Ruby trunk - Bug #13000

Implement Set#include? with Hash#include?

12/02/2016 04:25 PM - headius (Charles Nutter)

Status:	Feedback		
Priority:	Normal		
Assignee:	knu (Akinori MUSHA)		
Target version:			
ruby -v:		**Backport:**	2.1: UNKNOWN, 2.2: UNKNOWN, 2.3: UNKNOWN

Description

Why does Set#include? not call Hash#include?? Currently it calls Hash#[].

The protocol of Set already use Hash#include? for ==.

```
diff --git a/lib/set.rb b/lib/set.rb
index 43c388c..f3dbe2d 100644
--- a/lib/set.rb
+++ b/lib/set.rb
@@ -230,7 +230,7 @@ def flatten!
   #
   # See also Enumerable#include?
   def include?(o)
-    @hash[o]
+    @hash.include?(o)
   end
   alias member? include?
```

History

#1 - 12/21/2016 06:08 AM - shyouhei (Shyouhei Urabe)

- Assignee set to knu (Akinori MUSHA)

- Status changed from Open to Assigned

#2 - 12/21/2016 08:11 AM - knu (Akinori MUSHA)

It originally used Hash#include?, but changed to use Hash#[] to benefit from the optimized dispatch VM instruction for [] (opt_aref). ([Misc #10754])

Running a benchmark, I can observe that Hash#[] actually has an advantage over include? in performance (up to ~1.2x faster) but the "optimization" may only apply to CRuby. Do you think we should have a straightforward implementation for a library shared between Ruby implementations, or is it OK to leave this if I add a comment to explain why?

#3 - 10/21/2017 03:24 PM - knu (Akinori MUSHA)

- Status changed from Assigned to Feedback

#4 - 11/28/2017 05:45 AM - headius (Charles Nutter)

I would prefer the straightforward implementation, but I have some bias.

In JRuby, the [] method generally is *more* expensive, because it *might* be String#[] with a Regex, which needs to be able to set $~, so we deoptimize some things when [] is being called.

What's good for MRI here is bad for JRuby :-)

I guess the real question here is whether it would matter if JRuby just used Hash#include? in our version of the library. I think there might be some oddities around nil, but that already seems pretty odd in a Set.