Redmine - Defect #3722

Nested projects can get in disorder

2009-08-08 22:16 - Andreas Deininger

Status:	New	**Start date:**	2009-08-08
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	Projects	**Estimated time:**	0.00 hour
Target version:			
Resolution:		**Affected version:**	

Description

How to reproduce (using redmine SVN revision 2835):

- create project *Toplevel A*
- create project *Sublevel A* as child of project *Toplevel A*
- create project *Subsublevel A* as child of project *Sublevel A*
- create project *Toplevel B*
- copy project *Sublevel A*, name it *Toplevel C*. Leave the field *Subproject of* empty (that's also the default coming up)
- delete project *Sublevel A*
- now, delete project *Toplevel A*. You are prompted:

```
Toplevel A
Are you sure you want to delete this project and related data ?
Its subproject(s): Toplevel B, Toplevel C will be also deleted
```

This is definitely wrong, since Toplevel B was never a subproject of Toplevel A.

If I click *OK* project Toplevel B is **not** deleted, however, project *Toplevel C* now has a minus sign on its left, indicating that is has subprojects, though it has none.

Note:
My first encounter with that bug was when I wanted to delete all but one project from a set of deeply nested projects (~20 projects). I inadvertently deleted the project I wanted to keep. First I thought it was my fault, so I restored my data from a backup and tried again. Same effect, again the project was deleted. Then it became obvious to me that something must be wrong with the new functionality of nesting projects as deep as you want. I played around a bit and finally found the steps above that reproducibely lead to an inconsistent state of the nesting of the projects (as far as I can see).

Related issues:

Related to Redmine - Feature #18860: Replace awesome_nested_set gem with a cu...	**Closed**	

History

#1 - 2009-08-13 14:46 - Augusto Dias Pereira dos Santos

For those who had the nested set disordered in MySQL DataBase, I develop a procedure that recalculate the nested set base on the parent_id. It's not a solution, just a way to rearange the mess. It can be better.
Watch out to the max_sp_recursion_depth variable inside the procedure.
Here is:

```
-- Procedure BEGIN
DROP PROCEDURE `recalculateNetstedSet`//
CREATE PROCEDURE `recalculateNetstedSet`(parent INT, INOUT left_value INT)
BEGIN
  DECLARE done INT DEFAULT 0;
  DECLARE son_ID INTEGER;
  DECLARE sons CURSOR FOR SELECT id FROM projects WHERE parent_id IS NULL;
  DECLARE sons2 CURSOR FOR SELECT id FROM projects WHERE parent_id = parent;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
  set max_sp_recursion_depth=10;

  IF parent IS NOT NULL THEN
    UPDATE projects set lft = left_value where id = parent;
    SET left_value = left_value+1;
  END IF;
```

```
  OPEN sons;
  OPEN sons2;

  IF parent IS NULL THEN
    FETCH sons INTO son_ID;
    REPEAT
      IF son_ID IS NOT NULL THEN
        CALL recalculateNetstedSet(son_ID,left_value);
        SET left_value = left_value+1;
      END IF;
    FETCH sons INTO son_ID;
    UNTIL done END REPEAT;
  ELSE
    FETCH sons2 INTO son_ID;
    REPEAT
      IF son_ID IS NOT NULL THEN
        CALL recalculateNetstedSet(son_ID,left_value);
        SET left_value = left_value+1;
      END IF;
    FETCH sons2 INTO son_ID;
    UNTIL done END REPEAT;
  END IF;

  UPDATE projects set rgt = left_value where id = parent;

  CLOSE sons;
  CLOSE sons2;
END
-- Procedure END

To call the procedure:
set @parent = NULL;
set @left = 1;
CALL recalculateNetstedSet(@parent,@left);
```

#2 - 2009-08-13 14:53 - Augusto Dias Pereira dos Santos

I think the procedure got wrong for the wiki sintax. I will try again, just the procedure this time.
Sorry I should have seen the preview ;)

```
DROP PROCEDURE `recalculateNetstedSet`//
CREATE PROCEDURE `recalculateNetstedSet`(parent INT, INOUT left_value INT)
BEGIN
  DECLARE done INT DEFAULT 0;
  DECLARE son_ID INTEGER;
  DECLARE sons CURSOR FOR SELECT id FROM projects WHERE parent_id IS NULL;
  DECLARE sons2 CURSOR FOR SELECT id FROM projects WHERE parent_id = parent;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
  set max_sp_recursion_depth=10;

  IF parent IS NOT NULL THEN
    UPDATE projects set lft = left_value where id = parent;
    SET left_value = left_value+1;
  END IF;

  OPEN sons;
  OPEN sons2;

  IF parent IS NULL THEN
    FETCH sons INTO son_ID;
    REPEAT
      IF son_ID IS NOT NULL THEN
        CALL recalculateNetstedSet(son_ID,left_value);
        SET left_value = left_value+1;
      END IF;
    FETCH sons INTO son_ID;
    UNTIL done END REPEAT;
  ELSE
    FETCH sons2 INTO son_ID;
    REPEAT
      IF son_ID IS NOT NULL THEN
        CALL recalculateNetstedSet(son_ID,left_value);
        SET left_value = left_value+1;
      END IF;
    FETCH sons2 INTO son_ID;
```

```
    UNTIL done END REPEAT;
  END IF;

  UPDATE projects set rgt = left_value where id = parent;

  CLOSE sons;
  CLOSE sons2;
END
```

#3 - 2009-09-21 16:53 - Paul Groves

I too experienced this, I removed the "dependent_destroy" declaration from app/project.rb as I believe this is not applicable for the awesome_nested_set gem. This *seems* to have made things better.

app/project.rb Line 51

```
-- acts_as_nested_set :order => 'name', :dependent => :destroy
++   acts_as_nested_set :order => 'name'
```

Hope this helps,

Paul

#4 - 2010-01-14 11:43 - Chris Platts

Here's another version of the stored proc which tries to keep a sane alphabetical sort order on projects and subprojects. It also corrects the typo (recalculate **Netsted** Set) :)

```
CREATE DEFINER=`redmine`@`` PROCEDURE `recalculateNestedSet`(parent INT, INOUT left_value INT)
BEGIN
  DECLARE done INT DEFAULT 0;
  DECLARE son_ID INTEGER;
  DECLARE sons CURSOR FOR SELECT id FROM projects WHERE parent_id IS NULL ORDER BY name;
  DECLARE sons2 CURSOR FOR SELECT id FROM projects WHERE parent_id = parent ORDER BY name;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
  set max_sp_recursion_depth=10;
  IF parent IS NOT NULL THEN
    UPDATE projects set lft = left_value where id = parent;
    SET left_value = left_value+1;
  END IF;
  OPEN sons;
  OPEN sons2;
  IF parent IS NULL THEN
    FETCH sons INTO son_ID;
    REPEAT
      IF son_ID IS NOT NULL THEN
        CALL recalculateNestedSet(son_ID,left_value);
        SET left_value = left_value+1;
      END IF;
    FETCH sons INTO son_ID;
    UNTIL done END REPEAT;
  ELSE
    FETCH sons2 INTO son_ID;
    REPEAT
      IF son_ID IS NOT NULL THEN
        CALL recalculateNestedSet(son_ID,left_value);
        SET left_value = left_value+1;
      END IF;
    FETCH sons2 INTO son_ID;
    UNTIL done END REPEAT;
  END IF;
  UPDATE projects set rgt = left_value where id = parent;
  CLOSE sons;
  CLOSE sons2;
END
```

#5 - 2010-07-11 19:57 - Jean-Baptiste Barth

Why not: Project.update_all(:lft=>nil,:rgt=>nil); Project.rebuild! or something like that ?

Introducing stored procedure is limited to your own database system, while Redmine tries to support sqlite3, Postgresql and Mysql the same time.

#6 - 2011-06-08 20:01 - Sven Culley

This is what I did to get the order back to work:

1. Go to your database
2. Make a backup of your "projects" table
3. Set the value for "lft" and "rgt" to "0"
4. Go to you console and use "ruby script/runner -e production 'Project.rebuild!'" to rebuild the project order

In case anything goes wrong you still have the backup.

#7 - 2011-09-13 11:40 - provetza provetza

Hi all,

I did what Sven suggests (zero lft and rgt on projects and then Project.rebuild). Nested projects appear alphabetically inside toplevel projects, but toplevel projects are out of order. How can I make the toplevel projects appear in order?

Thanks!

#8 - 2012-05-23 06:09 - Toshiya TSURU

Sven Culley wrote:

> This is what I did to get the order back to work:
>
> 1. Go to your database
> 2. Make a backup of your "projects" table
> 3. Set the value for "lft" and "rgt" to "0"
> 4. Go to you console and use "ruby script/runner -e production 'Project.rebuild!'" to rebuild the project order
>
> In case anything goes wrong you still have the backup.

in redmine 2,

```
4. Go to you console and use "ruby script/runner -e production 'Project.rebuild!'" to rebuild the project orde
r
```

the command above doesn't work. maybe :

```
ruby script/rails runner -e production 'Project.rebuild!'
```

#9 - 2015-01-21 05:43 - Toshi MARUYAMA

- Related to Feature #18860: Replace awesome_nested_set gem with a custom implementation of nested sets added