（発起人、チーフオーガナイザー）、OSS活動（Albaなど）、登壇（RubyConfTw 2025, Rails World 2025, Friendly.rbなど）
- 好きなメソッド：`BasicObject#instance_eval`

RDoc
meets
YARD

RDoc

meets

YARD

RDocとは

- Rubyに標準添付されている（別途のインストールが不要な）ドキュメンテーションツール
- HTMLドキュメントを生成する`rdoc`コマンドとコマンドラインからドキュメントを閲覧する`ri`ツールを同梱
- Ruby on Railsで採用されている他、Ruby本体のドキュメントもRDoc記法で書かれている

Pages

- COPYING
- COPYING.ja
- LEGAL
- NEWS ►
- README.ja
- README
- bsearch
- bug_triaging
- case_mapping
- character_selectors
- command_injection
- contributing ►
- calendars
- dig_methods
- distribution
- dtrace_probes
- encodings
- exceptions
- extension.ja
- extension

Ruby Documentation

Welcome to the official Ruby programming language documentation.

Getting Started

New to Ruby? Start with our [Getting Started Guide](#).

Core Classes and Modules

Explore the essential classes and modules:

- [String](#) - Text manipulation and string utilities.
- [Symbol](#) - Named identifiers inside the Ruby interpreter.
- [Array](#) - Ordered collections of objects.
- [Hash](#) - Key-value pairs for efficient data retrieval.
- [Integer](#) - Integer number class.
- [Float](#) - Floating-point number class.
- [Enumerable](#) - Collection traversal and searching.
- [File](#) - File operations and handling.
- [IO](#) - Input/output functionality.
- [Time](#) - Time representation.
- [Regexp](#) - Regular expressions for pattern matching.
- [Range](#) - Representing a range of values.
- [Exception](#) - Base class for all exceptions.
- [Thread](#) - Multithreading and concurrency.

RDoc
meets
YARD

YARDとは

- gemのドキュメンテーションなどに広く使われているドキュメンテーションツール
- 標準ではないので別途インストールが必要
- ``@param``などを使った記法であり、独自の型記述も可能
- RDocもパースすることができる
- 個人的にもYARDを使うことが多い

Top Level Namespace
▼ Alba
Association < Object
ConditionalAttribute < Object
DefaultInflector
Deprecation
Error < StandardError
Layout < Object
NestedAttribute < Object
Railtie < Railtie
► Resource
Type < Object
TypedAttribute < Object
UnsupportedBackend < Error
UnsupportedType < Error

Module: Alba

Defined in: lib/alba.rb	more...
-------------------------	---------

Overview

This file includes public constants to prevent circular dependencies.

Defined Under Namespace

Modules: [DefaultInflector](#), [Deprecation](#), [Resource](#) **Classes:** [Association](#), [ConditionalAttribute](#), [Error](#), [Layout](#), [NestedAttribute](#), [Railtie](#), [Type](#), [TypedAttribute](#), [UnsupportedBackend](#), [UnsupportedType](#)

Constant Summary

collapse

Serializer =

[Resource](#)

REMOVE_KEY =

A constant to remove key from serialized JSON

[Object](#).[new](#).freeze

Class Attribute Summary

collapse

[.backend](#) ⇒ [Object](#)

Returns the value of attribute backend.

[.encoder](#) ⇒ [Object](#)

Returns the value of attribute encoder.

[.inflector](#) ⇒ [Object](#)

Getter for inflector, a module responsible for inflecting strings.

Class Method Summary

collapse

[collection?<T> \(List\) ⇒ Boolean](#)

[find](#)

ツールの分断

Schedule

[< Back](#)

Vinicius Stock

  @vinistock

Vinicius Stock is a Senior Software Developer working on the Ruby developer experience team at Shopify. Vini started his journey writing Ruby on Rails applications in 2015 and now dedicates his time to improve developer tools, language servers, gradual typing and debuggers in the Ruby ecosystem.

EN

The state of Ruby dev tooling

During the last few years, the Ruby community invested significant effort into improving developer tooling. A lot of this effort has been divergent; trying out many solutions to find out what works best and fits Rubyists expectations.

So where are we at this point? How do we compare to other ecosystems? Is it time to converge, unite efforts and reduce fragmentation? And where are we going next? Let's analyze the full picture of Ruby developer tooling and try to answer these questions together.

Presentation Material

2023年

自作 ドキュメンター ションツール

その時点でのおおくらの考え

- ドキュメントを書くのは楽しくない
- 既存のドキュメンテーションツールには問題がある
 - RDocもYARDもメンテナンス状況が良くない（最新のRubyの文法に対応し切れていないなど）
- 色々アイデアがある（RSpecからドキュメント作れない？とか）
- 新しいドキュメンテーションツールを作って全部統合だ！

やりたいこと：
ツールの統合と、
機能の追加

2024年

RubyKaigi 2024



Stan Lo (st0012)

“新しいドキュメンテーションツールを自分で作るより、RDocをメンテナンスすれば、どうやって使ってもらうかを考える必要がなくなるし、ツールが新しく増えるだけにはならない”

HOW STANDARDS PROLIFERATE:

(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC.)

SITUATION:
THERE ARE
14 COMPETING
STANDARDS.

14?! RIDICULOUS!
WE NEED TO DEVELOP
ONE UNIVERSAL STANDARD
THAT COVERS EVERYONE'S
USE CASES.



SOON:

SITUATION:
THERE ARE
15 COMPETING
STANDARDS.

...はい

「ツールの統合は
RDocを中心にし
て行おう」

YARDとは（再掲）

- gemのドキュメンテーションなどに広く使われているドキュメンテーションツール
- 標準ではないので別途インストールが必要
- `@param`などを使った記法であり、独自の型記述も可能
- RDocもパースすることができる
- 個人的にもYARDを使うことが多い

YARDで
いいのでは？

RDoc > YARD

なぜRDocなのか

- 標準添付されているので別途のインストールなどが不要
- RDocはなくなることはない
 - CRubyのソースコード内のドキュメントを解釈できるのはRDocのみ
- YARDのメンテナンス状況が不安
 - 大きめの改善も最近は少ないし、メンテナンスは基本個人がしている

2024年9月

プロジェクト採択

RDocへの拡張機能基盤の実装

プロジェクト概要

Rubyのドキュメンテーション関連ツールとしては標準添付のRDocと広く使われるYARDが存在しており、記法を含めて統一されていない状況が長く続いています。近年ではrbs-inlineも登場するなどし、ドキュメンテーションツールについての要求は高まる一方、対応は進んでいません。

本プロジェクトでは、RDocにドキュメンテーションツールを集約するという方針の下、RDocに拡張機構を実装することで様々な要望に応えられる基盤を作ります。

応募者名

大倉 雅史

擴張機能基盤

プロジェクト当初の考え

- 開発者によってドキュメンテーションツールに必要とする機能は異なるという仮説
- 様々なニーズに対して拡張機能基盤を作ることに対応
 - 将来的に様々な拡張機能を実装する予定
 - YARD形式のコメントをドキュメントに変換する機能も拡張機能の一つとして実装したい

課題：

拡張機能基盤の
インターフェース

Proposal: RDoc extension system #1257

okuramasafumi started this conversation in Ideas



okuramasafumi on Sep 15, 2024



Abstract

I'd like to propose an extension system for RDoc.

[YARD](#) has been popular for years and new comment usages such as [rbs-inline](#) has been introduced. We need to organize efforts around comment space so that we don't have to duplicate our effort.

The important thing is that RDoc is currently the only library bundled with Ruby that handles comments. RDoc is used to parse CRuby's C comment so it's not realistic to remove it in near future. Therefore, I think RDoc is a good place for collecting efforts around comments and documentation.

Then an extension system comes in. Because we have different needs. When each extension is implemented as a gem, it's possible for other gems to bundle their extension to extend RDoc behavior. For example, language servers can extend RDoc so that it can be used with editors.

Details

RDoc has three main components: parsers, store and generators. It would be beneficial to be able to extend all of them.

<https://github.com/ruby/rdoc/discussions/1257>

Proposal: RDoc extension system

当時考えていたことを雑に書いた ディスカッション

メンターの 須藤さんと 相談

拡張機能同士を
疎結合にしたい

当初のアイデア

- RDocc本体が各タイミングでイベントを発火
- 拡張機能はイベントを受け取り処理をする
- PubSubっぽい仕組み



<https://github.com/ruby/rdoc/pull/1321>

イベント駆動拡張機能基盤の実装、
具体的な拡張機能として簡易的な
YARDパーサーも実装

PoC for the event-based plugin system with YARD parsing plugin #1321 

 Draft okuramasafumi wants to merge 2 commits into `ruby:master` from `okuramasafumi:parse-yard` 

Conversation 55 Commits 2 Checks 30 Files changed 7

 **okuramasafumi** commented on Mar 21 · edited  Contributor 


This PR introduces the event-based plugin system along with a YARD-parsing plugin as an example of concrete plugins.

It demonstrates how we can build practical plugins while keeping the core code small. In this case, YARD-related logic is encapsulated in a plugin and used only when users specify with `--plugins` option.

This YARD plugin successfully parses <https://github.com/okuramasafumi/alba> comments written with YARD, but doesn't print YARD-specific information such as types since current RDoc doesn't support type information.


This PR is completely proof of concept, and is not meant to be merged now.
I'd like to discuss if this is the right way to extend RDoc for developers, and even for core team, and improve maintainability.

The original discussion is [#1257](#)



 **okuramasafumi** force-pushed the `parse-yard` branch from `f2bd262` to `5a2f695` 5 months ago [Compare](#)

Reviewers

-  kou
-  tompng
-  st0012
-  flavorjones
-  colby-swandale
-  vinistock

Assignees

No one assigned

Labels

None yet

大失敗

私がしてしまった失敗

- RDocメンテナの実際のニーズや懸念に応えていない
 - 拡張機能基盤のコンセプトはともかく、インターフェースは一度決めると変更が困難なので、軽々しく決定できない
 - 実際の拡張機能を先に作ってそこからインターフェースを決めたい
- 目の前の問題を解決できない
 - 拡張機能基盤は遠回りになる

どうすればよかったか

- 先にYARD対応を単体でやるべきだった
 - そこから知見を抽出できる
- もっと早めに動くべきだった
- 気づいたときには2025年の3月
 - プロジェクトはほぼ終わりの時期



2025年4月

RDdocのメンテナンス
と相談する

RubyKaigi 2025

コード“懇親会

@ktou

@st0012

@vinistock

このとき話したこと

- 拡張機能基盤どうする？
- YARD対応をどう進める？

Adding YARD document support to RDoc #1344 📌

🔗 Open



okuramasafumi opened on Apr 20 · edited by okuramasafumi

Edits ▾

Contributor



Background

I implemented a plugin system for RDoc in [#1321](#) and YARD parsing plugin is implemented.

In the discussion with [@st0012](#), [@kou](#) and [@vinistock](#) we decided that we should implement YARD parsing feature as a standalone feature without plugin.

The note exists in [#1257](#) ([comment](#))

Steps

There are some steps to have effective YARD parsing feature in RDoc.

- ☐ Adding basic framework to support YARD style document (`RDoc::Yard` class or similar) ...
- ☐ Adding support of YARD tags that already works with current RDoc such as `@yield` and `@private` ...
 - ☐ `@yield`
 - ☐ `@private`
- ☐ Adding support of YARD tags that needs simple modification of RDoc such `@deprecated` ...
 - ☐ Adding `:deprecated` directive
 - ☐ Adding `@deprecated`

https://github.com/ruby/rdoc/issues/1344

改めてRDocとしてほしいものをまとめ直したissue

このIssueで決まった方針

- 拡張機能基盤は一旦放棄する
- まずいくつかのYARDタグを先行して実装する
 - 残りのタグは先にRDoc側の機能を追加する必要があるため
- 型に関してはRBSのサポートを優先する
 - YARDの独自記法はサポートされない可能性もある

YARD実装の方針

- rdoc gemはyard gemに依存できない
- YARDタグのパーズは手書き
- RDocにはtomdocというフォーマットをサポートするための仕組みがあるので、流用できるかも

2025-08-26

feat(yard): add initial support of YARD tags #1416

🔗 Open okuramasafumi wants to merge 2 commits into `ruby:master` from `okuramasafumi:add-yard-yield-private-tag`

💬 Conversation 1

📄 Commits 2

📄 Checks 32

📄 Files changed 6



okuramasafumi commented 19 hours ago

Contributor ...

At this time, `@yield`, `@private` and `@api private` are supported.
Ref: [#1344](#)



feat(yard): add initial support of YARD tags ...

Verified ✖ 6781c04



okuramasafumi commented 19 hours ago

View reviewed changes

lib/rdoc/yard.rb

```
112 + # Extract parameter names from YARD type specification
113 + # e.g., "[String, Integer]" -> "value1, value2"
114 + # e.g., "[item, index]" -> "item, index"
115 + def extract_param_names(params_string)
```



okuramasafumi 19 hours ago

Contributor Author ...

This part is kind of too much, but YARD's `@yield` might cause confusion to use type name instead of param name. Here it replaces type name with temporary param name,



<https://github.com/ruby/rdoc/pull/1416>

`@yield`, `@private`, `@api private`を実装し、RDocの同等機能であるかのように振る舞うようにした

拡張性は特段考慮せず、一番短い実装を選択

CIが通った（今日）

今後の課題と
やりたいこと

RDocでやりたいこと

- YARD対応を完成させる
 - yard gemを使わないでYARD形式のコメントからHTMLドキュメントを生成できる状態にする
- 他のドキュメンテーションツールから良い所を抽出する
 - Sphinx, TypeDoc, RustDoc, Docusaurus, etc.
 - 特にdoctestはRubyコミュニティに受け入れられるのか興味あり

The image features a dark blue background with a network of glowing white lines and dots, resembling a circuit board or neural network. Scattered throughout are various white line-art icons: a human head profile with internal circuitry, a lightbulb, a gear, a microchip, a line graph, a computer monitor with a waveform, and a hexagon. In the center, a large, glowing purple and blue circle contains a white icon of a brain with circuitry, mounted on a clipboard. Below this circle, the text "GENERATIVE AI" is written in a bold, white, sans-serif font.

GENERATIVE AI

特に Claude Code
(2025年4月)

やれることが
増えた...？

生成AIが助けになりそうなこと

- 網羅的なテストスイートの作成
- 複雑な既存コードの読解補助
- 他のツールの調査

謝辭

メンターの須藤さん
Rubyアソシエーションの皆様

ありがとうござい
ました！！！！

今後もRDdocに
取り組んで
いきます