Ruby master - Feature #14059

Refine multiple classes in one call

10/26/2017 05:30 PM - kddeisz (Kevin Deisz)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

Consider the following problem - I would like to be able to validate that a JSON string contains all things that can represent positive integers. In this case, it works well with refinements, because I can:

```
class PositiveJSONValidator < ActiveModel::EachValidator
  using Module.new {
    refine String do
      def positive_integer?
        match?(/[1-9](?:[0-9])*/)
      end
    end

    refine Integer do
      def positive_integer?
        self > 0
      end
    end

    refine Float do
      def positive_integer?
        self == self.to_i
      end
    end

    refine NilClass do
      def positive_integer?
        false
      end
    end

    refine Array do
      def positive_integer?
        false
      end
    end

    refine Hash do
      def positive_integer?
        false
      end
    end
  }

  def validate_each(record, attribute, value)
    return if valid_positive_json?(value)
    record.errors[attribute] << options[:message] || 'is invalid'
  end

  private

  def valid_positive_json?(value)
    JSON.parse(value).all? do |key, value|
      key.positive_integer? && value.positive_integer?
    end
  rescue JSON::ParserError
```

```
    false
  end
end
```

However, this is kind of annoying because I need to specify each class manually. What I'd love to be able to do is group classes by passing all of the ones that need to be refined in the same way, as in:

```
refine NilClass, Array, Hash do
  def positive_integer?
    false
  end
end
```

Is this something that people would consider? It seems like a good use case for refinements because I just want to send a message to an object, so basically I want a default case. I could just refine Object but that feels wrong, I'd rather get an undefined method exception so that I know something unexpected occurred. If accepted I'd be happy to submit a patch.

History

#1 - 10/26/2017 08:51 PM - rrroybbbean (RRRoy BBBean)

On Thu, 2017-10-26 at 17:30 +0000, kevin.deisz@gmail.com wrote:

> ```
> refine NilClass, Array, Hash do
>   def positive_integer?
>     false
>   end
> end
> ```
>
> Is this something that people would consider? It seems like a good use case for refinements because I just want to send a message to an object, so basically I want a default case. I could just refine Object but that feels wrong, I'd rather get an undefined method exception so that I know something unexpected occurred. If accepted I'd be happy to submit a patch.

Yes. This is awesome and excellent. For curiosity, why not enclose the classes in []?

#2 - 10/26/2017 11:13 PM - duerst (Martin Dürst)

I'm really not sure about this. If we need this for refinements, why don't we have for classes in the first place? E.g.:

```
class Foo, Bar, Baz
  def positive_integer?
    false
  end
end
```

It seems strange to introduce syntactic sugar just for refinements.

#3 - 10/26/2017 11:27 PM - shevegen (Robert A. Heiler)

Perhaps it could be an Array input for Module#refinements?

There he could perhaps specify the classes that are to be modified. So the above example would then be:

```
refine [NilClass, Array, Hash] do
  def positive_integer?
    false
  end
end
```

instead.

http://ruby-doc.org/core-2.4.2/Module.html#method-i-refine

But I know too little about refinements and also too little about what Kevin wants to achieve, aside from him using less syntax for batch-modification, which I think is understandable. :)

#4 - 10/27/2017 01:26 AM - jeremyevans0 (Jeremy Evans)

I think the natural way to handle this is a loop:

```
[NilClass, Array, Hash].each do |c|
  refine c do
    def positive_integer?
      false
    end
  end
end
```

Because of that I don't think we need new syntax to support this.

#5 - 10/27/2017 01:37 AM - nobu (Nobuyoshi Nakada)

- Description updated

And also a Module.

```
using Module.new {
  m = Module.new {
    def positive_integer?
      false
    end
  }
  [NilClass, Array, Hash].each {|k| refine(k) {include m}}
}
```

#6 - 04/11/2018 02:46 PM - kddeisz (Kevin Deisz)

I agree with the points above, so this issue can be closed (or someone can teach me how to close these, I'm not entirely sure how).