Addons Cleanup Background

August 2019 Update: The Addons code clean up part is completed and documentation for what has been moved to Packages is at Packages that extend Tiki. Roles is currently being worked on and the work in progress will be demo'd at the August 15 roundtable meeting. Associated to Roles is the new Templated Groups feature (of which Organic Groups is a subset, since not all Templated groups are Organic, which means able to be created and managed by users, as sometimes they are created by admins, e.g. Classes - students and teachers). All work will be complete before October

Thank you for all your comments. Taking into account these comments, and also Ricardo has spoken with Jonny, the work is now underway.

The Addons (doc site: Addons) feature was originally developed by Nelson as part of a project that he was doing for his Masters program with a vision towards an Addons Marketplace where developers could build **custom code extensions to Tiki** to share or market depending on the license chosen, with the goal of attracting more developers to Tiki. But the Addons Marketplace never happened when Nelson got busy with work after graduation. Nevertheless, the company where Nelson works at uses Tiki and has been using the Addons feature internally as it finds the functionality useful.

But because it has not been used much, it has been put on the Endangered Features list and its removal has been added as a TODO for Tiki20. The rationale is not only to avoid maintaining code that is not used much, but also to avoid providing too many ways of Extending Tiki, as that can be confusing to users of Tiki.

Since the company where Nelson works at uses Addons, he had a look at how useful functionality can be preserved while the Addons feature is removed. The result of that earlier exercise is a now outdated Addons Cleanup Original Requirements which was discussed by Nelson with Marc in December 2018, and then subsequently with Ricardo in February 2019 to see if he could do the work after Nelson secured a budget from his company to fund it. After more in-depth discussions with Ricardo who did an investigation into the Addons code and also looked at some of the Addons that were developed at Nelson's workplace, the outcome is this document requesting comments from everyone.

Overall Strategy

First, we consider the general topic of Extending Tiki. Right now, if you want to modify the way that Tiki looks and feels, it is clear that you use Themes. If you want to add third-party **code**, then Packages is the main way.

However, even though Packages is great for adding code dependencies for Tiki, it is lacking bindings with Tiki core to facilitate installing **custom code extensions to Tiki** without introducing hacks that touch Tiki core folders and files.

Addons contain functionality that is really useful even if we forget about the idea of an Addons Marketplace. This functionality adds to the ability to easily install **custom code extensions to Tiki** without introducing hacks that touch Tiki core folders and files.

Therefore, it makes sense to remove Addons but save whatever functionality that is useful by moving it to Packages. At the same time, there is also some useful functionality in Addons that is related to Profiles which will be moved to Profiles core.

As part of the AddonsCleanup project, the Packages feature as well as other related existing (but still poorly documented) features such as Theme Install will be documented thoroughly. Addons documentation will be removed. Nelson will be dogfooding this by converting all the Addons that are in use in the company where he works to Packages.

As for the vision of the Addons Marketplace, I think it will have to be put to rest for now. However, if in the future it makes sense to revive the vision it can always be done as a Packages Marketplace, which actually makes much more sense anyway since Packages uses Composer which is a standard for PHP.

Details

The following describes each of the functionality that currently exists in Addons and how it will handled as part of the cleanup.

Ability to specify dependency addons

In Addons, it is possible to specify that an addon is a dependency on another addon. Packages is using Composer which already support specifying dependencies.

To remove this Addon functionality together with the removal of Addons

Ability to version Addons and keep track which version is currently installed.

Packages uses Composer which basically does this. Addons has a db table within Tiki that keeps track of which version is currently installed but this would not be needed using Packages.

To remove this Addon functionality together with the removal of Addons

Ability to run profiles when custom code is added

Right now, this is possible with Addons but not with Packages. When you are adding custom code that extends Tiki, you will often want to set some Tiki configurations or create some Tiki objects that the code depends on using Profiles. It is very neat to be able to package together certain profiles to be executed when custom code is introduced.

Extend Packages to support packaging together Profiles that execute when a package is installed. There should be a way to specify which Profile to execute as the main one (secondary profiles can be included in the main one using existing functionality mentioned below). The installer should not re-run Profiles already executed.

How about make it that profiles launch packages? Profiles already have tons of logic, and extensibility.

I agree, adding a Packages Profile Handler would be a far lighter way of achieving much the same, especially bearing in mind the versioning of Profiles proposed below

Ability to install a bunch of profiles together

This functionality is already possible within profiles itself. You can already link to other profiles, as part of one profile: https://profiles.tiki.org/Test_All_Features#Including_a_bunch_of_profiles

> To remove this Addon functionality together with the removal of Addons

Ability to check if a profile is meant for a particular or minimum Tiki version

Right now this is not possible outside of Addons, but it is useful.

> To make it possible to add meta information into profiles. This meta information can include specifying particular or minimum Tiki version(s) that the profile can be installed on. The profiles installer will do this check before installing a profile

This would help cleanup profiles.tiki.org which currently uses categories for this.

I suggest a new section in profiles (mandatory going forward) - something like:

```
profile:
 version: ^20.0
 type: test
 maintainer: jonny@tiki.org
```

Use the composer type versioning, not sure yet what the type property would be for, and we may need others (category maybe, but not just if it's a duplicate of the page category)

Ability to selectively forget or reapply previously installed Profiles when upgrading custom code

Right now, with Addons, it is possible to specify certain Profiles to reapply or forget when it is upgraded. For security reasons, there is checking to make sure that these are Profiles that came with that Addon. However, there is an easy alternative to "reapply" which is to make a new profile that has the exact same contents as the old one, and "forget" is not really useful.

> To remove this Addon functionality together with the removal of Addons

Ability to specify Tiki objects to delete when upgrading

custom code

Right now in Addons, it is possible to delete objects that have been created by the Profiles in that Addon when it is upgraded. For security reasons, there is checking to make sure that the object being created was in fact previously created by one of the Profiles of that particular Addon. The workaround would be to manually delete those Tiki objects, or to do that using custom code. I think the workaround is acceptable in order to not maintain extra functionality.

> To remove this Addon functionality together with the removal of Addons

Ability to remove Tiki objects that were created by that Addon when uninstalled.

Addons can be uninstalled and it is possible to specify what Tiki objects to delete as part of uninstall process when this happens. For security reasons, there is checking to make sure that the object being created was in fact previously created by one of the Profiles of that particular Addon. There is already Profile Rollback that works through the Action Log which kind of serves that purpose.

> To remove this Addon functionality together with the removal of Addons

I believe *Profile Rollback* relies on the Tiki logs, not the Action Log. These get cleaned now and then or they grow infinitely. This is nice for trying profiles and rolling back, but I fear uninstalling an addon might happen years after the fact and the Tiki Log might have lost the relevant information

Ability to avoid unreliable hardcoded object IDs in custom Smarty templates and wiki page content

Hardcoded IDs are unreliable in custom Smarty templates as well as wiki page content (e.g. in List Plugins) as they might vary from one instance of Tiki to another. As such, for these to be able to work without modification across different Tiki instances, one should refer to Tiki objects using their Profile Reference rather than their IDs.

When Tiki objects are created using Profiles, the resulting ID - Profile Reference mapping is stored in tiki_profile_symbols. Addons currently provide a way for custom Smarty templates to resolve the correct ID from a profile reference.

> This functionality is generically useful for Tiki as a whole and so should be moved to Tiki core in a simplified form. A wiki plugin in addition to the Smarty function should be created so that it can be used in Wiki Page content as well.

Indeed, this is a challenge: Divergent Preferences in Staging Development Production and some work was done: https://sourceforge.net/p/tikiwiki/code/67442

Pretty certain this is possible already without Addons - there is a table for `tiki_profile_symbols` and you can access these from Smarty templates (i just forget how currently)

Ability to include templates, css, img etc within custom code packages

While it may appear that these things belong in a Theme rather than in Packages, there are situations where they indeed belong in Packages rather than in a Theme. This is because templates, css, img etc might be part and parcel of the **functionality** of the custom code that you are installing, similar to templates, css and img in out-of-the-box Tiki core. In contrast, the purpose of the Theme controls the **look and feel** of the site. The following is critical to understand:

- It needs to be possible to install and use multiple different Packages at the same time, since they add functionality, but you can only use one Theme any point in time.
- You might be switching from one Theme to another to change the look and feel (you can modify the CSS or even templates) in each theme but you would want the same templates in the custom code that you added to not need to be copied to every theme you are using.
- Multiple Workspaces in Tiki can use different Themes, but similarly to above you would want the same templates in the custom code that you added to not need to be copied to every theme you are using.

Note though there is already a new Theme Install feature that can install the following into themes from a zip file:

1. Do a database schema update
2. Install or update a theme in themes/ folder
3. Run some profiles packaged together in the zip file (understandably useful as you can create modules or menus or set preferences, things common in themes)
4. Install some .ini config files

To refactor everything into shared code for both Packages and Theme Install to avoid duplication. Packages will be able to contain Themes too, so it would be possible to install a theme through Composer by putting it in a Package if you want to take advantage of versioning or to package custom code with it. Themes Install will be retained as a simpler Composer-less method to install Themes from a zip file but it will not support custom code, as it is important that code is versioned (and Composer provides that).

Ability to include lang files within custom code packages

It would be convenient to be able to include a lang file within a Package, since **custom code that extends Tiki** for a multilingual site will always contain language strings that need to be translated. It is much neater and makes it unnecessary to separately provide instructions how to add language strings into lang/xx/custom.php when installing **custom code that extends Tiki**.

To move this functionality to Packages

Also see plans for directory revamp, everything custom should go in the `_custom` directory in the long run

Also allow installation of additional languages as packages. Once this is available, we can spectacularly decrease the size of Tiki releases by only providing the few languages with have more than 50% coverage (or another percentage) and letting people install more later as they need them.

Ability to add custom prefs to custom code packages

Not just setting preferences (which can be done through Profiles), but extending Tiki with totally new preferences that control settings for how the custom code behaves, etc. Actually, Themes might benefit from this too as you might want to add preferences within Themes to be able to set different look and feel options, e.g. "show something or not, different layout options", in a more fine-grained way than just using Theme Option.

Currently in Addons, there is an admin screen within the Control Panel that displays the prefs introduced by each Addon where you can set their value. By convention, each Addon should provide a pref that turns the entire Addon off in case the site admin wants to do that for whatever reason.

> To move this functionality over to Packages, as well as make it possible for Themes to be able to make use of this functionality too. To refactor into shared code whatever shared functionality between Packages and Themes to avoid duplication

PHP Namespaces handling for custom code packages

There is some PHP namespace (http://php.net/manual/en/language.namespaces.php) management functionality in Addons so that when you add PHP code you can't clobber any Tiki core variables. Right now it is very prescriptive how the namespacing is done based on the Addon name, and there is code that enforces using the right namespace. However, so long a people writing custom code follow best practices in using PHP Namespaces, there should not be a need to be so prescriptive/restrictive.

> To remove this Addon functionality together with the removal of Addons but make sure that best practices when writing custom code that extends Tiki is documented well

Smarty 3 has better variables `scope` possibilities, so it's not as bad as it used to be

Ability to have own isolated Smarty instance within custom code packages

As Smarty doesn't support any equivalent of PHP namespaces, this was an attempt in Addons to achieve the same kind of isolation for Smarty variables, so that when you add custom Tiki templates you can't clobber any Tiki core Smarty variables. However, this is not supported in Themes either so if it's in Addons only it kind of creates a two-class situation. Moreover, this is the kind of functionality that Smarty should provide really and shouldn't need to be handled within Tiki. Also, there is a workaround that developers can use which is to use exclusively variables like $custompackagename.variable within the templates that they introduce if they are paranoid.

> To remove this Addon functionality together with the removal of Addons but make sure that best practices when writing custom code that extends Tiki is documented well

Ability to add custom Events to trigger on certain actions

Tiki has a pretty powerful Events handling system. When adding custom code, it will be really useful to be able to bind to it. For example, if a certain tracker is saved you can run your own custom code to do some custom functionality, integrate with external systems etc.

To move this functionality to Packages

Can't this be done already via `db/custom.xml`? (need to check)

Ability to add custom Search sources

Tiki has a really powerful ability to index all sorts of Search sources into whatever search engine it supports, e.g. Elasticsearch. When adding custom code, it will be really useful to be able to index anything you want and in any way into search, essentially adding a "Content Source" to "lib/core/Search/ContentSource. Of course it is possible to manually add files there and hack the core to bind it manually, but this kind of binding should be provided through a stable API.

To move this functionality to Packages

Also think this be done already via `db/custom.xml` (need to check)

Tracker/Token based Groups that can have unlimited Group information and other Tiki objects that belong to it

This functionality is used right now to enable Organic Groups, but it can conceivably be used in endless other use-cases as well.

The following functionality is provided:

- You can store all sorts of information about a Group in the Tracker depending on your needs. These are just some examples: Group URL, Group Image, Group Descriptions (not limited to one), Group Address/Location/Map etc...
- It provides a way to conveniently link Tiki objects (e.g. File Gallery, Forum, Category, Wiki Pages) to a group as belonging to it. For example, "John's Group" can have "John's File Gallery", "John's Forum", "John's Whiteboard" (a wiki page), "John's Notes" (another wiki page), etc. that "belong" to it.
- Group names can easily be renamed by editing the Tracker. Once renamed, it automatically applies to everywhere it is used, e.g. if you have a forum that belongs to "John's Group" called "Discussion Forum for John's Group" and you rename the group to "John and Jane's Group", the forum will appear automatically renamed as "Discussion Forum for John and Jane's Group".
- You won't accidentally clobber Group names already existing on the system. If someone tries to rename "John's Group" to "Jane's Group" when there is already a "Jane's Group", if you have set up the tracker to

allow it, there will now be two "Jane's Group" and it will still all work. You can also easily prevent duplicate names by setting the "unique" validation rule on the Tracker Field for the Group name.
- It provides the possibility to add new functionality to script the removal, re-categorization or other batch processes of all the objects that are related to a group. For example, you can delete "John's Group" which automatically deletes "John's File Gallery", "John's Forum", "John's Whiteboard" (a wiki page), "John's Notes" (another wiki page), etc. all at the same time. Some of this functionality will be done as part of the project as the company where Nelson works needs it.

This kind of functionality is not restricted to the Organic Groups use case. You can use it even for fixed groups if you have a use-case where you want to have easily editable information for a group, or need to assign all sorts of Tiki objects (such as Forums, File Galleries, Categories, Wiki Pages) to "belong" to the Group.

Right now, Addons uses a Smarty modifier to transform tokens to display names. It would be better if tokens (once configured, perhaps through a pref to specify token patterns in use) should be automagically transformed whenever used in Smarty variables. That would be cool as then anyone can use this feature without having to customize any code or templates at all.

> To make the functionality generic and well-documented so that it can be used in other use cases, even if you are not introducing any custom code or templates.

Custom templates in the theme and Group Tracker can do a lot of this already, the rest (roles) should be added into Tiki

Ability to add custom Navbar(s) for each Tracker/token-based group.

In addition to the other Tiki objects mentioned above that can be linked to "belong" to a Tracker/Token-based group, it is useful for each group to be able to have Navbars that can be used in Custom Smarty templates or elsewhere. For example, the Navbar for "John's Group" might have links to "John's Forum", "John's File Gallery", etc. Right now though, Addons do not use the Menu feature to implement Navbars but instead expects you to design the Navbar in a custom Smarty template. This is not good. It should just allow linking in Tiki Menus instead.

> Refactor to use Tiki Menus instead

All Tiki templates can be overridden to achieve similar functionality

Ability to easily create variants of shared custom code extensions to Tiki

The Organic Groups feature is a good example of something that everyone who is using it might want to customize it in their own way. As such, it would be more amenable to keep it as a Package (right now it's an Addon) rather than have it in Tiki core. The challenge is to be able to make it as easy to install as possible. Either we could push it to packagist (or composer.tiki.org) or give the instructions (add url to composer.json pointing to git) for people to install the composer package. Right now, I'm leading towards composer.tiki.org as it makes it official and potentially opens up possibilities for a UI that can look there for such packages to install. However, access to composer.tiki.org is limited right now as there is no easy way to contribute a package there. Packagist would be my close 2nd choice. Any ideas?

Convert Organic Groups from Addon to Package. To decide where to host it (composer.tiki.org? Packagist? Other?)

I agree that we should have a common and documented way to deploy and override PHP code. Here is what I know of:

- "Currently any custom PHP code required for your site can now be added to _custom/lib/setup/custom.php."
    - https://sourceforge.net/p/tikiwiki/code/HEAD/tree/trunk/_custom/readme.txt
    - And at https://dev.tiki.org/File-and-directory-structure-revamp, it is mentioned "src/ (for most of the .php files (no consensus: tiki-read_articles.php, etc renames as read_articles.php)"

But I disagree strongly to convert Organic Groups to a Package. **Because** it touches on many things and it's configurable, it **should** be part of Tiki core. Non core Organic Groups in Tiki have never worked. I have had this discussion with various people for over 10 years. Everyone who has tried has failed:

- AulaWiki with Javier Reyes
- TikiProject with Damian Parker
- Organic Groups Addon Proof of Concept with Nelson Ko. Nelson wrote "Unfortunately the Organic Groups feature will have to wait till tiki 16, as we plan to do implementation of Roles feature in Tiki before pursuing organic groups again. Organic groups addon is currently not ready for production use, as it will be undergoing major revamp after Roles feature is developed. The dependency is important to ensure long term viability of this project because without core Roles capabilities in Tiki, organic groups will be a nightmare to manage.".

And even if you did succeed in having Organic Groups as a Package. It would be utterly fragile. Inexorably, some of the features there will be added to the core. And then, you will end up with code/data which diverges from Tiki. As explained at https://community.redhat.com/blog/2015/03/upstream-first-turning-openstack-into-an-nfv-platform/, it's will be harder and harder to maintain.

So the first thing to do is solve roles / organic groups for the base / common use case. Have one common solution that everyone can use out-of-the-box. As a reference, we can look at what was done in the Tiki community (ex.: https://profiles.tiki.org/CartoGraf_15), but also beyond, with projects such as https://elgg.org/showcase/ And then, it makes sense to override / extend via themes, profiles, Smarty Templates or even PHP code.

Here is some related work: https://sourceforge.net/p/tikiwiki/code/42685/

Similar ideas to organic groups

- https://profiles.tiki.org/CartoGraf_15 is sort of like organic groups of users around maps. This organic group around an object type is interesting. This could be extended to H5P objects, Diagrams, etc.
- https://doc.tiki.org/Workspace-UI Combines multiple site features to create a workspace experience for workgroups.
    - Introducing workspace templates, which don't have much information yet, but adds non-admin permissions for creating workspaces