Ruby master - Feature #6840

YAML tag method

08/07/2012 09:30 AM - trans (Thomas Sawyer)

Status:	Rejected
Priority:	Normal
Assignee:	tenderlovemaking (Aaron Patterson)
Target version:	2.0.0

Description

=begin
When loading YAML documents that have a tag, there doesn't seem to be anyway to query for that information.

h = YAML.load("--- !foo\na: 1\nb: 2")
h #=> {"a"=>1, "b"=>2}
h.what_method? #=> "!foo"

I know about YAML.add_domian_tag and the like, but registering tags up front doesn't always fit the usecase. For instance, I am working on a project where I don't want the end users to have to use a support library to work with the data, and I certainly don't expect them to copy and paste dozens of lines of #add_domain_tag boilerplate to their projects. Yet the tags can be useful to them for type revison information, e.g.

--- !foo/2

So this is the second revision of the foo type. The information could be important to consumer apps to ensure they handle the data properly.

So I am proposing that Pysch add a method (maybe call it #yaml_tag), that can be used to get this information from an object when it has been loaded via YAML.
=end

History

#1 - 08/07/2012 01:58 PM - nobu (Nobuyoshi Nakada)

- Assignee set to tenderlovemaking (Aaron Patterson)

- Target version changed from 1.9.3 to 2.0.0

#2 - 08/08/2012 06:19 AM - tenderlovemaking (Aaron Patterson)

- Status changed from Open to Rejected

#3 - 08/08/2012 06:23 AM - Anonymous

- File noname added

On Tue, Aug 07, 2012 at 09:30:51AM +0900, trans (Thomas Sawyer) wrote:

> Issue #6840 has been reported by trans (Thomas Sawyer).
>
> ---
>
> Feature #6840: YAML tag method
> https://bugs.ruby-lang.org/issues/6840
>
> Author: trans (Thomas Sawyer)
> Status: Open
> Priority: Normal
> Assignee:
> Category: lib
> Target version: 1.9.3
>
> =begin
> When loading YAML documents that have a tag, there doesn't seem to be anyway to query for that information.
>
> h = YAML.load("--- !foo\na: 1\nb: 2")
> h #=> {"a"=>1, "b"=>2}
> h.what_method? #=> "!foo"

> I know about YAML.add_domian_tag and the like, but registering tags up front doesn't always fit the usecase. For instance, I am working on a project where I don't want the end users to have to use a support library to work with the data,

If you're defining custom data types (which it sounds like you are),
wouldn't you *want* users to install a support library? You'll probably
need to update and version those custom data types (I assume people will
probably want to upgrade some day).

> and I certainly don't expect them to copy and paste dozens of lines of #add_domain_tag boilerplate to their projects. Yet the tags can be useful to them for type revison information, e.g.
>
> --- !foo/2
>
> So this is the second revision of the foo type. The information could be important to consumer apps to ensure they handle the data properly.

This sounds like a complex object. What is a "foo" type? Who defines
the "foo" type? If the "foo" type is the same as a built in Ruby
object, why do we need the extra methods?

> So I am proposing that Pysch add a method (maybe call it #yaml_tag), that can be used to get this information from an object when it has been loaded via YAML.

I have to reject this. I don't want to add more methods to core objects
(especially with things specific to YAML), and this problem seems like
something easily solvable by installing a gem.

--
Aaron Patterson
http://tenderlovemaking.com/

#4 - 08/08/2012 09:52 PM - trans (Thomas Sawyer)

As I stated before, for my use case, I do *not* want users to use a support library. In fact, it's not just want, its a requirement. This format must be inter-operable. It doesn't mater if the user is programming in Perl, Python, Javascript or any other language. They need to be able to access the information just as readily. I'm not about to write a library for each possible language. The same holds for Ruby.

The "foo" type is a very specific YAML specification. If the user simply knows the name and version of the spec then they know what to expect. That's the most important thing. Any "Foo" library I write to support "foo" is just a convenience if it suites the purposes of the specifications user. But usually it will not, and should not.

But this isn't really about my specific use case. There is a more fundamental issue here. Having to define a global handler upfront makes it impossible to handle the incoming information dynamically. It is one thing to make something difficult, it's another to make it impossible. I realize it sucks to add core methods, but in this case its really a feature that's needed.

#5 - 08/09/2012 03:23 AM - Anonymous

- File noname added

On Wed, Aug 08, 2012 at 09:52:15PM +0900, trans (Thomas Sawyer) wrote:

> Issue #6840 has been updated by trans (Thomas Sawyer).
>
> As I stated before, for my use case, I do *not* want users to use a support library. In fact, it's not just want, its a requirement. This format must be inter-operable. It doesn't mater if the user is programming in Perl, Python, Javascript or any other language. They need to be able to access the information just as readily. I'm not about to write a library for each possible language. The same holds for Ruby.
>
> The "foo" type is a very specific YAML specification. If the user simply knows the name and version of the spec then they know what to expect. That's the most important thing. Any "Foo" library I write to support "foo" is just a convenience if it suites the purposes of the specifications user. But usually it will not, and should not.
>
> But this isn't really about my specific use case. There is a more fundamental issue here. Having to define a global handler upfront makes it impossible to handle the incoming information dynamically. It is one thing to make something difficult, it's another to make it impossible.

You're welcome to access the AST and translate yourself:

```
require 'psych'

Psych.parse_stream("---\n- !foo/2 omg\n- zomg") do |ast|
  ast.each do |node|
    if node.tag
      p [:tag, node.tag, node.to_ruby]
    else
      p [node.class, :notag, node.to_ruby]
    end
  end
end
```

You can be as dynamic as you want here.

> I realize it sucks to add core methods, but in this case its really a feature that's needed.

Sorry, I'm not convinced.

--
Aaron Patterson
http://tenderlovemaking.com/

Files

noname	500 Bytes	08/08/2012	Anonymous
noname	500 Bytes	08/09/2012	Anonymous