Ruby master - Feature #5010

Add Slop(-like) in stdlib and deprecate current OptionParser API

07/10/2011 08:01 AM - rosenfeld (Rodrigo Rosenfeld Rosas)

Status:	Closed
Priority:	Normal
Assignee:	matz (Yukihiro Matsumoto)
Target version:	2.6

Description

I always found the OptionParser API not as well designed as it could be.

I've just found this gem:

http://lee.jarvis.co/slop/

Much better API and I think we should integrate it to Ruby 2.0.

Take a look at the minimal example shown in OptionParser :

```
require 'optparse'

options = {}
OptionParser.new do |opts|
opts.banner = "Usage: example.rb [options]"

opts.on("-v", "--[no-]verbose", "Run verbosely") do |v|
  options[:verbose] = v
end

end.parse!

p options
p ARGV
```

This is the equivalent in Slop:

```
require 'slop'

opts = Slop.parse do
banner "Usage: example.rb [options]"
on :v, :verbose, "Run verbosely", :default => true
end

p opts.to_hash
```

History

#1 - 07/10/2011 08:23 AM - nobu (Nobuyoshi Nakada)

Hi,

At Sun, 10 Jul 2011 08:01:58 +0900,
Rodrigo Rosenfeld Rosas wrote in [ruby-core:37936]:

> This is the equivalent in Slop:
>
> ```
> require 'slop'
>
> opts = Slop.parse do
> banner "Usage: example.rb [options]"
> on :v, :verbose, "Run verbosely", :default => true
> end
>
> p opts.to_hash
> ```

Hidden instance_eval is cause of confusion. It's a way
OptionParser has thrown away.

--
Nobu Nakada

#2 - 07/10/2011 08:23 AM - rosenfeld (Rodrigo Rosenfeld Rosas)

> Hidden instance_eval is cause of confusion. It's a way
> OptionParser has thrown away.

Sorry, Nobu, I didn't get it. Could you explain it better?

#3 - 07/10/2011 10:23 AM - drbrain (Eric Hodel)

On Jul 9, 2011, at 4:19 PM, Rodrigo Rosenfeld Rosas wrote:

> Em 09-07-2011 20:13, Nobuyoshi Nakada escreveu:
>
> > At Sun, 10 Jul 2011 08:01:58 +0900,
> > Rodrigo Rosenfeld Rosas wrote in [ruby-core:37936]:
> >
> > > This is the equivalent in Slop:
> > >
> > > require 'slop'
> > >
> > > opts

#4 - 07/10/2011 11:08 AM - trans (Thomas Sawyer)

Would much rather see CLAP(-like) in standard library.

It's so simple. Maybe integrate into Shellwords module.

#5 - 07/10/2011 01:23 PM - neleai (Ondrej Bilka)

On Sun, Jul 10, 2011 at 10:16:25AM +0900, Eric Hodel wrote:

> On Jul 9, 2011, at 4:19 PM, Rodrigo Rosenfeld Rosas wrote:
>
> > Em 09-07-2011 20:13, Nobuyoshi Nakada escreveu:
> >
> > Sorry, Nobu, I didn't get it. Could you explain it better?
>
> I think Nobu means that formerly OptionParser used instance_eval like slop does inside the parse method.
>
> This was changed due to confusion of scoping and methods available inside and outside the instance_eval.
>
> I'd prefer not to have hidden instance_eval for option parsing. It's too often that I like to refer to items in a scope the execution environment doesn't have.
> This reminds me that I instead instance_eval rely on method injection.
> What I currently do is temporary add singleton method_missing to
> delegate methods. It has relatively sane scoping.
> I am wondering if there is way that is not as ugly as my current
> implementation.

```
def redef(&b)
b.binding.eval "
class <<self
if self.instance_methods.include? \"method_missing\"
@r=true
alias_method :mm2,:method_missing
end
def method_missing(n,*a)
Del.send(n,*a) if Del.respond_to? n
```

```
mm2(n,*a) if self.respond_to? :mm2
end
end"
b.call
b.binding.eval "
class <<self
if @r
alias_method :method_missing,:mm2
else
remove_method :method_missing
end
end
"
end
```

--

Your Pentium has a heating problem - try cooling it with ice cold water.(Do not turn of your computer, you do not want to cool down the Pentium Chip while he isn't working, do you?)

#6 - 07/10/2011 11:10 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

Sorry, I still didn't get it. I understand that you don't want any DSL when you talk about instance_eval, right? Any reason why you find it confusing, since this is a common pattern when writing DSL's in Ruby? I mean, is there some real example showing how this could be confusing?

Anyway, if that's the concern, Slop also support this alternative:

```
opts = Slop.parse do |s|
s.banner "Usage: example.rb [options]"
s.on :v, :verbose, "Run verbosely", :default => true
end
```

I don't really care about using instance_eval or not, but I still find that the OptionParser API could be better written.

#7 - 10/18/2011 09:16 AM - naruse (Yui NARUSE)

- Project changed from Ruby master to CommonRuby

- Target version deleted (3.0)

#8 - 10/23/2011 05:21 PM - naruse (Yui NARUSE)

- Project changed from CommonRuby to Ruby master

#9 - 11/08/2011 11:33 AM - injekt (Lee Jarvis)

Hi,

I'm the author behind Slop. Although I never considered Slop to ever hit Ruby trunk I of course think it's a great idea. That said, there's already two option parsing libraries in stdlib and I think adding another is just bloat.

For those of you worried any instance_eval, it's completely optional. Slop also supports the following:

- Parsing an optspec: https://github.com/injekt/slop/wiki/Optspec
- Auto creating options: https://github.com/injekt/slop/wiki/Auto-Create
- Commands (nested Slop instances): https://github.com/injekt/slop/wiki/Commands

And multiple methods of creating options: https://github.com/injekt/slop/wiki/Creating-Options

Slop is also very well documented and extremely well tested, and fits within a concise 500 or less LOC: https://github.com/injekt/slop

Again, I'm not so sure about integrating Slop into stdlib (I'd like my bug fixes and features changes instantly available to those who use my gem), but I wanted to clear up any concerns anyway.

#10 - 03/18/2012 06:46 PM - shyouhei (Shyouhei Urabe)

- Status changed from Open to Assigned

#11 - 11/20/2012 09:25 PM - mame (Yusuke Endoh)

- Target version set to 2.6

#12 - 11/20/2012 09:47 PM - trans (Thomas Sawyer)

I actually think it would be better to remove the option parser libraries from Ruby's stdlib alltogther. There are a number of really good option parser gems out there, but they get little use b/c developers tend to "play it safe" and use the built-in library even if it is less optimal then a 3rd party gem.

This in turns hurts the option parser "market" b/c developers aren't putting the available libraries to the test, nor submitting bug reports or patches to improve them. Remove the standard libs and we'd see this area of API development flourish.

#13 - 11/20/2012 11:04 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

Thomas, I like the idea but there is a shortcoming to this approach when people are using Ruby for performing shell scripting, like sysadmin scripts. It is not fair to force them to install any gem for some common task like this...

#14 - 11/21/2012 12:16 AM - trans (Thomas Sawyer)

=begin
rosenfeld (Rodrigo Rosenfeld Rosas) It's not? Maybe a little. But one can always parse ARGV by hand for simple shell scripts. It's not that hard. In fact, a simple helper makes it pretty easy.

```
def ARGV.option(opt)
  if i = ARGV.index(opt)
    ARGV.index(i+1)
  end
end
```

So I don't think an option parser library is needed in standard libraries. If there ((*has*)) to be something more, then only a very simple library like CLAP (https://github.com/soveran/clap/blob/master/lib/clap.rb) extending ARGV itself, makes the most sense to me.
=end

#15 - 11/21/2012 04:09 AM - drbrain (Eric Hodel)

RDoc and RubyGems both depend on OptionParser so it cannot be removed.

#16 - 11/21/2012 05:59 AM - trans (Thomas Sawyer)

Not so sure RDoc should be a standard library either. But in any case, "cannot" is a rather strong term. It would take a little work, but they could be adapted. Lord knows RubyGems' command code is almost a framework in itself anyway.

#17 - 11/21/2012 09:53 AM - duerst (Martin Dürst)

I agree with Rodrigo. If I have to use an option parser, I don't want to waste time shopping around.

Also, if no option parser outside optparse gets significant traction outside the Ruby standard library, this means that none of them is significantly better than the current one for a significant percentage of Ruby users.

I think the best thing is for the people who really care to get together and merge their work, and then come here with a replacement proposal.

Regards, Martin.

#18 - 11/21/2012 10:00 AM - duerst (Martin Dürst)

trans (Thomas Sawyer) wrote:

> But in any case, "cannot" is a rather strong term. It would take a little work, but they could be adapted.

Are you ready to provide a patch?

#19 - 11/21/2012 10:57 AM - shyouhei (Shyouhei Urabe)

"I don't like this shit let's just remove" doesn't sound productive to me. Is it hard to make it better instead?

#20 - 11/22/2012 12:48 AM - rosenfeld (Rodrigo Rosenfeld Rosas)

Shyouhei, what if we create a separate mailing list to discuss and propose a new gem to replace optparser? The idea would be to invite designers and users of the current optparser alternatives (slop, clap, thor, etc) and try to come up with some gem that would satisfy most of us and put the new API in a new gem for consideration by ruby-core members.

What restrictions should such a gem have to be part of stdlib? Should it remain backward compatible or could we just forget about the current optparser API? Maybe we can't get much traction from others if the former is required. I would also prefer not having to design a backward compatible API.

If this idea is accepted and no one steps up to create the mailing list I could create a Google Group and invite developers to discuss a new API extracting the good parts of the alternative solutions out there and start some discussion about the pros and drawbacks of each solution and try to figure out some API that would satisfy most of us.

What do you think, shyouhei?

#21 - 11/22/2012 01:23 AM - zzak (Zachary Scott)

optparse is used by many programs, including other stdlib gems.

In my opinion, it would be best to keep backwards compatibility.

Why create a new gem, instead of improving the current standard library?

#22 - 11/22/2012 01:53 AM - judofyr (Magnus Holm)

On Wed, Nov 21, 2012 at 5:16 PM, Zachary Scott zachary@zacharyscott.net wrote:

> optparse is used by many programs, including other stdlib gems.
>
> In my opinion, it would be best to keep backwards compatibility.
>
> Why create a new gem, instead of improving the current standard library?

I agree. Can we identify where optparse isn't optimal and tweak it?

We already have two argument parsing library in stdlib (getoptlong and optparse).

#23 - 11/22/2012 02:52 AM - trans (Thomas Sawyer)

duerst (Martin Dürst)

> I agree with Rodrigo. If I have to use an option parser, I don't want to waste time shopping around.
>
> Also, if no option parser outside optparse gets significant traction outside the Ruby standard library, this means that none of them is significantly better than the current one for a significant percentage of Ruby users.

You just contradicted yourself.

And your first argument is exactly the problem with Ruby rubber stamping an official option parser.

> Are you ready to provide a patch?

Yes, I could do that.

#24 - 11/22/2012 03:11 AM - trans (Thomas Sawyer)

shyouhei (Shyouhei Urabe)

> I don't like this shit let's just remove" doesn't sound productive to me.

No one said that.

> Is it hard to make it better instead?

Hmmm... seems to me I tried that once. It was rejected. Unfortunately it was so long ago now I can't find anything about it.

But in any case I think it's missing the point. For who's to say what the best option parser is? There isn't just one perfect parser out there that if we all just work together we can pull down from the world of perfect forms. There are all sorts of approaches: We have very simple mechanisms like CLAP, traditional systems like getoptlong and optparse, "on steroids" variations of those like slop and trollop, DSLs like thor and even command to object mappings like executable. All of these have various merits, and might be more suitable to one application or one developer's way thinking. That's why I think it is better to encourage diversity (as matz said in his last keynote), rather than try to lock Ruby further into a "one right way".

#25 - 11/22/2012 09:53 AM - jonforums (Jon Forums)

> ...rather than try to lock Ruby further into a "one right way".

Use an existing alternative or roll your own if you don't like ruby's (needed) baseline impl. The situation is not even remotely close to lock-in.

This discussion smells like what may have occurred in Python. They weren't able to find a backwards compatible update to optparse and ended up with argparse-using-deprecated-but-still-supported-optparse. I believe Python now has *3* option parsers in stdlib.

http://www.doughellmann.com/PyMOTW/argparse/

Perhaps it made sense for Python given it's realities (i.e. easy_install/distribute/pip vs. rubygems), but if this request is going to compete for limited ruby-core committer time, I agree with Magnus.

#26 - 11/22/2012 10:25 AM - shyouhei (Shyouhei Urabe)

What made this thread long was the request to deprecate optparse, I think. No one is arguing about its lacking documents, being non-intuitive, being not cool modern sexy urbane taste, and so on. And no one is arguing that other libs (like Slop) have metits.

That said, to deprecate something is still a hard job. You need a good reason to remove it, not just because it is broken (if it is why not just fix). And I think I have not yet heard about that. Why you think we should abandon optparse? You can't have 128 different option parser standard libs at once, doesn't mean you should have zero standard lib. Right?

#27 - 11/22/2012 01:37 PM - duerst (Martin Dürst)

trans (Thomas Sawyer) wrote:

> duerst (Martin Dürst)
>
> > I agree with Rodrigo. If I have to use an option parser, I don't want to waste time shopping around.
> >
> > Also, if no option parser outside optparse gets significant traction outside the Ruby standard library, this means that none of them is significantly better than the current one for a significant percentage of Ruby users.
>
> You just contradicted yourself.

Can you explain? The fact that I'm not interested in wasting time shopping around doesn't mean that others won't do that if they meet problems with the current ones in the standard library or are otherwise interested in something better.

> And your first argument is exactly the problem with Ruby rubber stamping an official option parser.

Anybody can use any option parser they want. Ruby already has two. Ruby isn't rubber stamping, but just providing something for those who don't want to shop around and deal with the hassles of installation. Even if the two currently in Ruby are not perfect, they may be good enough. And they were available at a time when the more modern ones were not yet available. Anyway, the fact that there are so many out there seems to indicate to me that it's too early for Ruby to pick up a third one, or replace one of the existing ones in the standard library.

#28 - 06/26/2014 01:46 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

Ok, what about introducing new methods to make it easier to work with OptionParser?

Like:

```
opts = OptionParser.parse do |p|
  p.banner "Usage: example.rb [options]"
  p.on :v, :verbose, "Run verbosely", :default => true
end

p opts.to_hash
```

If you prefer I can create a new ticket to discuss the improvements and close this one.

#29 - 06/30/2014 02:57 AM - shyouhei (Shyouhei Urabe)

I'm neutral about that proposed #to_hash (so far, bit vague), but is definitely far better than removing optparse. Can you let your proposal be a new ticket? This thread is already too long to read through.

Rodrigo Rosenfeld Rosas wrote:

> Ok, what about introducing new methods to make it easier to work with OptionParser?
>
> Like:
>
> ```
> opts = OptionParser.parse do |p|
> ```

```
  p.banner "Usage: example.rb [options]"
  p.on :v, :verbose, "Run verbosely", :default => true
end

p opts.to_hash
```

If you prefer I can create a new ticket to discuss the improvements and close this one.

#30 - 06/30/2014 10:10 AM - rosenfeld (Rodrigo Rosenfeld Rosas)

Sure, feel free to close this one.

#31 - 06/30/2014 11:33 AM - shyouhei (Shyouhei Urabe)

- Status changed from Assigned to Closed