Ruby - Feature #19089

Load bundler/setup in gem_prelude.rb when "bundle exec" is used

10/28/2022 07:03 AM - mame (Yusuke Endoh)

Status:	Closed
Priority:	Normal
Assignee:	
Target version:	

Description

Problem

Currently, we cannot specify the version of did_you_mean by using Gemfile.

For example, Ruby master bundles with did_you_mean 1.6.1 by default:

```
$ ruby -e 'p DidYouMean::VERSION'
"1.6.1"
```

Consider that we want to use did_you_mean 1.5.0 with this version of ruby:

```
$ cat Gemfile
source "https://rubygems.org"
gem "did_you_mean", "= 1.5.0"
```

But the attempt fails:

```
$ bundle exec ruby -e 'p DidYouMean::VERSION'
/home/mame/work/ruby/local/lib/ruby/gems/3.2.0+3/gems/bundler-2.3.19/lib/bundler/runtime.rb:308:in
 `check_for_activated_spec!': You have already activated did_you_mean 1.6.1, but your Gemfile requ
ires did_you_mean 1.5.0. Since did_you_mean is a default gem, you can either remove your dependenc
y on it or try updating to a newer version of bundler that supports did_you_mean as a default gem.
 (Gem::LoadError)
...
```

This issue is not only with did_you_mean, but also with error_highlight and syntax_suggest which are automatically loaded at the interpreter startup.

This example is specifying an older version of did_you_mean, but typically you will want to specify a newer version. Actually, in https://github.com/rails/rails/pull/45818, I wanted to use error_highlight 0.4.0 with Ruby 3.1. (Note that Ruby 3.1 bundles with error_highlight 0.3.0.)

The cause of this problem is that bundle exec makes the interpreter load bundler/setup using RUBYOPT=-rbundler/setup, but this load is too late.

Proposed solution

Let's load bundler/setup in gem_prelude.rb when bundle exec is used.

```
diff --git a/gem_prelude.rb b/gem_prelude.rb
index f382021ca3..825508f571 100644
--- a/gem_prelude.rb
+++ b/gem_prelude.rb
@@ -6,6 +6,10 @@
   warn "`RubyGems' were not loaded."
 end if defined?(Gem)

+if ENV["BUNDLE_BIN_PATH"]
+  require File.join(File.dirname(ENV["BUNDLE_BIN_PATH"], 2), "lib/bundler/setup")
+end
+
 begin
   require 'error_highlight'
```

```
 rescue LoadError
```

The key is that bundler/setup is loaded immediately after rubygems is loaded, and before error_highlight and did_you_mean are loaded. This patch allows to specify the version of did_you_mean gem by Gemfile:

```
$ cat Gemfile
source "https://rubygems.org"
gem "did_you_mean", "= 1.5.0"

$ bundle exec ruby -e 'p DidYouMean::VERSION'
"1.5.0"
```

@deivid What do you think?

History

#1 - 10/28/2022 08:07 AM - mame (Yusuke Endoh)

- Description updated

#2 - 10/28/2022 08:40 AM - vo.x (Vit Ondruch)

Let me provide you different perspective. did_you_mean is development dependency, which have no benefit for runtime, where there should be error free code. From that POV, it would be much better if it was possible to disable did_you_mean autoloading. Tighter integration of Bundler into Ruby is mistake IMHO.

#3 - 10/28/2022 09:46 AM - deivid (David Rodríguez)

@mame (Yusuke Endoh) That's indeed a long standing issue for us and I'm very happy if it can get fixed! I'm not sure the proposed approach will work when Bundler is installed as a default gem though? We could set a different variable with the proper path to bundler's lib for this feature, or maybe we could leverage RubyVM.resolve_feature_path?

#4 - 10/28/2022 11:30 AM - mame (Yusuke Endoh)

@deivid Thank you for your reply!

deivid (David Rodríguez) wrote in #note-3:

> I'm not sure the proposed approach will work when Bundler is installed as a default gem though?

Oops, you are right. My current patch does not work with bunlder as a default gem. It would be fixable by parsing RUBYOPT=-r.../bundler/setup ... to get the path to bundler/setup. It is very ugly, though. Alternatively, if we don't care about the case where multiple versions of bundler are installed, we may just require "bundler/setup" if ENV["BUNDLE_BIN_PATH"].

However,

> We could set a different variable with the proper path to bundler's lib for this feature

I think it is the best. For example, how about this patch?

```
diff --git a/gem_prelude.rb b/gem_prelude.rb
index f382021ca3..8dd1e53c16 100644
--- a/gem_prelude.rb
+++ b/gem_prelude.rb
@@ -6,6 +6,8 @@
   warn "`RubyGems' were not loaded."
 end if defined?(Gem)

+require ENV["BUNDLE_SETUP"] if ENV["BUNDLE_SETUP"]
+
 begin
   require 'error_highlight'
 rescue LoadError
```

I am fine with a different environment variable name from BUNDLE_SETUP.

#5 - 10/28/2022 11:40 AM - mame (Yusuke Endoh)

@nobu (Nobuyoshi Nakada) pointed out that we can load bundler/setup in lib/rubygems.rb instead of gem_prelude.rb.

```
diff --git a/lib/rubygems.rb b/lib/rubygems.rb
index 915a899f38..b61d21050d 100644
```

```
--- a/lib/rubygems.rb
+++ b/lib/rubygems.rb
@@ -1348,3 +1348,5 @@ def default_gem_load_paths
 require_relative "rubygems/core_ext/kernel_gem"
 require_relative "rubygems/core_ext/kernel_require"
 require_relative "rubygems/core_ext/kernel_warn"
+
+require $1 if ENV["RUBYOPT"] && ENV["RUBYOPT"] =~ /-r(.*?\/bundler\/setup\b)/
```

(This patch is just a proof-of-concept. Parsing RUBYOPT should be written more carefully!)

Either would be fine to me.

#6 - 10/28/2022 02:50 PM - jeremyevans0 (Jeremy Evans)

vo.x (Vit Ondruch) wrote in #note-2:

> From that POV, it would be much better if it was possible to disable did_you_mean autoloading.

That's already possible via --disable-did_you_mean

> Tighter integration of Bundler into Ruby is mistake IMHO.

I agree with this 100%.

Is this fixable without any changes to Ruby if bundler sets RUBYLIB to include the path to the specific version of bundler in use, in addition to setting RUBYOPT?

#7 - 10/28/2022 04:36 PM - mame (Yusuke Endoh)

Thank you all.

All I ask is that did_you_mean, etc. be loaded by default and that their versions be controllable by Gemfile. I don't have a strong preference on how to fix it.

Now I agree that modifying Ruby is overkill for this issue. As @nobu (Nobuyoshi Nakada) proposed, it is fixable only in the side of rubygems/bundler. I have created a PR: https://github.com/rubygems/rubygems/pull/6025. I think this is simple and enough.

Note that this change does not force Ruby users to use bundler. Ruby should work just as before if you don't use bundle exec.

#8 - 10/28/2022 05:20 PM - Eregon (Benoit Daloze)

I believe this feature hurts startup time optimizations, so from that point of view it's not great.
For instance TruffleRuby loads did_you_mean ahead of time, during context pre-initialization.
Similarly, CRuby could for instance store the bytecode of did_you_mean/error_highlight in the CRuby binary or some more other efficient access than parsing Ruby code.
But those optimizations cannot be applied if another version of did_you_mean/error_highlight can be loaded.

I think it would make more sense to only allow loading a different did_you_mean / error_highlight version when
--disable-did-you-mean/--disable-error-highlight/--disable-gem is passed.
Then it's clear it's loaded later.

#9 - 10/28/2022 06:12 PM - vo.x (Vit Ondruch)

jeremyevans0 (Jeremy Evans) wrote in #note-6:

> vo.x (Vit Ondruch) wrote in #note-2:
>
> > From that POV, it would be much better if it was possible to disable did_you_mean autoloading.
>
> That's already possible via --disable-did_you_mean

Ah, right. I have not realized this. Thx for pointing this out.

Eregon (Benoit Daloze) wrote in #note-8:

I completely agree with your assertions. While I appreciate that Ruby wants to improve the development experience, I think that all these gems should not be used for runtime and from that POV, the defaults are wrong. To the extreme, enabling did_you_mean for production (which is the majority of Ruby runs, right?) is like executing the application test suite before each run.

#10 - 10/29/2022 01:55 AM - mame (Yusuke Endoh)

- Status changed from Open to Closed

According to @k0kubun (Takashi Kokubun), some people actually use the hack to work around this problem by using --disable-gems or --disable-error_highlight.

However, this is arguably far from ideal. The option --disable-gems is not recommended for use (https://bugs.ruby-lang.org/issues/17684). Also --disable-error_highlight is for those who want to disable error_highlight, not for such a hack to manually load error_highlight later. This is why I created this ticket.

This ticket is no longer necessary if the issue is fixed only in rubygems/bundler, so closing it once. I would like to hear from @deivid's decision. If something is discovered that is difficult to fix in rubygems side, I may reopen this ticket.

#11 - 11/02/2022 01:11 PM - deivid (David Rodríguez)

I agree with @mame (Yusuke Endoh) that using --disable-gems or --disable-did_you_mean to workaround this is far from ideal, because these flags are intended to disable the functionality, not to use a different version of it. So I think this should be fixed and work just like any other gem works.

I also agree that RubyGems & Bundler seems like a better place to fix this, although I'm not yet sure of what the best fix is. I will research further.

#12 - 11/21/2022 03:32 AM - naruse (Yui NARUSE)

People shouldn't use and recommend others to use --disable-gem as a workaround for a bug. It's only for debugging.
If people use --disable-gem for such use case widely, we need to remove the option.

#13 - 11/21/2022 10:54 AM - Eregon (Benoit Daloze)

naruse (Yui NARUSE) wrote in #note-12:

> People shouldn't use and recommend others to use --disable-gem as a workaround for a bug. It's only for debugging.
> If people use --disable-gem for such use case widely, we need to remove the option.

There is no need for --disable-gem here.
--disable-did-you-mean / --disable-error-highlight are different and are supported AFAIK (e.g., some found that some error_highlight version was significantly slowing down Ruby in production).

As I said above, supporting this means giving up on some significant startup optimizations.
IMHO it's not worth it by default, probably very few people need to use a newer did_you_mean/error_highlight via Gemfile.
Also error_highlight would IMHO make more sense to be implemented in core (so e.g. it can always reliably find the source and not hope it can reread from disk) and then it's obvious it can't be updated.