Redmine - Patch #36691
Background job and dedicated status for project deletion
2022-02-23 10:17 - Jens Krämer

Due to the deletion of dependent objects (issues etc) and child projects, project deletion may take a long time.

This patch, which was extracted from [Planio](https://plan.io/redmine-hosting), moves the actual project deletion into an ActiveJob job. It also introduces a new project status (SCHEDULED_FOR_DELETION) that is used to effectively hide the project that is about to be deleted (and any potential descendant projects) from the system immediately.

A security notification is sent out to the user who deleted the project, informing about success / failure.

The projects list is extended to be able to filter for the new status, so in case of a failure, the project can still be accessed for examination.

Related issues:
Related to Redmine - Patch #31076: Issues CSV / PDF export via ActiveJob

History
#1 - 2022-02-23 22:58 - Marius BALTEANU
- Target version set to Candidate for next major release

Nice feature!

#2 - 2022-02-23 23:00 - Marius BALTEANU
- Related to Patch #31076: Issues CSV / PDF export via ActiveJob added

#3 - 2022-02-23 23:07 - Marius BALTEANU
Jens, should we take into consideration also #34987?

#4 - 2022-02-24 00:24 - James H
maybe my group members issue #36696 can benefit from this as well

#5 - 2022-02-24 03:43 - Jens Krämer
Marius BALTEANU wrote:

Jens, should we take into consideration also #3498??

Ah, that's interesting. If I understand correctly, that would mean the project record (plus potential child projects' records) would be deleted synchronously, but removal of all dependent objects would be pushed to the job queue.

I see few problems with that:

1. Data integrity. Before the job is finished, we have an inconsistent database (i.e. issues referencing a nonexisting project). I am not sure we handle such a situation gracefully (i.e. resulting in an HTTP 404 vs a 500) in all cases.
2. Two transactions instead of one. Currently, if the project deletion fails due to a problem somewhere along the way (due to a failing callback for example), the transaction is rolled back and everything is as if nothing happened. Splitting the process in two separate transactions (one for the project, one later for the dependent objects), means the project record will be gone even if some dependent object insists on not being destroyed. I do not think core has any such callbacks, but plugins might.
3. Does not work at all when there are foreign keys referencing `projects.id`. We do not have such in core, but again, there may be plugins that do.

#6 - 2022-02-24 08:46 - Marius BALTEANU

Thanks Jens for your quick response.