ChiliProject - Feature # 590: Multilanguage for Status of Issues, Issue Categories and Enumerations

Status:	Open	**Priority:**	Low
Author:	Ferdinand Thommes	**Category:**	Translations
Created:	2011-08-22	**Assignee:**	
Updated:	2011-08-24	**Due date:**	
Remote issue URL:			
Affected version:			

Description: We feel, the following Items should be multilanguage:

Status of issues:

<pre>
1□ id□ int(11)
2□ project_id□ int(11)
3□ name□ varchar(30)
4□ assigned_to_id□ int(11)
new□ language_code varchar(3)
</pre>

New Table language_codes
<pre>
1□ code□ varchar(3)□
2□ name□ varchar(30)
</pre>

Table Issue Categories:

Table issue_categories should be multi-language

<pre>
1□ id□ int(11)
2□ project_id□ int(11)
3□ name□ varchar(30)
4□ assigned_to_id□ int(11)
new□ language_code varchar(3)
</pre>

New Table language_codes
<pre>
1□ code□ varchar(3)□
2□ name□ varchar(30)
</pre>

*Enumerations should be multilanuage:

Table enumerations
<pre>
#□ Name□ Type
1□ id□ int(11)
2□ name□ varchar(30)
3□ position□ int(11)
4□ is_default□ tinyint(1)
5□ type□ varchar(255)

6 active tinyint(1)
7 project_id int(11)
8 parent_id int(11)
new language_code varchar(3)
</pre>

New Table language_codes

<pre>
1 code varchar(3)
2 name varchar(30)
</pre>

Associated revisions

2008-02-03 03:38 pm - Jean-Philippe Lang

ProjectsController#add_news moved to NewsController#new.
Preview added when adding/editing a news (#590).

git-svn-id: http://redmine.rubyforge.org/svn/trunk@1111 e93f8b46-1217-0410-a6f0-8f06a7374b81

History

2011-08-22 10:30 am - Felix Schäfer

Sadly it's not that easy: How do you recognize that 2 categories are the same but in different languages? Who can add/change the names of the categories? If you have multiple entries in the categories table for the same category, it makes filtering/searching more difficult (you have to remember to filter/search for all the categories that have the same meaningâ€¦), and so on.

The status quo for this has been that as they are user-defined inputs (either the project manager or the ChiliProject admin), teams should agree on a base language those non-localized names are in. Changing this is rather complex, and we have other more important matters to attend to for the moment, sorry.

2011-08-23 11:49 am - Alf Gaida

* Positive: It could be that easy.
* Negative: This means, you'll have to change it systemwide. Database (tables and indices), underlying methods and front.

I've implemented this in MS Dynamics NAV several times that way.

IMHO the first step could be the implementation in tables and indices. That doesn't change any functionality. A unique index foo (fieldx,fieldy) unique is in the first step equivalent to index bar (fieldx) unique, if fieldy is ''. ;) The implementation of methods like method foo (par1,par2) instead of method bar (par1) can be done later.

In our special case we will change the default language completely to english. Because of many german users it would be very nice to have ML with a complete german front. denglish isn't so nice.

2011-08-23 09:30 pm - Felix Schäfer

Alf Gaida wrote:
> * Positive: It could be that easy.

It is neither easy nor good practice or database design, at least not in the rails world and for people who care about database normalization. So what, I should first make a query to whatever store no one has talked about to find out which categories are the same but just in different locales and then I can do whatever query I need with (for example) @WHERE category_id IN (1,2,3)@ instead @WHERE category_id = 1@? Not good.

> * Negative: This means, you'll have to change it systemwide. Database (tables and indices), underlying methods and front.

Doesn't sound very object oriented, everything that deals with categories now has to know that they have split personalities? Again, not good.

I didn't say it was extremely difficult to realize, in fact just throwing in a table with @category_id@, @localized_name@ and @locale@, removing the @name@ from the @categories@ table and making a @Category@ object return the @localized_name@ corresponding to it's @id@ and with the right locale would probably do the trick, the difficulty here is to make it efficient. Go query the name each time you need it? Lots of queries. Grab all the names every time and decide which one you want later? Lots of object creation for the request. Just load the table on start in memory and cache it? Needs a lot of extra care.

And again, past those considerations, who will have permission to change what? Each user only for the language his interface is in? And how do you make sure someone doesn't translate "important" with "low" just to piss everyone of?

As I said, the problem isn't trivial and we have more pressing matters on our hands, I'll happily review any patch with tests posted here, but I (I can't speak for my fellow devs) don't have time for more, sorry.

> I've implemented this in MS Dynamics NAV several times that way.

Then I guess I'm sorry four you and whomever has to maintain this.

> In our special case we will change the default language completely to english. Because of many german users it would be very nice to have ML with a complete german front. denglish isn't so nice.

I don't want to start a flamewar or endless discussion here, but where do you stop translating everything? Say the UI is localized, some of the system-wide and maybe project-wide attributes too, will you want extra fields to translate the issue titles too? Forum posts? Commit messages?

2011-08-23 11:59 pm - Alf Gaida

- Status changed from Open to Closed

> It is neither easy nor good practice or database design.

False.

> Doesn't sound very object oriented.

A sql database isn't really objectorientated. So you have to add the fields, modify indices and triggers to. After that you can go on and work oo-based.

> Then I guess I'm sorry four you and whomever has to maintain this.

It was not so bad at all. Nearly all of your points are right. We had a lot of discussions, how to do the translation process right. In the end we make a wise descision: wait. In our case the next official version has multilanguage partly integrated. So we implemented the dirty rest and earn some good money doing this. ;) And we had a lot of horrible translations. But it wasn't our business. The customers themselve translate most of the stuff. The old principle of shit in shit out.

>but where do you stop translating everything?

We made translations possible for nearly all of our objects. All Objecs have a fallback method to default language and that it. No underlying translation -> fallback. So our customers translated only stuff they really needed.

As i wrote, it would be nice to have. Other points are more important. Eventually ML is a issue for chili 4 or 5 or 10. We made the descision to switch back to english. It was a hard descision, because we have 170 german speaking users vs. 5 english speaking users. But this will change hopefully.

(Wenn ich kÃ¶nnte und die Zeit dafÃ¼r hÃ¤tte, also wirklich nichts wichtiges mehr anliegen wÃ¼rde, dann wÃ¼rde ich mich hinsetzen und das da reinhacken. Das das bei uns aber Ã¤hnlich aussieht im Projekt: Sei's drum. Ein funktionierender, stabiler und komfortabler Tracker ohne Ãœberraschungen in englisch ist mir sehr viel lieber als larifari in multi. Macht weiter mit Eurer Arbeit, Chili wird von Release zu Release besser und das finde(n) ich (wir) toll.) Ich setz das einfach mal auf zu.

2011-08-24 06:54 am - Felix Schäfer

- Priority changed from Normal to Low

- Category set to Translations

- Status changed from Closed to Open

Alf Gaida wrote:
> > It is neither easy nor good practice or database design.
>
> False.

For completeness, let's add the rest of the sentence: "at least not in the rails world and for people who care about database normalization". Anyway, I don't know how to make that more clear without going into DB theory, but having _multiple records_ in the DB representing in the end _the same thing_ goes against DB normalization (you have to work on multiple records to change "one thing") and the ruby and rails principle of DRYness (Don't Repeat Yourself).

The important thing is: you want to localize _the name_ not _the whole record_, so ripping out the @name@ column from tables holding objects for which you want to internationalize the name and storing it in an extra table is the way to go.

> > Doesn't sound very object oriented.
>
> A sql database isn't really objectorientated. So you have to add the fields, modify indices and triggers to. After that you can go on and work oo-based.

It isn't, and that's why rails uses a db as a "dumb" data store with a nice query interface, no triggers, no foreign key constraints. Per default you only get primary key uniqueness and can add indexes yourself, that's all. That's also why we need "kludges" like the nested set or materialized path patterns for trees and the acts_as_list plugin to store ordered lists in the DB, but those are very hard to get transactionally safe (e.g.: creating a sub-issue and double-clicking create will if the server takes the requests fast enough result in an inconsistent DB because the server will try to insert 2 data sets at the same place in the issue tree at the same time). To be honest, I'd like some way to define at least foreign key constraints in the DB, but that's not possible in "vanilla" rails that I know of.

(Without wanting to go too much into detail, there's some reasons to that too, ranging from having to support multiple SQL servers (MySQL, PostgreSQL, SQLite, SQLServer, â€¦) which all have slightly different interpretations of how SQL works (Oracle writes NIL to a string field you write and empty string to, not quite the same ting <notextile>*sigh*</notextile>), ActiveRecord (the rails library, not the design pattern) trying to not be limited to SQL, and so on).

> >but where do you stop translating everything?
>
> We made translations possible for nearly all of our objects. All Objecs have a fallback method to default language and that it. No underlying translation -> fallback. So our customers translated only stuff they really needed.

Yes, the fallback is not a problem, the i18n gem now provides a fallback mechanism too and I think with Chili 3.0 we will activate it (not sure it's possible yet if we still support "old" i18n versions), that sill doesn't decide "where" to stop localizing. The usual split point is "user input" vs. "stuff that's already in the program/package", i.e. categories and so on are "user input" (privileged users, but still users) and don't need to be localized, but I guess that could be lowered to "normal user input" vs. "structural user input and what's already in the program/package".

> As i wrote, it would be nice to have. Other points are more important. Eventually ML is a issue for chili 4 or 5 or 10.

Now that I've spent so much thinking about it, it might even happen in 5 rather than in 10, but no promises.

(For the record, my ideas:
* create a @ChiliProject@ namespace in the i18n store,
* read the DB and populate said namespace either at boot time (rather easy) or patch i18n to fallback to reading the DB and populating itself if it can't find stuff in its store,
* the keys under ChiliProject would be @[klass.name.underscore][id]@, which should guarantee key uniqueness,
* that would introduce i18n stuff to the models though, so not sure how good that works out)

> (Wenn ich kÃ¶nnte und die Zeit dafÃ¼r hÃ¤tte, also wirklich nichts wichtiges mehr anliegen wÃ¼rde, dann wÃ¼rde ich mich hinsetzen und das da reinhacken. Das das bei uns aber Ã¤hnlich aussieht im Projekt: Sei's drum. Ein funktionierender, stabiler und komfortabler Tracker ohne Ãœberraschungen in englisch ist mir sehr viel lieber als larifari in multi. Macht weiter mit Eurer Arbeit, Chili wird von Release zu Release besser und das finde(n) ich (wir) toll.) Ich setz das einfach mal auf zu.

Thanks, and no need to close it, it is a valid feature request and one I wouldn't mind having, but the priority isn't so high :-)