redmineorg-copy202205 - Vote #68369

anonymized issue numbers

2022/05/09 15:04 - Admin Redmine

ステータス:	New	**開始日:**	2010/10/12
優先度:	通常	**期日:**	
担当者:		**進捗率:**	0%
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/6642	**status_id:**	1
category_id:	2	**tracker_id:**	2
version_id:	0	**plus1:**	0
issue_org_id:	6642	**affected_version:**	
author_id:	21787	**closed_on:**	
assigned_to_id:	0	**affected_version_id:**	
comments:	12		

説明

Hi

This is a minor issue..

I'd like to see issue numbers anonymized, so that the number of issues opened in project X does not pollute the "id space" of the numbers assigned to issues in project Y.

Proposition 1

For example, instead of a sequential auto incrementor, issues could be numbered with a unique, random 128-bit value, which is then represented by a Youtube-like number in the URL, for example using one of the algorithms here:

http://radius.yubico.com/demo/Modhex_Calculator.php?srcfmt=N&srctext=12345678901

Perhaps it should be an optional feature, there could be people that would prefer sequential issue numbering across projects.

Proposition 2

Issues could be assigned per-project sequential numbering, so that within each project, the first issue is numbered 1, then 2, 3 etc.

journals

Proposition 2 has been discussed at great lengths already. Long story short: Issue IDs need to be unique in redmine, period.

Regarding proposition 1: Is there any specific reason or security concern you have with sequential IDs?

Anyway, the sequential IDs are provided by the DB, convince your DB to throw out randomized IDs and your problem is solved. The problem with randomized IDs though is that you might have unpredictable complexity, i.e. for each ID you create, you have to check if it doesn't exist in the DB already, wheres a sequential ID has a pretty predictable complexity of O(1).

> Proposition 2 has been discussed at great lengths already.
> Long story short: Issue IDs need to be unique in redmine, period.

Fair enough.

> Regarding proposition 1: Is there any specific reason or security
> concern you have with sequential IDs?

Yeah, currently leaking stuff such as:

- the total number of issues we handle for one organizational unit leaks to another
- the rate of cases handled (just create a new issue once in a while to read it out)
- how many cases we handle (which can be extrapolated into other kinds of data points)
- etc...

The short of the long is probably that we're using Redmine in a more business-minded way than the standard usage, which I presume is for open source projects. (Meaning that this issue may be entirely non-critical to everyone else.)

> Anyway, the sequential IDs are provided by the DB, convince your DB
> to throw out randomized IDs and your problem is solved.

Okay, awesome.

Would a patch to enable this as an option be accepted, you think?

> The problem with randomized IDs though is that you might
> have unpredictable complexity, i.e. for each ID you create,
> you have to check if it doesn't exist in the DB already,

If there was a non-trivial repetition rate in the database's random function leading to collisions, then it would be a bad random function. I think this part is OK.

(If there was a problem, it could be solved by a background process that keeps a buffer of random identifiers ready to go in a table or whatever. But now we're delving into a tangent that will never be relevant, I think. Back on track..)

> wheres a sequential ID has a pretty predictable complexity of O(1).

Okay, I can see where this might be a problem. If the source code that makes up Redmine has a notion that it can traverse issues by starting at number 1 and working upward, then we have a problem.

A similar problem arises if somewhere a bunch of issues is stuffed into an array, with the array index corresponding to the issue number.

Do things like this happen in the Redmine code base?

(I'm assuming that arrays in Ruby are not hash maps like they are in PHP.. Please excuse my poor Ruby-fu.)

Albert Rosenfield wrote:

> Regarding proposition 1: Is there any specific reason or security
> concern you have with sequential IDs?

> Yeah, currently leaking stuff such as:
>
> - the total number of issues we handle for one organizational unit leaks to another
> - the rate of cases handled (just create a new issue once in a while to read it out)
> - how many cases we handle (which can be extrapolated into other kinds of data points)
> - etc...
>
> The short of the long is probably that we're using Redmine in a more business-minded way than the standard usage, which I presume is for open source projects. (Meaning that this issue may be entirely non-critical to everyone else.)

If you have such trust issues, you might be better off having one redmine per OU altogether. Off the top of my head, I can think up a lot of other things one could find out by crafting URLs (project identifiers are unique, navigate to one you think is there, and you get a 403 forbidden if it's there, a 404 not found if not, same thing with wiki links, which contain the name of the page, and so on), so everyone sharing one redmine might not be along your guidelines/fears.

> Anyway, the sequential IDs are provided by the DB, convince your DB

> to throw out randomized IDs and your problem is solved.

Okay, awesome.

Would a patch to enable this as an option be accepted, you think?

I think this would rather be a DB-side change, not something that will have to change in redmine. This is most unlikely to get in core even it is anyway, but a plugin/patch and a mention of it in the docs would help anyone looking for something similar in the future. See the paragraph about the logger config in the [[RedmineInstall#Logger-Configuration]].

> Okay, I can see where this might be a problem. If the source code that makes up Redmine has a notion that it can traverse issues by starting at number 1 and working upward, then we have a problem.
>
> A similar problem arises if somewhere a bunch of issues is stuffed into an array, with the array index corresponding to the issue number.
>
> Do things like this happen in the Redmine code base?
>
> (I'm assuming that arrays in Ruby are not hash maps like they are in PHP.. Please excuse my poor Ruby-fu.)

Ruby has arrays and hashes (the first references by numerical position, the second by "any object you want" but without order), but that aside, I don't think there are places where the IDs are handled as something else than a reference to the object. The only time I can think of ordering is relevant is when you order issues by ID in the issue list (sic). You'd need a full code review to be sure though.

> If you have such trust issues, you might be better off
> having one redmine per OU altogether.

Good idea, but too much trouble. (The OUs change quite often, making it cumbersome.)

> Off the top of my head, I can think up a lot of other
> things one could find out by crafting URLs

Interesting :-)

> (project identifiers are unique, navigate to one you
> think is there, and you get a 403 forbidden if it's
> there, a 404 not found if not,

Meh, the search space is big enough that it is less than trivial. Anyone willing to spend that much effort could extract the information by equally difficult means (social engineering with various attack vectors, for example). Good point, though.

> same thing with wiki links, which contain the name of the page, and so on)

Haven't really started using the wiki yet, so that's okay for now.. :-)

> This is most unlikely to get in core even it is anyway, but a plugin/patch [...]
> would help anyone looking for something similar in the future.

It is not possible to trivially uninstall/install a plugin that alters the core Redmine database in this way, is it?

> See the paragraph about the logger config in the RedmineInstall.

Will do.

> I don't think there are places where the IDs are handled as something else than a reference to the object.

Excellent. I'm going to explore this path further, then.

> The only time I can think of ordering is relevant is when you order issues by ID in the issue list (sic). You'd need a full code review to be sure though.

Easier to ask someone who has everything in his/her head :-)

> See the paragraph about the logger config in the RedmineInstall.

Just did..

The idea is to make a patch to core, but continue using the sequential numbering per default. Then with an option hidden away in the config file, random numbering can be enabled.

Did I understand this correct?

Albert Rosenfield wrote:

> > See the paragraph about the logger config in the RedmineInstall.
>
> Just did..
>
> The idea is to make a patch to core, but continue using the sequential numbering per default. Then with an option hidden away in the config file, random numbering can be enabled.
>
> Did I understand this correct?

Not quite. That was just an illustration of how we can extend existing docs to reflect your changes.

I just had a quick look at some docs though, and it seems most DB will only do incremental IDs OOTB. Ok, so this will imply a change to redmine, which should be doable in a plugin too, as ruby has open classes. Off the top of my head, I'd say use the @before_save@ callback from ActiveRecord in the @User@ model (@app/models/user.rb@), and authoritatively set the id there. "Setting" the ID there would imply generating some sort of random ID or hash, and check that it is unique wrt the current DB state. You'd have one "risk" left with that if someone writes an issue to the DB with the random ID you just checked to be unique, you could alleviate that by throwing everything into a transaction, but I'm not sure how complicated that would be. I think the worst case would be a failed write, the new issue page getting shown again with everything still in there, and you'd just need to re-submit the form.

Anyway, I've put a little too much time in this already, I'd be happy to continue to help, though I'd prefer IRC or some other form of more "direct" communication.

Had a short discussion about this on IRC.

Came up with 3 different solutions:

1)
Discussed one solution with a plugin that alters the issue primary key to contain for example a 128-bit UUID. It has a couple of problems, one being that only new cases get fixed (it's not uniform unless the plugin is installed exactly when the system is created - inflexible), another problem is that it would be difficult to uninstall the plugin again.

2)
Discussed assigning a random ID in before_create of app/models/issue.rb. The problem here is that the database is not generating the random number at insertion time, so the number might collide with fx. a random number chosen by another user at the same time, or an existing row in the table.

3)
The core could stay as-is, and a minimal shim added. The shim translates from core numbering and into whatever the user needs (through a plugin). The core would have to call two functions to translate the primary identifier to a shim value. One is called when putting an issue id in an email or a hyperlink, and another is called when an incoming id needs to be translated to a real value.

(The default shim functions would do nothing, aka return the original value. A plugin would hook the shim functions and keep its own table of UUID-to-id or whatever the author wants to do.)

Reading the discussions linked to #74 I see that there was no intention to fix this. I'm from yet-another redmine-evaluating project that was surprised the 'feature' of sharing ID space across all projects. Previously I have used JIRA and YouTrack, where issue IDs like CALC-29 are well adopted. I cannot imagine a scenario when one would need to move issue from one project to another.

The only option I see possible to achieve separate ID spaces is to run an instance of Redmine per project, but this is not good either.

I think this is important. Any updates for this story for 2012?

Here is a solution for proposition 2: http://projects.andriylesyuk.com/project/redmine/issue-id

The difference is that it requires the project key in addition to the issue number, so you get #X-1, #Y-1 and so on.

---

related_issues

relates,Closed,1926,Issues identifiers
relates,Closed,74,Projects share issue numbers
relates,New,2885,A segregated numbering per project
relates,New,25625,Turn issue numbers into UUIDs

履歴

#1 - 2022/05/10 17:21 - Admin Redmine

- カテゴリ を Issues_2 にセット