今日で分かる！
カスタムコップの作り方

株式会社YOUTRUST
寺井 省吾

自己紹介

- 寺井 省吾 (@krpk1900_dev)
- 株式会社YOUTRUST
  - SRE (パフォーマンス改善など)
- Kaigi on Rails Organizer
- 個人開発
  - 席替えメーカー
  - 大谷メーター
  - 世界一セキュリティが堅牢なアプリ

@krpk1900_dev

CFPの内容(30分)

1. カスタムコップ導入のモチベーションとメリット (2分)
2. [デモ] 簡単なカスタムコップ作成の例 (12分)
    1. ASTやnodeといった用語の説明
    2. パターンマッチの方法
    3. カスタムコップの雛形
    4. テストの書き方
    5. auto-correctの実装
3. [実例1] 参照系アクションをリードレプリカに向けるカスタムコップ (7分)
    1. 導入のモチベーションとメリット
    2. 実装の過程
    3. 導入の方法、rubocop_todo.ymlへの記述
    4. 効果
4. [実例2] クラス群Aからクラス群Bの呼び出しを禁止するカスタムコップ(7分)
    1. 導入のモチベーションとメリット
    2. 実装の過程
    3. 導入の方法、rubocop_todo.ymlへの記述
    4. 効果
5. まとめ (2分)

今日の内容(30分→10分)

1. カスタムコップ導入のモチベーションとメリット (2分)
2. **[デモ] 簡単なカスタムコップ作成の例 (12分)**
    1. **ASTやnodeといった用語の説明**
    2. **パターンマッチの方法**
    3. **カスタムコップの雛形**
    4. **テストの書き方**
    5. **auto-correctの実装**
3. **[実例1] 参照系アクションをリードレプリカに向けるカスタムコップ (7分)**
    1. 導入のモチベーションとメリット
    2. **実装の過程**
    3. **導入の方法**、rubocop_todo.ymlへの記述
    4. 効果
4. [実例2] クラス群Aからクラス群Bの呼び出しを禁止するカスタムコップ(7分)
    1. 導入のモチベーションとメリット
    2. 実装の過程
    3. 導入の方法、rubocop_todo.ymlへの記述
    4. 効果
5. まとめ (2分)

作りたいカスタムコップ

- 参照系のアクションに対して、接続先をリードレプリカに変更する
  コールバックの書き忘れを警告するカスタムコップ

```ruby
class Api::UsersController < Api::ApplicationController
  around_action :with_reader, only: %i(index show)

  def index
    # 処理内容
  end

  def show
    # 処理内容
  end
end
```

Controllerの例

with_readerの定義

```ruby
module TreatDatabase
  extend ActiveSupport::Concern

  included do
    private

    def with_reader(&block)
      ActiveRecord::Base.connected_to(role: :reading, &block)
    end
  end
end
```

カスタムコップ作成の流れ

1. ファイルを作成する
2. 実装する
3. テストを書く
4. 有効化する
5. 実行してみる
6. auto correctを実装する

①ファイルを作成する

- lib/rubocop/cop/lint/作成するカスタムコップ名.rb

```ruby
module RuboCop
  module Cop
    module Lint
      class WithReaderCop < Base
        # TODO: 処理内容を書く


      end
    end
  end
end
```

②実装する

- アルゴリズムを考える
  1. Controllerであるかを確認
  2. indexかshowが定義されているかを確認
  3. around_action :with_reader, only: %i() の引数を取得
  4. 取得した%i()の引数にindexかshowがなければ警告

```ruby
class Api::UsersController < Api::ApplicationController
  around_action :with_reader, only: %i(index show)

  def index
    # 処理内容
  end

  def show
    # 処理内容
  end
end
```

②実装する

- アルゴリズムを考える
  1. **Controllerであるかを確認**
  2. indexかshowが定義されているかを確認
  3. around_action :with_reader, only: %i() の引数を取得
  4. 取得した%i()の引数にindexかshowがなければ警告

②-1. Controllerであるかを確認

- RuboCopでは、特定のノードに対応したコールバックが用意されている
  - on_send(node)
    - メソッド呼び出しのノードに対して処理を行う
  - **on_class(node)**
    - **クラスのノードに対して処理を行う**

②-1. Controllerであるかを確認

- on_class(node)
  - クラスのノードに対して処理を行う

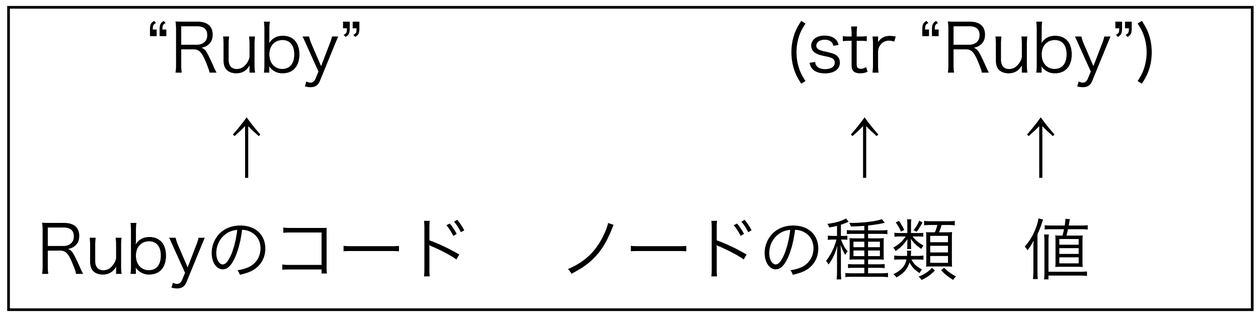
```ruby
def on_class(node)

  # ①Controllerでなければ終了
  return unless controller_class?(node)

  # TODO: 処理内容
end

private

# Controllerであるかを判定する
def controller_class?(node)
  node.identifier.const_name.end_with?('Controller')
end
```

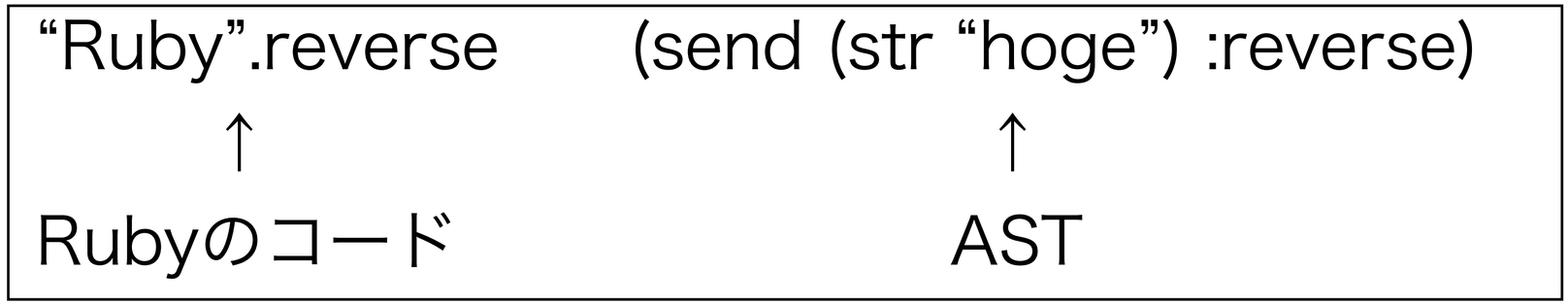
②実装する

- アルゴリズムを考える
  1. Controllerであるかを確認
  2. **indexかshowが定義されているかを確認**
  3. around_action :with_reader, only: %i() の引数を取得
  4. 取得した%i()の引数にindexかshowがなければ警告

②-2. indexかshowが定義されているかを確認

- ノードパターン
  - ASTに対する正規表現のようなもので、特定のコードパターンに
    マッチするコードの探索や取得ができる





②-2. indexかshowが定義されているかを確認

```ruby
# indexアクションの定義のノード
def_node_search :index_node, '(def :index ...)'

# showアクションの定義のノード
def_node_search :show_node, '(def :show ...)'
```

def_node_search
　　探索メソッドを定義している。
　　パターンにマッチするノードを探索し、
　　マッチしたノードを返す

```ruby
def on_class(node)
  # Controllerでなければ終了
  return unless controller_class?(node)

  index_action = index_node(node)
  show_action = show_node(node)

  # indexアクションもshowアクションも定義されていなければ終了
  return if !index_action && !show_action

  # TODO: 処理内容
end
```

②実装する

- アルゴリズムを考える
  1. Controllerであるかを確認
  2. indexかshowが定義されているかを確認
  3. **around_action :with_reader, only: %i() の引数を取得**
  4. 取得した%i()の引数にindexかshowがなければ警告

②-3. :with_reader, only: %i() の引数を取得

- ノードパターンで取得

```
# around_action :with_reader, only: %i(hoge fuga)のとき、hogeとfugaのノード
def_node_search :around_action_with_reader_node, <<~PATTERN
  (send nil? :around_action (sym :with_reader)
    (hash
      (pair
        (sym :only)
        $(array ...))))
PATTERN
```

nil?	レシーバが存在しない
:around_action	メソッド名がaround_action
array …	任意の要素の配列
$()	一致するノードをキャプチャ

```
def on_class(node)
  # Controllerでなければ終了
  return unless controller_class?(node)

  index_action = index_node(node)
  show_action = show_node(node)

  # indexアクションもshowアクションも定義されていなければ終了
  return if !index_action && !show_action

  # around_action :with_reader, only: %i()の中身を取得する
  actions = around_action_with_reader_node(node)

  # TODO: 処理内容
end
```

```
class Api::UsersController < Api::ApplicationController
  around_action :with_reader, only: %i(index show)

  def index
    # 処理内容
  end

  def show
    # 処理内容
  end
end
```

②実装する

- アルゴリズムを考える
  1. Controllerであるかを確認
  2. indexかshowが定義されているかを確認
  3. around_action :with_reader, only: %i() の引数を取得
  4. **取得した%i()の引数にindexかshowがなければ警告**

②-4. 引数にindexかshowがなければ警告

- add_offenceで警告を出す

```ruby
# action_nameがactionsに含まれているかどうかをチェックする
def check_action(action_name, actions, node)
  # actionsが空であれば、警告する
  if actions.none?
    add_offense(node, message: 'around_action :with_reader must be defined with the :only option including specific actions.')
  else # actionsが存在するが、action_nameが含まれていない場合
    # action_nameがactionsに含まれていれば終了
    return if actions.any? do |around_action_node|
      around_action_contains_action?(around_action_node, action_name)
    end

    # action_nameがactionsに含まれていなければ、警告する
    add_offense(actions.first.location.expression, message: sprintf(MSG, action: action_name))
  end
end

def around_action_contains_action?(around_action_node, action_name)
  # around_action_nodeがaction_name(indexなど)を含んでいるかどうかをチェックする
  around_action_node.children.any? { |child_node| child_node.type == :sym && child_node.children[0] == action_name }
end
```

②-4. 引数にindexかshowがなければ警告

- 最終的な形

```
def on_class(node)
  # Controllerでなければ終了
  return unless controller_class?(node)

  index_action = index_node(node)
  show_action = show_node(node)

  # indexアクションもshowアクションも定義されていなければ終了
  return if !index_action && !show_action

  # around_action :with_reader, only: %i()の中身を取得する
  actions = around_action_with_reader_node(node)

  # indexアクションがControllerに定義されているとき、around_action :with_reader, only: %i()に含まれているかどうかをチェック
  check_action(:index, actions, node) if index_action
  # showアクションがControllerに定義されているとき、around_action :with_reader, only: %i()に含まれているかどうかをチェック
  check_action(:show, actions, node) if show_action
end
```

③テストを書く

- 警告を出さないとき

```ruby
require 'rails_helper'
require 'rubocop'
require 'rubocop/rspec/support'
```

```ruby
context 'Controller内でindexとshowの両方が定義されているとき' do
  context 'around_action :with_reader, only: %i()でindexとshowの両方が指定されているとき' do
    it '警告しない' do
      expect_no_offenses(<<~RUBY)
        class TestController < ApplicationController
          around_action :with_reader, only: %i(index show)

          def index; end
          def show; end
        end
      RUBY
    end
  end
end
```

③テストを書く

- 警告を出すとき

```ruby
context 'around_action :with_reader, only: %i()でindexとshowの片方が指定されていないとき' do
  it '警告する' do
    expect_offense(<<~RUBY)
      class TestController < ApplicationController
        around_action :with_reader, only: %i(index)
                                          ^^^^^^^^^ around_action :with_reader must include `show` action in the `:only` array.

        def index; end
        def show; end
      end
    RUBY
  end
end
```

```ruby
context 'around_action :with_reader が記述されていないとき' do
  it '警告する' do
    expect_offense(<<~RUBY)
      class TestController < ApplicationController
      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ around_action :with_reader must be defined with the :only option including specific actions.

        def index; end
        def show; end
      end
    RUBY
  end
end
```

③テストを書く

- テスト一覧

```
RuboCop::Cop::Lint::WithReaderCop
  Controller内でindexとshowの両方が定義されているとき
    around_action :with_reader, only: %i()でindexとshowの両方が指定されているとき
      警告しない
    around_action :with_reader, only: %i()でindexとshowの片方が指定されていないとき
      警告する
    around_action :with_reader が記述されていないとき
      警告する
  Controller内でindexとshowの片方(index)が定義されているとき
    around_action :with_reader, only: %i()でindexが指定されているとき
      警告しない
    around_action :with_reader が記述されていないとき
      警告する
  Controller内でindexとshowの両方が定義されていないとき
    around_action :with_reader, only: %i()でindexとshowの両方が指定されていないとき
      警告しない

Finished in 0.17967 seconds (files took 3.44 seconds to load)
6 examples, 0 failures
```

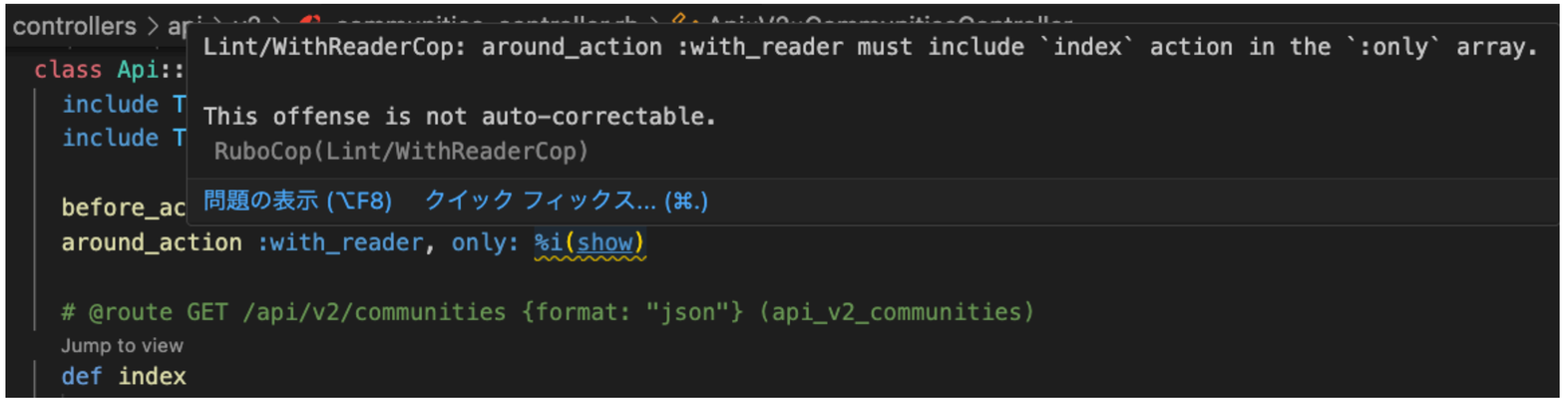
④有効化してみる

- .rubocop.yml を修正

```
require:
  - rubocop-rails
  - rubocop-rspec
  - rubocop-factory_bot
  - ./lib/rubocop/cop/lint/no_use_case_invocation_in_command.rb
  - ./lib/rubocop/cop/lint/no_command_invocation_in_command.rb
  - ./lib/rubocop/cop/lint/no_command_invocation_in_controller.rb
  - ./lib/rubocop/cop/lint/with_reader_cop.rb
```

```
Lint/WithReaderCop:
  Enabled: true
```

⑤実行してみる



Ruby LSPを導入しているため、
エディタ上で警告が出る

```
youtrust-webapp on  master [$!] via  desktop-linux via  v20.15.0 via  v3.3.5
❯ docker compose exec app bundle exec rubocop app/controllers/api/v2/communities_controller.rb
Inspecting 1 file
W

Offenses:

app/controllers/api/v2/communities_controller.rb:6:37: W: Lint/WithReaderCop: around_action :with_reader must include show action in the :only array.
  around_action :with_reader, only: %i(index)
                                    ^^^^^^^^

1 file inspected, 1 offense detected
```

ターミナル上で実行

⑥auto correctを実装する

- add_offenceにブロックを与えることによってauto correctを実現できる

```ruby
# action_nameがaround_actionsに含まれていなければ、警告する
add_offense(node, message: MSG) do |corrector|
  correct_around_action(corrector, actions.first, action_name)
end
```

```
youtrust-webapp on ?master [$!] via 🐳 desktop-linux via ?v20.15.0 via 💎 v3.3.5
❯ docker compose exec app bundle exec rubocop app/controllers/api/v2/communities_controller.rb
Inspecting 1 file
W

Offenses:

app/controllers/api/v2/communities_controller.rb:1:1: W: [Correctable] Lint/WithReaderCop: around_action :with_reader must be defined with the :only option including specific actions.
class Api::V2::CommunitiesController < Api::ApplicationController ...
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

1 file inspected, 1 offense detected, 1 offense autocorrectable
```

Correctableになった

まとめ

静的解析で解決できるような特有のルールは
カスタムコップで解決しよう！