TikiDesignProblems

Tikiwiki has a lot of features, does a lot of things, slowly it is gaining acceptance ... but it has some problems.

Tiki is BIG

There are several ways in which Tiki is big.

Latest version is near 20Mb installed, an issue that could cause problems in most hostings offers. And with new additions, it will only grow even bigger. Even in compressed size, even without documentation (galaxia documentation is a separate download, and current Tiki documentation is in this site, a bit mixed with other content) it's a somewhat big download. Some time ago someone packed a smaller version of tiki, removing avatar image library, most themes, languages, and things like that to make it smaller. Release size could be decreased, but the removed content should have an easy way to be downloaded and installed.

The size matters also in rendered size. Actual html file could not be so big (home of tikiwiki.org (using moreneat theme) has like 40k, but the .css of moreneat adds 50k more, tiki-js adds 8k, but could be more information, like graphics and other kind of resources. In not so low bandwidth environments such sizes could make problems. Outputting just the relevant sections of the theme css file to the actual page could decrease total page size, but it could take more cpu (?) and anyway, if the css is cached should not have so big impact. The problem is that Tiki sites will make a first bad impression of slowness (when nothing is cached yet) and not always client computers do a good caching. For recomendations about page sizes related to usability, check this note on Jakob Nielsen's site.

- It takes a lot of resources. By default PHP memory settings should be increased to make it run. The refactoring of tikilib decreased a lot the required memory, but anyway, it still grabs a lot. Could the memory required by the application be decreased? Probably the real developers could have a hint on this 🙂. Refactoring language files could be a start (if possible, of course) but I guess that there are more areas in Tiki where a rationalization, or refactoring, or things like that, could lower required memory. And CPU is another area where resources are heavily used, that combined with the LeftColBug (if is caused sending at a relatively slow rate the rendered pages) could make sites unusables in not very fast servers.

- It does a lot of database access. I read somewhere about Tiki that it do like 70 queries to the database (*see bottom bar, tiki 1.8 displays number of queries per page now - ohertel*) to render the home page, probably because in the database is stored the configuration, user data, all content, etc. Probably with MySQL 4 that could cache results of queries things could improve, but a more generic solution could be that the db abstraction layer does the caching, at least for the info relevant to where i'm displaying if not full records.

- But it also **feels** big, with so much things on screen that look complex. Even if most features could be disabled (and/or the application menu disabled and very few modules) to have less options on screen, it still feel very big and complex for administrators, for the amount of options and features that s/he can configure and see). There are several approaches to solve or at least minimize this problem. Properly implementing WYSIWYCA (for users and admins) will decrease the perceived size. Having profiles (like in TikiProfilesDev) will be a help too. But what will make a great change here is to move most features to external modules, with a way to install it separately, and being integrated to the system, a step that could be done slowly adapting the existing system or going to a model where you have a tiki core and most features as extensions (this will help in more problems discussed here), see DaytonTikiCore, TikiCoreWishlist, TikiWhatShouldBeInCore, TikiCorePrototype.

Mixed design approachs

Tiki was done by a lot of people, so having a lot of views on how things should be done, so is natural to see that, but the dual point of view, even if its not so evident always, could have negative impact on installed base.

Editing vs presentation environment: Tiki is presented as a Content Management System, as a Portal system, with permissions that in a way could separate visitors from editors, but for some things the way to access the content is just the editing environment, there is no separate presentation content from them, or at least no hiding that it is the editing environment. Wiki pages are an example, even if you can't edit you can't forget that it is an editing environment, you can see the page history, look at similar pages, be aware of unedited pages, even for visitors, and the only way to hide that kind of things is disabling them for all users, or changing templates for all users. Trackers is another example, they have no just-viewing interface, and problably are more.

Filesystem vs Databases: Storing all in a sql database have a lot of advantages, like remote maintenance of content and options, not being dependent on remote filesystem permissions or restrictions on hosting company, you don't need a shell to edit options, or having a local copy of the configuration file and upload it every time you do a change, and you can backup the entire system just doing a dump of the database. There are things that are stored in the filesystem that should be that way, backups of the database should be in the filesystem, the cache of generated pages, even file/image galleries if you specifically selected to store them in the filesystem instead of the database. But there are content stored in the filesystem that is not so obvious, like illustrations and probably some Galaxia components, that need to be considered when doing a system backup or things like that. Storing those things in the database, and for rendering/running storing them maybe temporally in the filesystem could be a solution for some of those exceptions.

Admin vs Programmer: The main flexibility of Tiki is for programmers more than for admins or end users. For open source programs always is that, but in the case of Tiki the system have little flexibility "as is", probably the most used sites (excluding tikiwiki.org, that should make tiki itself) have some tweaking to modules or add they own modules to make they system usable or that deliver what the admin meant to deliver. And Galaxia is meant to be used this way. This, for a system that try to store all in a database so the admin should not dig in text configuration files is a bit confusing. Maybe an scripting language over tiki features could add flexibility to the suite, make Galaxia more integrated and natural to use, it even could be an existing scripting language with some predefined (hidden) headers, a very simple api, etc, like i.e. Zope uses Python or things like that, and those scripts be in the database as main storage, and on demand written/cached on the filesystem to be run.

Random user base vs editing team: the system enables that anyone can register to the system, but this, added to the ability of editing wiki pages, and users be able to have their own home pages could make problems, discussed with other name problems.

Name problems

I talked about it several times, but for completeness of this article I will include that here.

There is a dual view on when things have a name (i.e. a Wiki Page) and a number (in most objects, the name is little more than a comment attached to it, referring to them should be done with a number). The system navigation could be maybe more clear if functional names, maybe with the Wiki style of names, are used, is not the same to bookmark or read tiki-view_forum.php?forumId=3 than ?forum=Developers.

For the things that have names, there is a problem because all have a flat namespace. Two pages can't have the same name, so if two or more users want to have an AboutMe page, they can't. If two documentation projects, maybe different categories, are in the same system, then they can't have both an Introduction page. Even a simple user can create a page called AboutThisSite, so later the admin can't create it and it will look as the "authoritative" page that describes the site. This could be fixed adding an implied prefix to the names, i.e. userlogin.AboutMe or mydocproject/Introduction, and for simplicity they could be assumed if they are in the same context (the others user or category page could call the name without the prefix if it is the same).

Me too problems

Tiki has a lot of features, and many of them are modeled like the top applications for that feature, i.e. for forums or image galleries. This has the problem that put there the features of a stand alone system without taking in account what is already done and available in the system, and leads to a low integrated system with a lot of things that can be reused or derived from and are not, and a system harder to use as it have a bit different user interface than the rest of the system . In TikiIntegration2 i put some examples on how could be a more integration between Tiki features, but the point here is try to see and develop the system as a whole, not just glued up parts that looks like other standalone systems.

There are things that look and/or work alike that could be in some way consolidated. Image and File galleries could be more similar or at least have a common code base, the same for articles and blogs, or forums and comments. Text entry could have the wiki syntax all over the place, image and file attachments could have the galleries code behind, and the same for comments.

As a "political" note, I think that Tikiwiki should have its own identity, its own way to do things, not because it should look different from other systems, but because its own set of features, that could make more easier or meaningful do things in a way different than most alternatives.

Consistency problems

Tiki have a lot of different features, so is natural that have several users interfaces. But differences goes a bit deeper, and that make the system harder to understand as a whole for users.

The first example that comes to my mind is the text entry (at least, for text boxes, not one-line entries). In some works the wiki syntax and in others no, in some works html, in others must enable it specifically (or even can be disabled by admin), and probably in others none of this is allowed. Also the editor that at the very least have an option to explain i.e. the wiki syntax is not used in all places where wiki syntax is allowed. File attachments, image insertions/uploading are good candidates for universally enable in all text entry, but could be a lot more, like the CopyrightManagementSystem .

But there are features that apply for some kind of information and not for others that look or behave in a similar way. Security is one of them, you can put security in individual user pages, but in some information types things don't go so deep. But there are a lot, like
printing to PDF that don't work for things that are not wiki pages, or features that works for forums but not for comments, and so on.

The basis of this is that what an user learn for a section of the system works for every other section, of course taking in account the particularities of that section (i.e. for objects that are not editable is worthless to have an history of changes), conceptually similar tasks should have similar features, dialogs, behaviour, etc.

Consistency also means that everything works in a similar way wherever you are. In forums when you write a post start a thread, creates a new level inside the form, when you reply to it, you reply comes in there, but if you reply a reply dont get a subthread, but it belongs to the same level than the previous thread. Following replies in comments also have this one-sublevel-fits-all problem.

Permission System Problems

The actual permission system defines global actions for the system but enables to add permissions to objects that overrides the system permissions. In mostly open systems where the access is mostly enabled for all but you want to disable access to some part to someone, but things could be less intuitive when all is closed and want to open some section or add some extra permission to and user or group to particular objects, i.e. if group A (and maybe more) can see/edit wiki pages in general and group B (and maybe more, again) no, but if you add that group B can see/edit a particular page, group A could not see it.

The permission system could have a redesign soon if the people behind PhpGaclDev comes with a happy ending, but whatever implementation it gets I would hide the implementation details from the user interface (i.e. no tiki_p_view, but maybe "View", so future implementations are hidden from end users) and the system do the conversions/adaptations to the object and security implementation behind) and will choose a semantic more intuitive like have "implicit" groups like Everyone, have an order between permissions, and have the global permissions attached at the end of object permission, this way I could say in the previous example for the page that group B have edit permission, and for everything else global permissions, like group A can edit, and Everyone not.

UserPagegmuslera