TikiCorePrototype

Overview

Below are notes I (zaufi) wrote during code *Tiki core prototype*. You may see the source code in **$(CVSROOT)/tests/core**.

See also

- TikiCoreWishlist
- TikiPackager
- TikiPackageRemover
- TikiInstallFeatureDev
- GongosViewOnCoreAndTiki

Implementing Tiki Core Prototype

System Init

- The main purpose is to initialize 'low level' components during core construct... It is like *runlevel startup scripts* in linux
- Initialization sequence can be extended by adding scripts into init.scripts directory
    - Name of script file should have the following format: *NN-name.php*, where *NN* is a 2 digit number used to define execution order
    - 00-name.php is executed first; 99-name.php is executed last
- Typical examples of such scripts:
    - add include paths for PHP
    - database connectivity and low level init
    - inherit and make smarty instance
- All of above will replace tiki-db.php (with local.php), setup.php and some *(most? -terence)* parts of tiki-setup_base.php/tiki-setup.php into well organized and logically independent ordered execution scripts

Tiki Objects Tree

- All objects in Tiki organized in the Tree
- Every node of the Tiki Objects Tree (TOT) have associated ACL (Access Control List)
- Unique key of the object in Tiki system is the pair of objectID and objectType
- Object type should be registered by special API
    - ID of object type assigned by programmer — not by the system (because objectType should remain the same after registering/unregistering)
- Any object in system should be in the Tree, else it is impossible to determinate ACL for that object
- core objectTypes like user and group cannot be unregistered
- There is API to manage objects in the Tree
    - Object Types API
        - register/unregister object type

        - list registered object types
        - check if given object type registered
    - Objects Management
        - add/remove object from the Tree
        - get/set parent for object
        - get/set 1st level childs for object
    - Rights (Permission) Management
        - grant/revoke one object's right to perform an action on another object
        - check if given an object has a certain right
        - examples of actions: read/write/lock/delete/undo/upload, etc.
        - not all actions apply to each object; some objects are not even "active"
- Consider using phpGACL as 'low level' layer

Core Object Types

- 'Core object types' mean that it is impossible to unregister such object types and they are exists just after installation. They are exists always ('till system exists .
- There is attributes of 'object type' entity in Tiki
    - object_type — numbered ID assigned by programmer
    - name — human readable name
    - description — hint what this type is.
    - can_contain — list of object types which can be contained (be child) of instances of this object type
- List of 'core' objects ...
  [+]
- Maybe sometimes in future we will implement full featured 'Tiki Object Schema' with inheritance relations among Object Types...

Core objects

- by default the only object of type 'User' is present... it called 'admin' — user with default password and admin rights on TOT top (root).
- another bunch of 'default' Tiki Objects is a core 'Extensions' objects like
    - Admin interface
    - Extension to handle User objects
    - Extension to handle Group objects
    - Extension to handle Container objects
    - Maybe 'Workplace Layout and Theme Control' can be extension(s) too
- To make 'core objects' uninstallable it is enough not to provide corresponding scripts so system can't invoke it and can't remove this extensions — just a little trick — of couse admin interface will show this like 'uninstallable extensions'
- Default location of all objects is top of the Tiki Objects Tree

Extensions Management

- Lifecycle of Tiki Extension
  [+]
- What registering of extension is?

    - It is the way to tell what an extension is ready to be enabled and configured — i.e. all installation procedures done and extension can be used
    - actualy happened: add record into 'extensions' table with the following info
        - main extension file name (i.e. *mymegacoolextension.php*)
        - extension class name
            - should be child of TikiExtension core class
        - version info (number?) — can be asked from class
        - description text (to display in interface) — can be asked from class
        - smth else?
- What is the registering of page?
    - assume that extension register page (URL) like '*tiki-name.php*'... this mean that core should create such file and place inclusion of core files and make a call to extension after some checks...
    - ... also extension needs to provide method name core should call if this page requested ...

... 2b continued ...