SimpleDelegator活用のご提案

平成Ruby会議 01

Daisuke Fujimura

2019-12-13

自己紹介

藤村大介

- https://twitter.com/ffu_
- https://github.com/fujimura
- スタートアップ数社 ⇨ マチマチCTO ⇨ フリーランス
- 現在は数社で技術顧問のお仕事
- RubyとHaskellが好き
- WEB+DB PRESS Vol.110 名前付け大全を書いた

今日のお話

SimpleDelegatorは便利なのにあまり使われていない印象があるので宣伝したい！

SimpleDelegatorとは

- オブジェクト指向でいうところの「委譲」ができるライブラリ
- Ruby標準ライブラリ ‘delegate’ に含まれる

基本的な使用例

```ruby
class User
  def born_on
    Date.new(1989, 9, 10)
  end
end

class UserDecorator < SimpleDelegator
  def birth_year
    born_on.year
  end
end

decorated_user = UserDecorator.new(User.new)
decorated_user.birth_year  #=> 1989
decorated_user.__getobj__  #=> #<User: ...>
```

出典: https://ruby-doc.org/stdlib-2.6.5/libdoc/delegate/rdoc/SimpleDelegator.html

何が便利なの

- 局所的な振る舞いを後から追加することができる
  - *部分的にしか使わない実装で元のクラスの実装を膨らませないで済む*
- デコレーターを簡単に書ける
- その他、小技がいくつか

具体例

例：APIのレスポンスとカラム名を揃えたい

- 既存のオブジェクトのインターフェイスが期待しているものと異なる
- APIのレスポンスがオブジェクトで、モデルとインターフェイスが違うケース

```
class Entry < SimpleDelegator
  def title
    subject
  end
end

entries = api.get('/entries/').map { |r| Entry.new(r) }
puts entries.first.title # => `subject`の値
```

- 簡単にデコレーターを書けました

例：ソート順を変えたい

- 特定の箇所で既存のオブジェクトのソート順を変えたい

```
class UserSortedByBirthday < SimpleDelegator
  def <=>(other)
    birthday > other.birthday
  end
end
users = User.limit(10).map {|u| UserSortedByBirthday.new(u) }
users.sort #=> 誕生日で並べ替えられる
```

- デコレーターはインターフェイスだけでなく挙動を拡張することもできます

例：アクセサを足したい

```ruby
class UserWithToken < SimpleDelegator
  attr_accessor :token
end

token = generate_token
users = User.limit(10).map {|user|
  u = UserWithTimestamp.new(user)
  u.token = token
  u
}

users.first.token #=> 生成したトークン
```

- 元のクラスにアクセサは定義したくない
  - *クラスを読み下していって「これのアクセサは一体...？」となりがち*
  - *後にそのアクセサが悪用されて副作用パズルが発生したりするけど、これなら防げる*
  - *局所化は最高*

例：現在のユーザーによってオブジェクトの値を変えたい

- オブジェクトの振る舞いを他のオブジェクトによって変えたい

```
class PersonalizedPost < SimpleDelegator
  def initialize(post, user)
    @user = user
    __setobj__(post)
  end

  def comments
    __getobj__.comments.filter {|c| !c.author.blocked_by?(@user) }
  end
end

posts = Post.first(10).map {|p| PersonalizedPost.new(p, current_user) }
posts.first.comments # ブロックされたユーザーのコメントは現れない
```

- `post` に状態をもたせる実装よりも影響範囲が少ない
- `Post#comments_for(user)` という実装も考えられるけど、パーソナライズする箇所が増えるといちいち渡すのが面倒
  - *局所化は最高*

例：クエリオブジェクト作るの面倒

```
class DailyNewComment < SimpleDelegator
  def initialize(user, start_at)
    comments = user.visible_comments.where(
      'created_at >= :from AND created_at < :to',
      from: start_at,
      to: 24.hours.since(start_at)
    )

    __setobj__(comments)
  end
end

comments = DailyNewComment.new(user, start_at)
```

- これは完全に小技
- `DailyNewCommentQuery.new(...).result` というようなクエリオブジェクトあるある実装より簡潔

まとめ

- SimpleDelegatorは便利
- 局所化進めていきましょう

補足

- 遅いってマジ？
- DelegateClassとの違いは？わからん！誰か教えてくれ！