Tracker Roadmap

Trackers are one of the most important features in Tiki. For many of the larger installations, they are a central component. Over the years, the user interface was refined and the end result is a good feature to build on for many of the Tiki users. Many ran into issues by pushing them too far, and too often, the issues could not be resolved. For all of the extensibility and flexibility available to the end users, the code is monolithic, rigid and brittle. Few developers have put enough effort to understand the code well enough to fix even minor issues reported and it likely turns away many others.

It has been known for a long time that important refactoring is required in the code. Many venues were suggested. Most often, a complete rewrite is proposed. However, a complete rewrite would not allow current users to migrate their installation because there is no way the behavior could be preserved. The code is the specifications, and there is too much of it to replicate. This roadmap proposes an incremental approach based on moving the codebase towards a better design.

Considerations

Deprecation

Just like the Wiki feature, the extensive use of trackers lead to extensive list filtering options. The rich functionality is a blessing for the users, but the SQL queries generated bend the relational model too far and cannot be executed efficiently. While they might work on smaller installations with little data or traffic, larger ones will simply crumble under the weight.

The Unified Index can perform all the required filtering without causing abnormal stress to the database server. Most of the stress on the server is caused by simple queries done during the indexing period. To obtain the same functionality provided by the trackerfilter plugin, some work remains to be done in the user interface, but the following refactoring will ease this work.

The trackerlist and trackerfilter plugins should be considered **deprecated**. As they act mostly on the database directly, they should remain functional for a transition period, but moving to alternatives must be the recommended option.

To be discussed : For long-term compatibility reasons those plugins might be kept but their code would, in this case, simply call the LIST plugin with the right syntax. There is also another argument to keep them : the user interface. To define how the list will be rendered and with which fields, it will probably still be easier to use their own plugin editing popup until there is an elaborated Wizard/GUI to do this for the LIST plugin.

- LIST's parameters are easier to understand than the trackerlist ones. There are less edge cases to understand.
- There won't really be a direct mapping for all the parameters in trackerlist. The filters simply work differently.

Freeze

Some work was performed in Tiki7, but much of what remains will be performed in trunk. Large changes in trackerlib for Tiki7 need to be avoided as significant changes will happen in trunk. Minor changes in the handlers can be performed to fix issues, but outside of those. Changes must be kept minimal to avoid merging pains.

Already, fixes made for Tiki6 are hard to replicate in Tiki7 and trunk as the code structure is now very different. Two distinct patches are required. As much as possible, large sites using trackers should be encouraged to test with Tiki7 and migrate to the newer versions. Some issues are expected along the way, but support for older trackers should be limited to critical issues to avoid resource split.

What was done

During TikiFestBoston and the weeks that followed, initial work was made on the trackers. As a first step, trackerlib was converted to use the database helpers when possible, cleaning-up significant portions of the code. As a second step, the rendering aspect of the fields was completely replaced. What was previously *tracker_item_field_input.tpl* and *tracker_item_field_value.tpl*, along with various initialization code scattered in multiple files, was entirely removed and replaced with handler classes for each field type. The logic did not change significantly in the code. The effort was only to remove duplication and collect the logic specific to each field in separate, isolated, entities. The weeks following the sprint unveiled multiple issues that were caused by the refactoring, but many issues and inconsistencies were resolved along the way.

Those changes have been made before the release of Tiki7 and will be used as a foundation for further work.

Phase 1: Continue extracting field logic

Still, a lot of work remains to be done. During the refactoring, very little of the processing in trackerlib has been removed. Many methods still contain field-specific logic, especially in item saving, retrieval, import and export. Those methods need to be refactored further and delegate much of the work performed to the handler classes.

In many cases, those methods use queries using joins to extract the tracker field information they need and the data simultaneously. It should be noted that there are now facilities to extract the tracker information and fields in a single entity that is preserved during runtime. Joins are no longer required as the needed information is likely to be already retrieved using simpler queries.

Along the way, some of the logic in item storage should be extracted to events to reduce the complexity of the method.

The benefits of extracted field logic:

- A cohesive structure for the field types allowing to improve the user interface without breaking everything else.
- The ability to adapt the search filters based on the specific field types.
- To allow customizers with niche requirements to have their own types with specific handling for specific cases, and contribute back easily when desired.
- To allow enabling or disabling of field types.

Phase 2: Separate Storage

In too many areas, the data structures used in the code are plainly those provided by the database, and the arguments passed to functions that update those records are nothing more than the direct parameters. These issues apply for multiple aspects of trackers:

- Tracker definitions
- Tracker field definitions
- Item values

Using independent structures has multiple benefits for the code. Among others,

- Testability would be increased by adding the possibility to run test suites on trackers without accessing the database.
- Reusability would be increased as different features in the code could use the tracker definitions to

build forms.
- The separation of concerns would be clear and easier to understand for new developers and what is expected as a parameter would be more explicit.
- People using Tiki as a framework would have a better toolkit to build on.
- Encapsulated information prevents reliance on globals and the side effects that comes with it.

As the algorithms in trackers evolve to use those static structures, it would be possible to add even more functionality, like revision history. Considering the tracker definitions and items are under version control, comparing any version of an item would simply be about extracting the appropriate tracker structures and comparing them.

Alternate storage methods also become an option in the future.

Phase 3: Clean up

There are many flags and options on field types at this time. The purpose and implementation of these options will need to be revisited.

For example, many of them affect the default listings. Those values are part of a different concept: the view, which has a purpose, but should be separated from the actual tracker definition. There is no reason why it would not be possible to have multiple views for a single tracker and the current implementation of Pretty Trackers is a reflection of this. Just like pretty trackers, the unified search also brings new possibilities on how to format the results.

With categories and especially transitions, the purpose of the tracker status is challenged.

Phase N: Improve gradually

With the responsibilities better divided in the code, gradual improvement of the trackers will be possible. Adding new field types will not be as painful. Address fields and multivalued fields come to mind. As the search will be based on an independent indexer, it will not be required to keep the values clean in the database. Serializing complex values will be an option as the indexer will handle it properly through the field type handlers.

There is a lot of room for improvement which will be possible:

- Additional, higher level field types
    - ex: address which would be smart about multi-line addresses and zip codes...
- User interface enhancements on existing types
- Additional right management on whom can modify the fields, handled globally rather than with a series of exceptions
- Improved logging, notifications and reporting
- Improved connectivity to other features

Cross Feature Integration

MLP:The current integration of trackers and wiki could be strengthened and potential integrations with calendar, spreadsheet, users, forum, and article.
examples:

- the registration or user tracker is a common feature but the users table data is not associated with or incorporated in the tracker data. A revised tracker feature could be "always on" and tracker#1 is the user tracker.
- Calendar items are also forms with a different ui. Why not instead just have a tracker#2 associated with the default calendar objects.

An integrated tracker could have an ObjectID field type, effectively joining it to the table of wiki pages, articles, other trackers or items.

geoff: some detailed usage ideas on this topic are being added at Cross Feature Integration.

Closing Remark

How about changing the name of the feature altogether. The French translation already uses forms. The term tracker is confusing at best as the feature outgrew the original purpose a long time ago.

Names of similar apps

- Zoho Creator
- Google forms
- Drupal Content Construction Kit (CCK)
- JotForm
- DataGrid for Zend Framework

Related links

- Batch actions on the bug tracker
- BugTrackerCleanup
- bugtrackerv2_tpl
- Calendar to Tracker association
- Deployment of Category Jail for Trackers
- DevTrackerStructure
- Mathematical calculation tracker field
- Merge Tiki Spreadsheet into Tiki Trackers
- Mirror Trackers
- Project Management - Tracker Integration
- TKM Proposed Framework: Tracker Macros
- Tracker Ajax Services
- Tracker Calendar Timezones
- Tracker Field Types
- Tracker Field Visibility
- Tracker Issues 7.x
- Tracker performance
- Tracker Query
- Tracker Reports
- Tracker Roadmap
- Tracker Tabular
- Tracker to Tracker association
- Tracker-to-tracker Index List use case
- TrackerField UI Revamp
- TrackerFilter not working for Wishlist
- Trackers
- Trackers DB schema
- Trackers Examples
- Trackers Revamp
- TrackerToGanttChart

alias

- Trackers Roadmap