TikiDesintegration

In TikiIntegrationDev I talked about how to make things more integrated, for the point of view of the user. Here I will talk about some ideas about taking out pieces of tiki to external modules, that could be distributed separated from Tiki.

The idea behind this is to make Tiki to support modules, and move there most of functionality, and leaving behind a somewhat minimum core, enough to ensure that it is Tiki, loading of modules, interoperability, things like that.

Why do this?

- Size: By now Tiki is huge, even without the documentation. Even the proposed "lite" version is 2+Mb in a compressed format, and the trend is that it will grow.
- Speed of change: in 9 months it evolved a lot from 0.9. The most notorious changes are the "big" features, surveys, trackers, workflow, calendars, mytiki, and themes, that seems to not be very integrated with the initial core. The new features in non-core modules could be responsible for the high frequency of releases.
- Developers: this could make Tiki more open from external contributions
- Different speed of development: modules can have a speed of change separated from the speed of change of the core. If something big changes in a module, there is no need to wait a new release of Tiki
- more to come

What can be in the core? What you can say safely that is needed for all possible modules and what could be replaced by a module?

- Users: User permission system
- Editing: Generic "wiki" editor/viewer
- Galleries: Generic file gallery
- Forums/Blogs/Comments: Generic threaded discussions
- Time: Generic Calendar
- Presentation: smarty, theme support, menus
- Other: Trackers

I think that over this most of actual tiki functionality can be built

There may be a few external modules bundled as example and to make Tiki usable by itself, like Wiki + some discussion system and a few things more.

How this can be done? I don't very fluent in how other systems enables the development of external modules, but some of my ideas is to have:

- extmodules/ the head of the tree of modules
- modulename/ have some subfolders, like its own tree of languages (translations for their own messages), themes (with its own modifications of specific global themes), db (the database addition/modifications for they own tree of templates, and the actual code.
- for each module, should be some communication protocol, some that say to Tiki functions, security categories, categorized entry points for integrating with other modules, menu options for the tiki menu, modules for the actual module, and it read from Tiki some state, the things that that Tiki supports to

match with its own capabilities, available functions to use from core and other modules and things like that. The idea is that for a module and a Tiki developed both after the external module protocol definition should not be version problems, all should be interoperables.

With a bit of luck, maybe even can be done a second layer module that converts modules for other programs to Tiki, talking with the other program in the protocol it is done for, and with tiki with their own protocol, and win a lot of functionality making available to tiki a lot of existing external modules.

Of course, this ideas are very "green", a lot should be added and knowing how this situation is already solved in other projects is a plus, but anyway, could be a nice thing to advance in that direction, have some idea on the communication protocol, start to see how to put out modules of Tiki in this format

---

See also

- TikiInstallFeatureDev
- TikiPackager
- TikiPackageRemover
