Redmine - Defect #11035

Doesn't patch "User" model in a plugin

2012-05-29 04:03 - Vladimir Pitin

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	Plugin API	**Estimated time:**	0.00 hour
Target version:			
Resolution:		**Affected version:**	2.0.0

Description

Sorry for my English :)

I have a plugin that works in "Redmine 1.3.0.stable.24228 (MySQL)"
This plugin extend a model "User". And it worked fine.

Now we have installed Redmine 2.0.0
I try rewrite my plugin, but extending model "User" call error "Expected /usr/share/srv-redmine/redmine-2.0/app/models/user.rb to define User"

This init.rb code:

```
Redmine::Plugin.register :advanced_roadmap do
  name 'Advanced Roadmap plugin'
  author 'Author name'
  description 'This is a plugin for Redmine'
  version '0.0.1'
  url 'http://example.com/path/to/plugin'
  author_url 'http://example.com/about'
end

#require 'user'

ActionDispatch::Callbacks.to_prepare do
    User.send(:include, AdvancedRoadmap::UserPatch)
end
```

If I add

```
require 'user'
```

I have error:

```
ActionView::Template::Error (undefined method `association_class' for nil:NilClass):
    40:         <%= hidden_field_tag(controller.default_search_scope, 1, :id => nil) if controller
.default_search_scope %>
    41:         <label for='q'>
    42:           <%= link_to l(:label_search), {:controller => 'search', :action => 'index', :id
=> @project}, :accesskey => accesskey(:search) %>:
    43:         </label>
    44:         <%= text_field_tag 'q', @question, :size => 20, :class => 'small', :accesskey => a
ccesskey(:quick_search) %>
    45:         <% end %>
    46:         <%= render_project_jump_box %>
  app/helpers/application_helper.rb:240:in `render_project_jump_box'
  app/views/layouts/base.html.erb:43:in `_app_views_layouts_base_html_erb___374889370_87506190'
  app/controllers/projects_controller.rb:168:in `show'
```

This's content of file "lib/advanced_roadmap/user_patch.rb"

```
require_dependency "user"
```

```
module AdvancedRoadmap
  module UserPatch
    def self.included(base)
        base.extend(ClassMethods)
        base.send(:include, InstanceMethods)

      base.class_eval do
      end

    end

    module ClassMethods
    end

    module InstanceMethods
    end

  end
end
```

It's my settings

```
root@redmine:/usr/share/srv-redmine/redmine-2.0/log# RAILS_ENV=development rake about
(in /usr/share/srv-redmine/redmine-2.0)
About your application's environment
Ruby version              1.9.3 (i686-linux)
RubyGems version          1.8.24
Rack version              1.4
Rails version             3.2.3
Active Record version     3.2.3
Action Pack version       3.2.3
Active Resource version   3.2.3
Action Mailer version     3.2.3
Active Support version    3.2.3
Middleware                ActionDispatch::Static, Rack::Lock, #<ActiveSupport::Cache::Strategy::Lo
calCache::Middleware:0x9b232d0>, Rack::Runtime, Rack::MethodOverride, ActionDispatch::RequestId, R
ails::Rack::Logger, ActionDispatch::ShowExceptions, ActionDispatch::DebugExceptions, ActionDispatc
h::RemoteIp, ActionDispatch::Reloader, ActionDispatch::Callbacks, ActiveRecord::ConnectionAdapters
::ConnectionManagement, ActiveRecord::QueryCache, ActionDispatch::Cookies, ActionDispatch::Session
::CookieStore, ActionDispatch::Flash, ActionDispatch::ParamsParser, ActionDispatch::Head, Rack::Co
nditionalGet, Rack::ETag, ActionDispatch::BestStandardsSupport, OpenIdAuthentication
Application root          /usr/share/srv-redmine/redmine-2.0
Environment               development
Database adapter          mysql2
Database schema version   20120422150750
```

Also I use: RVM (version 3.*), OS - Ubuntu 10.10

Any help is appreciated.

History

#1 - 2012-05-29 06:03 - Alex Shulgin

> #require 'user'
>
> ActionDispatch::Callbacks.to_prepare do
> User.send(:include, AdvancedRoadmap::UserPatch)
> end

Hm, did you try this instead:

```
require 'dispatcher'

Dispatcher.to_prepare do
```

```
  require_dependency 'user'
  User.send(:include, AdvancedRoadmap::UserPatch)
end
```

That seems pretty idiomatic to me.

#2 - 2012-05-29 06:11 - Vladimir Pitin

If i do this

```
require 'dispatcher'

Dispatcher.to_prepare do
  require_dependency 'user'
  User.send(:include, AdvancedRoadmap::UserPatch)
end
```

I cause error

```
cannot load such file -- dispatcher
```

#3 - 2012-05-29 09:49 - Etienne Massip

First code seemed OK, related to this rails issue if it is an issue; try to clear your caches as suggested?

#4 - 2012-05-29 09:49 - Etienne Massip

- Category changed from Plugin Request to Plugin API

- Affected version (unused) set to 2.0.0

- Affected version set to 2.0.0

#5 - 2012-05-29 10:08 - Fabrice ROBIN

Vladimir Pitin wrote:

> If i do this
>
> [...]
>
> I cause error
>
> [...]

In fact, 'dispatcher' does not exist anymore in rails 3.
Redmine documentation for plugin development needs to be updated.
Try this instead:

```
Rails.configuration.to_prepare do
  require_dependency 'user'
  User.send(:include, AdvancedRoadmap::UserPatch)
end
```

#6 - 2012-05-29 10:42 - Etienne Massip

Fabrice ROBIN wrote:

> Redmine documentation for plugin development needs to be updated.

I don't think that Redmine documentation actually refers to the use of dispatcher?

#7 - 2012-05-29 11:22 - Fabrice ROBIN

Exact, I was convinced that was the case.

But it might be a good idea to add something in redmine wiki.
The to_prepare method is a must have when a plugin needs to patch redmine core,
mainly in development mode.

Here is the documentation about the to_prepare method

```
Add a preparation callback. Preparation callbacks are run before every
```

```
request in development mode, and before the first request in production
mode
```

#8 - 2012-05-29 11:31 - Etienne Massip

This is described in RoR guides, mainly in http://guides.rubyonrails.org/configuring.html#initializers.

Vladimir is right when he replaces the dispatcher requirement with ActionDispatch::Callbacks.to_prepare, see http://guides.rubyonrails.org/configuring.html#configuring-action-dispatch , this is the most accurate match.

#9 - 2012-05-30 00:47 - Vladimir Pitin

If I try it

```
Rails.configuration.to_prepare do
  require_dependency 'user'
  User.send(:include, AdvancedRoadmap::UserPatch)
end
```

I have same error

```
Expected /usr/share/srv-redmine/redmine-2.0/app/models/user.rb to define User
```

#10 - 2012-05-30 00:56 - Vladimir Pitin

> First code seemed OK, related to this rails issue if it is an issue; try to clear your caches as suggested?

No.How can I clear my cashes?

#11 - 2012-05-30 01:01 - Vladimir Pitin

My folder /tmp/cache is empty. If it's meant?

#12 - 2012-05-30 01:37 - Vladimir Pitin

Upgrade to Redmine 2.0.1 had no affect
Commands:

```
rake tmp:cache:clear
rake tmp:sessions:clear
```

had no affect, too

#13 - 2012-05-30 08:33 - Daniel Munn

This works:
lib/redmine_test/patches/user_patch.rb

```
require_dependency 'principal'
require_dependency 'user'
module RedmineTest
  module Patches
    module UserPatch
      def self.included(base)
        base.send(:extend, ClassMethods)
        base.send(:include, InstanceMethods)
        base.class_eval do
          unloadable
        end
      end

      module ClassMethods
        def echoTest
          STDOUT.print "echo Test\n"
          STDOUT.flush
          nil
        end

      end

      module InstanceMethods
```

```
      end
    end
  end
end

ActionDispatch::Callbacks.to_prepare do
  User.send(:include, RedmineTest::Patches::UserPatch)
end
```

After which the User.echoTest method exists - you need to add require_dependency 'principal' before require_dependency 'user'

#14 - 2012-05-30 09:29 - Vladimir Pitin

init.rb

```
require 'redmine'
require 'advanced_roadmap/user_patch'
Rails.logger.info 'Starting Advanced Roadmap plugin for Redmine'

Redmine::Plugin.register :advanced_roadmap do
  name 'Advanced Roadmap plugin'
  author 'Author name'
  description 'This is a plugin for Redmine'
  version '0.0.1'
  url 'http://example.com/path/to/plugin'
  author_url 'http://example.com/about'
end
```

user_patch.rb

```
#require_dependency "user"

require_dependency 'principal'
require_dependency 'user'
module AdvancedRoadmap
    module UserPatch
      def self.included(base)
        base.send(:extend, ClassMethods)
        base.send(:include, InstanceMethods)
        base.class_eval do
          unloadable
        end
      end

      module ClassMethods
        def echoTest
          STDOUT.print "echo Test\n"
          STDOUT.flush
          nil
        end

      end

      module InstanceMethods
      end
    end
end

ActionDispatch::Callbacks.to_prepare do
  User.send(:include, AdvancedRoadmap::UserPatch)
end
```

It's not work. error's is same

```
ActionView::Template::Error (undefined method `association_class' for nil:NilClass):
    40:         <%= hidden_field_tag(controller.default_search_scope, 1, :id => nil) if controller.default_sea
rch_scope %>
    41:         <label for='q'>
    42:           <%= link_to l(:label_search), {:controller => 'search', :action => 'index', :id => @project}
, :accesskey => accesskey(:search) %>:
    43:         </label>
    44:         <%= text_field_tag 'q', @question, :size => 20, :class => 'small', :accesskey => accesskey(:qu
ick_search) %>
    45:         <% end %>
```

```
    46:         <%= render_project_jump_box %>
  app/helpers/application_helper.rb:240:in `render_project_jump_box'
  app/views/layouts/base.html.erb:43:in `_app_views_layouts_base_html_erb__947349986_81730330'
```

#15 - 2012-05-30 19:08 - Daniel Munn

Hi, looks like an un-resolved dependency on the project model (via member model):

```
require_dependency 'project'
require_dependency 'principal'
require_dependency 'user'
```

Seems to allow standard output, please confirm.

#16 - 2012-05-31 00:17 - Vladimir Pitin

It seems that it works fine. I will test it in more detail. In any case, this variant doesn't cause error.
Thanks!

#17 - 2012-05-31 13:59 - Oleg Kandaurov

Daniel, Thanks a lot. You soulution works for me.

#18 - 2012-06-10 21:49 - Jean-Baptiste Barth

I confirm it works too, but it feels like black magic to me. If anybody knows why we should declare explicitly those dependencies manually, it would be great.

#19 - 2012-06-13 02:20 - Vladimir Pitin

Jean-Baptiste Barth wrote:

> I confirm it works too, but it feels like black magic to me. If anybody knows why we should declare explicitly those dependencies manually, it would be great.

+1

#20 - 2012-07-27 23:58 - John Kubiatowicz

Is there a way to have something work with both Rails 2 and 3? I tried the ActionDispatch::Callbacks idiom in a Rails 2.3.14 installation and it doesn't seem to work.

What does seem to work in 2.3.14 is to use

```
config.to_prepare do
    Monkey Patch things
end
```

Will this work in 3.0+? I'm just wondering if this is an idiom that might be usable in general (sorry if this is a dumb question -- I am bit hazy on the startup internals of Rails and plugins).

#21 - 2012-07-28 01:34 - John Kubiatowicz

Actually, in fact, what about avoiding the callback entirely? If you do a:

```
require_dependency 'my_patch_file'
```

in the init.rb file and then leave your User.send by itself at the end of the patch file, shouldn't this do the right thing (i.e. loaded once in production mode and every requires in development mode)? It certainly seems to work in production mode....

Forgive me if this sounds clueless, just trying to understand options here...

#22 - 2012-11-07 01:43 - Vladimir Pitin

For information

If I write

```
require_dependency 'project'
require_dependency 'principal'
require_dependency 'user'
```

in my plugin, the plugin

http://www.redmine.org/plugins/extended_fields
stops working, because validation

```
validates_inclusion_of :field_format, :in => Redmine::CustomFieldFormat.available_formats
```

doesn't pass.

#23 - 2013-03-20 17:50 - Anonymous

It seems to me that the issue reported here has been overcome already; the remainder of discussion seems to resolve about a debate how to implement Monkey Patching in a way that work on both Rails 2 and 3 (although I don't understand why you would bother, RoR 3 is the future, RoR 4 is coming soon), and how to do it most elegantly etc.

So it appears this issue can be closed, can't it?

#24 - 2013-05-02 00:58 - Jean-Baptiste Barth

Max: not really, the underlying issue (why we need to declare dependencies explicitly) isn't resolved

John: yes, you should use config.to_prepare { <block> } in Redmine 1.x/Rails 2.x, and ActionDispatch::Callbacks.to_prepare { <block> } above. Note that since we started this thread, support for Redmine 1.x has been removed, and only Redmine 2.3.x is supported at the moment (hence Rails 3.2+).

Vladimir: that's unexpected, it should work