Data representation

First I will discuss what requires to be done to actual database. I think that at data level the changes are minimal.

Page aliases

This is not a central argument in my idea, but have some nice side effects, and is having another table with "aliaeses", some place to describe that some page and/or object is know also by another name. Something with a structure like

- Alias
- Real Name
- Parameters

The parameters component are extra parameters for the rendered of the real name. That could be used for multilingual sites (i.e. NombresEnTiki be an alias of TikiNames with parameter lang=es or something like that) or other ways to see the same resource . Is not very critical for what I discussing here, but I think that for completness and future should be included.

If some page is created after with the name of an alias, the real name will be solved before than the alias, but would be good to remove the alias in that moment.

See also: WikiPageRenameDev

Names worldwide

I think that all objects, or most of them, could have names. Thru the actual system, I think the only kind of object that have a proper name are just wiki pages, but forums, blogs, polls, menus, trackers, etc, already have a name component that is not used. I think that names should be more widely used, maybe with a universal parameter for all scripts that is "name" and that could be used instead of Id or whatever is used to refer to a particular instance of an object. The names also should be required to be URLizables, i.e. no "?" or other kind of reserved chars in them

A particular requeriment for this idea: **should be a single namespace for all objects** . At a given level, no two objects of whatever kind could be called in the same way. And also, I think that names should be case-insensitive.

Data dictionary

For that single namespace, another table will be required, something with a structure like

- Object ID
- Object Type (look at actual object permission table for an example of what could be here)
- Object Name (the actual and unique name for an object for the system)

The object id could be used in other tables instead of the name for repetitive calls of that name (i.e. for permissions that we will see later here), and the 2 more important justitifacion for them for me is size (to not have long fields like the one that should hold the object name in every table) and unicity (so I can rename a resource and all links still point to it in tables)

Namespaces

Well, we are reaching more polemic fields. My proposal for namespaces is a simple one, choose a notation for namespaces and they will be defined by the component objects (of course, the Object Name in the previous example contains all the "path" that conform their namespace).

For notation there are several alternatives, either for separators (i.e. category.page, category/page, category:page) and for order (page.category or cateegory.page). Just to illustrate what will follow of this page, I will take "/" as separator, and first container and then contained object, i.e. category/page. This notation could enable some clever URL games to make site references more clear and easier to remember, but any other notation could be chosen, is not critical to what is exposed here.

With this in account, lets make some definitions. There are a "root" category, a place where all content in a site belongs. Also we can talk about qualified names (the ones with the full path to root) and unqualified names (names without path/categories or local to the actual category/position.

Also there are objects that not have its own name, either because there is no way to name it or because it could have little sense. For that kind of objects (forum/articles/blogs posts, trackers records, etc) an implicit name could be considered, a numeric one i.e. myforum/34 for the 34th post to some forum, or mytracker/23 to say the 23th record of mytracker, or even mywikipage/12 for the 12th version of my wiki page.

Permissions

Exist already an user object permissions table. With the data dictionary table, we can have a object_permissions table with just

- Object ID
- group
- permission name

(as a side note, instead of group field could be an "owner" field be instead? something where groups are notated with a starting @ or else be the normal user name? anyway, is not central to this article)

Implications

Ok, enough from data definition, it is a good way to store things, but... what this kind of things could imply? The data representation sounded easy and very innocent, for programs to take account of this new data don't look so hard, but this new way to look things could enable interesting things.

Embedding

If we have an object called /a/b/c/d/myfinalobject, is easy to traverse all the path to the root. If we can do that parsing, we can see as object all the intermediate results. But /a/b/c could be a wiki page, and then d and d/myfinalobject would be "embedded" inside that wiki page. For namespaces we could use not just "contentless" categories, but also actual objects. If there are no special data structure for categories, just names, could be possible to embed objects inside objects.

And,what kind of objects could be embedded here? Not just "normal" content objects, I can put inside a wiki page a poll, banners, menus, modules, etc.

Permission checkings

Ok, we have a table that say the generic permissions for a groups, and another that say the permissions for objects. To see if an user can access to an object, first we should check the group permissions to see if the user can access that type of object, that is ouside the scope of this page, but, for object permission? For an object /a/b/c/d/myobject the checking will be check if there are permission for his groups for the myobject object, then for /a/b/c/d, then for /a/b/c, then /a/b and then /a.... or in the backward order. You could put permissions in any part of the path, and is O(1) to check all the involved permissions. I'm not sure, but maybe the permission checking could be stopped when the first positive permission is found.

But also it could give more granularity to the permission system. As a name could be for even a tracker item or a forum post, permissions could be that deep.

As permissions can be done to any part of the path, category permissions are contempled here. Can be added to a category permissions for groups of users to edit material, create certain kinds of objects, and so on.

Aliases

An object have a "fixed" location, but with aliases we can categorize it in several ways, put it in several places of the tree Also, for permission checking could, or could not (not entirely sure what is the right choice) check only the aliased page permission or to include the aliased name permission in the checking.

If some part of the path is not existing, the alias table should be checked also. This could slow a bit things, but will make things more clear.

Users

Ok, we have names for objects, no? And users have a name, so we can have objects named /myuser/UserPage that don't conflict with others UserPages in the system. By default, pages created by an user, contextless, will have /myuser/ as implicit path. Or maybe a generic implicit path called Users, from where all users have their content (i.e. /users/gmuslera).

Of course that some new permission should be created, i.e. move_object_to and move_object_from , and maybe create_alias_to and create_alias_from, for modify the path of objects from/to places in the tree, or create alias.

Existing content

The programs that renders links to objects should resolve the context on which those objects are referenced. For unqualified object names (i.e. wiki page names) could be checked if in current context/path and in root if that page exist and render links with the full name. Actual categories and structures can be used to categorize existing content and generate the corresponding aliases.

Anyway, a "name=" parameter should be contempled tor all scripts to say that the object ireferenced is one where the namespaces, paths, etc is taked in account, old "page=PageName" could still be used in wiki scripts for backward compatibilty. And scripts could gradually gain support for the name parameter.

Include plugin

As names could be now universal, the include plugin idea could be implemented more as standard wiki syntax, and support for different kind of object types that could be done in a gradual way. Instead of having a tag for including rss, or banners, or dynamic contents, etc, could have a {include objectname} implicit in the wiki syntax. minitemplates to show embedded content could be done for blogs, trackers, etc.

WYSIWYCA

Ok, we have already currently in memory user permission settings, and checking object permissions is already very defined. A function to check if some object have some permission could be done, it only should traverse the tree implicit on the object's name. Before rendering a link, showing a record, etc, the permission of that object could be checked (it will impact on site performance, maybe a system setting called "wysiwyca check" 🙂 could be added if the admin want that this kind of things be checked.

Wrapper

A wrapper to see an object could be developed and use that to link internal resources. As it will know the kind of object is, will be enabled to link to the rigth script. External links to the site could be rewritten to http://mysite/mycategory/myobject" and menu entries to sections where information is shown could be put in a particular syntax or option, and do checking for "view" on that object.

Object treeÂ navigation

Trying to see a category could list the component objects (maybe a permission for enable/disable that for certain groups could be useful, i.e. listing users with content could be desired or not). If some object includes "environmental" objects (menus, modules, polls) maybe they could be added to one of the side columns (i.e. adding a meta-module for content-related modules) or just be included in text areas with things like the generic include plugin.

Conclusion

I think I have yet not written all the idea, but most of it is here. It requires a little work in the tables, and a not so big work at the programming level, that mostly could be done gradually. I think that this could solve in a simple way some of the problems of actual tiki, and open a very wide window of possibilities for it. This is not focused in dialogs and a bit in programs, was mostly in data and implications over that data, but I think it will not be so difficult to implement.