Ruby master - Feature #17156

Refinements per directory tree

09/06/2020 01:59 AM - Dan0042 (Daniel DeLorme)

Status:	Open	
Priority:	Normal	
Assignee:		
Target version:		

Description

One of the main use case for refinements is to enable extensions to core classes for a given codebase (app or gem) without impacting the other codebases. Let's call it sandboxed monkey-patching. Note that you usually want this for an entire codebase, not specific files.

But refinements can only be enabled on a per-module or per-file basis. So if you want refinements in your entire project, you need to add the "using" boilerplate to every single file. I find this too bothersome. I have some core extensions that I would love to convert into refinements, but adding a "using" statement to every file is too much of a pain; I find it not worth the trouble, especially considering that the risk of conflicts due to monkey-patching is very low to start with.

So I would like the ability to enable refinements for all the files in a project, that is, on a whole directory tree.

I'd like to keep it simple. So my idea is to have something like using M, subdirs: true that would be equivalent to using M at the top of every file loaded after that point within the directory tree of the file with the using statement. So the typical case would be, for gem "foobar" which is loaded via require "foobar", you'd put the using statement at the top of the gem's main foobar.rb file, and then every file loaded within the gem's directory would benefit from those refinements. Similarly, if you want to use certain refinements throughout your app, put them in a file in the root dir of the app and load it early in the initialization process.

By restricting this feature to the directory of the current file, we limit the possibility that someone can apply refinements globally by writing using M, subdirs: "/"

Also this would allow refinements to apply within erb templates, or any other code loaded via eval(code, nil, filename, lineno) where filename is within the targeted directory tree.

History

#1 - 09/15/2020 01:17 AM - Dan0042 (Daniel DeLorme)

- Description updated

#2 - 10/22/2020 03:14 AM - Dan0042 (Daniel DeLorme)

shugo (Shugo Maeda), may I ask for your opinion on this?

#3 - 12/07/2020 05:19 AM - marcandre (Marc-Andre Lafortune)

I believe a good solution would be a configurable RuboCop cop that could add it for you. It should be easily more customizable than any solution in core, while maintaining Matz' idea to avoid too much magic "hey where did this method come from".

If this interests you, please check https://github.com/rubocop-hq/rubocop/issues/9180