redmineorg-copy202205 - Vote #69242

Patch providing issue journal rollback

2022/05/09 15:22 - Admin Redmine

ステータス:	New	**開始日:**	2011/02/11
優先度:	通常	**期日:**	
担当者:		**進捗率:**	100%
カテゴリ:	Issues_2	**予定工数:**	0.00時間
対象バージョン:		**作業時間:**	0.00時間
Redmineorg_URL:	https://www.redmine.org/issues/7610	**status_id:**	1
category_id:	2	**tracker_id:**	3
version_id:	0	**plus1:**	1
issue_org_id:	7610	**affected_version:**	
author_id:	26118	**closed_on:**	
assigned_to_id:	0	**affected_version_id:**	
comments:	33		

説明

Has someone ever made a mistake that left a journal turd behind? Have you ever wished that you could just get rid of the journal turds?

This patch provides a rollback icon next to the number of the last journal entry. When you click on the rollback icon, the journal entry is destroyed, and the issue is restored to the state it was in before the entry was created.

The rollback action is guarded by a new permission, 'Rollback issues'.

Note that there is no validation check when rolling back an journal entry. So, for example, if 'Target Version' is changed, and the old target version is deleted, when that journal entry is rolled back, the 'Target Version' then becomes nil. Someone can probably come up with a better icon, but I just re-used the 'cancel' icon. Also, rollback cannot recover the deletion of an attachment. Rollback on an attachment deletion journal will simply remove the journal entry (and rollback any accompanying changes).

Use the rollback action carefully!

journals

related to #1725

related to #1665

todo: provide seperate permission to rollback own issue updates

h3. Improved1

- moved code from issue controller to journal controller
- improved error handling
- improved permissions
- improved code quality

h3. Permissions

- Users with 'edit issue notes' permission and 'rollback issue notes' permission can rollback anyone's journal entry.
- Users with 'edit own issue notes' permission and 'rollback issue notes' permission can rollback their own journal entry.
- Attempting to rollback any journal but the last journal will result in a 'notice_locking_conflict' failure.

* Attempting to rollback a journal without permission will result in a 403 error page.

Note that there is no validation check when rolling back an journal entry.

Isn't it very dangerous?

Looks promising, but do we know all the edge cases and security related issues it could introduce ? Could you easily turn this into a plugin (didn't look at the code) ?

Ivan:
Yes, I'm sure there are still things to work out. The idea is that you don't really want rollback to fail. In the case of an invalidation, it's better to just remove the journal entry and move forward. The tricky part is that some rollbacks should succeed, and others that cause invalidation should just fail. There's no easy way to do this at the moment, but I'll get to it when I have time. The disclaimer I give at the bottom does say 'Use the rollback action carefully!'.

Jean-Baptiste:
I don't have the desire nor the time to learn how to do plugins. I don't see any security related issues. It uses permissions which I assume are well-tested.

Edge cases that will likely cause problems are rollbacks that occur on properties that have values that may no longer be valid. Cases I can think of:

- a parent issue moved to another project
  ** not a high risk: will create confusing heirarchies and pull issues into other projects, but probability is low (rare use case)
- a tracker disabled for a project
  ** not a high risk: will cause a problem with overviews and queries(?), but probability is low (rare use case)
- a category/version/other property value deleted
  ** not a problem: deleted property values have ids that are not reused, the property will read '-', which I suppose is expected
- permissions removed for a user
  ** not a problem: the rollback permission should also be removed

I suppose a proper solution for this is to use the validate command before saving. On validation failure, scan the errors for property ids and set the failed properties to their old values. This preserves rollback for properties that can be rolled back, while maintaining the current values for properties that cannot be rolled back. It would also be nice to modify the errors to say 'not rolled back', to indicate which properties rollback failed for. However, I'd like to get more feedback from the community before going down this path.

I suppose there could be potential security issue when it comes to 'View own issues' permission that is in the works. This, in combination with the assignee field can allow someone who is no longer an assignee view a changed description (since descriptions are not journal'd).

Possible workflow:

- issue assigned to assignee1
- issue assigned to assignee2
- description updated with secure information only supposed to be viewed by assignee2 (and others with 'view all issues' permission
- rollback occurs, assigning issue back to assignee1 (description is not rolled back, since it is not journal'd)
- assignee1 now can see description with secure information

However, putting secure information inside of description is not a good practice. Secure information should be kept in an isolated project with only authorized employees as members, and cross-project issue relations can be used for linking secure information.

My main concern regarding security is traceability. I like your idea, but I don't like the fact everything is erased, after that we don't know who rolled back on the issue nor why.

I'd love to see a "revert" option instead: it would pre-fill all the fields with the inverse transitions.

Finally, same as you, I'd like to get more feedback from the community before going down

this path. ;-) Good job anyway!

Ahh... I see. I suppose a 'rollback' flag could be introduced for journal entries. Journal entries that have been rolled back won't be displayed. A rollback comment is required to be entered when rolling back, and this will create a rolledback journal entry that contains the comment (which also won't be displayed). There could be a button (perhaps an expansion '+') somewhere that would expand the journal entries to display rolled back journal entries as well.

Personally, I prefer that all traces are removed. This allows someone to truly remove information that should not be in the history at all. After all, it is a separate permission - I consider it sort of a power user action for someone with a high level of responsibility. If it presents a security concern for a particular role/company, simply disable the permission. If you just want to revert the state of the issue and need history, just edit the issue - I don't really see the difficulty in editing.

Badda-bing, badda-boom!

Finally got around to fixing your concerns. I added a user preference that determines whether or not rolled-back issue notes are displayed or not. If they're displayed, they're displayed with a strikethrough. The rolled-back notes are not removed from the database, and are simply flagged with a boolean.

I would really love to see this moved into the trunk for 2.1.2 or 2.2.0.

Thanks!

Oops, forgot to correctly implement 'last_valid_journal' in app/models/journal.rb.

The patch currently does this:

```
def last_valid_journal?(journals)
    self == journals.last
end
```

It *should* do this:

```
def last_valid_journal?(journals)
    self == journals.reject{|j| j.rolled_back}.last
end
```

Optionally, it could be changed to this:

```
def self.valid(journals)
    journals.reject{|j| j.rolled_back}
end
```

And the check in app/views/issues/_history.html.erb would be this:

```
if journal == Journal.valid(journals).last && journal.can_rollback?
```

Also, the comments in app/models/issue.rb in function rollback should be updated. I believe they still reference that the rolled-back journal is destroyed/deleted.

Sorry for the spam. I just realized when I was copying the patch file, that I introduced a typo. Here's the real patch file with the typo fixed.

+1

I'm trying to use this patch but running into some unexpected issues.

After upgrading my Redmine 2.0.0 installation:

cp redmine-2.0.0/config/database.yml redmine-2.1.0/config

cp redmine-2.0.0/config/configuration.yml redmine-2.1.0/config

cp -R redmine-2.0.0/files redmine-2.1.0/files

cd redmine-2.1.0; bundle install; rake generate_secret_token

rake db:migrate RAILS_ENV=production; rake tmp:cache:clear; rake tmp:sessions:clear

I then patched the source, which was successful:

```
$ patch -p0 < rollback-2.1.0-fixed.patch
patching file redmine-2.1.0/app/controllers/journals_controller.rb
patching file redmine-2.1.0/app/models/issue.rb
patching file redmine-2.1.0/app/models/journal.rb
patching file redmine-2.1.0/app/models/user_preference.rb
patching file redmine-2.1.0/app/views/issues/_history.html.erb
patching file redmine-2.1.0/app/views/users/_preferences.html.erb
patching file redmine-2.1.0/config/locales/en.yml
```

I then hupped apache (httpd-2.2.15-15.0.1.el6.x86_64; passenger-3.0.12; ruby 1.9.3p194; rails 3.2.11; RubyGems 1.8.23) and attempting to access Redmine results in this:

```
*** Exception SyntaxError in PhusionPassenger::ClassicRails::ApplicationSpawner (/var/www/redmine-2.1.0/app/models/issue.rb:612: syntax error, unexpected ',', expecting ')'
                    when 'attr'; send ("#{d.prop_key}=", d.old_value)
                                                                    ^
/var/www/redmine-2.1.0/app/models/issue.rb:612: syntax error, unexpected ')', expecting keyword_end
/var/www/redmine-2.1.0/app/models/issue.rb:616: syntax error, unexpected keyword_when, expecting keyword_end
                    when 'attachment'; attachments.ea...
                         ^
/var/www/redmine-2.1.0/app/models/issue.rb:784: syntax error, unexpected keyword_do_block, expecting keyword_end
                issues.each do |issue|
                               ^
/var/www/redmine-2.1.0/app/models/issue.rb:785: syntax error, unexpected tSTRING_BEG, expecting keyword_end
...      issue.instance_variable_set "@relations", relations.select...
...                                                 ^
/var/www/redmine-2.1.0/app/models/issue.rb:785: syntax error, unexpected ',', expecting keyword_end
...ance_variable_set "@relations", relations.select {|r| r.issu...
...                                                 ^
/var/www/redmine-2.1.0/app/models/issue.rb:788: syntax error, unexpected keyword_end, expecting $end)
```

If I run a ruby -c on redmine-2.1.0/app/models/issue.rb, I get similar syntax errors:

```
redmine-2.1.0/app/models/issue.rb:612: syntax error, unexpected ',', expecting ')'
                    when 'attr'; send ("#{d.prop_key}=", d.old_value)
```

So I try and remove the parentheses from redmine-2.1.0/app/models/issue.rb line 612, changing this:

```
                    when 'attr'; send ("#{d.prop_key}=", d.old_value)
```

to this:

```
                    when 'attr'; send "#{d.prop_key}=", d.old_value
```

Pages start rendering! Great! That is, until I try to view an issue with a journal entry, at which point I get a Redmine 500 error, and this in the production log.

```
Started GET "/issues/20" for 10.135.101.192 at 2013-01-25 12:18:51 -0700
Processing by IssuesController#show as HTML
  Parameters: {"id"=>"20"}
```

```
  Current user: joeuser@joeuser.com (id=4)
  Rendered issues/_action_menu.html.erb (5.0ms)
  Rendered issue_relations/_form.html.erb (1.6ms)
  Rendered issues/_relations.html.erb (3.0ms)
  Rendered issues/_history.html.erb (1.6ms)
  Rendered issues/show.html.erb within layouts/base (469.0ms)
Completed 500 Internal Server Error in 536ms

ActionView::Template::Error (undefined method `rolled_back' for #):
    1: <% reply_links = authorize_for('issues', 'edit') -%>
    2: <% for journal in journals %>
    3:   <% if journal.show? %>
    4:
    5:   <% if journal.rolled_back? %>
    6:
  app/models/journal.rb:75:in `show?'
  app/views/issues/_history.html.erb:3:in `block in _app_views_issues__history_html_erb___2997522374147565881_58424420
'
  app/views/issues/_history.html.erb:2:in `each'
  app/views/issues/_history.html.erb:2:in `_app_views_issues__history_html_erb___2997522374147565881_58424420'
  app/views/issues/show.html.erb:122:in `_app_views_issues_show_html_erb___3007417919998968854_52189840'
  app/controllers/issues_controller.rb:117:in `block (2 levels) in show'
  app/controllers/issues_controller.rb:114:in `show'
```

Any idea what's broken? I'm definitely a Ruby newb, and it's unclear to me what "rolled_back" method Rails is complaining about.

Sorry I don't have the time to dig into this patch, please don't re-assign this issue to me.

If you want this to be integrated in the core, you should at least address the following points:

- how do you ensure everything is still traceable ?
- add tests so we're sure we don't break it in the future

As said earlier, I think a plugin would be a better place for this functionnality if left as is.

Wallace Winfrey wrote:

> Any idea what's broken? I'm definitely a Ruby newb, and it's unclear to me what "rolled_back" method Rails is complaining about.

It looks like you forgot to migrate the database after applying the patch. rolled_back is an attribute of journal, it appears you're missing the 'rolled_back' column in your journal table (which should be added after the db:migrate).

Brian Lindahl wrote:

> It looks like you forgot to migrate the database after applying the patch.

I rolled everything back, then started again:

@% cp redmine-2.0.0/config/database.yml redmine-2.1.0/config@

@% cp redmine-2.0.0/config/configuration.yml redmine-2.1.0/config@

@% cp -R redmine-2.0.0/files redmine-2.1.0/files@

@% cd redmine-2.1.0; bundle install; rake generate_secret_token@

@% rake db:migrate RAILS_ENV=production; rake tmp:cache:clear; rake tmp:sessions:clear@

@% cd ..; patch -p0 < rollback-2.1.0-fixed.patch@

@% cd redmine-2.1.0; rake db:migrate RAILS_ENV=production@

At this point, the rake db:migrate fails due to the parentheses bug on line 612 of redmine-2.1.0/app/models/issue.rb; I fix it and re-run the db migration:

@sed -i -e 's/when 'attr'; send ("#{d.prop_key}=", d.old_value)/when 'attr'; send "#{d.prop_key}=", d.old_value/' app/models/issue.rb@

@rake db:migrate RAILS_ENV=production@

Curiously, there is no output to the rake db:migrate, and sure enough, when I examine the table, there's no rolled_back column in journals:

```
mysql>  desc  journals;
+-----------------+-------------+------+-----+---------+----------------+
| Field           | Type        | Null | Key | Default | Extra          |
+-----------------+-------------+------+-----+---------+----------------+
| id              | int(11)     | NO   | PRI | NULL    | auto_increment |
| journalized_id  | int(11)     | NO   | MUL | 0       |                |
| journalized_type| varchar(30) | NO   |     |         |                |
| user_id         | int(11)     | NO   | MUL | 0       |                |
| notes           | text        | YES  |     | NULL    |                |
| created_on      | datetime    | NO   | MUL | NULL    |                |
+-----------------+-------------+------+-----+---------+----------------+
```

Is there another step I forgot to take, or did I do the post-patch db:migrate at the wrong time?

Looks like the migration code was never in the patch - sorry about that. Try this patch - should fix that 'send' problem, as well as add the proper migration.

Thanks so much! Is this patch compatible with only Redmine v2.1.0 ?

After applying the patch, I found that:

There were no rollback icons available next to issue updates.

Adding a new update to an existing issue did not have a rollback icon either.

Newly-created issues show up in the Issues listing & Gannt chart, but clicking on an issue created post-patch, or manually navigating to it, resulted in a 404.

Odd, I've never had any of these problems before, even during development. Are you sure you're using 2.1 stable?

The only rollback icon that should appear is on the last update (you can only rollback the most recent update).

Yes, absolutely sure, this version, specifically: http://rubyforge.org/frs/download.php/76448/redmine-2.1.0.tar.gz

Fixed the problems that Wallace talks about in the new 2.4.4 patch file.

The old 2.1.0 patch was generated incorrectly. It was missing the config/routes.rb modification and the lib/redmine.rb modification.

Wallace's issues 1) and 2) is fixed by the lib/redmine.rb modification, which adds the rollback permission (why no rollback icons appeared). Wallace's issue 3) is fixed by the config/routes.rb modification, which adds the route for journals/rollback/<#id> (why there were 404 errors). These modifications can easily be added on top of the 2.1.0 patch. I also noticed, in the 2.1.0 patch, there may be some 'missing translation' errors, which should be relatively easy to fix up.

The 2.4.4 patch fixes all of the above problems and should be working perfectly.

Oops, use this patch file instead. The one I just uploaded had the problem referenced in note 12.

Hah.. didn't upload the fixed patch. I really wish I could delete my own attachments. Anyway, here's the correct patch file.

Brian, I have used your patch, and it works perfectly.
However, if the journal is shown in reverse order, the "rollback" icon is shown in the first entry...

Neither usability or purpose are broken, but maybe you would like to verify this situation.

Patch against redmine 2.6.3 with newer version of Ruby.

Updated some deprecated functions and order problem I was having with postgresql.

Patch against redmine 4.0 with various fixes
dry-run first (test, may have some hunk offsets but it's working):

```
patch -p1 --dry-run < 0001-Rollback-functionality.patch
```

Ok? then apply:

```
patch -p1 < 0001-Rollback-functionality.patch
```

I made some mistakes in last patch, this one corrects everything. Apply this instead.

Patch against redmine 4.0 with fixes
dry-run first (test, may have some hunk offsets but it's working):

```
patch -p1 --dry-run < 0003-Rollback-functionality.patch
```

Ok? then apply:

```
patch -p1 < 0003-Rollback-functionality.patch
```

This should probably be added as a setting for "paranoid mode"...

related_issues

relates,Closed,1725,Delete button on comments
relates,Closed,1665,Ability to delete tracker comments
relates,New,12388,diffs for editions of issue/notes entries

履歴

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

- カテゴリ を Issues_2 にセット