Ruby master - Feature #15903

Move RubyVM.resolve_feature_path to Kernel.resolve_feature_path

06/05/2019 08:13 PM - Eregon (Benoit Daloze)

Status: Closed
Priority: Normal
Assignee: matz (Yukihiro Matsumoto)
Target version: 2.7

Description

RubyVM contains mostly MRI-specific features but resolve_feature_path is clearly not MRI-specific.

So I propose to move it as a class method of Kernel.
I think this makes sense given the related load and require are defined in Kernel too.

Moreover, moving this method outside RubyVM is necessary for other Ruby implementations to implement it, and keep the clean separation that RubyVM is only defined on MRI (see #15752).

So, can I move RubyVM.resolve_feature_path to Kernel.resolve_feature_path?

Do we need to keep the method on RubyVM (and deprecate it), or can we just remove it since anyway API under RubyVM is not stable?

cc mame (Yusuke Endoh)

Related issues:
Related to Ruby master - Feature #15230: RubyVM.resolve_feature_path Closed
Related to Ruby master - Feature #15752: A dedicated module for experimental ... Feedback

Associated revisions

Revision d77b84ca - 07/11/2019 05:05 AM - nobu (Nobuyoshi Nakada)
$LOAD_PATH.resolve_feature_path

Moved from RubyVM. [Feature #15903]

History

#1 - 06/05/2019 08:14 PM - Eregon (Benoit Daloze)
- Related to Feature #15230: RubyVM.resolve_feature_path added

#2 - 06/05/2019 08:14 PM - Eregon (Benoit Daloze)
- Related to Feature #15752: A dedicated module for experimental features added

#3 - 06/05/2019 08:17 PM - Eregon (Benoit Daloze)
- Assignee set to Eregon (Benoit Daloze)

#4 - 06/05/2019 10:51 PM - byroot (Jean Boussier)
This is kinda tangential, so sorry if it's shifting the discussion. But if resolve_feature_path is to be made a first class public API, I wonder if it could be the occasion to make Kernel#require invoke Kernel#resolve_feature_path under the hood.

The reasoning is similar to [#11140] which made Kernel#autoload invoke Kernel#require and allowed tools like bootscale / bootsnap.

If the feature resolution logic was swappable it could make these tools much simpler, and open the door to avoiding $LOAD_PATH entirely.

#5 - 06/05/2019 11:11 PM - mame (Yusuke Endoh)
(I'm an author of RubyVM.resolve_feature_path.)

Sorry but I'm not so positive. From perspective of module design, I agree that Kernel module looks the best place to add the method. However, we can't be too careful to add anything to Kernel nowadays. At least, I don't want to do that until we receive an actual request to make the method available in production. Currently, I have no reason to move it to Kernel, except module design consistency.

This is just my opinion. It is all right if matz accepted this.
mame (Yusuke Endoh) wrote:

However, we can't be too careful to add anything to Kernel nowadays.

I propose only as a class method, not an instance method, so I think there is literally no chance for conflicts. What's your concern?

At least, I don't want to do that until we receive an actual request to make the method available in production.

We very rarely receive this, e.g., even for RubyVM::InstructionSequence which is now used in production (bootsnap). I think it is not a good criteria, it's just too easy to use RubyVM in user code.

I understand we should have an actual use-case, but we already have since the feature was introduced. It would be useful when wanting to have more control over loading files (e.g., I guess this could be useful in RubyGems), and potentially bootsnap as byroot (Jean Boussier) just said above.

Currently, I have no reason to move it to Kernel, except module design consistency.

I think that's a good enough reason on its own. RubyVM shouldn't become a random collections of classes & methods of which part of it are MRI-specific and part not, part stable and part not. That's just so messy, so I'd like to fix that. This issue is a trivial fix for I think an obvious case that does not belong under RubyVM.

This is just my opinion. It is all right if matz accepted this.

OK, I'll assign to him and add to the developer meeting's agenda.

byroot (Jean Boussier) wrote:

This is kinda tangential, so sorry if it's shifting the discussion. But if resolve_feature_path is to be made a first class public API, I wonder if it could be the occasion to make Kernel#require invoke Kernel#resolve_feature_path under the hood.

Could you file a separate feature request and show or explain how bootsnap would use it?

deep-cover might be interested by this too, cc marcandre (Marc-Andre Lafortune)

mame (Yusuke Endoh) wrote:

However, we can't be too careful to add anything to Kernel nowadays.

I propose only as a class method, not an instance method

Oh sorry I missed the point. Fair enough. I'll ask matz's opinion at the next meeting.

At least, I don't want to do that until we receive an actual request to make the method available in production.

We very rarely receive this, e.g., even for RubyVM::InstructionSequence which is now used in production (bootsnap). I think it is not a good criteria, it's just too easy to use RubyVM in user code.

Let's try to remove it and see how many people are killed. (Joke)

nobu (Nobuyoshi Nakada) wrote:

How about $LOAD_PATH.resolve_feature_path?
#11 - 06/13/2019 09:33 AM - Eregon (Benoit Daloze)
nobu (Nobuyoshi Nakada) wrote:

How about $LOAD_PATH.resolve_feature_path?

As a singleton method, and because $LOAD_PATH cannot be re-assigned?
It's a fun idea, although I suspect this will be very hard to find and document properly, so I think it's better on Kernel as a class method.

#12 - 06/14/2019 02:51 AM - mame (Yusuke Endoh)
This ticket was discussed at yesterday dev meeting. Currently there is no singleton method to Kernel, so some people were reluctant. Nobu counterproposed $LOAD_PATH as above, and matz said he waits for eregon's response to the counterproposal.

#13 - 06/14/2019 09:45 PM - Eregon (Benoit Daloze)
Thanks for discussing the issue at the meeting.

I think having singleton-only methods on Kernel would be OK, and probably most of us agree having the instance method is not warranted for a rarely-used method.

Kernel makes sense to me, because it's where require and load are defined, and resolve_feature_path is a subset of those methods.

My main concern with $LOAD_PATH is documentation (e.g., where would it be listed on https://docs.ruby-lang.org/, there is no class, globals.rdoc doesn't seem right) and discoverability (hard to find it when searching for it), but otherwise it sounds fine.

Another concern with defining it on $LOAD_PATH is resolve_feature_path does not depend only on $LOAD_PATH but also on other values such as Dir.pwd, and maybe other things if the feature lookup changes behavior in the future.

Dear matz (Yukihiro Matsumoto), could you decide between Kernel and $LOAD_PATH?
I would be happy with either, and I believe that would be much better than RubyVM (as explained in the description).

#14 - 07/11/2019 04:53 AM - matz (Yukihiro Matsumoto)
I vote for $LOAD_PATH.resolve_feature_path. We need to improve the documentation as well.

Matz.

#15 - 07/11/2019 05:12 AM - nobu (Nobuyoshi Nakada)
- Status changed from Open to Closed

Applied in changeset gitid77b84ca82a1cef10ef06776a207151ef864b3ca.

$LOAD_PATH.resolve_feature_path

Moved from RubyVM. [Feature #15903]

#16 - 07/13/2019 10:37 AM - Eregon (Benoit Daloze)
Thanks for the decision, and thanks to nobu (Nobuyoshi Nakada) for already moving the method.
I noticed the documentation is still on RubyVM, I'll try to fix that.

#17 - 07/13/2019 01:37 PM - Eregon (Benoit Daloze)
I documented the new method in globals.rdoc and added a NEWS entry.