Ruby master - Feature #13847

Gem activated problem for default gems

08/29/2017 08:53 AM - hsbt (Hiroshi SHIBATA)

Status:	Assigned
Priority:	Normal
Assignee:	
Target version:	3.0

Description

If you try to use some default gems with a fixed version using Bundler, there are cases where the current RubyGems/Bundler/Ruby specification can not be used with the version specified by the user.

For example

```
$ ruby -v
ruby 2.4.1p111 (2017-03-22 revision 58053) [x86_64-darwin17]
$ gem list | grep openssl
openssl (2.0.5, 2.0.4, default: 2.0.3)
```

In the environment such as require 'openssl', the version that is activated when openssl is searched with openssl is the version found first, ie 2.0.5.

```
$ ruby -ropenssl -e 'p OpenSSL::VERSION'
"2.0.5"
```

At this time, for example, suppose the user really wants to use openssl 2.0.4 and wrote the following Gemfile.

```
> cat Gemfile
# frozen_string_literal: true
source "https://rubygems.org"

gem 'openssl', '2.0.4'
```

Unfortunately, since rubygems has required openssl before the bundler runs it will result in an activated error like this:

```
> bundle exec ruby -ropenssl -e 'p OpenSSL::VERSION'
/path/to/2.4.1/lib/ruby/gems/2.4.0/gems/bundler-1.15.4/lib/bundler/runtime.rb:317:in `check_for_ac
tivated_spec!': You have already activated openssl 2.0.5, but your Gemfile requires openssl 2.0.4.
 Prepending `bundle exec` to your command may solve this. (Gem::LoadError)
```

This problem can be avoided by bundling it as a vendoring library under bundler's repository if it is a default gem implemented with pure ruby.

Https://github.com/bundler/bundler/blob/master/lib/bundler/vendor/fileutils/lib/fileutils.rb

In the case of bundler, by separating the namespace as Bundler::FileUtils, even the version specified by the user is made available without conflict at the time of activate. However, this method can not be used with C extension library.

Since we want to use json/psych from the bundler team with rubygems/bundler to serialize data, we need about whether we can implement a way to avoid some kind of C extension on Ruby itself.

I discussed with indirect (André Arko) who is maintainer of RubyGems/Bundler. We can resolve this problem like following feature of ruby.

```
require_for_bundler 'json', '2.0.2'
```

When we declared above require_for_bundler, We put a json-2.0.2 to placed in a namespace like Bundler::JSON. There were similar issues in the past as well.

https://bugs.ruby-lang.org/issues/10320

I think that the way of writing require 'json', version: '2.0.2', into: :Bundler which extended the method like this issue seems like that. Also, in this use case, it seems to be enough to use require 'json', version: :default, into: :Bundler which forces the use of default gem.

Matz, How do you think about this feature?

Related issues:

Related to Ruby master - Feature #10320: require into module	**Open**

History

#1 - 08/29/2017 08:54 AM - hsbt (Hiroshi SHIBATA)

- Target version set to 2.5

- Subject changed from gem activated problem for default gems to Gem activated problem for default gems

#2 - 08/29/2017 08:57 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #10320: require into module added

#3 - 08/29/2017 09:27 AM - rhenium (Kazuki Yamaguchi)

hsbt (Hiroshi SHIBATA) wrote:

> Unfortunately, since rubygems has required openssl before the bundler runs it will result in an activated error like this:
>
> ```
> > bundle exec ruby -ropenssl -e 'p OpenSSL::VERSION'
> /path/to/2.4.1/lib/ruby/gems/2.4.0/gems/bundler-1.15.4/lib/bundler/runtime.rb:317:in `check_for_activated_
> spec!': You have already activated openssl 2.0.5, but your Gemfile requires openssl 2.0.4. Prepending `bun
> dle exec` to your command may solve this. (Gem::LoadError)
> ```

Off-topic, but in this specific case, the root of the problem is that bundler/setup is require-ed after openssl (or whatever gem that is installed globally, and Gemfile has a different version in it), because options in the RUBYOPT environment variable are processed after the ones specified in the command line. ruby -rbundler/setup -ropenssl -e1 or bundle exec ruby -e'require "openssl"' should work.

#4 - 08/29/2017 11:33 AM - shevegen (Robert A. Heiler)

> Also, in this use case, it seems to be enough to use [..]
>
> require 'json', version: :default, into: :Bundler

Matz will decide but I wanted to comment on what Hiroshi Shibata wrote.

Kernel require presently accepts no additional arguments:

https://ruby-doc.org/core-2.4.1/Kernel.html#method-i-require

So I guess the proposal also means to extend Kernel#require if
I understood it correctly.

I may be wrong but I think that matz may wish to keep require
simple.

I myself have, since some time, wanted to have some more useful
or powerful ways for ruby to handle add-ons in general stored
in different files. Some time ago I noticed that load() has
a second argument (boolean - true or false), and if false then
the loaded file will not modify the namespace. So it will be
an anonymous module. That was quite interesting to me. There
are also some tricks to be able to access the module still
such as on:

http://iceruby.blogspot.co.at/2014/07/or-how-to-get-anonymous-module-from.html

(In case the above link no longer works at some point in
time, and you don't want to search via the wayback www archive,
I'll very briefly mention what they do:

They use catch/throw trick such as:

```
throw :wrapper, Module.nesting.last
```

and catch the error that way

```
catch :wrapper do
  load File.expand_path('b.rb'), true
```

Which was an interesting idea.)

Anyway, to come back to Hiroshi Shibata, I actually agree with the
proposal, although I come from another direction (in general to have
more power and control about how to load up add-ons and external
code, so this is why I am in support of it).

In Gemfiles, please correct me if I am wrong but there are ways to
speciically load certain versions such as via:

```
gem 'openssl', '2.0.4'
```

So in my opinion, I believe that it may make sense for require
to accept a Hash too. This would be consistent then with gem
and bundler, and being able to version. I should, however had,
also mention that I presently do not use bundler at all and I
would actually also like to be able to NOT have to use it, so
while I am in support of the proposal, the only concern I have
if people would be REQUIRED to use bundler; it's perfectly fine
if this would not be the case, and one can be in support of
the proposal even without bundler too of course. I am only
concerned when it would force to change my workflow, not if
things can stay the same - and I think with the proposal
things would stay about the same right? Since it would only
give require() more options, which I think is good. Anyway,
I guess this may require some discussion nonetheless. :))

This Hash could allow for additional parameters that would allow
us rubyists for more finer-grained control.

Hiroshi gave the example:

```
require 'json', version: '2.0.2', into: :Bundler
```

Perhaps we could also re-define importable namespaces, a bit similar
as to load() but actually attaching onto the main namespace of the
current ruby gem.

For example, if I have a project called foobar, with module Foobar,
and I am including an external project called bla, that has a toplevel
module namespace called Bla, then I could perhaps say that the namespace
Bla never "exists" but instead becomes attached towards module Foobar
at once.

Syntax proposals:

require 'bla', version: '2.0.2', attach_as_namespace :Foobar

or something like this. However, please do not get too distracted
by my suggestion above; it's just what I personally would like to
be able to have. It is not so important for me whether this is
part of require, or load, or some other way, I am more interested
in the functionality, since I needed it a few times already.

I do not want to make Hiroshi's proposal too complicated, so I
want to end that I agree with his proposal.

+1

#5 - 09/01/2017 03:18 PM - vo.x (Vit Ondruch)

Shouldn't the focus be on why the OpenSSL is loaded at the first place?

#6 - 09/06/2017 05:34 AM - hsbt (Hiroshi SHIBATA)

We discussed this issue on last Developer Meeting.

Matz has some concerns about ruby internal.

1. Conflicts versions of a shared library like libyaml-*, *libssl-*, libffi, etc...
2. Order of LOADED_FEATURE

So, This feature is difficult to implement on current ruby specification now.

#7 - 12/12/2017 08:09 AM - hsbt (Hiroshi SHIBATA)

- Target version changed from 2.5 to 3.0

- Status changed from Open to Assigned