# Ruby master - Feature #12655

## Accessing the method visibility

08/04/2016 11:26 AM - mathieujobin (Mathieu Jobin)

| | | |
|---|---|---|
| **Status:** | Feedback | |
| **Priority:** | Normal | |
| **Assignee:** | | |
| **Target version:** | | |

**Description**

I took on the task to make the looksee gem work with ruby 2.3 and 2.4. Unfortunately, some features were not directly accessible in ruby, so a C extension was made, which includes some ruby internals.

For ruby 2.2 support, internal.h and method.h were included. For ruby 2.3, I found I need id_table.h at least, but I did not fully succeed making it work.

In a short email thread, koichi san suggested it would be better to request to add the necessary public interface to ruby so that looksee would not need internals.

looksee creates these new methods from ruby internals:

```
rb_define_method(mMRI, "internal_superclass", Looksee_internal_superclass, 1);
rb_define_method(mMRI, "internal_class", Looksee_internal_class, 1);
rb_define_method(mMRI, "internal_public_instance_methods",
Looksee_internal_public_instance_methods, 1);
rb_define_method(mMRI, "internal_protected_instance_methods",
Looksee_internal_protected_instance_methods, 1);
rb_define_method(mMRI, "internal_private_instance_methods",
Looksee_internal_private_instance_methods, 1);
rb_define_method(mMRI, "internal_undefined_instance_methods",
Looksee_internal_undefined_instance_methods, 1);
rb_define_method(mMRI, "included_class?", Looksee_included_class_p, 1);
rb_define_method(mMRI, "singleton_class?", Looksee_singleton_class_p, 1);
rb_define_method(mMRI, "singleton_instance", Looksee_singleton_instance, 1);
rb_define_method(mMRI, "real_module", Looksee_real_module, 1);
rb_define_method(mMRI, "module_name", Looksee_module_name, 1);
```

It uses the following macros to find the method visibility and whether it has been redefined:

```
UNDEFINED_METHOD_ENTRY_P(me)
METHOD_ENTRY_VISI(me)
```

Ideally, a ruby method that returns visibility should return one of the following values:

```
[:public, :protected, :private, :undefined, :overridden]
```

We are using other ruby macros to find where and within which module or class the method is defined.

```
RCLASS_SUPER(internal_class)
CLASS_OF(object)
RCLASS_M_TBL(klass)
SPECIAL_CONST_P(object)
BUILTIN_TYPE(object)
FL_TEST(singleton_class, FL_SINGLETON)
RCLASS_IV_TBL(singleton_class)
RBASIC(module_or_included_class)->klass
```

You can see what I have tried for ruby 2.3: https://github.com/oggy/looksee/pull/36/files#diff-d5ef4b0cfbd5a6712f37dfa7ffbe2130

I cannot use rb_id_table_foreach, which suggests that I need to import much ruby code into the extension. I prefer not to include more C into this gem.

So I am trying to extract st_table from struct rb_id_table*, but I am getting a deferencing incomplete type error.

Please help.

## History

#### #1 - 08/05/2016 01:32 AM - shyouhei (Shyouhei Urabe)

I think it should be possible to implement locksee in pure-ruby.  To me it seems OK to have a way to access visibility of a method.

It is not clear if locksee cannot be implemented without exposing IClass, though.

#### #2 - 08/06/2016 12:26 AM - nobu (Nobuyoshi Nakada)

*- Status changed from Open to Feedback*

*- Description updated*

Mathieu Jobin wrote:

> it uses the following macros to find the method visibility and if it has been redefined
>
> ```
> UNDEFINED_METHOD_ENTRY_P(me)
> METHOD_ENTRY_VISI(me)
> ```

I'd love to add the functions, but are they really needed?

> Ideally, a ruby method that would return its visibility, should return one of the following value
>
> ```
> [:public, :protected, :private, :undefined, :overridden]
> ```

I think :overridden doesn't make sense in ruby.
What do you expect by it?

> we are using other ruby macros to find where the method is define, which module or class.
>
> ```
> RCLASS_SUPER(internal_class)
> CLASS_OF(object)
> SPECIAL_CONST_P(object)
> BUILTIN_TYPE(object)
> FL_TEST(singleton_class, FL_SINGLETON)
> ```

These would be safe.

> ```
> RCLASS_M_TBL(klass)
> RCLASS_IV_TBL(singleton_class)
> ```

It's not recommended to directly access internal members, m_tbl, iv_tbl, etc.

> ```
> RBASIC(module_or_included_class)->klass
> ```

It is preferable to use RBASIC_CLASS() instead of RBASIC()->klass.

> that is mostly it
>
> you can see what I have tried for ruby 2.3
> https://github.com/oggy/looksee/pull/36/files#diff-d5ef4b0cfbd5a6712f37dfa7ffbe2130
>
> I'm unable to use rb_id_table_foreach which seems like I would need to import too much of ruby code inside the extension... which I would prefer not including more C into this gem

The reason to use id_table seems to collect methods for each visibilities.
What differs from rb_class_public_instance_methods etc?

What you need are rb_class_undefined_instance_methods and
rb_class_singleton_object (provisional names)?

#### #3 - 08/07/2016 05:18 PM - oggy (George Ogata)

Hi,

I'm the original author of looksee.

I agree that we don't need everything in the Looksee extension in ruby. Looksee was written back in 2009 when I think the situation was a little different, but now I believe Module#ancestors should suffice to get the chain of modules.

Nobuyoshi Nakada wrote:

> What you need are rb_class_undefined_instance_methods and
> rb_class_singleton_object (provisonal names)?

I think this is exactly right. Would you consider exposing these at the ruby level?

**#4 - 08/10/2016 04:45 AM - shyouhei (Shyouhei Urabe)**

We briefly looked at this issue at yesterday's developer meeting and roughly agreed that locksee should be able to be done in pure-ruby.

Sad news is we had no time to have a deeper look at it so what is actually needed was not made clear at the meeting.

**#5 - 08/11/2016 06:19 AM - nobu (Nobuyoshi Nakada)**

https://github.com/ruby/ruby/compare/trunk...nobu:feature/12655

**#6 - 09/06/2016 11:23 PM - oggy (George Ogata)**

Thanks Shyouhei! I believe I need the 2 methods Nobu has implemented in this patch here to implement Looksee entirely in ruby, which I would love to do. In the latest version of looksee I have reduced the MRI extension down to just these 2 methods, so if this were in trunk I could swap those in & try it.

(Or if you are reluctant to expose these at the ruby level, is there any chance they could be made available to extensions?)

**#7 - 03/27/2019 08:31 PM - mathieujobin (Mathieu Jobin)**

I know its been two years already.
but do you think we could get those new methods merged into trunk?

**#8 - 12/23/2019 02:29 PM - mathieujobin (Mathieu Jobin)**

mri/2.x.0/internal.h:1363:11: error: conflicting types for ‘rb_st_copy’
1363 | st_table *rb_st_copy(VALUE obj, struct st_table *orig_tbl);

looks like this has changed in ruby 2.7

we would have appreciated the public methods from nobu to be merged in...

**#9 - 12/24/2019 02:50 AM - nobu (Nobuyoshi Nakada)**

Sorry, I've forgotten completely.
Updated the patch, but it is sorry to be too late for 2.7.

**#10 - 12/24/2019 04:42 AM - sawa (Tsuyoshi Sawada)**

*- Description updated*

*- Subject changed from accessing a method visibility to Accessing the method visibility*