Redmine - Defect #25867

Assignable users should respect database collation

2017-05-16 22:10 - Pavel Rosický

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	Database	**Estimated time:**	0.00 hour
Target version:	Candidate for next major release		
Resolution:		**Affected version:**	

Description

mysql collation: utf8_general_ci
['a','u','č'] should be sorted as ['a','č','u'] but because of ruby sort it's reordered back as ['a','u','č']

```
  def assignable_users
    users = project.assignable_users(tracker).to_a
    users << author if author && author.active?
    if assigned_to_id_was.present? && assignee = Principal.find_by_id(assigned_to_id_was)
      users << assignee
    end

    users.uniq.sort
  end
```

I can provide a patch, are you interested or is it desired behaviour?

```
Environment (not important, all redmine versions and databases are affected):
  Redmine version                3.3.3.devel.16557
  Ruby version                   2.1.5-p273 (2014-11-13) [x64-mingw32]
  Rails version                  4.2.8
  Environment                    production
  Database adapter               Mysql2
SCM:
  Subversion                     1.9.5
  Git                            2.11.0
  Filesystem
Redmine plugins:
  no plugin installed
```

History

#1 - 2017-05-18 21:58 - Pavel Rosický

- File issue.rb.patch added

- File issue_test.rb.patch added

#2 - 2018-07-26 11:56 - Pavel Rosický

It's been a year and the problem is still reproducible.

#3 - 2018-12-02 04:55 - Go MAEDA

- Target version set to Candidate for next major release

#4 - 2022-12-08 09:51 - Go MAEDA

- File 25867.patch added

Updated the patch for the current trunk (r21987).

#5 - 2022-12-21 08:47 - Go MAEDA

- Target version changed from Candidate for next major release to 5.1.0

Setting the target version to 5.1.0.

#6 - 2022-12-21 15:21 - Holger Just

In the updated patch, you removed the line return @assignable_users unless @assignable_users.nil? from the original patch in #25867#note-1. Without this line, the caching of the result in @assignable_users becomes useless.

I'm actually unsure if it's worthwhile to introduce caching here at all. ~~I tend to say: we do not need it as the method does not appear to be regularly called multiple times per request. As such, I think, we can get rid of the caching and its associated possibility for inconsistencies.~~ *Turns out, it is called multiple times in the issues/_attributes.html.erb partial. Thus, we still might want caching...* In any case though, we should either remove the instance variable caching completely, or use it if present.

As a slight improvement, it might also be useful to also remove the to_a at the end and to return a query object. That way, callers might chain other query refinements to it without affecting the current use-case.

Finally, it might also be useful to extract the fetching of the (unsorted) user ids into a separate method, e.g. assignable_user_ids, which might make checks such as those in the Issue model to check if the assignee is allowed less expensive by avoiding the final fetch of the Principal objects. Only these ids might then possible be cached?

#7 - 2023-10-21 03:00 - Go MAEDA

- Target version changed from 5.1.0 to 6.0.0

#8 - 2024-10-28 21:50 - Marius BĂLTEANU

- Target version changed from 6.0.0 to Candidate for next major release

Files

issue_test.rb.patch	883 Bytes	2017-05-18	Pavel Rosický
issue.rb.patch	1.16 KB	2017-05-18	Pavel Rosický
25867.patch	2.01 KB	2022-12-08	Go MAEDA