

Redmine - Defect #36245

ActiveSupport::Reloader.to_prepare not working in trunk 21287

2021-11-25 09:00 - Alexander Meindl

Status: Resolved	Start date:
Priority: Normal	Due date:
Assignee:	% Done: 0%
Category: Plugin API	Estimated time: 0.00 hour
Target version:	Affected version:
Resolution:	
Description	
<p>With r21283 in trunk ActiveSupport::Reloader.to_prepare and Rails.configuration.to_prepare is not fired anymore.</p> <p>ActiveSupport::Reloader.to_prepare is required for plugins (as an example), if some code should loaded after all plugins (other plugins) are loaded.</p> <p>Maybe there is another possibility with Rails 6 or Zeitwerk for doing this, but imho ActiveSupport::Reloader.to_prepare should work within plugins, too. See https://guides.rubyonrails.org/autoloading_and_reloading_constants.html#autoloading-when-the-application-boots</p>	
Related issues:	
Related to Redmine - Patch # 34072: Hook after plugins were loaded	Closed
Related to Redmine - Feature # 32938: Rails 6: Zeitwerk support	Closed

History

#1 - 2021-11-25 09:20 - Alexander Meindl

The defect is for the usage of ActiveSupport::Reloader.to_prepare in a plugin - not in Redmine itself (just to make it clear).

I did not find a solution to fix this behavior until now.

Here is an example for a plugin init.rb:

```
Redmine::Plugin.register :my_plugin do
  name 'My plugin'
  version '0.01'
end
```

```
ActiveSupport::Reloader.to_prepare do
  raise 'this is never called'
end
```

```
Rails.configuration.to_prepare do
  raise 'this is never called, too'
end
```

#2 - 2021-12-01 13:53 - Takashi Kato

I apologize for the delay in responding.

Before the introduction of zeitwerk, the autoloader loads Redmine plugins on initializing.

After the introduction of zeitwerk, to make the Redmine::Plugin class manageable for zeitwerk, Redmine::PluginLoader runs the "init.rb" for all plugins inside the "Rails.configuration.to_prepare" block (and run on every reload).

https://www.redmine.org/projects/redmine/repository/entry/trunk/lib/redmine/plugin_loader.rb#L108

Now processing written inside the "Rails.configuration.to_prepare" block in "init.rb" can be written directly in "init.rb".
Let me know if there are any problems in creating plugins.

#3 - 2021-12-01 14:52 - Takashi Kato

Alexander

It may not be a plugin you want to fix, but I found your plugin on GitHub and made it compatible with zeitwerk.

https://github.com/tohosaku/redmine_emojibutton/commits/zeitwerk

#4 - 2021-12-01 16:49 - Alexander Meindl

Hi Takashi,

thanks for your answer. The problem is, you cannot use Classes from Plugin B with Plugin A - because Plugin B is not initialized at this moment. Because of this, till now without zeitwerk, the solution was to use this Classes after all plugins are initialized (with Rails.configuration.to_prepare).

If Redmine::PluginLoader loads all plugins in "Rails.configuration.to_prepare" block, it is not possible to call an "Rails.configuration.to_prepare" block in a plugin again. This would be a "Rails.configuration.to_prepare" block in a "Rails.configuration.to_prepare" block - and this does not work - as it looks at the moment.

Here are some examples: https://github.com/AlphaNodes/additional_tags/blob/master/init.rb or https://github.com/AlphaNodes/redmine_saml/blob/master/init.rb or https://github.com/AlphaNodes/redmine_sudo/blob/master/init.rb (we build an [plugin loader](#) for that)

I am not sure, if you get me right. I try to explain, that if you have dependencies between plugins (which we have a lot), there is no way (or I do not know it), how we can run code from a plugin, after all plugins are initialized.

An example: *Plugin B* requires *Plugin C*. You cannot use *Plugin C* code in *Plugin B*, till it is initialized - and this worked perfectly with Rails.configuration.to_prepare before zeitwerk. Maybe to provide a hook after initializing all plugins could be a solution.

#5 - 2021-12-01 17:09 - Ko Nagase

Hi Alexander,

I still haven't tried Redmine trunk yet, but I encountered the similar situation which needs to control plugins load orders in Redmine 4.2-stable branch.
https://github.com/gtt-project/redmine_gtt/pull/130

From glance of Takashi's comment,

| https://www.redmine.org/projects/redmine/repository/entry/trunk/lib/redmine/plugin_loader.rb#L108

I noticed that there seems to be after_plugins_loaded hook which seems to be called when all plugins are loaded, so I guess that we can try to use this way as a workaround.

#6 - 2021-12-01 17:32 - Alexander Meindl

Hi Ko,

indeed I found `after_plugins_loaded` hook some hours ago. But the problem with that is, you cannot use a patched method in `Redmine::Plugin.register` block (e.g. to add a link to menu for special conditions, which requires a patch, which is applied later with `after_plugins_loaded` hook).

Maybe the way with `after_plugins_loaded` hook is the right direction. Not sure, if there are more problems with that. But dispense with `Rails.configuration.to_prepare` means a lot of rework/adjustments in plugins.

#7 - 2021-12-02 07:43 - Alexander Meindl

- Status changed from New to Resolved

`after_plugins_loaded` hook works for me as a replacement for `Rails.configuration.to_prepare`

#8 - 2021-12-02 09:22 - Go MAEDA

- Related to Patch #34072: Hook after plugins were loaded added

#9 - 2021-12-02 09:23 - Go MAEDA

- Related to Feature #32938: Rails 6: Zeitwerk support added

#10 - 2021-12-02 14:16 - Takashi Kato

Hi Alexander,

Glad that this is solved!

Thank you for your very meaningful report as it was a case that I hadn't really anticipated.

#11 - 2022-01-06 06:32 - Ko Nagase

Sorry for the very late reply.

I tried to check Zeitwerk plugin load sequence by "puts" debug on the latest 4.2-stable and master (trunk) branches with using ruby 2.7.4, and the difference was as follows:

	4.2-stable	master (trunk)
code		
puts 'MyPlugin -		
puts 'MyPlugin::Plugin.register		
myplugin::Plugin.register		
Redmine::Plugin.register MyPlugin		
'MyPlugin::Plugin.register' Authen		
MyPlugin.plugin_authentications		
for Redmine::Plugin::Authentications		
to_prepare example.com/path/to/plugin		
"http://example.com/path/to/plugin		
"http://example.com/about"endActi		
vs		

after_plugins_loaded		
Rails::Application.reloader.to_prepare		
ActiveSupport::Reloader.to_prepare		
Rails::Configuration.to_prepare		
Rails.application.reloader.to_prepare		
Rails::Application.config.to_prepare		
ActiveSupport::Reloader.to_prepare		
reMyPlugin -		
Rails.application.reloader.to_prepare		
are		

In master (trunk) branch, I had to change the Hook class definition, because of the following error when reloading.
(Thanks tohosaku and @matobaa for the @redmine_id_rize plugin's commit!)

TypeError (superclass mismatch for class AfterPluginsLoadedHook):

```
plugins/my_plugin/init.rb:29:in `<top (required)>'
lib/redmine/plugin_loader.rb:31:in `load'
lib/redmine/plugin_loader.rb:31:in `run_initializer'
lib/redmine/plugin_loader.rb:108:in `each'
lib/redmine/plugin_loader.rb:108:in `block in load'
```

In master (trunk) branch,

- "ActiveSupport::Reloader.to_prepare" and "Rails.application.reloader.to_prepare" are actually called when reloading, but not called at initialization.
- "ActiveSupport::Reloader.to_prepare" and "Rails.application.reloader.to_prepare" seem to be called multiple times after 2nd reloading, and I think that this behavior needs to be fixed.
- "after_plugins_loaded" hook called timing is different between 4.2-stable branch (only once) and master (trunk) (every load/reload), and I think that same called timing is preferable (especially when supporting both Redmine 4.2 and 5.0 in the plugin).

2022-04-04: Added "Rails.application.config.after_initialize" in above table

#12 - 2022-01-06 10:14 - Ko Nagase

"after_plugins_loaded" hook called timing is different between 4.2-stable branch (only once) and master (trunk) (every load/reload), and I think that same called timing is preferable (especially when supporting both Redmine 4.2 and 5.0 in the plugin).

Well, about this, just separating the event handler to "Rails.application.config.after_initialize" seems to be enough.

```
--- a/lib/redmine/plugin_loader.rb
+++ b/lib/redmine/plugin_loader.rb
@@ -106,7 +106,9 @@ module Redmine

  Rails.application.config.to_prepare do
    PluginLoader.directories.each(&:run_initializer)
  + end

  + Rails.application.config.after_initialize do
    Redmine::Hook.call_hook :after_plugins_loaded
  end
end
```

```
}}
```

Also, in current master (trunk), "Rails.application.config.after_initialize" event handler (in the plugin's "init.rb") seems to be called at every load/reload timing without duplicate call, so this can be also used instead of other ".to_prepare" functions of Redmine <= 4.2. (But I am not sure whether this is expected result...){collapse

```
+ Rails.application.config.after_initialize do
+ puts 'MyPlugin - Rails.application.config.after_initialize'
+ end
```

```
}}
```

Here is the combination result from above diffs.{collapse

```
# Init server
MyPlugin - init.rb
MyPlugin - Redmine::Plugin.register
MyPlugin - after_plugins_loaded hook
MyPlugin - Rails.application.config.after_initialize

# Reload 1st after editing "plugins/my_plugin/config/locales/en.yml"
MyPlugin - init.rb
MyPlugin - Redmine::Plugin.register
MyPlugin - Rails.application.config.after_initialize
MyPlugin - ActiveSupport::Reloader.to_prepare
MyPlugin - Rails.application.reloader.to_prepare

# Reload 2nd after editing "plugins/my_plugin/config/locales/en.yml", again
MyPlugin - init.rb
MyPlugin - Redmine::Plugin.register
MyPlugin - Rails.application.config.after_initialize
MyPlugin - ActiveSupport::Reloader.to_prepare
MyPlugin - Rails.application.reloader.to_prepare
MyPlugin - ActiveSupport::Reloader.to_prepare
MyPlugin - Rails.application.reloader.to_prepare
```

```
}}
```

FYI

#13 - 2022-01-06 13:20 - Ko Nagase

"ActiveSupport::Reloader.to_prepare" and "Rails.application.reloader.to_prepare" seem to be called multiple times after 2nd reloading, and I think that this behavior needs to be fixed.

About this, just adding once execution guard by class variable may be enough.{collapse

```
--- a/lib/redmine/plugin_loader.rb
+++ b/lib/redmine/plugin_loader.rb
@@ -86,6 +86,8 @@ module Redmine
   attr_accessor :directory
```

```

self.directory = Rails.root.join('plugins')

+ @@initialized = false
+
# Absolute path to the public directory where plugins assets are copied
catrr_accessor :public_directory
self.public_directory = Rails.root.join('public/plugin_assets')
@@ -105,9 +107,12 @@ module Redmine
  add_autoload_paths

  Rails.application.config.to_prepare do
-   PluginLoader.directories.each(&:run_initializer)
+   if !@@initialized
+     PluginLoader.directories.each(&:run_initializer)

-   Redmine::Hook.call_hook :after_plugins_loaded
+   Redmine::Hook.call_hook :after_plugins_loaded
+   @@initialized = true
+   end
  end
end
end

}}

```

With this change, the result becomes as follows, and I think that this is the most similar with past (Redmine <= 4.2) sequence.{{collapse

```

# Init server
MyPlugin - init.rb
MyPlugin - Redmine::Plugin.register
MyPlugin - after_plugins_loaded hook
MyPlugin - Rails.application.config.after_initialize

# Reload 1st after editing "plugins/my_plugin/config/locales/en.yml"
MyPlugin - ActiveSupport::Reloader.to_prepare
MyPlugin - Rails.application.reloader.to_prepare

# Reload 2nd after editing "plugins/my_plugin/config/locales/en.yml", again
MyPlugin - ActiveSupport::Reloader.to_prepare
MyPlugin - Rails.application.reloader.to_prepare

}}

```

#14 - 2022-01-06 16:54 - Ko Nagase

Sorry, above note-13 comment seemed to be completely wrong...

Now, I am using "Rails.application.config.after_initialize" with current master (trunk) branch, and it seems to be no problem.

https://github.com/gtt-project/redmine_custom_fields_groups/pull/14