

Ruby - Feature #6308

Eliminate delegation from WeakRef

04/17/2012 05:02 PM - headius (Charles Nutter)

Status:	Assigned	
Priority:	Normal	
Assignee:	matz (Yukihiro Matsumoto)	
Target version:		
Description		
<p>WeakRef's delegation features are a really awful pattern that should not be allowed in future versions of Ruby.</p> <p>WeakRef makes no guarantees as to the liveness of its contained object. It can be collected at any time if there are no strong references to it.</p> <p>WeakRef currently uses delegation to pass method calls through to the contained object. This encourages a pattern where a WeakRef is passed to methods that expect to have a reference to the underlying object, making it appear to be that object.</p> <p>Unfortunately, this is <i>never</i> a good idea. Because the object can be collected at any time, you may get a nil reference from getobj <i>arbitrarily</i> in code that tries to call methods against the given WeakRef. That means using WeakRef as a delegate will always result in unreliable code, and errors may happen for inexplicable reasons.</p> <p>I believe Ruby 2.0 should eliminate WeakRef's delegation features and make it a simple reference holder. There's no safe way to use a weak reference except to grab a reference to the object, check that it is alive (non-nil) and then proceed with the use of the object, as follows:</p> <pre>obj = weakref.getobj raise ArgumentError unless obj obj.do_something obj.do_something_else</pre> <p>Along with eliminating delegation, I would recommend simply making the get method #get, since the uglier #getobj is only named that way because it is not delegated.</p>		
Related issues:		
Related to Ruby - Feature #6309: Add a reference queue for weak references		Assigned

History

#1 - 04/17/2012 08:01 PM - nobu (Nobuyoshi Nakada)

- Category set to lib

It should be another new class, I think.

#2 - 04/18/2012 01:43 AM - headius (Charles Nutter)

Perhaps under the GC module. Seems better than under ObjectSpace

I also think WeakMap should perhaps be moved under GC.

GC::WeakRef
GC::WeakMap
GC::ReferenceQueue

The top-level WeakRef library should probably be deprecated, then, and implemented in terms of GC::WeakRef.

#3 - 04/20/2012 04:53 AM - mame (Yusuke Endoh)

- Status changed from Open to Feedback

Hello,

2012/4/17, headius (Charles Nutter) headius@headius.com:

WeakRef's delegation features are a really awful pattern that should not be allowed in future versions of Ruby.

Maybe I understood your subject, but your proposal is not clear.
Could you please make it concrete?

"change the behavior of lib/weakref.rb in 2.0"

=> Impossible, because of matz's 2.0 compatibility policy.

"change the behavior of lib/weakref.rb in 3.0"

=> Maybe possible if matz accepts. Please create a patch that warn a user when the delegation features are used.
Then, I'll set this ticket as 3.0 issue.

"deprecate lib/weakref.rb and add an alternative library"

=> Please show us the alternative first.

--

Yusuke Endoh mame@tsg.ne.jp

#4 - 04/28/2012 03:00 PM - headius (Charles Nutter)

My proposal is that at some time deemed acceptable by ruby-core and Matz, the delegate-based WeakRef should go away, and that in Ruby 2.0 a "preferred" non-delegate WeakRef be added (ideally along with the reference queue support in my other bug).

Here is a patch that adds a non-delegate WeakRef implementation: <https://gist.github.com/2516417>

This is modeled after the Java WeakRef class, which can only be traversed or cleared. Again, I hope that this would be implemented in an efficient way as part of the GC subsystem, but this implementation should show what I'm looking for. I still whole-heartedly believe that using WeakRef as a delegate is an awful, awful pattern that should be discouraged now and unavailable in the future. This patch would force users to retrieve the weak reference into a hard reference, check it for liveness, and *then* proceed to use the object, which is the only safe way to deal with weak references.

#5 - 05/03/2012 11:52 AM - mame (Yusuke Endoh)

- Status changed from Feedback to Assigned

- Assignee set to matz (Yukihiro Matsumoto)

Thanks, please wait a "ruling" of matz.

--

Yusuke Endoh mame@tsg.ne.jp

#6 - 11/17/2012 12:59 AM - headius (Charles Nutter)

Seven months and no activity. I still would like to see delegate-based Weakref go away, but I know the official plan is to have no backward compatibility. Unfortunately, delegate-based Weakref is still a bad, broken implementation, and I think it should go away. 2.0 is as good a time as any.

#7 - 11/24/2012 10:49 AM - mame (Yusuke Endoh)

- Target version set to 2.6

#8 - 03/16/2013 02:26 AM - headius (Charles Nutter)

I request a ruling by matz for making the backward-incompatible change of having Weakref no longer be a Delegate. Alternatively, make it a Delegate where all delegated methods warn that you shouldn't use it as a Delegate.

If necessary I will move this feature request to CommonRuby.

#9 - 04/17/2013 02:36 AM - headius (Charles Nutter)

- Project changed from Ruby to 14

- Category deleted (lib)

- Target version deleted (2.6)

Moving into CommonRuby

#10 - 04/17/2013 02:53 AM - headius (Charles Nutter)

Here's a patch that removes delegation from Weakref: <https://github.com/headius/ruby/commit/431b971a147daf8a9b2185d117580a842f3c8dfc>

One test was no longer really relevant (and not adding anything), and others were modified to reflect this change.

#11 - 09/27/2013 08:17 PM - headius (Charles Nutter)

I believe this is just waiting on approval by matz. Can we make this change in 2.1, please?

#12 - 10/01/2013 09:38 AM - headius (Charles Nutter)

- Target version set to Ruby 2.1.0

#13 - 10/01/2013 09:57 AM - headius (Charles Nutter)

Put my patch into an updated pull request: <https://github.com/ruby/ruby/pull/406>

#14 - 10/01/2013 11:49 AM - nobu (Nobuyoshi Nakada)

My opinion is still unchanged.
"It should be another new class".

#15 - 10/01/2013 01:51 PM - headius (Charles Nutter)

nobu (Nobuyoshi Nakada) wrote:

My opinion is still unchanged.
"It should be another new class".

In case my opinion wasn't clear, or has been fogged over the last few months, I'll make it clear again.

WeakRef is broken when used as a delegate, because any method at any time might raise an error the code is not prepared to handle. This is filed as a feature, but it's actually a bug fix to eliminate the buggy behavior of transparently delegating method calls through WeakRef to its object.

Introduce another class called what? UndelegatedWeakRef? WeakRefThatIsNotBroken?
UseThisWeakRefBecauseTheOtherOneWillRaiseErrorsRandomly? :-)

Introducing another class will just add confusion to stdlib and allow the current broken WeakRef to perpetuate. We did not want any breaking changes in 2.0, but I think we really need to do this in 2.1.

#16 - 10/31/2013 02:07 PM - nobu (Nobuyoshi Nakada)

- File 0001-weakref.rb-non-delegation.patch added

headius (Charles Nutter) wrote:

Introducing another class will just add confusion to stdlib and allow the current broken WeakRef to perpetuate. We did not want any breaking changes in 2.0, but I think we really need to do this in 2.1.

Changing existing class drastically breaks existing code all.
It's not acceptable.

For new name, what about WeakRef.[]?

#17 - 11/01/2013 02:19 AM - nobu (Nobuyoshi Nakada)

Chained WeakRef (WeakRef to WeakRef) doesn't work well without delegation.

```
w = WeakRef.new(obj)
w = WeakRef.new(w)
w.foo
```

and

```
w = WeakRefWithoutDelegation.new(obj)
w = WeakRefWithoutDelegation.new(w)
w.get.get.foo
```

How do you think?

#18 - 11/01/2013 04:05 AM - headius (Charles Nutter)

nobu (Nobuyoshi Nakada) wrote:

Chained WeakRef (WeakRef to WeakRef) doesn't work well without delegation.

```
w = WeakRef.new(obj)
w = WeakRef.new(w)
```

w.foo

I can't say I've ever had a need to wrap a weakref in a weakref. What's the use case?

and

```
w = WeakRefWithoutDelegation.new(obj)
w = WeakRefWithoutDelegation.new(w)
w.get.get.foo
```

How do you think?

This looks fine to me. WeakRef is a reference holder, so calling get is correct.

Let me summarize reasons why delegation is bad:

- Calls can fail at any time.
- Error handling is not possible because other consumers of the WeakRef won't know it's a WeakRef.
- On earlier versions of Ruby (pre-2.0) another object could get the same object_id and you'd call against the wrong object.

The bottom line is that delegating through a WeakRef is a big potential bug if you're not constantly checking that the referenced object still exists.

nobu (Nobuyoshi Nakada) wrote:

Changing existing class drastically breaks existing code all.
It's not acceptable.

Many behaviors change in breaking ways in x.y releases. I contend that WeakRef transparently delegating is a bug that needs to be fixed.

For new name, what about WeakRef.[]?

I'll say it one more time: I don't believe *anyone* should be using the existing WeakRef behavior. Adding another class won't fix code that's out there doing things wrong (and delegating through WeakRef is *wrong*).

If we have to introduce a new class (I strongly protest) something like SimpleWeakRef might be better than a symbolic name.

#19 - 11/01/2013 04:06 AM - headius (Charles Nutter)

headius (Charles Nutter) wrote:

If we have to introduce a new class (I strongly protest) something like SimpleWeakRef might be better than a symbolic name.

Or how about WeakReference, and we add a warning saying WeakRef is deprecated? I can accept that.

#20 - 11/01/2013 04:28 AM - headius (Charles Nutter)

I have updated my PR to put the new non-delegating weak reference in a WeakReference class and add a deprecation warning to WeakRef.

<https://github.com/ruby/ruby/pull/406>

#21 - 12/23/2021 11:40 PM - hsbt (Hiroshi SHIBATA)

- Project changed from 14 to Ruby
- Target version deleted (Ruby 2.1.0)

Files

0001-weakref.rb-non-delegation.patch	1.63 KB	10/31/2013	nobu (Nobuyoshi Nakada)
--------------------------------------	---------	------------	-------------------------