Ruby master - Feature #14927

Loading multiple files at once

07/20/2018 06:13 PM - Anonymous

Status:	Open	
Priority:	Normal	
Assignee:	matz (Yukihiro Matsumoto)	
Target version:		

Description

Just a proof concept I wanted to share. Maybe it could be useful?

Say you want to load all the .rb files in your lib directory:

```
Dir['lib/**/*.rb'].each { |file| load(file) }
```

This approach may not work if your files have dependencies like that:

```
# lib/foo.rb

class Foo < Bar

end

# lib/bar.rb

class Bar

end
```

Foo class needs Bar class. You will get a NameError (uninitialized constant Bar).

So in my personal projects, I use this algorithm to load all my files and to automatically take care of dependencies (class/include):

```
def boot(files)
  i = 0
  while i < files.length
    begin
      load(files[i])
    rescue NameError
      i += 1
    else
      while i > 0
        files.push(files.shift)
        i -= 1
      end
      files.shift
    end
  end
end

boot Dir['lib/**/*.rb'] # It works! foo.rb and bar.rb are properly loaded.
```

My point is: it would be cool if Kernel#load could receive an array of filenames (to load all these files in the proper order). So we could load all our libs with just a single line:

```
load Dir['{path1,path2}/**/*.rb']
```

History

#1 - 07/20/2018 07:17 PM - shevegen (Robert A. Heiler)

I wanted to propose a more sophisticated load-process in ruby some time ago, but I never got around it. I am even thinking of being able to load files based on abbreviations/shortcuts, without necessiting a hardcoded path (e. g. require needs the path, whereas with an

abbreviation we could only refer to that abbreviation, and an internal list keeps track of where the actual file resides instead). But it's not so simple to suggest something that has a real chance of inclusion. I am glad to see other people have somewhat similar ideas - of course your suggestion is quite different from my idea, but you tap into a very similar situation:

- Handling multiple files.

This is especially useful for larger ruby projects. For smaller projects it is not so important perhaps but when you have like +50 .rb files and growing, making things easier in regards to handling files, would be great.

To the suggestion - I think several ruby hackers may benefit from better handling of files.

I am not sure if there is a big chance to see load() and require() itself being changed, but I also don't know. I think we should ask matz, but there may be a chance that they may not be changed, possibly due to backwards compatibility (if there is a problem). In the long run we may want to consider using alternative means. For example, the require-family, such as require_relative(). I don't mean require_relative in itself, but something related to require_*.

require_relative also handles location to other files, just relative to the directory at hand.

By the way, I also understand this use case:

> This approach may not work if your files have dependencies like that:

And it is related to another use case which isn't a lot of fun:

- Circular dependencies + warnings about this

I also thought about this with my never-written proposal... :D

Circular dependency warnings are not a lot of fun IMO.

I think Hiroshi Shibata also had a suggestion in regards to ... require, I think, some months or a few years ago, but I don't remember what it was exactly.

Anyway, before I write way too much and digress from the suggestion, I am in general in favour of your suggestion. I don't have any particular opinion on your proposed solution - another API may be fine or perhaps a new method... load_files() ? Hmm... may not be an ideal name either. But I think the specific API may be a detail. The more important aspect is whether ruby can provide easier means for ruby users to load or require a batch of files. Perhaps load() and require() will remain as they are, for simplicity and backwards compatibility, but in such a case we could think about better ways to handle the task of "pulling all necessary files" into a project. This may also help people when they create gems.

In my own larger gems I do something very similar as to what Sébastien Durand showed, e. g. I also do Dir['*.rb'] often on a per-directory basis. That way I don't have to specify the names of the individual .rb files.

Anyway, +1 from me.

#2 - 07/20/2018 08:23 PM - ahorek (Pavel Rosický)

Dir glob has to find all files, sort them, create objects.
Then require loads them again from the filesystem...

I think Dir.glob + require is a very common pattern and if we have a function like require_directory / require_tree? some of these unnecessary steps could be skipped and simplified.

#3 - 09/08/2018 02:35 PM - shevegen (Robert A. Heiler)

I thought about creating a new issue but then I remembered that the issue here refers to a similar use case that I wanted to show.

Take the following link as an example:

https://github.com/jordansissel/fpm/blob/master/lib/fpm.rb

In the event that the project may be relocated, here is the copy/paste outcome of that code:

```
require "fpm/namespace"

require "fpm/package"
require "fpm/package/dir"
require "fpm/package/gem"
require "fpm/package/deb"
require "fpm/package/npm"
require "fpm/package/rpm"
require "fpm/package/tar"
require "fpm/package/cpan"
require "fpm/package/pear"
require "fpm/package/empty"
require "fpm/package/puppet"
require "fpm/package/python"
require "fpm/package/osxpkg"
require "fpm/package/solaris"
require "fpm/package/p5p"
require "fpm/package/pkgin"
require "fpm/package/freebsd"
require "fpm/package/apk"
```

As you can see, there are several require statements for the subdirectory at fpm/package/.

I think this is a very common use case. I encounter it myself a lot in (almost) daily writing of ruby code, where I have to load code stored in .rb files spread out.

Of course there are workarounds over the above, e. g. the Dir[] or Dir.glob example that was given here (and the former I use a lot). But I think it may be nicer to have an official API support this as well.

The name could be:

```
require_files
```

The first argument could be the path to the subdirectory at hand; the second argument could be an options Hash that allows more fine-tuning, such as traversing subdirectories, handling .so files as well, or exclusively, and so on and so forth.

I believe it may fit into the "require" family, since that already has e. g. require_relative.

In the long run it would be nice to even be able to refer to .rb files without having to use any hardcoded path at all - but for the time being, any support for requiring/loading files helps a lot.

(To the issue of dependencies in said .rb files, I usually batch-load the .rb files, and if I get some error about an uninitialized constant, I add it into that .rb file at hand. It's a bit cumbersome but I understand that this part is not easy to change presently.)

I think require_directory() is a better name that require_tree() but I also like require_files().

The more important part is to want to convince that this is a common pattern, which is also why I added an example from a quite popular ruby project (fpm presently has ~7.3 million downloads on rubygems.org).

What I encounter myself doing is that, for my larger projects in ruby, I end up creating a subdirectory called requires/ and in that directory I put .rb files that handle loading of require-related activities, including subdirectories and external dependencies.

#4 - 09/09/2018 04:33 AM - nobu (Nobuyoshi Nakada)

shevegen (Robert A. Heiler) wrote:

> In the event that the project may be relocated, here is the copy/paste outcome of that code:
>
> ```
> require "fpm/namespace"
>
> require "fpm/package"
> require "fpm/package/dir"
> require "fpm/package/gem"
> require "fpm/package/deb"
> require "fpm/package/npm"
> require "fpm/package/rpm"
> require "fpm/package/tar"
> require "fpm/package/cpan"
> require "fpm/package/pear"
> require "fpm/package/empty"
> require "fpm/package/puppet"
> require "fpm/package/python"
> require "fpm/package/osxpkg"
> require "fpm/package/solaris"
> require "fpm/package/p5p"
> require "fpm/package/pkgin"
> require "fpm/package/freebsd"
> require "fpm/package/apk"
> ```

Doesn't the order matter?

#5 - 09/09/2018 05:28 AM - marcandre (Marc-Andre Lafortune)

nobu (Nobuyoshi Nakada) wrote:

> Doesn't the order matter?

Very often, it does not. If it does, one can always require the one that's needed first, say, then require the whole directory; require won't load the same file twice so this works fine.

We wrote a small method for this in deep-cover:
https://github.com/deep-cover/deep-cover/blob/master/core_gem/lib/deep_cover/tools/require_relative_dir.rb

One thing I really like about it is that it makes it clear that the whole directory is loaded. If there's a missing require "fpm/package/something" in the list above, it can take a while to notice.

My opinion on the feature request: not great for load, but could be useful for multiple require_relative.

#6 - 09/18/2018 12:38 AM - marcandre (Marc-Andre Lafortune)

- Assignee set to matz (Yukihiro Matsumoto)

Until we convince Matz, I pushed a gem require_relative_dir which hopefully can be helpful to others.