

# あなたの知らない **RuboCop**の設定

@vividmuimui

2019/04/03

社内LT

~~あなたの知らない~~  
~~RuboCop~~の設定  
僕の知らなかった  
RuboCopの設定

# 話すこと

便利copとかの話ではないです。  
継承(inherit)に関する3点の話です。

- 初級: URLでinherit
- 中級: inherit\_mode
- 上級: 相対パスはどこから？

初級: URLでinherit

# inherit(継承)

RuboCop では、共通設定ファイルを利用する方法がいくつかある。

- `inherit_from`
  - ローカルファイル
  - `.rubocop_todo.yml` が代表的
- `inherit_gem`
  - Gem
  - `onkcop`, `meowcop` が代表的

```
inherit_from: .rubocop_todo.yml
```

```
inherit_gem:  
  onkcop:  
    - "config/rubocop.yml"  
    - "config/rails.yml"
```

# `inherit_from` には**URL**もかける！

```
inherit_from:  
- http://www.example.com/rubocop.yml
```

会社共通の設定とかはGitHubにあげて

`https://raw.githubusercontent.com/xxx` を参照するのが手っ取り早そう？

Rubyのバージョンが最新に追従できてない、とかもあると思うので、現実的にはGemで提供するのが正しそう。

中級: **inherit\_mode**

# 前提1

子ファイル(例: `.rubocop.yml`)で親ファイル(例: `.rubocop_todo.yml`)の設定を継承するときに、  
子ファイルで親ファイルの設定を上書きできる

```
# .rubocop_todo.yml
Style/CollectionMethods:
  PreferredMethods:
    collect: map

# .rubocop.yml
inherit_from: '.rubocop_todo.yml'

Style/CollectionMethods:
  PreferredMethods:
    collect: collect # こっちの設定が有効
```




# 前提2

RuboCopの設定は、`Hash` と `Array` で設定される。

```
Style/CollectionMethods:
  # hashの例
  PreferredMethods:
    collect: map
  # arrayの例
  Exclude:
    - 'foo/bar.rb'
```

継承したときに、

- Hashは設定がMergeされる
- Arrayは設定がOverrideされる

 ArrayがOverrideなのは、そうじゃないと親の設定を子で無効化できないから。MergeだとYamlで空Arrayを書いても、親の設定が残ってしまう。Hashの場合は`~`(Yamlのnull)で、親の設定をキャンセルできる。

# inherit\_mode

inherit\_modeでArrayの継承時の挙動を変更できる。  
OverrideからMergeに変更できる

```
inherit_from:  
  - 'shared.yml'
```

```
inherit_mode:  
  merge:  
    - Exclude
```

```
Style/For:  
  inherit_mode:  
    override:  
      - Exclude
```

上級: 相対パスはどこから？

# パスを書ける部分

- `inherit_from`
  - 継承するファイル
- `Include/Exclude`
  - Copの対象とするファイル

```
inherit_from: .rubocop_todo.yml
```

```
Style/CollectionMethods:
```

```
  Exclude:
```

```
    - 'foo/bar.rb'
```

どちらも、パスは絶対パス・相対パスどちらも書ける。  
相対パスはどこからの相対なのか？

# inherit\_fromの相対パス

inherit\_from に書く相対パスは、inherit\_from が書いてあるファイルからの相対パス

```
# .rubocop.yml  
inherit_from: '../.rubocop_todo.yml'
```

こう書いた場合、

.rubocop.yml からみて ../ にある .rubocop\_todo.yml を読み込む。

# Include/Excludeの相対パス

```
Style/CollectionMethods:  
  Exclude:  
    - 'foo/bar.rb'
```

これが難しい

Include/Exclude の設定が書かれている  
ファイル名によって変わる！



# Include/Exclude の相対パス

- `.rubocop` で始まる設定ファイルの場合
  - その設定ファイルからの相対パス
  - `.rubocop.yml` や `.rubocop_todo.yml` など
- それ以外の設定ファイルの場合
  - `rubocop` コマンドを実行したディレクトリからの相対パス
  - `.my_rubocop_config.yml` や `rubocop.yml` (ドットなし) など

```
.  
├── config  
│   ├── .my_rubocop_config.yml  
│   ├── .rubocop_todo.yml  
│   └── .rubocop.yml  
└── src  
    └── sample.rb
```

```
# .rubocop.yml  
inherit_from:  
  - .my_rubocop_config.yml  
  - .rubocop_todo.yml
```

このような状況とします。

rubocopコマンドは、ルートディレクトリで以下のように実行するとします

```
$ rubocop -c config/.rubocop.yml
```

```
.
├── config
│   ├── .my_rubocop_config.yml
│   ├── .rubocop_todo.yml
│   └── .rubocop.yml
└── src
    └── sample.rb
```

```
# config/.rubocop.yml
# config/.rubocop_todo.yml
Style/CollectionMethods:
  Exclude:
    # .rubocopで始まるので、設定ファイルからの相対
    - '../src/sample.rb'

# config/.my_rubocop_config.yml
Style/CollectionMethods:
  Exclude:
    # .rubocopではないので、実行ディレクトリからの相対
    - 'src/sample.rb'
```

`.rubocop.yml` がリポジトリルートに置いてあるケースでは、ほぼ問題にならない。

ただ、`tools/rubocop/` などに設定ファイル(`.rubocop.yml` など)をおいてるときには注意しなければならない。

例えば、以下のようなケースの場合、`.rubocop` で始めないファイル名のほうが楽。

```
.
├── src
│   ├── server
│   │   └── test
│   │       └── foo.rb
└── tool
    ├── rubocop
    │   └── config
    │       └── rubocopの設定ファイル
```

rubocopの設定ファイルがルートになく、  
`src` ディレクトリでrubocopを動かすようなケース

実行コマンドが次のようになるケース

```
$ rubocop -c ../tool/rubocop/config/{rubocopの設定フ
```

コマンドの実行ディレクトリが設定ファイルと同じ場所の場合は話が別です。

( `tools/rubocop/config` でrubocopコマンドを動かすようなケース。そういうケースはあまりなさそうだが。。)

# `.rubocop` で始めないファイル名のほうが楽な理由

例えば、`test` ディレクトリはまるっと無視したいCopがあったとき、

```
# rubocopの設定ファイル
Foo/BarCop:
  Exclude:
    - '**/test/**/*.*rb'
```

というように書きたくなるが、

- `.rubocop` で始まるファイルの場合
  - その設定ファイルからの相対パスなので、`**/` が効かない。
- `.rubocop` で始まらないファイルの場合
  - `rubocop` コマンドの実行ディレクトリからの相対パスなので、上記の設定で問題ない。
  - 実行ディレクトリが、プロジェクトルートでも、`src`でも、`src/server`でも問題ない。

# まとめ



# まとめ

- 初級: URLでinherit
  - `inherit_from` にはURLが書ける
- 中級: `inherit_mode`
  - 子ファイルで設定を上書きするとき、HashはMerge, ArrayはOverride
  - `inherit_mode` でArrayの上書き設定をMergeに変更できる
- 上級: 相対パスはどこから？
  - `inherit_from` に書くパスは、`inherit_from` が書かれた設定ファイルからの相対
  - `Include/Exclude` に書くパスは
    - `.rubocop` で始まる場合は、設定ファイルからの相対
    - それ以外は、実行コマンドからの相対
    - 設定ファイルを `tools/` などに置くときは、`.rubocop` で始まらないファイル名のほうが楽