Workspace

Intro

Workspaces were added to Tiki4 and further improved in Tiki5. In Tiki10 a GUI was added for some basic features.

Workspace is a large project which may or may not impact Tiki as a whole. Previous efforts like AulaWiki built a parallel structure with highly useful features for those who need workspaces. The main issue with it is that the rest of the community lived on without any of the changes. This can be seen in two ways: optional features should not affect those who do not use it, or as a lack of collaboration on the project leading to little used, brittle code.

Many people already use Tiki to collaborate on projects as small teams without workspaces or AulaWiki. Be it through implicit trust that others will not play in their sandbox or creating multiple instances of Tiki. Both of these solutions currently have scalability issue. How can we improve the experience without changing everything?

This roadmap changes the fundamental question asked in development from:

~~What can I build to solve the workspace issue?~~

to:

What can we improve in order to get closer to workspaces? and **By improving *X*, what will those who have no interest in workspaces gain from it?**

Workspaces should not be about building another pile of code, but assembling some of the many existing functionalities. If it means improving every single piece along the way, that's what it should be. While there are deadlines in play, the project does not end with the summer and it would be better to have tangible improvements on existing functionalities, making workspaces closer to reality, than a new unstable feature.

The remainder of this roadmap will attempt to explain the incremental improvements required to Tiki in order to achieve workspaces.

GUI

Development of the Graphical User Interface (GUI) for workspaces started during TikiFestBarcelona3. A quick demo of the progress achieved by then can be seen here:

http://blip.tv/xavi-de-pedro/tutorial-on-the-new-workspace-gui-for-tiki10-august-2012-6323818

Issues to resolve

Workspace users only want to see what is relevant to their workspace

Information overload is always a problem. Even on small collaboration sites, the recent changes can quickly grow to a level where the list is too long for anyone to bother looking at. Category trees become so large that no one can find what they are looking for. Information has to be filtered. It has nothing to do with access rights, it's a matter of personal choice.

Perspectives allow the user to select which point of view he desires to have on Tiki. By changing the perspective, the user selects which workspace he is working on right now and which information he finds relevant.

By themselves, the perspectives don't do much. They are implemented by overriding the global preferences, thus creating multiple sets of global preferences. Actual preferences need to be implemented to filter the visible data and multiple components need to be updated to take into account the perspective's preferences. However, perspectives are entirely transparent. Only the preferences need to

be considered.

One such new preference that would be required is **category jail**, similar to the UNIX concept limiting what the user is allowed to view in the filesystem by hiding the higher sections of the filesystem tree from them. Adapted to categories, a category jail would allow to limit the visible portion of the category tree to a certain category.

A crucial component of the UI (that is not handled through preferences) is the modules. Module visibility is typically restricted by groups. Limiting the visibility of modules could either be done by changing the **active groups** or adding a **perspective filter** on the module. Both can be done without significant changes. In the first case, the perspective would contain a preference listing the relevant groups in the perspective and the list would serve as a mask on the user's group list when selecting the modules to display. The second one would add a generic parameter to modules and only display when the perspective is active.

Some contexts require privacy and to hide the work of a workgroup

Tiki has very fined-grained global & object permissions. In regards to category permissions, Tiki is currently (3.x) limited in terms of flexibility. The lack of fine-grained permissions make understanding the impact of granting permissions hard to understand. For this reason, a revamp of the **category permission system** is underway. The complete granularity will be available to categories.

This issue resolved, rethinking how large amounts of categories should be handled becomes possible. At this time, one of the major limitations of categories is that anyone with edit rights on a page (or the equivalent for other objects) can change categories on the page. In a world where categories are used for categorization, this makes perfect sense. However, when changing categories grants or revokes permissions, **category security** is required.

Ultimately, who is allowed to add an object to a category or remove one from it is specific to the category itself, not to the object. The permissions that apply to the categories must be sorted out and deployed in the code base.

Because **perspectives** may point to views in which the user has no rights, visibility permissions on the perspectives may also be desired.

Another issue relevant to security is **permission auditing**, which is to be able to view which permissions apply to an object, and from where they were granted. The revamp of the permission system will make this task easier, but interfaces to audit the permissions and compare them may be required.

Administrators want to ~~manage~~ delegate management of workspaces

In large organizations, creating workspaces will be a day to day task. Creating a new workspace has to be quick and easy. Through **data channels**, workspace templates can be created. Effectively, this would allow to configure a new workspace based on a local configuration. The workspace template could create a set of categories, a perspective, new groups and set-up all the category permissions required to get the workspace up and running. Data channels are not the only option to create new workspaces, however, creating a dedicated interface for this task is likely to be time consuming and should be postponed to later releases.

In fact, because data channels rely on groups to determine "execution" rights, it may be possible to keep the global administrator entirely out of the loop.

Afterwards, all that would be required is to add the users to the specific groups. Considering the data channel has a parameter to specify the workspace leader, the leader could then be responsible for adding the members. Adding and removing members from groups currently requires administrator privileges (tiki_p_adminusers) at the instance level. In an **emergent group** context (aka Organic groups), this is

unfortunate. To resolve this, finer grained permissions on groups would be required. Effectively, this would require treating groups as object themselves and to grant permissions on them.

Workspaces need to have a life of their own

Especially for large workgroups and emergent contexts, it's important to reduce the effort required by the group leader(s) and to let them delegate tasks within the workspace. It should be possible to delegate simple tasks (like approving new members or suspending troublemakers) to moderators. While it would be possible to simply grant member administration rights to these people, in many cases this would seem too wide. The introduction of **group transitions** would allow to specify paths between two groups and to grant rights on the transition to a group of moderators.

It should be implemented with perms on group
These transitions would simplify the management of workspaces and define community-wide policies on how to handle workspace management. By reducing the complexity of group management, you allow less technical people in the organization to lead a workspace.

Similarly, the concept of transitions may be adapted to **category transitions**, which would enable workflow-like patterns for document management. These may be required when coordinating specifications between multiple independent entities where an approval process is required. Just like normal categories, categories part of a transition set would have permissions assigned to them, limiting the ability to edit for example, and permissions on the transitions themselves.

A sample use case would be the approval of engineering documents.

Workspace leaders may need to customize the configurations locally

Perspectives allow to override any preference. Because they are created through profiles, the burden of selecting which one is appropriate is left to the administrator. A perspective management UI would narrow down the set of available preferences. The same way the administrator does not really want to handle the day to day management of the workspaces, he may want to delegate some of the configurations of the perspective to the workspace leaders. Such configuration may be enabling or disabling the forums, changing the theme or any other relevant configuration.

To be done efficiently, Preferences would be required. Essentially, the type of field and validation rules for each preference have to be defined at a higher level to be able to generate dynamic interfaces from a list of preferences without having to duplicate and maintain large amounts of code.

Features

The previous sections introduced the features and how they fit in the global picture. This section details each of the features, their impact on the rest of the project, the dependencies among them and their current state. Ideally, each of these is seen as a separate development effort and provides a worthwhile improvement to Tiki as a whole.

Perspectives

Perspectives were introduced in trunk on July 19th. They are in three parts:

1. Application of the preference overrides in tiki-setup.php
2. Creation of new perspectives from profiles
3. Perspective switching module

These three components provide a usable base to work from. The only change required for workspaces is the introduction of a *tiki_p_view_perspective* permission currently pending the merge of *perms-take2* in

trunk. The introduction of the permission would only affect the perspective switch module and the companion tiki-switch_perspective.php file. No changes are required in tiki-setup.php or in the profile. Adding permissions on the perspective would be handled through the standard object permission handling in profiles.

Outside of workspaces, perspectives would be useful to create micro-sites with a different visual appearance (with Site Identity preferences, not just Theme Control Center), the introduction of an Administrator perspective to allow the site's administrator to switch between a normal view of the site and a perspective that would contain more administrator controls.

In the future, a better interface to manage perspectives may be desired. Ideally, the perspective itself would define which preferences can be updated within it and a customized interface could be built through **Preferences**. Combined with the idea of delegating configuration, two additional permissions could be introduced: tiki_p_edit_perspective and tiki_p_edit_perspective_full.

Breakdown

- Perspective base - 4.0 **done**
- Perspective view permissions - 4.0 **done**
- Perspective UI - 5.0 **started, depends on Preferences**

See:

- Perspective

Category jail

The category jail preference can theoretically be used without perspectives, however, the use is limited. It could be used to limit the visible categories to a limited set of content-related categories and hide the permission management ones. However, in the context of workspaces, the category jail is a pivot concept and enables the real purpose of perspectives.

Rather than a jail, it can be deployed as a suggestion. By default, it could display only the jailed categories, but an option could allow to show all categories instead.

The jail could also force new objects to be created to belong to the category as well. Wether this respects the user's permissions has to be evaluated.

Technically, the jail is implemented as a preference which will contain a category ID. Any object listing or category listing should restrict the displayed items to the category in the preference and child nodes. To speed up the lookup, the complete list of sub-categories could be stored in preferences.

The exact work required to achieve the category jail has yet to be analyzed. The creation of the preference is trivial, however, multiple components will need to be updated.

Breakdown

- Category jail - 4.0
  - Categorize.php/tpl to enable category jail on all objects **done**
  - Update modules listing objects, like recent changes **ready to work on** (see Deployment of Category Jail)

Updating all modules is a significant task. It might be a good idea to flag modules that are *jail aware* in the documentation of the module. Enabling category jail on the module may also be an option.

Reference:
Updating all modules

Perspective filter

Optional: Alternative to **Active groups**

The perspective filter is a local modification in the modules execution path. There are already multiple module arguments that can be used to filter the visible modules, including pages, sections, themes and many others. Adding an additional argument and the handling is a localized change that requires little work.

Instead of filtering on perspective, it may also be possible to filter on categories and use the category jail filter. However, directly on the perspective may make it easier to grasp.

Breakdown

- Perspective filter - 4.0 **done**

Active groups (Cancelled)

[+]
Unlike the perspective filter, this approach reuses the group filtering on modules. Conceptually, it changes the groups the user is a member of to a subset of groups relevant to the workspace. If applied only to module filtering, the effect is the same as the perspective filter. However, if applied globally, it may change the meaning of global roles.

More analysis is required. There may be a dependency problem between the resolving of the perspective preferences and the resolving of the permissions.

Breakdown

- Local to modules - 4.0 **ready to work on**
- Global - Analysis required

Category permission system

The *perms-take2* branch modifies how category permissions are resolved. It provides the full permission granularity on categories and provides more efficient ways of resolving permissions.

These changes to the category permissions impact the permission management user interfaces. The current category permission interface is no longer suitable. Instead, the object permission interface is closer to being more suitable, but needs to be adapted to display the category permissions correctly when setting permissions on categories.

At this time, to remain efficient when fetching permissions, no inheritance applies to category permissions. Only the permissions applied directly on the category are considered. To accommodate this, the interface must allow bulk modification of the permissions to child categories. An alternative would be to update the database schema to allow efficient lookup of parent categories. This can be done with a parent category table that needs to be updated and kept consistent with the hierarchy or by entirely modifying the category structure to nested sets.

Breakdown

- Permission lookup - 4.0 **done**
- Update of the permission interface - 4.0 **in progress** jonnyb, luciash
- Category permission inheritance - 6.0 **pending category structure redesign** *Inheritance can wait for 6.0, but copying perms from parent to children, or at least from one category to multiple others, cannot - this is a fundamental requirement of category functionality and has been done in 5.0*

Related wishes

WYSIWYCA for all permissions : feature_check in Table: users_permissions
Mass assignment of permissions, especially for wiki pages
Better/Easier reporting of item/object permissions which override category and group permissions
When creating a page, how to inherit permissions from source page?
Item/Object perms: copy permissions from another object. (especially for wiki pages and categories)
Permissions: when assigning permissions to item, an option to start with current general permissions
Add green & yellow permission keys on tiki-listpages.php

Category security

One of the current limitations of the category system in Tiki is that only the administrator can modify the category tree. Workspaces need to allow for emergent category creation to suit the needs of the workgroup. Moreover, the visibility of categories is handled globally, which is well suited for small trees, but becomes impossible on larger trees. Filtering is required.

The solution here is to have category-specific permissions that affect the categories themselves, and not the objects contained in them. Possible permissions are:

- tiki_p_view_category to indicate if the category and its subcategories are visible to the user
- tiki_p_add_object to indicate if the user is allowed to add objects in the category
- tiki_p_remove_object to indicate if the user is allowed to remove objects from the category
- tiki_p_create_category to create new categories

Additionally, an object specific permission may be desired

- tiki_p_modify_object_categories to lock the categories on an object altogether, regardless of the permissions on the categories.

This change mostly impacts the categorize php/tpl component which allows to change the categories on the object. Effectively, it will need to be updated so that visibility rules apply, but also so that the permissions on each category applies. For example, if a user is not allowed to remove a category, adding or changing other categories would not affect the one category he is not allowed to modify.

Permissions like tiki_p_create_category would also need to be scoped to the category that grants it. For example, if a user has the permission on Workspaces > Chemistry > Autumn 09, he would only be allowed to create categories under it. Because there is no inheritance in phase 1, only the category itself has to be considered.

Breakdown

- Visibility - 4.0 **done**
- Object management - 4.0 **done**
- Category creation - 4.0 or 5.0 **ready to work on**

Permission auditing

As a companion to effective permission management, the ability to quickly view who has which permission on which object allows to increase confidence in the system.

Permission auditing can be seen either as a dashboard for administrators and workspace leaders and as additional information presented on the object's permission page. Within the permission page, allowing to display permissions from different levels, like category level or global level, would allow the permission assigner to have a baseline on the permissions to grant and make educated decisions.

It must answer the following questions:

- Which of global, category or object permissions apply?
- If category permissions, which categories provide permissions?
- Are permissions more open or more restrictive than the parent level?
- Compared to another object, are the permissions more open or more restrictive?

Breakdown

- Displaying information in edit permission interface - 4.0 **ready to work on**
- Dashboard and comparison - 5.0

Data channels

Workspaces are a combination of multiple features providing an experience of workgroup and focus, not a feature in itself. While it may be possible to create different interfaces to set-up the groups, categories, permissions, perspectives, transitions and all sorts of objects, using profiles greatly simplifies the task and provides a good starting point to prepare workspaces and test out the improvements to be made.

A certain load of work is required by the administrator to set-up the data channel and profiles initially, but once the template is defined, instantiation can be made multiple times and easily. Typical workspace templates can be distributed through profiles.tiki.org.

Profiles have been mostly tested out and are stable. Data channels on the other hand are in their infancy and have a single use case at this time. Some modifications may be required to get them to work correctly with all datatypes. Some conflicts may occur in the handlers when called multiple times.

A convenient interface to call data channels and input data would be useful.

Breakdown

- Create templates as profiles - 4.0 **ready to work on**
- Data channel UI - 4.0 **done** PluginDataChannel

Reference:
http://profiles.tiki.org/Data+Channels

Organic / Emergent groups

As part of the workspaces, groups will primarily be created through data channels and follow a standard template. However, for larger workspaces, one may want to create subgroups to grant special permissions to a few people or simply to identify them (participants to a workshop, experts, ...).

Without emergent groups, the attention of the administrator is required to create new groups. Granting rights to create new groups allows for the possibility to create groups named in a way that expresses a much larger meaning, which may not be required. Consider someone creating a group called "Experts" for their workspace. Groups are defined globally, so the "Experts" group would be available globally, even though it was created for a subfield and a team of 10 people in a 2000 person organization. Just like category creation would be limited to the category on which the permission was granted, it could be possible to impose a prefix to the group based on which category the right was granted on.

Generally, it's important for a group leader to be able to manage users in his workspace. By treating groups as objects, permissions can be assigned to groups for other groups. For example, a lead could be allowed to add and remove members to a group he controls.

Permissions on groups

- tiki_p_add_member to allow someone to add group members
- tiki_p_remove_member to allow someone to remove group members

- tiki_p_join_group to allow someone to join or leave a group on his own
- tiki_p_remove_group to allow someone to destroy the group

Permissions on categories

- tiki_p_create_group to create groups under the category, although the group is not categorized itself

Breakdown

- Member management - 4.0 **done**
- Self join (modify current implementation) - 4.0 **done**
- Creation and removal - 5.0

Questions

- Should not be possible to have group perms on Anonymous group
- Need a link somewhere to assign perms to group (already works at: tiki-objectpermissions.php?objectId=Registered&objectName=Registered&objectType=group&permType=group)
- Do these group perms apply globally/in a category as well? (or just on specific group?)

See also:
Organic Groups

Group transitions

(As reference, see graphic above)

Adding and removing group members to promote them is a tedious and error prone task. By introducing transitions between groups, which can be triggered based on a permission, group administration can be delegated in a safe way and reducing the complexity of the task.

This feature is useful for self-managing groups within workspaces, but it could also be used to simplify the code in the Tiki registration process and the multiple approval and validation steps. Correcting the blocked validation states would simply be a group change for the administrator and it would allow installations to customize their validation process.

This feature would require the introduction of a new permission to trigger transitions, a table to store the transitions and some user interface modifications to allow triggering of transitions.

Breakdown

- Support for transitions - 4.0 **done**
- Deployment of transitions in the registration process - 4.0/5.0

Category transitions

Category transitions are an extension of the group transitions, the same concept could be applied to categories, allowing groups of users to trigger transitions on objects, effectively allowing them to change the applicable categories on the object even if they would not usually be able to change those categories due to **category security**.

Depending on how it is implemented, it could share the implementation with **group transitions**. This could evolve into an Approval Workflow.

Breakdown

- Category transitions - 5.0 **done**

Dynamic preferences

Preferences are not a workspace feature by themselves, but they would allow to enable for the workspace leader to customize the perspective's configuration. However, the scope of this feature is much larger and long term. Globally, this feature would allow:

- The simplification of the current administration panel's code.
- The creation of restricted administration panels with a subset of options to create lesser administrators. For example, a wiki farm administrator may want to only allow Tiki administrators to enable stable features
- The creation of a configuration search for those times you know what you are searching for, but don't know where it was arbitrarily placed in the administrator panel.
- The creation of a perspective configuration panel.
- To build the preferences in a profile editor.

The downside is that Preferences are a massive task. The type of field and validation rules for each preference have to be defined. A prior initiative called Magic aimed to do this. However, complexity of the code and departure of the effort's lead caused the initiative to abort. A CSV file contains some of the information required, but is now outdated.

Breakdown

- Define general structure and interface - 5.0 **in progress in trunk for 4.0**
- Document features / import data - 5.0 **mostly done**

Category structure redesign

In order to support permission inheritance in category permissions, which would simplify administration, modifications are required in the category structure. Without such modifications, it is not possible to fetch permissions efficiently. One of the alternatives is to create a table to contain the relationship from each category to all its parents, allowing to join categories with it and obtain all parents in a single query. The alternative is to change the structure entirely and use nested sets instead.

The conversion to nested sets is a much larger effort, however, it would allow for much more flexibility in the future. A similar change would also be required in structures.

Breakdown

- Conversion of the category structure - 6.0

Short term notes

Jotting things down to remember to do

- "Default category assigned to uncategorized objects edited by a user with this default group:" in tiki-admingroups.php should be reworded and transferred to perspectives
- http://demo.tiki.org/trunk/tiki-searchresults.php?highlight=cgcom&boolean=on&search=Go shows a result when logged as admin but not as anonymous. Yet, this file gallery has view perms for Anon.

Related links

- Workspace Helper
- Workspace Ideas

alias

- Workspaces RoadMap
- Workspace RoadMap

- Workspaces