Redmine - Defect #10813
Rails 3. Redmine plugins are not engines anymore, are they?

2012-05-02 10:47 - John Yani

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
Resolution:	Invalid	**Affected version:**	

Description

Seems like Redmine maintainers have decided not to use rails engines mechanism. Is it true?
Or it just require a little fix to make redmine plugin a Rails 3 Engine?

History

#1 - 2012-05-02 18:25 - Jean-Philippe Lang

- Status changed from New to Closed

- Resolution set to Invalid

John Yani wrote:

> *Seems like Redmine maintainers have decided not to use rails engines mechanism. Is it true?*

Yes, Redmine plugins do not rely on Rails engines.

#2 - 2012-05-02 19:21 - John Yani

Is it a temporary solution?
Do you have any statistics on which Redmine plugins rely on Rails Engines?
What's the point of making Redmine plugins independent from Rails Engines?
Plugin maintainer will still be required to make changes to support Rails 3.

Jean-Philippe Lang wrote:

> *John Yani wrote:*
>
> > *When plugins were in /vendor/plugins things were so much easier. Why do you think that just changing plugins directory will help plugin maintainers to upgrade their plugins?*
>
> *It's not supposed to make the upgrade process easier but they were moved to /plugins for these 2 reasons: - having Rails plugins used by the core and Redmine plugins mixed in the same directory was a mess. Some people were just copying the entire old content from vendor/plugins when upgrading*
>
> *- having plugins in vendor/plugins is deprecated in Rails 3.2 and will be removed in a future Rails release*

Ok, so these changes were for the future Rails 4.0 release.
Now plugin maintainers have 3 options:

- make their plugin independent from Rails Engine and ask users to put their plugin to the plugins/ directory
- ask users to put their plugin to the lib/ directory

- make their plugin a Railtie and ask users to change their Gemfile

All options require making Rails 3 compatibility changes.

First option require more work for both Redmine and Redmine plugins maintainers. However it has an advantage, like when Rails X will be released, Redmine won't have to be changed to support plugins. But copying rails engine loading code seems like a duplication to me.

#3 - 2012-05-02 21:12 - Jean-Philippe Lang

John Yani wrote:

> *Is it a temporary solution?*

No it's not.

> *Do you have any statistics on which Redmine plugins rely on Rails Engines?*

I don't know any plugin that **depends** on Rails::Engine but I don't know all plugins. So few I guess.

> *What's the point of making Redmine plugins independent from Rails Engines?*

To make 1.x plugins easier to upgrade. Redmine 2.0 offers pretty much the same features for plugins as Redmine 1.x, models/controllers/views are loaded just like before, custom routes, locales, tasks, assets and migrations are handled. And Redmine::Plugin API did not change.

> *Plugin maintainer will still be required to make changes to support Rails 3.*

Of course, code that runs with Rails 3 has to be Rails 3 compatible.

> *Ok, so these changes were for the future Rails 4.0 release.*

...and for reason 1.

> *Now plugin maintainers have 3 options:*
> - *make their plugin independent from Rails Engine and ask users to put their plugin to the plugins/ directory*
> - *ask users to put their plugin to the lib/ directory*
> - *make their plugin a Railtie and ask users to change their Gemfile*
>
> *All options require making Rails 3 compatibility changes.*

Option 2 is irrelevant. And again, I can't see how Redmine could run with a plugin whose code is not Rails 3 compatible.

> *First option require more work for both Redmine and Redmine plugins maintainers. However it has an advantage, like when Rails X will be released, Redmine won't have to be changed to support plugins. But copying rails engine loading code seems like a duplication to me.*

I disagree, option 1 does not require more work than making an existing plugin a Railtie. I have ported a few plugins myself and it was pretty

straightforward. Rewriting routes with the new Rails3 route syntax was most of the work. You can have a look at the sample plugin in the repository (source:trunk/extra/sample_plugin), it was ported to Redmine 2.0 without many changes.

Of course, if the internal of the plugin uses a lot of the Rails2 API that is not compatible with Rails3, a bit more work will be required. But I can't see how we can prevent that.