TrackerField UI Revamp

Background

I actually hadn't planned on tackling this until I started looking at Tracker 2.0 (conceptually, an overhaul of the coding and internals of the tracker system) next year, but as I began development after TikiFest Montreal 2008 on the Tracker Macros feature, it became apparent to me that the interface, as it currently existed, really didn't lend itself to easily supporting the addition of a new field type with complex input parameters.

The current interface has been a peeve of mine for a while anyway, with the combining of options in a comma-separated list in a text field...it's just not very user-friendly, and often can be confusing to inexperienced users.

Solution

While what I've come up with is still somewhat fluid in concept, the general gist of it revolves around a three point model that involves re-coding the field type structure to make it more scalable, re-designing the tracker field admin form to get rid of the text field for the options and replace it with graphical representatives of those same options, and adding a layer of abstraction so as to translate from the GUI options to store the data in the database in its native comma-separated list.

Re-code the Field Type Structure

The old structure had unused data in it, had some confusing data in it, and some of the help text I'd put in made things hard to read on some displays, so taking a suggestion from lph and taking a step back to consider what I wanted to do, I combined that idea with how I saw the form processing handled, and came up with the following:

```php
$type['<field_type_identifier>'] = array(
    label   => string,  // How type displays in drop-down.
    desc    => string,  // Description of what the field type is for.
    options => array(
        array(
            name    => string,  // Identifier for this option.
            type    => string,  // HTML form field type.
            default => mixed,   // Default value for the field.
            size    => int,     // Visible size of the field, 0=default.
            max     => int,     // Max size of the field, 0=default.
            prompt  => string,  // How the identifier is displayed.
            help    => string   // Help text for the option.
        ),
        array(
            name => string,
            ...  
        ),
    ),
)
```

_Caveat:_ I'm not sure the way I have the arrays setup will work as intended, but I haven't coded anything
yet to test this out. The idea is that 'options' => will be 'options' => 0, 1, etc.

Example of New Text Field Structure

/**
 * Field Type 't' - Text Field
 *
 * Array definition for the tracker Text Field type. This file gets
 * included in the TrackerLib class, within the field_types function.
 *
 * @package TikiWiki
 * @subpackage Trackers
 * @author Luis Argerich, Garland Foster, Eduardo Polidor, et. al.
 * @copyright Copyright (c) 2002-2008 All Rights Reserved.
 * @copyright See copyright.txt for details and a complete list of authors.
 * @license LGPL - See license.txt for details.
 * @version SVN: $Rev$
 * @filesize
 * @link http://dev.tiki.org/Trackers
 * @since Always
 */

$type['t'] = array(
    // Text that shows up in the field type drop-down.
    'label' => tra("Text Field"),

    // Descriptive phrase used in the help display to indicate what this field
    // type is for.
    'desc' => tra("Allows alphanumeric text input in a one-line field of arbitrary
    size."),

    // A list of all the possible options available for this field type.
    // Possible types for options are standard HTML form elements with an
    // additional option of CUSTOM. Custom types get processed with custom
    // code in the tracker scripts.
    'options' => array(
        array(
            // Standard HTML form element options
            'name' => 'samerow',
            'type' => 'checkbox',
            'default' => false,
            'size' => 0, // Ignored.
            'max' => 0, // Ignored.
            'prompt' => tra("Next Field on Same Row"),
            'help' => tra("will display the next field or checkbox in the same row if
    selected")
        ),
        array(
            'name' => 'size',
            'type' => 'numeric',
            'default' => '',
            'size' => 3,
            'max' => 3,
            'prompt' => tra("Visible Field Size"),
            'help' => tra("is the visible length of the field in characters")
    )

);
The first obvious implication of this is that some crucial code will need to be recoded in the tiki-tracker_admin_fields.php script, and in the .tpl file, which is fine since we're going to be making changes in there anyway for the better. The second implication is that by adding additional customization to these options, it reduces coding and special treatments, and makes it a lot easier to clean up the interface.

The custom option type does allow developers to step outside the box for complex options (like user lists, or the macro buttons) to process special cases differently than standard options requiring checkbox, text fields, numeric fields, or radio buttons. custom options can also entertain additional array elements to store custom data that needs to make it to the form. [eg. 'list' => $all_users or 'list' => $this->get_flags(true, true, true)]

As a result of some of these changes, and at the same time attempting to shore up the code documentation, I feel in the interests of code readability, that it is of benefit to refactor and remove the $type assignments from the field_types() function in trackerlib.php, and put them in their own files. Each assignment would then be replaced by an include_once() entry and comment explaining what the option was. This also makes it a lot easier to re-order the list of field types, or quickly remove a type from the list without deleting all of its code if its buggy. The caveat to this is cleanup, but as long as it is not impacting anything that's a smaller concern.

I have already begun to do this on my 2.5 experimental branch as a case-study to see how feasible this is.
In the end, it may be a bad idea to remove this code, but time will tell. On the upside, it's harder to pull out of something than to add back in, so undoing this is relatively easy.

While my testing is being done on 2.5, I am eager to implement some of these changes into an experimental trunk branch for folks to test and assist with.

Re-Design the tiki-tracker_admin_field Form

My initial redesign of the UI is two-fold: first to redesign the actual field options part of the display, but also to tweak the field list display so that it doesn't blow up the screen real estate and force you to use the horizontal scrollbar. Here are a couple of shots of initial changes I've made to the template. Note that these changes do not yet reflect the full GUI changes to the options, but only the general layout. Completion of converting the old field type structure to the new field type structure is required before the GUI changes can be implemented. Unfortunately the conversion is quite labour-intensive, so it will take some time. Volunteers are welcome. 😊

Abstraction Conversion Layer

As we now have graphical representations of our form options, replacing the old text field with comma-separated values, we have to keep in mind that the data being stored in the database has not changed, just how we come about that data. So for every option that is a checkbox, we need to convert that to a 0 or 1, for radio buttons they provide options beyond 0 and 1, and likewise for text fields and numeric fields. The order that the options appear within the field type structure dictate the order they appear in the database. So for example, in the Country Selector, we have two possible options: *name~flag* and *sort.* *name~flag* can be 0 to display both the country name and the flag (which is default), 1 to display only the name, or 2 to display just the flag. *sort* can be either 0 or 1 to sort by the translated country name, or by the english name, respectively. So we turn these into a radio option with three choices: Both, Country Only, and Flag Only, representing 0, 1 and 2. Sort becomes a checkbox, either off or on. When it comes time to save this field data, the abstraction layer converts this data and stores it as we would normally see it in the database as "0,0" or "1,0" or "2,0" or "2,1", etc.

It should be noted that *custom* options are not processed in this way because those custom options represent special data and will get treated as such, which hasn't changed.

Critical Steps

- Finish new data structure. (need to do: fjixhSerlwmMqUGsCpA)
- Incorporate new data structure into tiki-tracker_admin_fields.php and template.
- Finish work on handling "custom" options.
- Test, test, test.
- Incorporate display nuances back into standard class elements, or create new standard class elements. (I am by NO means a CSS whiz, so anybody with mad CSS skills that can help with the aesthetic appearance of the tables and such and make things standard CSS and not my spaghetti CSS would be a Godsend!)
Who

This is a list of people who have signed up to assist with this endeavour.

- Mike Kerr (kernel22)

alias

- TrackerFieldUIRevamp