Redmine - Defect #15551

Validating a Setting with invalid name triggers an error

2013-11-27 00:28 - Guido Heymann

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Jean-Philippe Lang	**% Done:**	0%
Category:	Code cleanup/refactoring	**Estimated time:**	0.00 hour
Target version:	2.5.0		
Resolution:	Fixed	**Affected version:**	2.4.0

Description

In case you want directly add a setting by insert through the model you get an error

insert the following test case into setting_test.rb

```
def test_insert_name_not_in_settings_yml
 setting  = Setting.new(:name => "does_not_exist", :value => "should_not_be_allowed" )
 assert !setting.save
end
```

Then you get the errors below:

```
test_insert_name_not_in_settings_yml(SettingTest):
NoMethodError: undefined method `[]' for nil:NilClass
    /redmine/app/models/setting.rb:86:in `block in <class:Setting>'
    /usr/lib64/ruby/gems/1.9.1/gems/activesupport-3.2.15/lib/active_support/callbacks.rb:425:in `_
run__2686202268941026958__validate__1184212570148052443__callbacks'
    /usr/lib64/ruby/gems/1.9.1/gems/activesupport-3.2.15/lib/active_support/callbacks.rb:405:in `_
_run_callback'
    /usr/lib64/ruby/gems/1.9.1/gems/activesupport-3.2.15/lib/active_support/callbacks.rb:385:in `_
run_validate_callbacks'
    /usr/lib64/ruby/gems/1.9.1/gems/activesupport-3.2.15/lib/active_support/callbacks.rb:81:in `ru
n_callbacks'
    /usr/lib64/ruby/gems/1.9.1/gems/activemodel-3.2.15/lib/active_model/validations.rb:228:in `run
_validations!'
    /usr/lib64/ruby/gems/1.9.1/gems/activemodel-3.2.15/lib/active_model/validations/callbacks.rb:5
3:in `block in run_validations!'
    /usr/lib64/ruby/gems/1.9.1/gems/activesupport-3.2.15/lib/active_support/callbacks.rb:403:in `_
run__2686202268941026958__validation__1184212570148052443__callbacks'
    /usr/lib64/ruby/gems/1.9.1/gems/activesupport-3.2.15/lib/active_support/callbacks.rb:405:in `_
_run_callback'
    /usr/lib64/ruby/gems/1.9.1/gems/activesupport-3.2.15/lib/active_support/callbacks.rb:385:in `_
run_validation_callbacks'
    /usr/lib64/ruby/gems/1.9.1/gems/activesupport-3.2.15/lib/active_support/callbacks.rb:81:in `ru
n_callbacks'
    /usr/lib64/ruby/gems/1.9.1/gems/activemodel-3.2.15/lib/active_model/validations/callbacks.rb:5
3:in `run_validations!'
    /usr/lib64/ruby/gems/1.9.1/gems/activemodel-3.2.15/lib/active_model/validations.rb:195:in `val
id?'
    /usr/lib64/ruby/gems/1.9.1/gems/activerecord-3.2.15/lib/active_record/validations.rb:69:in `va
lid?'
    /usr/lib64/ruby/gems/1.9.1/gems/activerecord-3.2.15/lib/active_record/validations.rb:77:in `pe
rform_validations'
    /usr/lib64/ruby/gems/1.9.1/gems/activerecord-3.2.15/lib/active_record/validations.rb:50:in `sa
ve'
    /usr/lib64/ruby/gems/1.9.1/gems/activerecord-3.2.15/lib/active_record/attribute_methods/dirty.
rb:22:in `save'
    /usr/lib64/ruby/gems/1.9.1/gems/activerecord-3.2.15/lib/active_record/transactions.rb:259:in `
block (2 levels) in save'
    /usr/lib64/ruby/gems/1.9.1/gems/activerecord-3.2.15/lib/active_record/transactions.rb:313:in `
block in with_transaction_returning_status'
```

```
    /usr/lib64/ruby/gems/1.9.1/gems/activerecord-3.2.15/lib/active_record/connection_adapters/abst
ract/database_statements.rb:192:in `transaction'
    /usr/lib64/ruby/gems/1.9.1/gems/activerecord-3.2.15/lib/active_record/transactions.rb:208:in `
transaction'
    /usr/lib64/ruby/gems/1.9.1/gems/activerecord-3.2.15/lib/active_record/transactions.rb:311:in `
with_transaction_returning_status'
    /usr/lib64/ruby/gems/1.9.1/gems/activerecord-3.2.15/lib/active_record/transactions.rb:259:in `
block in save'
    /usr/lib64/ruby/gems/1.9.1/gems/activerecord-3.2.15/lib/active_record/transactions.rb:270:in `
rollback_active_record_state!'
    /usr/lib64/ruby/gems/1.9.1/gems/activerecord-3.2.15/lib/active_record/transactions.rb:258:in `
save'
    test/unit/setting_test.rb:93:in `test_insert_name_not_in_settings_yml'
    /usr/lib64/ruby/gems/1.9.1/gems/mocha-0.14.0/lib/mocha/integration/mini_test/version_230_to_21
01.rb:36:in `run'
```

Same if you call the following on console

```
s = Setting.new(:name => "does_not_exist", :value => "should_not_be_allowed" )
s.valid?
```

The reason is in the app/model/setting.rb

```
validates_numericality_of :value, :only_integer => true, :if => Proc.new { |setting| @@available_s
ettings[setting.name]['format'] == 'int' }
```

The Proc breaks if name (the key) is not in the hash.

Fix for it is to check if key is in hash inside the proc before using the non existing key so that check of numericality is skipped when key does not exist:

```
validates_numericality_of :value, :only_integer => true, :if => Proc.new { |setting| @@available_s
ettings.has_key?(setting) ? @@available_settings[setting.name]['format'] == 'int' : false}
```

Associated revisions

Revision 12348 - 2013-11-29 22:18 - Jean-Philippe Lang

Fixed that validating a Setting with invalid name triggers an error (#15551).

History

#1 - 2013-11-27 08:18 - Jean-Philippe Lang

Yes, the fix is trivial. Could you attach a patch with the fix and the test ?

#2 - 2013-11-27 22:29 - Guido Heymann

- File Defect15551.diff added

Sorry that I forgot that, Jean-Philippe:
Attached you can find the patch.

#3 - 2013-11-29 00:33 - Jean-Philippe Lang

- Category changed from Project settings to Code cleanup/refactoring

- Status changed from New to Confirmed

- Assignee set to Jean-Philippe Lang

- Target version set to 2.5.0

#4 - 2013-11-29 22:20 - Jean-Philippe Lang

- Subject changed from Error on validation in Setting model to Validating a Setting with invalid name triggers an error

- Status changed from Confirmed to Closed

- Resolution set to Fixed

With this patch applied, integer settings are no longer validated:
@@available_settings.has_key?(setting) always returns false, should be @@available_settings.has_key?(setting.name) instead.

It's fixed with an additional test in r12348, thanks for pointing this out.

Files

Defect15551.diff	1.14 KB	2013-11-27	Guido Heymann