Ruby master - Bug #16959

Weakmap has specs and third-party usage despite being a private API

06/12/2020 10:09 PM - headius (Charles Nutter)

Status:	Open		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:		**Backport:**	2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN

Description

Weakmap is still described as an internal API, and the documentation points users at WeakRef as the official public API:

https://github.com/ruby/ruby/blob/1fb16dbb6e28b9f32f92554d29e646e088b21a98/gc.c#L11928-L11936

However there are now specs for its current set of features, even though those features have never been discussed or approved as a public API:

https://github.com/ruby/spec/tree/dd8437628a6f2de5b74b338d4960682bb1590a60/core/objectspace/weakmap

And we are starting to see it being used by the community:

- https://github.com/jruby/jruby/issues/6267
- https://github.com/rsim/oracle-enhanced/issues/2027
- https://github.com/rails/rails/pull/39121

One of two things needs to happen:

- Weakmap is made a public API after some discussion. It would be an official public feature only in 2.8/3.0 or higher.
- ~~The specs are be removed and Weakmap remains a private API not to be used by the community. I suspect the addition of the specs led to folks starting to use this private API.~~

(edit: The Rails PR was merged after the specs, but the change is actually a year old, as mentioned below. In any case there's plenty of in-the-wild uses of WeakMap that go back even further.)

Personally, I'm in much more in favor of making WeakRef support all the features necessary to implement Weakmap in pure Ruby, rather than the other way around:

https://bugs.ruby-lang.org/issues/6309

But whatever happens it needs to happen soon, since this use case is now a merged feature in Rails master.

History

#1 - 06/12/2020 10:18 PM - headius (Charles Nutter)

- Description updated

#2 - 06/12/2020 11:19 PM - chrisseaton (Chris Seaton)

> I suspect the addition of the specs led to folks starting to use this private API.

I think it's the other way around - the specs were added because people were using it - https://github.com/oracle/truffleruby/issues/1385 and https://github.com/oracle/truffleruby/issues/1958 is what led to the specs being written.

The example you give which all lead back to the Rails PR was written well *before* the specs - April 2019, and the specs were just written I believe, but maybe there is more context to why it was merged that is not referenced?

So I don't think it's right to blame the specs here. The specs reflect reality.

But I do think it could be worth having a conversation about moving some specs to an optional namespace - the C extension API is in there. It'd mark intent at least.

#3 - 06/13/2020 12:41 AM - headius (Charles Nutter)

> So I don't think it's right to blame the specs here. The specs reflect reality.

I'm not sure I'm seeing what you're seeing, since the specs appear to have been created and merged in mid-April and the Rails PR was created in May.

https://github.com/ruby/spec/commits/dd8437628a6f2de5b74b338d4960682bb1590a60/core/objectspace/weakmap

I can't tell if the merging of those specs in any way led to the Rails usage, so I'll retract that theory. There certainly are many other uses of WeakMap that predate the specs, including the Mustermann usage that (presumably) led to the specs being created.

The specs should reflect official, public Ruby APIs... not private APIs that were added unilaterally by one implementation without any discussion. We should always be vigilant about hidden features of Ruby becoming de facto standards, given the mess that RubyVM has become for alternative Rubies.

JRuby isn't entirely blameless here, since we implemented WeakMap shortly after it was introduced almost a decade ago (mostly in hopes of sharing the pure-Ruby weakref.rb, which didn't end up panning out). WeakMap itself was actually introduced because I and others proved that the previous implementation of weakref.rb was fundamentally broken. We did not push for specs then largely because discussions were still ongoing about how to support weak references in Ruby.

We should really make Weakref a core API and build WeakMap on top of it (in pure Ruby)... not the other way around.

#4 - 06/13/2020 12:46 AM - headius (Charles Nutter)

I think it's also worth pointing out that the gc.c comment I linked above does propagate to the official documentation, which states that users should go to Weakref for the public API:

https://docs.ruby-lang.org/en/2.7.0/ObjectSpace/WeakMap.html

We should avoid adding specs for APIs documented as non-public... and we should also avoid adding "private" APIs to Ruby without at least a "require" opt-in.

#5 - 06/13/2020 12:55 AM - chrisseaton (Chris Seaton)

The PR was created recently, but the code was written in April 2019 https://github.com/rails/rails/pull/39121/commits/77f7b2df3aa3b7eb318cc830f239fd5ec2a7f28f, a year before it was specified. Sorry if it seems pedantic and thanks for retracting but I'm carefully pointing out that the specs definitely came second here because I don't want people to get the idea that the process of writing specs by itself cause problems like this or that people writing specs is a bad thing.

> We should avoid adding specs for APIs documented as non-public...

Given that people are using it in code in the wild, TruffleRuby needs specs or tests to figure out what MRI is doing. They could go into a private part of the TruffleRuby repo, not upstreamed to ruby/specs, but then does every implementation independently spec these things that we're not supposed to spec but we all have to as there's code using it?

But overall what you're saying does make sense and I can see your problem.

Maybe new specs should go through more PRs and community oversight? At the moment they do just tend to appear from downstream implementations. So then you'd have been able to see the new specs and have said 'wait are people using this, is this the behaviour we wanted, who is using this and why'.

#6 - 06/13/2020 01:17 AM - headius (Charles Nutter)

> Maybe new specs should go through more PRs and community oversight?

Well, clearly they should if they're not public APIs. Perhaps the spec author did not realize that was the case here.

This does illustrate a problem with each impl keeping their own copy of the specs and syncing periodically; it circumvents all review processes designed to ensure we're only adding "good" specs. Maybe we should revisit using something like git submodules for ruby/spec in alternative Ruby repos. Then we have the option to roll forward the submodule to pick up new specs *after* they're approved and merged.

#7 - 06/13/2020 01:18 AM - headius (Charles Nutter)

> or that people writing specs is a bad thing

I clearly didn't say that, unless you mean writing specs for private features...

#8 - 06/13/2020 01:20 AM - headius (Charles Nutter)

- Description updated

#9 - 06/13/2020 09:28 AM - zverok (Victor Shepelev)

Probably not only specs are "guilty", but also Ruby 2.7's changelog (and maybe, a small bit, my rendering of it): it included changes in "private" class as a part of official changelog, and people start noticing it can be used for all kind of stuff...

Though, the ticket #16035 that caused the changelog entry implies that people was already using it. And Matz's answer there goes as:

> I think this is the required behavior for WeakMap to implement a cache for example. Accepted.

(Also implying the class is publicly usable.)

I had clarifying this on my post-2.7 TODO list, but haven't acted on it, yet.

#10 - 06/13/2020 11:07 AM - Eregon (Benoit Daloze)

When APIs like this are used in the wild, for many years, the docs can state "private API" (well actually these docs don't, they just suggest using weakref.rb, which has its own issues) but it's simply not accurate anymore.

mustermann has been using ObjectSpace::WeakMap since 2016, and the "tool" gem before since 2014 or earlier: https://github.com/rkh/tool/issues/2.
There was a fallback with a Hash but that's been clearly incompatible in many subtle ways.

JRuby implemented ObjectSpace::WeakMap so that further made it "API any gem can use".

And indeed recently there has been a few improvements to ObjectSpace::WeakMap discussed on this tracker (#16035, #16038), which clearly shows this API is considered public.
And it was also mentioned as part of the 2.7 NEWS file: https://github.com/ruby/ruby/blob/master/doc/NEWS-2.7.0

The specs are reviewed by me during synchronization with upstream ruby/spec.
I think it makes perfect sense to add specs for ObjectSpace::WeakMap and e.g., other ObjectSpace features as long as >1 Ruby implementation implements them, and it's not specifically annotated as MRI-only (currently, that's only RubyVM).

So concretely, what problems do you see with adapting the documentation of ObjectSpace::WeakMap to mark it "non-private"?
Any issue in the API?
AFAIK the only problem is the documentation doesn't clearly mention what's weak.

#11 - 06/13/2020 11:07 AM - Eregon (Benoit Daloze)

- Description updated

#12 - 06/13/2020 11:12 AM - Eregon (Benoit Daloze)

In any case, having specs in ruby/spec doesn't mean it's public APIs.
That's why e.g., some private methods are also tested in ruby/spec.
Users should refer to documentation to find out if an API is public.

IMHO, every public constant and method is public API, or at least it is in practice with enough time.
RubyVM is the only module that is clearly documented as MRI-specific, might break APIs anytime, everything there is experimental.

#13 - 06/13/2020 06:10 PM - headius (Charles Nutter)

Please don't edit my description.

I personally do not believe the specs should be removed. But I also do not feel that WeakMap is the right abstraction. If Weakref had support for a reference queue, then WeakMap would be a pure-Ruby library that all implementations could share. Further, there would not just be the one WeakMap... Users could build their own weak arrays and other data structures. What I am asking for here is that we don't just accept this private API as official public API without going through the process that every other API must go through. This API was never discussed for public use, and has limitations that could be addressed quickly by making Weakref a bit more robust.

The bottom line is that WeakMap was added as a private API to make Weakref work better, but was never discussed and approved to become a public API. Yes, people have been using it, and that's not good. Yes, those uses predate the recent specs, but the specs and the 2.7 changelog make it even more likely this will become a de facto standard API, completely circumventing all processes for adding public APIs.

#14 - 06/13/2020 08:54 PM - Eregon (Benoit Daloze)

I don't see much the value of having the WeakMap implementation shared between Ruby implements, it's a tiny class, and CRuby/JRuby/TruffleRuby already all have their own WeakMap implementation.

So I guess your main point is having some sort of reference queue in addition of the existing WeakRef would be more flexible and allow building other weak data structures.
That makes sense to me.
I'd suggest to restart the discussion/review the PR on #6309 then.

WeakMap itself is useful in its own right, that's why a handful gems use it, and so I think there is no hope to remove it.

IMHO ObjectSpace::WeakMap became public API as soon as it was introduced, i.e., nothing recent.
Although to be fair I don't know the exact history of how ObjectSpace::WeakMap was added in CRuby.

> completely circumventing all processes for adding public APIs.

What's that process exactly?
As far as I know, once it's implemented in CRuby and in a release it's pretty much public API, discussed or not.
I agree discussion would be useful, but AFAIK there is no formal process to introduce new APIs in Ruby, just needs an approval from matz.