Rigorを入れろ今すぐに

Install Rigor as soon as possible



pixiv Inc.
USAMI Kenta

pixiv

2026-10-19 #afterkor2026 pixiv
Kaigi on Rails 2026 After Event from ピクシブ

お前誰よ



- うさみけんた（@tadsan）/ **Zonu.EXE** / **にゃんだーすわん**
- **ピクシブ株式会社** Platform Div > WebTechnology Team **PHPer**
  - 2012年末から現職、APIとかCIとかいろいろなところを見つめてきました
  - 最近チームが再編されてインフラっぽい仕事もしてます
- Emacs PHP Modeを開発しています（2017年-）
- プログラミング言語にちょっとこだわりのある素人（spcamp2010）

PHPの「型」はどこまで戦えるか

Can PHP Types Still Compete?
Pushing the Limits of Static Analysis.



pixiv Inc.
USAMI Kenta

pixiv

1

2026-10-03 #phpcon ehime
PHPカンファレンス愛媛 @愛媛大学 城北キャンパス

pixiv

PHPには型を書きたい

「自分の未来予測を信じてちょっと意地を張ってみる」　まつもとゆきひろ氏がRubyに型宣言を入れない理由

コピー　ブックマーク



pixiv

型宣言がなくても静的型宣言のメリットは実現できる

例えば、Ruby3.0から静的型チェックを導入するようになったんですけれども、型宣言はなかったんですね。つまり、型情報はそのプログラムとは別のところに持っていて、その情報を使って型チェックを行おうと。ないものについては推論をしようというアプローチですね。

もちろん、今のRubyの静的型チェックは誕生したばかりでまだ未成熟なんですけれども、原理的には、型宣言なしで型チェックの恩恵をある程度受けることができると言われています。がんばっている途中ですね。ただ、型宣言はないので、ドキュメントとしての役割は果たせません。

ただ、そうはいっても型情報はいろいろあるので、そこによる推論結果みたいなものをIDEで表示することによって、ドキュメントとしての役割をある程度果たせますし、Rubyには、インラインドキュメント。つまり型がなくてもいずれにしても振る舞いは書かないといけないのでインラインドキュメントをする習慣があって、型情報を書いたらそれを取り込んでくる。型情報として取り込んでくるというツールも今開発中です。

Rubyに型宣言を入れない理由

ただ、一度言語に型宣言を持たせると、もう取り除けないわけですよ。型宣言のあるプログラムがもうそこにたくさんできちゃうわけです。さらに、コミュニティの中におそらくですが、型警察みたいなものが出てくると思うんですね。

ジェムとかライブラリを作っている人のところに行って、「お前の使っているこのライブラリに型宣言がないんだけど、型宣言は絶対にあるべきだ」みたいに言って、型宣言を付けていないプログラムに対して文句をつけまくるみたいな人たちがコミュニティに登場しそうな気がするんですね。実際によその言語ではそういうことが発生していると聞くんですね。

2020年にRBSが導入されてから型は普及したか？

RBS? RBI?
よくわからん…

PHPには型を書きたい

Rubyでも書きたい人は書けばいいと思っていた

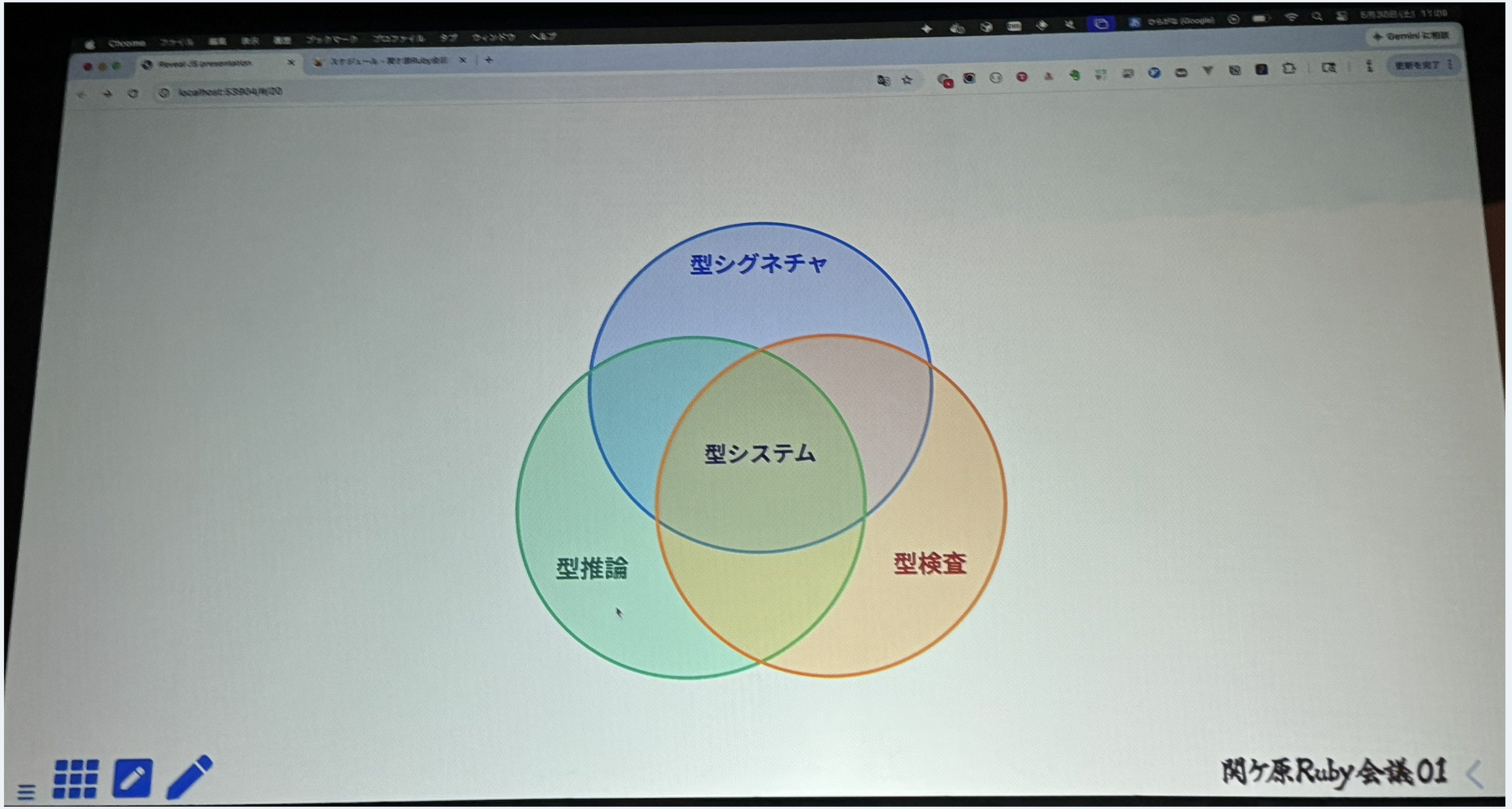


関ケ原Ruby会議01

拙者、
『型は欲しいが型は書きたくない』
者たちとの和睦を結び、
るびぃにおける型の領地安堵を
実現せんと欲す者也

大江森塚三矢大坂守小輔次郎真年
@sanfrecce_osaka
2026/05/30 #sekigahara01

pixiv


型シグネチャ
型システム
型推論
型検査
関ケ原Ruby会議01

型に関するみんなのお気持ち

- （シグネチャ）コードに無駄な記述を書きたい人は多くない
  - →
- （型検査）明記しなくても有用な静的検査ができるなら反対する人も少ない
  - →
- （型推論）型を書かなくても型推論で型が絞り込めるなら
  - →

型に関するみんなのお気持ち

- （シグネチャ）コードに無駄な記述を書きたい人は多くない
  - → 冗長だけど有用な記述をすべきか、という点で意見が分かれる
- （型検査）明記しなくても有用な静的検査ができるなら反対する人も少ない
  - → 偽陽性に煩わされるくらいなら検査なんか、するだけ無駄では？
- （型推論）型を書かなくても型推論で型が絞り込めるなら
  - → 型推論の結果が十分に絞り込まれていないと、却って冗長になる

導入が難しい…
運用が難しい…
概念が難しい…

PIXIV FANBOX

クリエイター



にゃんだーすわん

クリエイターページ

投稿する

ダッシュボード

投稿管理

プラン管理

クリエイター設定

ファン一覧

おたより

支援金管理/振込

FANBOXプリント

FANBOXプリントとは

コンテンツ提供者申請

クリエイター優待



ちょっとすごいRubyの型チェッカーを作ってます

編集

2026年5月25日 23:20・全体公開

これは自画自賛しても許されるくらいちゃんとすごい技術だと思っているのですが…

私はLispやRubyのような動的言語が大好きで、2016年くらいからPHPに型をつけながら暮らしている者です。もともとは2010〜2014年くらい前までShibuya.rbやRuby会議に顔を出していたものの、仕事でPHPをやっていることもあってPHP方面に軸足を移していました。

pixiv

なぜ新しい型チェッカーを書きはじめたのか

How I Ended Up Writing a New Type Checker for Ruby



pixiv Inc.
USAMI Kenta

pixiv

1

2026-09-05 #rubykaigi followup
RubyKaigi 2026 follow up

pixiv

難しいことを
オブラートに包む

チームメンバーは
型とか書かなくていい

AIエージェントに
SKILLで判断させる

Rigorの思想

Rails 向け Rigor: mise を使ったステップバイステップセットアップ

このウォークスルーでは、Rails プロジェクトをゼロから最初の `rigor check` 実行まで導きます。`mise` を使って Ruby 4.0 と合わせて Rigor をインストールし、解析器をプロジェクトの `Gemfile` の外に置きます。

セットアップの進め方は 2 通りあります。

	アプローチ	向いているケース
A	`rigor-project-init` スキル	ほとんどのプロジェクト。スキルがスタックを検出し、プラグインを提案し、設定を書いてくれます。
B	手動のステップバイステップ	各決定を明示的に制御したい場合。

どちらも同じ結果になります。**迷ったら Path A を選んでください。**

pixiv

お使いの言語でセットアップ

上のプロンプトはただの自然言語です。どの言語で頼んでも、エージェントは同じリンク先の手順に従います。つまりセットアップの対話全体を母語で進められます。下のすぐ使えるプロンプトはそれぞれ、エージェントにその言語で対話しながらセットアップを一緒に進めるよう明示的に依頼します:

> **お使いの言語でセットアップ**
>
> > **[ja] 日本語**
> >
> > ```
> > 日本語で対話しながら、このプロジェクトに Rigor をインストールする作業を一緒に進めてください。次の手順
> > https://raw.githubusercontent.com/rigortype/rigor/refs/heads/master/docs/install.md
> > ```
> >
> > **[zh-Hans] 简体中文**
> >
> > **[zh-Hant] 繁體中文**
> >
> > **[ko] 한국어**
> >
> > **[pt-BR] Português (Brasil)**
> >
> > **[pt] Português**

pixiv

PHPStanで学んだ経験則

ユーザーは適切な設定を選べない

無理してmaxレベルを目指そうとする

PHPStan自体も発展途上なのでコードを曲げることになりかねない

偽陽性に対して過剰なコードを書かされる

呼び出し方

AIコーディングエージェントに次のいずれかを伝えてください。

> “Set up Rigor in this project.” “Configure Rigor for this Rails app.” “Add type checking.”

エージェントは次のコマンドを実行して応答するはずです。

```
rigor skill rigor-project-init
```

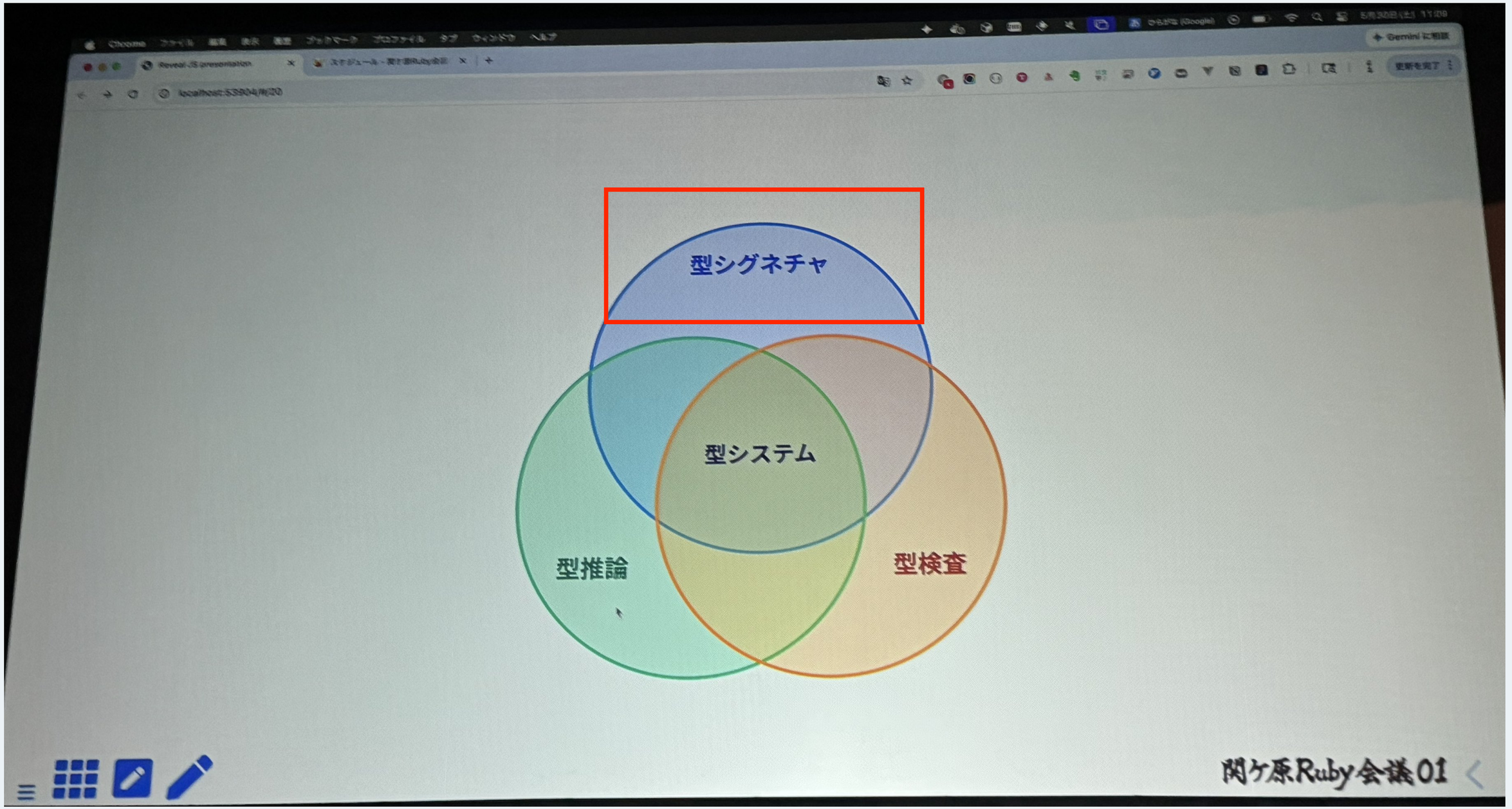
このコマンドはSKILL定義を標準出力に出力します。SKILLファイルとその `references/` ディレクトリの絶対パスを含む短いヘッダー、続いてSKILL本文です。エージェントはその手順に従い、ヘッダーが示すディレクトリから `references/NN-*.md` ファイルを順に読み込みます。

エージェントが自動的にコマンドを実行しない場合は、明示的に依頼してください: “**`rigor skill rigor-project-init` を実行して、表示される手順に従ってください。**”

このフローは任意のバンドルスキルに対して使用できます。

pixiv

プロジェクトの段階によって無視していいエラーも存在する


型シグネチャ
型システム
型推論
型検査
関ケ原Ruby会議01

実行時にとりうる
予測可能な状態

Rubyには構文上
要求される型宣言は
ないのだから
どう書いてもいいはず

RailsのCoCも
人間が期待する型

型チェッカーが
人間の期待に
応えられないのが悪い

メンテナンスとオーサリング

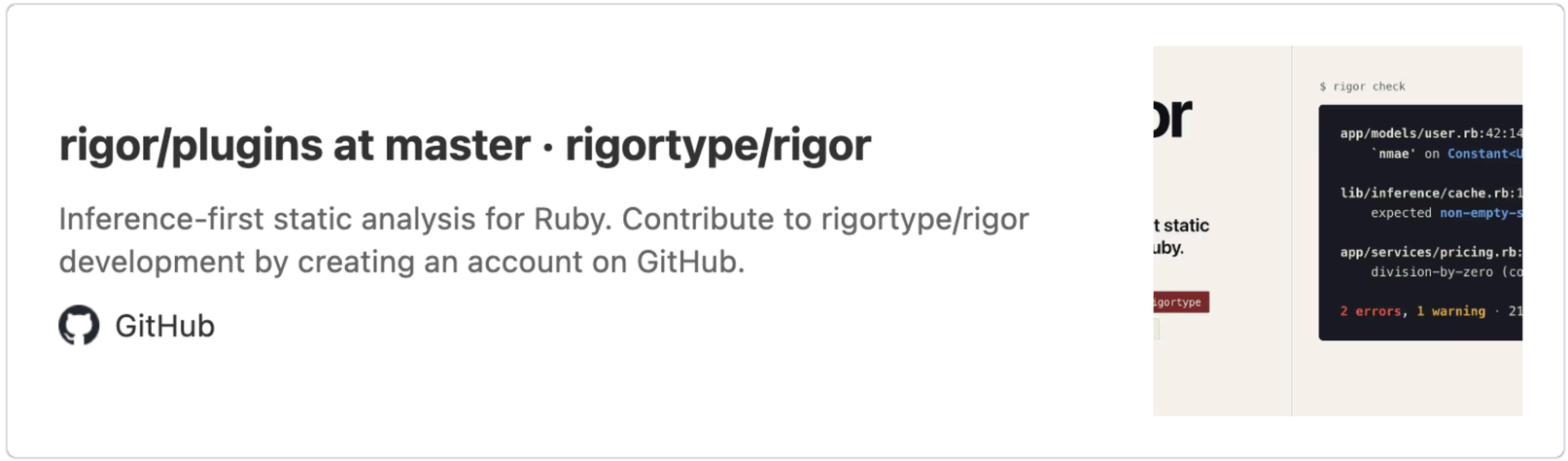
- `rigor-upgrade`：新しいRigorバージョンをクリーンに採用します。ベースラインとの差分を取り、本物の新規キャッチをsig品質の偽陽性から仕分けし、再生成します。
- `rigor-doctor`：セットアップが健全であること（設定が解決し、プラグインがロードされ、ベースラインが新鮮で、RBS環境が壊れていないこと）を検証します。トラブルシューティングを参照してください。
- `rigor-plugin-author`：自分のリポジトリに新しいRigorプラグインをスキャフォールドして、Rigorが推論できないアプリケーションDSLやメタプログラミングパターンについて教えます。バンドルされたプラグインがプロジェクトの依存するフレームワークやDSLをカバーしていないときに使ってください。

pixiv

理屈上は任意の項に型をつけられる

型推論がうまくいくのは単機能のちっちゃいスクリプトだけでは…？ と思われるかもしれませんが、現実的なプロジェクトでもちゃんとしっかり型がつくように頑張っています。

秘訣はPHPStan譲りの**プラグインシステム**です。
正攻法で型を推論できないメタプログラミングやDSLは全部これでカバーできます。



サンプルコードも結構いっぱい考えていて、Lispみたいな式を静的に評価したり、Rack的なマイクロフレームワークのコントローラに暗黙的なインターフェイス(プロトコル)を与えて、パラメータに型をつけるなんてもこともできます。

プラグインの書き方は覚えなくていい

プログラムがとりうる作用は戻り値だけではない

pixiv

Rigorによる持続可能なRailsの静的型解析

Speaker

USAMI Kenta

Format

Short session (15 min)

Track

General

Abstract

筆者が開発する「Rigor」は、人力での運用と型の知識をほとんど必要としない、Rubyのための新しい静的解析ツールです。

Rubyにおける型エコシステムが発展を続ける一方、導入を検討したものの「型定義管理の重さ」「ActiveRecordをはじめとするDSLやメタプログラミングとの相性」「チームメンバーへの学習コスト」といった現実的な運用の壁に直面することも少なくないかと思います。

筆者が提案するのは、手動による型定義のメンテナンスを避け、型推論による決定論的な型定義の生成とコーディングエージェントを活用することで、プロジェクトの実情に応じた的確なセットアップを半自動で行うアプローチです。

Rigorは既に「動作している」実績を持つRubyコードに対して、過剰な警告を避けながら徹底的な型推論を行います。型定義を手動でメンテナンスすることなく、AI時代においてもRubyの良さをスポイルせずに「ガードレール」としての型を享受できるよう設計されています。

本セッションでは、Rigorの誕生背景と現実のRailsプロジェクトへ型理論を適用する具体的なアプローチについてご紹介します。

pixiv

Rigorによる持続可能なRailsの静的型解析

Speaker

USAMI Kenta

Format

Short session (15 min)

Track

General

Withdraw Proposal

Status: NOT ACCEPTED Updated: 3ヶ月 ago

Abstract

筆者が開発する「Rigor」は、人力での運用と型の知識をほとんど必要としない、Rubyのための新しい静的解析ツールです。

Rubyにおける型エコシステムが発展を続ける一方、導入を検討したものの「型定義管理の重さ」「ActiveRecordをはじめとするDSLやメタプログラミングとの相性」「チームメンバーへの学習コスト」といった現実的な運用の壁に直面することも少なくないかと思います。

筆者が提案するのは、手動による型定義のメンテナンスを避け、型推論による決定論的な型定義の生成とコーディングエージェントを活用することで、プロジェクトの実情に応じた的確なセットアップを半自動で行うアプローチです。

Rigorは既に「動作している」実績を持つRubyコードに対して、過剰な警告を避けながら徹底的な型推論を行います。型定義を手動でメンテナンスすることなく、AI時代においてもRubyの良さをスポイルせずに「ガードレール」としての型を享受できるよう設計されています。

本セッションでは、Rigorの誕生背景と現実のRailsプロジェクトへ型理論を適用する具体的なアプローチについてご紹介します。

pixiv

実際に使っている人の
フィードバックが欲しい！

もっというと
マジレスが欲しい！

作り始めたばかりなので乞うご期待

pixiv

Rubyに型を
書きたくないのに
Docに型を書かされる

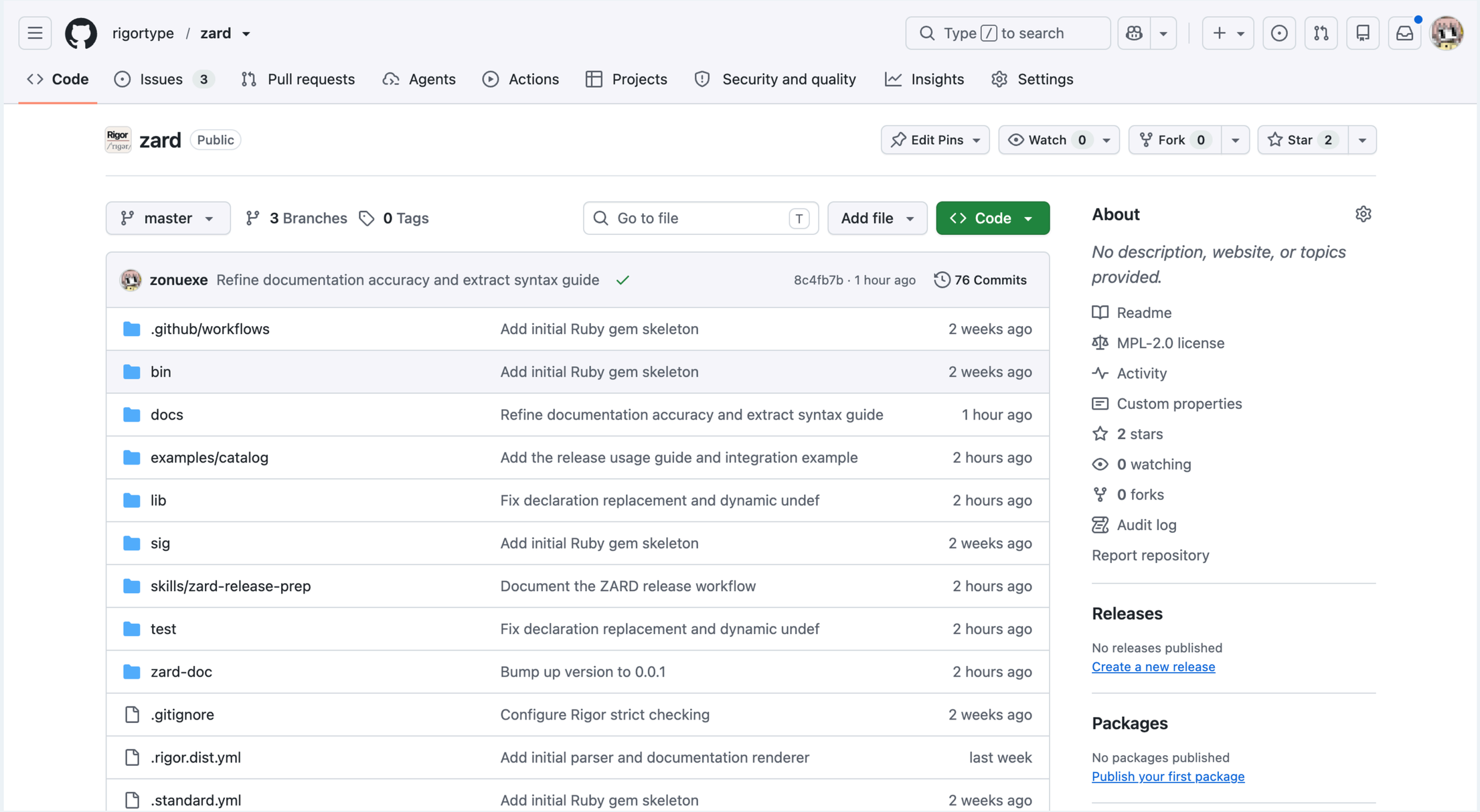
ZARD

Yet another YARDoc.



pixiv Inc.
USAMI Kenta

pixiv


rigortype / zard
Type / to search
Code
Issues 3
Pull requests
Agents
Actions
Projects
Security and quality
Insights
Settings
zard
Public
Edit Pins
Watch 0
Fork 0
Star 2
master
3 Branches
0 Tags
Go to file
T
Add file
Code
zonuexe Refine documentation accuracy and extract syntax guide
8c4fb7b · 1 hour ago
76 Commits
.github/workflows Add initial Ruby gem skeleton 2 weeks ago
bin Add initial Ruby gem skeleton 2 weeks ago
docs Refine documentation accuracy and extract syntax guide 1 hour ago
examples/catalog Add the release usage guide and integration example 2 hours ago
lib Fix declaration replacement and dynamic undef 2 hours ago
sig Add initial Ruby gem skeleton 2 weeks ago
skills/zard-release-prep Document the ZARD release workflow 2 hours ago
test Fix declaration replacement and dynamic undef 2 hours ago
zard-doc Bump up version to 0.0.1 2 hours ago
.gitignore Configure Rigor strict checking 2 weeks ago
.rigor.dist.yml Add initial parser and documentation renderer last week
.standard.yml Add initial Ruby gem skeleton 2 weeks ago
About
No description, website, or topics provided.
Readme
MPL-2.0 license
Activity
Custom properties
2 stars
0 watching
0 forks
Audit log
Report repository
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package

pixiv



pixiv



pixiv