Intro

While developers may not all agree (any constructive opinions are welcome 😃 with the details in this proposal, is it fair to say that a Tiki Core and Extension API is badly needed? Since this is a time-consuming and difficult undertaking, we need to discuss this, come to a conclusion, and begin a new CVS branch to implement this ASAP. **The longer we wait, the more bloated and disorganized Tiki becomes, and the more difficult the core implementation!**

Tiki Core Components

Note: Core services do not offer any user interface — just API. All user interface managed by corresponding extension... or some core extension — it is extension like all others.

Extensions Management

- Scan for new/removed extensions
- Maintain extensions table — i.e. sync DB table and actualy present files
- Execute extension installation/uninstallation routinues
  - Install can include DB operations (adding new entries to existing or create new tables)
  - Also EM can install/uninstall packaged extension (what format?)
    [+]
  - Maybe some consistency checking? I.e. detect broken installation of extension package or core files and warn user.
  - Even more crazy: check signature (package or individual files) — for high secured Tiki sites
- API for extensions
  - get list of installed/enabled extensions
  - is_installed('extension') / is_enabled("extension")
- There is no other way except core API for one extension to get (and work with) instance of another
- List of capabilities of extensions, like functions that give extensions to the system for the use of other sections
- Prerequisites (things that have to be installed to make it work, i.e. other extensions)
- Conflicting extensions/capabilities (something that would be nice to avoid, but probably is not possible, i.e. 2 extensions that give the same capability but in a different way, or use the same

table/field in a exclusive way)
- smth else?

Objects and Permissions Management

- Every extension may 'register' new object type it work with (can produce)
- Tiki Core Objects — can't be unregistered (existed even if no extensions installed)
  [+]
- Every object in system belongs to some category (possible more than one). By default it is equal to Root (Top) and maybe reassigned by admin later.
- All Tiki objects belongs to))GlobalObjectsTree((
  - There is special containers in this tree
    - Users/Groups — core objects represent users and goups. Assigning permissions to whole container mean that user granted rights to manage other users or gorups.
    - Installed Extensions — list all installed extensions here. Grant permissions mean that user can manage (configure) some extension (or all of them if container permissions applied)
    - Categorized Objects — application objects (like wiki pages, articles, FAQs, etc.) and categories. This is a categories tree in nowdays Tiki.
- Permissions can be granted on Root of GOT or any other node/leaf of this tree
- Permissions can be granted to user or group (role, smth else?)
- Permissions given on some container will be inherited by all containers and objects below
- Permissions can be revoked on some low level even if upper level grant such permisions... So we r talking about effective permissions != assigned.
- Maintain ACLs as internal run-time data structures
- Evaluate user permissions on given object
  [+]
- API for extensions
  - manage user/group permissions on objects or categories ...)/revoke(...
  - check user permissions (is_granted(...)), evaluate permissions
  - list of permissions with description
  - extend core rights with extension's addon
  - *object_type* (user/group) and *object_id* (maybe *name* will be enough) should be passed as arguments to all permission functions, so API can be used for 'effective permissions' evaluation (i.e. w/o relogin admin can check what permissions granted to another user/group — of couse if admin interface will allow this
- API for user management
  - create/delete users
  - enum users
  - get/set user info (core attributes like name, smth else...)
  - get/set user preferences — map key => value
    [+]
- API for goup management
  - create/delete gorups
  - list groups
  - add/remove/list members
  - get/set group info (core attributes)
- Work with core objects and permissions from extension should be possible **only** through API so future changes (even rewrite) of PM subsystem possible, with unchanged external API interface of couse.
- More effective instead of permissions is to use ***rights***...
  [+]
- ... and take Tiki to another level of ~~permissions~~ rights management

[+]

- Mass user operations
    - Admin (user with 'a' on users container) can set/change properties of more than one user at one operation...
    - ... so admin can assign any property to user(s)... from module in interface to change theme...
    - Is it reasonable to talk about 'policies' (like Windowz policies) applied to user?

Integrated Debugging/Logging Facility

- Need to have builtin debugging API
    - Good candidates are non-interface methods of Debugger class.
- Is smth like `strace` can be good? (to trace API calls sequence)
- Maybe some performance (profiler) couters can be implemented... to measure how long we execute 'core calls' and 'user calls'
- Is it reasonable to add log to Tiki?
    - Admin may want to see whats happening (in details) while he sleep — so Tiki may record user activity to log...
        - it may be not just 'wiki page updated' or 'forum created'...
        - but some admin option changed (to help to investigate 'who touch that checkbox'
    - Also (depending on verbose level) some extensions may spam info here...
    - Can be implemented as separate file (to helps grep/mail it or as DB table

Core Abstractions

DB Layer

- Handle queries to configured DB and return results to PHP
    - ***DB abstraction team*** plz append more details here 😃

Extension

- Every custom extension should inherit from))TikiExtension((class
- Can have install/uninstall procedures
- Can be enabled/disabled
- Can provide entries to dynamic menu
- Can draw extension configuration page when required, load and save configuration data
- Can draw 'module' (yes it is module box in nowdays Tiki) — it can be the only that extension do 😃— then))TikiExtensioncalledTikiModule((
- Can provide API for other extensions
- Can use API of other extensions (get another extensions instance before via core API)
- 'Core' Extensions — builtin extensions
  [+]

Extension Examples

[+]

Architecture notes

- Of couse OO used to implement all core services
- Class *TikiCore* contain no valued code! It is just ***facade class*** (see design patterns) for core components (like EM, OPM, DBL, ...) which is of couse classes too...

Design Notes

- It is needed to specify ***core CSS styles*** and ***common layout*** to make Tiki interface in one styled design. Common elements can be:
    - forms (dialogs)
    - tables with result records (with 'even'/'odd' TD styles now)
    - 'modules' (boxes at left or right)
    - smth else?
- ***WYSIWYCA*** — ***W***hat ***Y***ou ***S***ee ***I***s ***W***hat ***Y***ou ***C***an ***A***ccess — as defined by mose, and with what I fully agree...
    - Nothing on screen should appear that user can't click (or if click got error)
    - To implement strict permissions checking needed everywhere
        - checking ***rights***(TikiCoreWhisheslist) can be much easier 🙂

See also

- TikiDesintegration — more ideas from gmuslera
- UserPagezaufi — another bunch of ideas
- BeastRiderIdeas — Content Management from an Elder's Perspective
- TikiCorePrototype — random thoughts by zaufi on core
- TikiInstallFeatureDev — a package installer/upgrader (gongo)
- TikiPackager — a package creator (especially helpful for developers that want to make packages) (gongo)
- TikiPackageRemover — thoughts on how to remove installed packages later-on (gongo)