redmineorg-copy202205 - Vote #69245

Generated test instances may share the same attribute value object

2022/05/09 15:22 - Admin Redmine

ステータス:	Closed	**開始日:**	2011/02/12
優先度:	高め	**期日:**	
担当者:		**進捗率:**	0%
カテゴリ:	Code cleanup/refactoring_30	**予定工数:**	0.00時間
対象バージョン:	1.2.2_38	**作業時間:**	0.00時間
Redmineorg_URL:	https://www.redmine.org/issues/7613	**status_id:**	5
category_id:	30	**tracker_id:**	1
version_id:	38	**plus1:**	0
issue_org_id:	7613	**affected_version:**	
author_id:	6508	**closed_on:**	
assigned_to_id:	0	**affected_version_id:**	
comments:	17		

説明

Actually, this test code will fail :

```
p1 = Project.generate!
p2 = Project.generate!

assert_not_equal p1.identifier, p2.identifier
```

That's because ObjectDaddy Project identifier generator actually returns the same object each time.

Thus, in source:trunk/test/exemplars/project_exemplar.rb :

```
    # Project#next_identifier is defined on Redmine
    def self.next_identifier_from_object_daddy
        @last_identifier ||= 'project-0000'
        @last_identifier.succ!
        @last_identifier
    end
```

Should be :

```
    # Project#next_identifier is defined on Redmine
    def self.next_identifier_from_object_daddy(last_identifier)
        last_identifier ||= 'project-0000'
        last_identifier.succ
    end
```

I'll post a patch soon.

journals

Here comes the patch.

All tests pass.

This issue was blocking me writting tests for #7456.

Sorry I don't really understand how your patched version works. Does ObjectDaddy pass the last_ as an argument implictly ?

But I see the problem in actual version. Why not a @@@last_identifier@ instead of @@last_identifier@ ? I think the problem is here since it's a class method, an instance variable doesn't make much sense. And it should solve your first problem.

What do you think ?

Jean-Baptiste Barth wrote:

> But I see the problem in actual version. Why not a @@@last_identifier@ instead of @@last_identifier@ ? I think the problem is here since it's a class method, an instance variable doesn't make much sense. And it should solve your first problem.

Nope, I've been a bit too much sybilline in my description :

The problem is that the generator returns the same object instance each time (@@last_name@), and that this instance is set by ObjectDaddy as the attribute value of the newly spawned Project instance.

That is to say :

```
p1.identifier.object_id == p2.identifier.object_id == @last_name.object_id
```

So, doing a @@last_name.succ!@ when generating p2 actually also change the p1.identifier value, which is not the desired effect.

If @@last_name.succ!@ was replaced by @@last_name = @last_name.succ@, the test would pass, but using a class variable would not change anything.

> Does ObjectDaddy pass the last_ as an argument implictly ?

Absolutly, it passes the previous value if generator method/block arity is 1 ; using this, IMHO, is the most elegant way to fix this issue.

Okay, I'll have a deeper look at it, thanks for the infos.

I confirm the problem in current implementation but according to object_daddy documentation, there's a cleaner way to declare generators. The following patch fixes the problem for project generators:

```
Index: test/exemplars/project_exemplar.rb
===================================================================
--- test/exemplars/project_exemplar.rb (revision 4892)
+++ test/exemplars/project_exemplar.rb (working copy)
@@ -1,22 +1,9 @@
 class Project < ActiveRecord::Base
-  generator_for :name, :method => :next_name
-  generator_for :identifier, :method => :next_identifier_from_object_daddy
+  generator_for :name, :start => 'Project 0'
+  generator_for :identifier, :start => 'project-0000'
   generator_for :enabled_modules, :method => :all_modules
   generator_for :trackers, :method => :next_tracker

-  def self.next_name
-    @last_name ||= 'Project 0'
-    @last_name.succ!
-    @last_name
-  end
-
-  # Project#next_identifier is defined on Redmine
-  def self.next_identifier_from_object_daddy
-    @last_identifier ||= 'project-0000'
-    @last_identifier.succ!
-    @last_identifier
-  end
-
   def self.all_modules
     [].tap do |modules|
       Redmine::AccessControl.available_project_modules.each do |name|
```

Yes, saw that too.

I was not at ease with so much change and I thought Eric was not ignorant of this way to do when he committed the exemplars in the first place, and that he chose to write full generator methods on purpose.

But really, no idea why he didn't do that.

Also, there is possibility that the @Issue.generate_for_project!()@ method in source:trunk/test/object_daddy_helpers.rb#L30 could have been written to supersede this issue, as it is called many times in a row in @gantt_test.rb@ and that @generate!()@ already deals with a block argument.

I don't think so. I think the problem hasn't been noticed before.

Anyway, I don't see any reason not to clean up these generators.

I do agree, the more clean, the better !

Still needs merging.

履歴

#1 - 2022/05/10 17:19 - Admin Redmine

- カテゴリ を Code cleanup/refactoring_30 にセット

- 対象バージョン を 1.2.2_38 にセット