Pretty Trackers

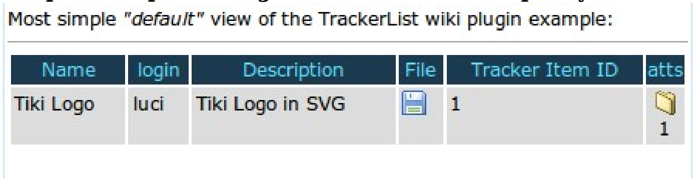
Tiki Pretty Tracker allow a site manager to design forms using tracker fields (profiled or not) as well as Tiki registration fields and have the results of each field displayed in customizable way on a Wiki page. The concept of a pretty tracker is similar to the functionality of Drupal's CCK (Content Construction Kit) or MediaWiki infoboxes.

e.g. Suppose you have a movie review site. You can create a tracker that asks for:

- Title of Movie (text field)
- Review (textarea field)
- Rating (numeric field)
- poster or picture (image field)

Trackers have the ability to change the order and size of these fields, and if you try you can get your tracker to display a reasonably clean arrangement. But (as always) you want more. You want to use all your html and css skills to arrange and style these reviews in your tracker exactly like so. Thats what pretty trackers are for.

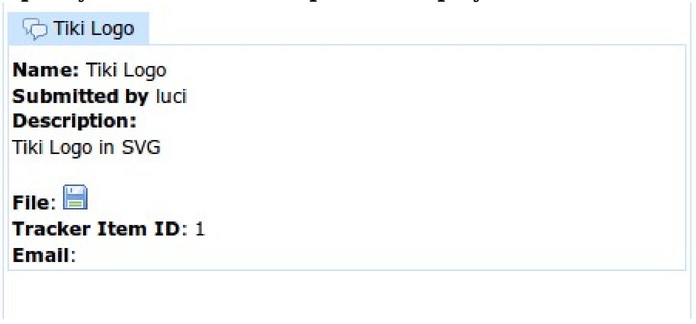
Simple example of PluginTrackerlist without pretty tracker



Most simple *"default"* view of the TrackerList wiki plugin example:

Name	login	Description	File	Tracker Item ID	atts
Tiki Logo	luci	Tiki Logo in SVG	💾	1	1

A pretty tracker uses a template to display the item



Tiki Logo

Name: Tiki Logo
Submitted by luci
Description:
Tiki Logo in SVG

File: 💾
Tracker Item ID: 1
Email:

Inside a Tracker or in a Wiki Page ?

There are are two ways in which pretty trackers are used.

- A Pretty Tracker (in the tracker feature) is one that uses a template to customize the display of tracker items inside the tracker feature.
  *My 2¢: **never** use this method, it means you can never see you data properly without accessing the database directly. Personally i think the feature should be removed*
- A Pretty Tracker (in the wiki) does the same thing within wiki pages, including via the PluginTrackerlist.
- PluginTracker can write directly to wiki pages using a template (discarding the tracker item after the page was created).

In all cases it is about displaying tracker items using a template.

Quick Step by Step Guide to a Pretty Tracker

Please read this **Pretty Tracker HowTo** first.

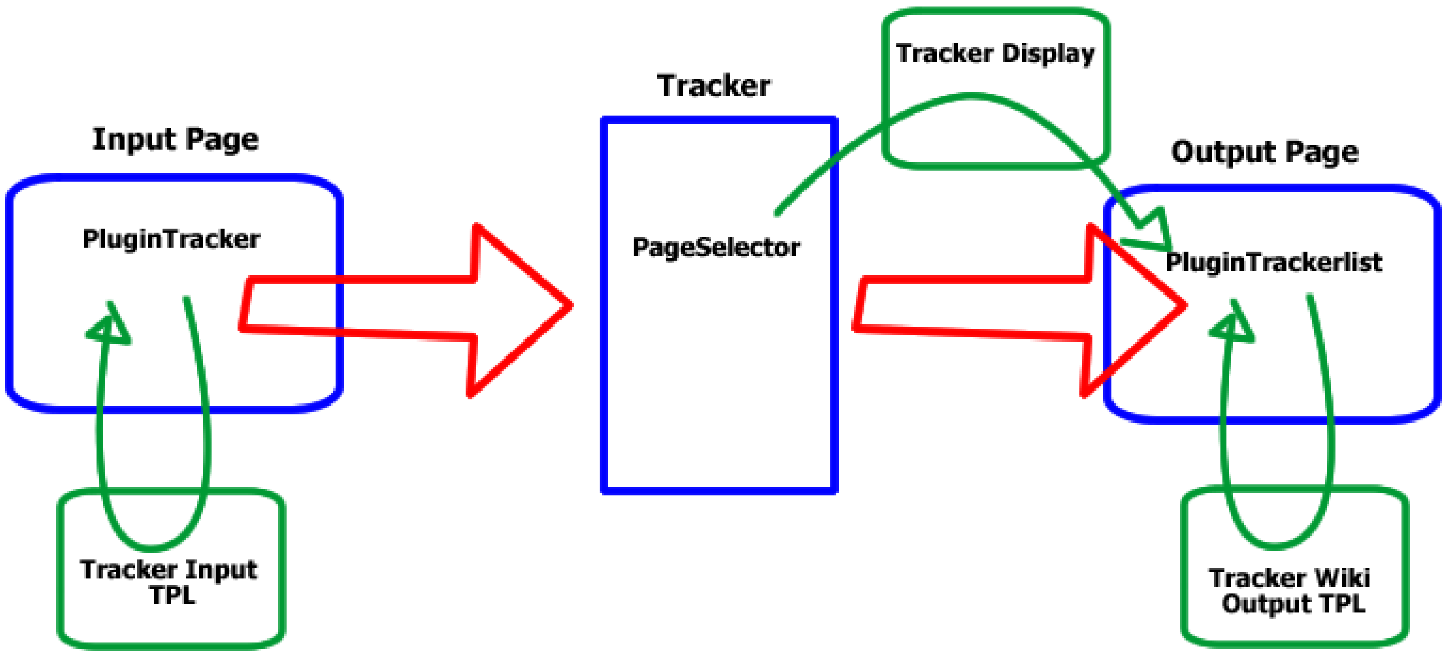
Parts of a Pretty tracker

A pretty tracker has several moving parts that are chained together. Not all will require all the parts below but this is the whole list. In the table below R=Required, Y=Can be used, N=Not used.

Part	**Used in Wiki**	**Used in Tracker**	Description	How many?
Tracker	R	R	The tracker is the table that holds the data	Only one.
Tracker Fields info	R	R	the columns of the table / fields of the form	More than one
PluginTracker	Y	Y	If used replaces the "create a new item" in the tracker function.	1 for each input location (can be more than one)
Tracker Input TPL	Y	N	If PluginTracker is used this can controls how the input form should look	1 for each PluginTracker
Output Page	Y	N	A wiki page where the single item results will show in a wiki. Usually pagename = item name (the "links to item" field of the tracker)	1 per item
Tracker Item	Y	Y	All trackers will have items visible in the tracker feature. For the PTW this is redundant, or only used by admins.	1 per Item
Page Selector Field	Y	N	Must be used to make a PTW where each item is associated with a page.	1 per tracker
Display template	R	N	For wiki only - when using the pageselector you need to provide an example of a PluginTrackerlist to be copied to the output page	1 master copy per tracker)
Tracker Wiki Output TPL	R	N	A wiki page or tpl file used by the PluginTrackerList. Controls how the output for all items is displayed in various wiki pages.	1 per tracker, controls all items.
Tracker Item Output TPL	Y	Y	A wiki page or tpl file used by the tracker itself. Controls how the output for all items is displayed in the trackers item view.	1 per tracker, controls all items.

How it Fits Together

The following graphic depicts how a pretty tracker in a wiki page works, including how you can have a template for the input page as well as the output page.
Blue objects are seen by the end user. Green objects are behind the scenes, called by the plugins of the blue objects. Red arrows are representing the time sequence: input, processing, output.

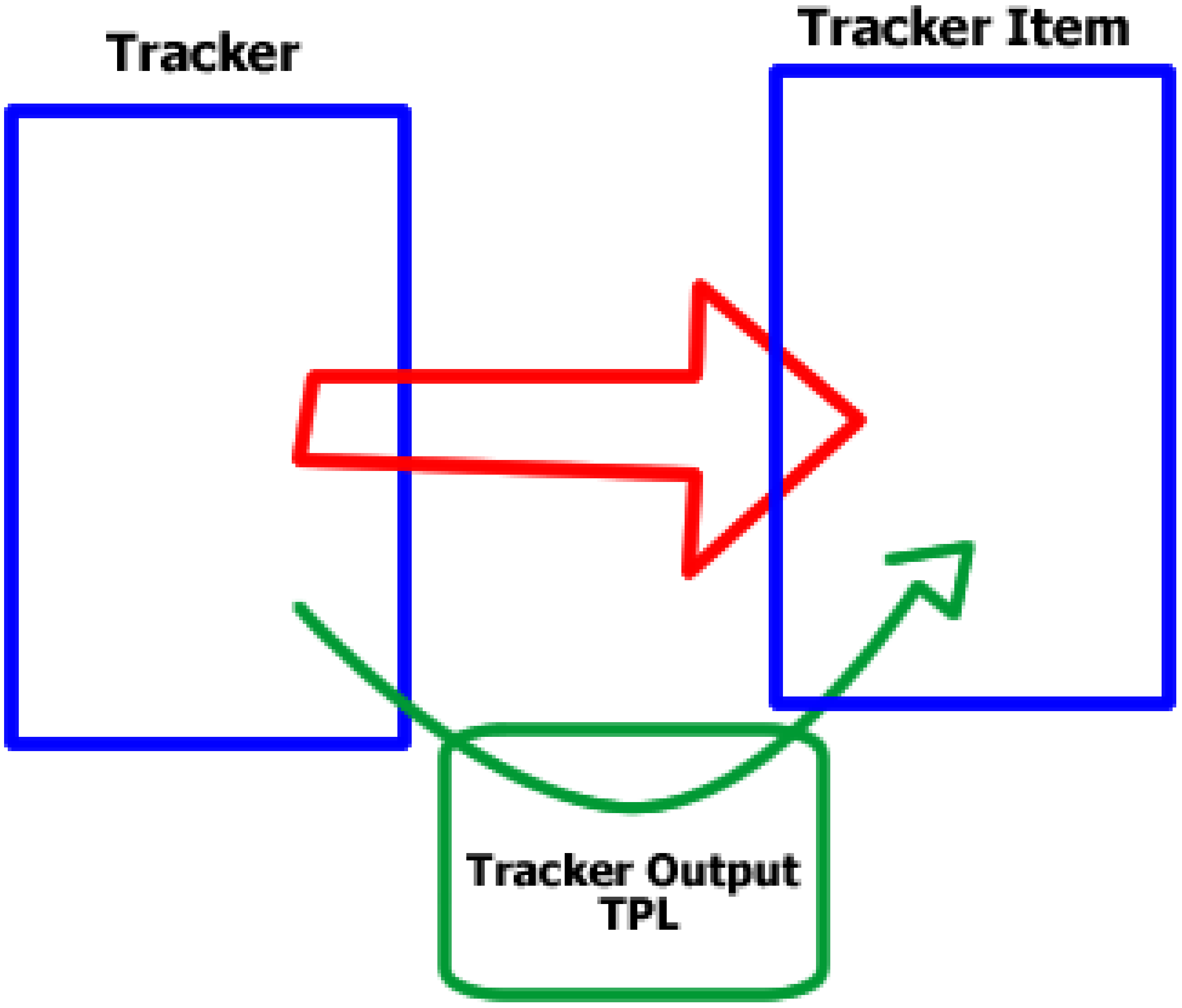


The Pretty Tracker In Action

When executed the sequence in the code works like this.
1. The *Input Page* calls the *PluginTracker*, which calls the Tracker Input TPL
2. The *Input Page* is rendered in the browser as "pretty"
3. The End user supplies the input for the tracker's fields - in other words, fills out the form, which includes providing the name to the *Page Selector* that will become the title of a new wiki page.
4. The tracker item is created in the background.
5. If the tracker has an Item Display TPL, that is called to render the tracker item.
6. If a page selector exists a wiki page is created and filled with the content of the *Tracker Display*, which is normally just the pre-configured *PluginTrackerList*.
7. When the wiki page is visited, *PluginTrackerlist* calls up the fields from the tracker, and the *Tracker Output TPL* and combines them together to render the "pretty" page as seen by the end user.

Tutorial - A Simple Pretty Tracker.

The step-by-step tutorial that follows will allow you to create a **pretty tracker within the trackers function**. This is a recommended first step to creating a more complicated pretty tracker. In this tutorial you will display a pretty tracker item using an *Item Output TPL*. If you get that far, you can use the template made here as the basis for a Pretty Tracker in a Wiki Output TPL.



Note: This tutorial assumes you know how to set up a tracker.

1. Create a normal tracker. You can "prettify" an existing tracker if you like. Please consult the docs on trackers to learn how if necessary. Here the Tracker name is *Simple Tracker*
   * Don't fill in the "Wiki page to display an item" yet (this is the last option in the *edit tracker* panel). We will fill that later.
   * Note what tracker ID you have created. Here we will say it is *trackerID=8*
2. Create fields for your tracker. For now, don't spend time trying to change how they are displayed, or what order they are in or what size they are.
3. Create a few test items for your tracker if they don't already exist. The point here is to get to step 4 as quickly as possible. You can always add more fields later.
4. Go back and take a look at *tiki-view_tracker.php?trackerId=8* (substitute your tracker ID in the url instead of □ to see your tracker's items. All there? Good.
5. **List your Tracker Field ID's** Open a second browser and navigate back to *tiki-admin_tracker_fields.php?trackerId=8* by clicking the *Fields* button. Keep this page open while you work on the next steps.
6. **Creating the Output TPL** Create a new wiki page called *Simple Tracker Output TPL* (replace 'Simple Tracker' with the name of your tracker. Don't be creative, use this naming convention to avoid confusion later.
7. In the new wiki page, key in a list of variables and field names using all the fieldID's you see in that other broswer window. Type in the fieldIDs as variables just like they are below.

{$f_12} Username {$f_13} City {$f_14} Photo {$f_15} Join Date {$f_16} Wants to Buy {$f_17} Wants to Sell {$f_18} GroupName

So what we have here are placeholders that will be replaced by the information belonging to the individual tracker item field. The curly bracket things are the variables, that is what matters. You can delete the names after, but you need them for now to avoid confusion.

8. Now go nuts with your HTML, Smarty, and CSS skills. By editing *Simple Tracker Output TPL* you are creating a web page that presents all the fields in your tracker in a nice and pretty way. Copy and paste each variable from your list into the appropriate spot. Save.
9. Add special object permissions to this page. Assign the perm *tiki_p_use_as_template* to anonymous and registered, or whatever groups will be viewing the tracker. Save. You may have already set this using categories or global permissions, but now you are sure it will work.
10. Now (is your other window still open? Hope So). Go back to *tiki-admin_trackers.php?trackerId=8* by clicking on the *Trackers* button, then click on the action wrench icon for tracker 8 and click on *Properties* in the popup. Within the modal popup click on *Section Format* to expand that section and perform the following:
    1. Set the *Section format* field to *Configured*
    2. Set "Template to display an item" to be *wiki:Simple Tracker Output TPL*. (you replaced 'Simple Tracker' with your trackers' name right?)
       *My 2¢: **never** use this method in production, put this back to flat or tabs and use a wiki page to display the items otherwise it means you can never see you data properly without accessing the database directly.*
    3. Save
11. Now go to *tiki-view_tracker.php?trackerId=8*, and select an item. How is this item displayed now? Kinda Pretty? you will probably have to edit your template page a couple more times to get the best result. You may also have to visit *tiki-admin_tracker_fields.php?trackerId=8* to resize the fields you created (it doesn't make much sense to spend time on that till now.)
12. Tweak until satisfied.

Dynamic Pretty Trackers

So far the tracker inputs and outputs have been rather static. It is not possible to change a field according to the value of another field, for example.

But what if you want to have different questions or different answers displayed in your template based on the information in that item? Basic conditional display is covered by PluginTrackerToggle. Using pretty trackers and JavaScript, more advanced behavior is possible for those who know HTML, DOM and JavaScript. The following form uses the jQuery library and PluginJq:

Log in as registered user to see the jquery in action in this "Dynamic tracker example".

The layout of this pretty tracker is defined in the template page Dynamic tracker example template, which also contains the code to make the form dynamic. The code below will use field numbers to control these field.

Field number	Name
121	Content type
122	File

123	URL
124	Did you ensure this does not breach policy?

Conditional fields

The form contains two alternative fields which are initially hidden. When a radio button is selected, the corresponding field appears. Since none of these fields is always required, the tracker fields are not defined as mandatory. But this is emulated using a JavaScript check when submitting the form.

The initial state is achieved with the following code:

```
$("#fileprompt").hide(); $("#urlprompt").hide();
```

fileprompt and *urlprompt* are the identifiers of DIV elements defined in the template to allow controlling the display of the 2 fields and their label:

```
{DIV(type="div",id="fileprompt")} File: {$f_122} {DIV} {DIV(type="div",id="urlprompt")} URL:
{$f_123} {DIV}
```

The following code handles selection of a radio button, showing a field and making sure the other one is hidden:

```
$("input[name='track_121']").change(function(){ if ($(this).filter(":checked").val() === 'File') {
$("#urlprompt").hide(); $("#fileprompt").fadeIn(); } else { $("#fileprompt").hide();
$("#urlprompt").fadeIn(); } });
```

The function is executed when field 121 is changed. When the function is executed, one of the 2 blocks is run depending on the value of the selected (or "checked") radio button.

Display a message when a certain value is selected

The user needs to fill a boolean field, but one value is not normal. A message is displayed if the user selects "No". The following code does this adding a handler to the change event on the select element:

```
$("select[name='track_124']").change(function(){ if ($(this).find("option:checked").attr("value") ===
'No') { alert("Then go do your homework."); } });
```

Using a Pretty Tracker in a Wiki Page

see Pretty Tracker in a Wiki Page

Conditional display of input fields

- PluginJq

Attached to this wiki page as a downloadable PDF and MS Word files.

Syntax tips

Pre-defined variables

{$f_created}: created date
{$f_status_input}: status input field (for plugin tracker with `showstatus="y"`)
{$f_status}: status (output)
{$f_itemId}: the item id
{$f_lastmodif}: last modified date (this will display unix date, for human readable date look below)
(In Tiki 8 onwards) {$itemoff}: the iteration number of each item
{$tr_offset}: the offset of the item, i.e. this is the nth item of the total number of x items

Registration pre-defined variables

You can easily integrate registration information in your pretty tracker using the proper syntax
{$register_login}
{$register_email}
{$register_pass}
{$register_pass2}
{$register_passcode}
{$register_groupchoice}
{$register_antibot}

See http://doc.tiki.org/User-Tracker#Using_Pretty_Trackers_to_generate_the_Registration_Tracker

Syntax modifier

Modifiers can be applied using "|" (pipe).

Date modifier

{$f_lastmodif|tiki_short_date} will display human readable date only.
{$f_lastmodif|tiki_short_datetime} will display human readable date and time.
etc...

View / edit modifier

Show data in view mode (not edit mode)

    {$f_122|output}

This is reported not to work within a wiki table
Related commit: http://sourceforge.net/p/tikiwiki/code/31403

Limitations to using wiki pages

There are certain limitations to the use of pretty tracker veriables when using a wiki page as the template, for example:

- An $f_ variable won't be interpreted properly when used as a parameter value for certain plugins (at least for PluginList), due to the way the plugin is processed

The solution in this case is to use a Smarty template file as the template rather than a wiki page.

Related

See:

- tracker
- Pretty Tracker in a Wiki Page

- PluginTrackerList
- PluginList and PluginCustomSearch
- Semantic Alias to see how you can link something like ((Project:15)) to a pretty tracker page.

Aliases: PrettyTracker | PrettyTrackers | Pretty Trackers | Pretty+Tracker