Plugin CustomSearch

See also PluginList with editable fields

This wiki plugin is available from Tiki8 for the purpose of creating custom user interfaces to search for objects within Tiki. This plugin makes use of the functionality of PluginList to show the results of the search. This plugin is intended for rather advanced customization and you will need to have a good understanding of PluginList as well as Unified Index and good knowledge of HTML, CSS, and Smarty in order to make good use of it.

What happens when the wheel spins forever?
That is likely the sign of an AJAX failure.

If you use the Firefox Firebug plugin in the HTML Response you should be able to see what the AJAX responds with.

Parameters

*Introduced in Tiki 8.*
*Preferences required: wikiplugin_customsearch, wikiplugin_list, feature_search*

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Accepted Values</th>
<th>Description</th>
<th>Default</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>(body of plugin)</td>
<td></td>
<td>LIST plugin configuration information</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tpl</td>
<td></td>
<td>Smarty template (.tpl) file where search user interface template is found</td>
<td>12.2</td>
<td></td>
</tr>
<tr>
<td>callbackscript</td>
<td>pagename</td>
<td>The wiki page on which custom JavaScript is to be executed on return of Ajax results</td>
<td>8.0</td>
<td></td>
</tr>
<tr>
<td>destdiv</td>
<td>text</td>
<td>The id of an existing div to contain the search results</td>
<td>9.0</td>
<td></td>
</tr>
<tr>
<td>searchfadediv</td>
<td>text</td>
<td>The specific ID of the specific div to fade out when AJAX search is in progress, if not set will attempt to fade the whole area or if failing simply show the spinner</td>
<td>8.0</td>
<td></td>
</tr>
<tr>
<td>autosearchdelay</td>
<td>digits</td>
<td>Delay in milliseconds before automatically triggering search after change (disables and is the default)</td>
<td>0</td>
<td>8.0</td>
</tr>
<tr>
<td>id</td>
<td>alnum</td>
<td>A unique identifier to distinguish custom searches for storing of previous search criteria entered by users</td>
<td>0</td>
<td>8.0</td>
</tr>
</tbody>
</table>
Basic Usage

You need two wiki pages, one for the search interface itself, and the other for the template to be used at the interface. From Tiki 13 onwards, it is possible to specify a `.tpl` file on disk instead of a wiki page for the second wiki page.

1.1.1. Create Wiki page 1, entitled: "Custom Search"

```plaintext
{CUSTOMSEARCH(wiki="Custom Search Tpl")}
{list max="10"}
{filter field="tracker_id" content="1"}
{OUTPUT()}
{display name="title" format="objectlink"}
{OUTPUT}
{CUSTOMSEARCH}
```

Tiki 12.2 onwards, `.tpl` version

```plaintext
{CUSTOMSEARCH(tpl="CustomSearchTpl.tpl")}
```
1.1.2. Create Wiki page 2, entitled: "Custom SearchTpl"

In Tiki 12

{literal}

Any text: {input _filter="content" type="text"}
{input type="Submit" value="Search"}
{/literal}

In Tiki 15+

{literal}

Any text:</div><div class="col-xs-10">{input _filter="content" type="text" class="form-control"}</div>
{input type="Submit" value="Search" class="btn btn-primary"}
{/literal}

In case you want to limit the search to content found just on a single tracker field, you need to use the tracker field unique name to refer to that field (for instance in this example: my_permanent_name). This is because by default since Tiki8, unified search uses the permanent names for tracker field names instead of field IDs. Therefore, you could change this line:

Any text: {input _filter="content" type="text"}

for this other one with _field as tracker_field_ + my_permanent_name instead:

Any text: {input _filter="content" type="text" _field="tracker_field_my_permanent_name"}

1.1.3. Grant permission to the tpl page

You need to grant permissions to the tpl page "Custom SearchTpl" to be used as a tracker template (or a template for unified search) by anonymous or registered users (or the group of your choice). Click on permissions for that page, and change permissions accordingly. Remember to restrict edit permissions to only admin users since then to that page.

1.1.4. Enable unified search

Ensure that you have Unified search enabled, under "Control Panels > Search".

1.1.5. Rebuild search index

Rebuild your search index after you have created some content in tracker 1, so that this content can be found by the search engine through the search index.
1.1.6. Use your custom search interface

Go to page 1 to use your new custom search interface.

Notes:

- Assuming you have a tracker with id 1 with some items, that should work.
- In fact if you remove the line
  ```
  {filter field="tracker_id" content="1"}
  ```

- it will work as a search over all your content. In fact just this line on it's own in a page will work!
  ```
  {customsearch wiki="Custom Search Tpl"}
  ```

Working Custom Search Example

For a working example showing many of these features see this Custom Search Example page.

More advanced usage

Wiki syntax:

```
Custom Search
{CUSTOMSEARCH(wiki="Search-tpl" id="mysearch")}
  {output template="/var/www/html/templates/my-custom-searchresults.tpl" pagination="y"}
  <put whatever base filters and formatting things that you would do in the LIST plugin here>
{CUSTOMSEARCH}
```

Note that the pagination="y" is generally required so that you have the ability to paginate across multiple pages of search results. (There is a more advanced way of specifying the pagination manually within Smarty templates explained below).

Base filters are filters that you want applied regardless of what search parameters the user chooses in the search form.

See PluginList for more information about filters and formatting of output.

Creating your search interface

Creating the wiki page

You will need to create a template to be placed in the wiki page mentioned above. In this wiki page, you need to surround the contents with smarty literal tags so that the contents don't get parsed as Smarty tags.

```
{literal}
<contents of template>
{/literal}
```

In addition, if you want to use HTML in your template, you can use PluginHTML:
It is important to make sure that this page has Permissions set so that it cannot be edited by non-admins

Creating form elements

Creating form elements are similar to creating standard HTML form elements, combined with using the filter parameters found in PluginList. Specify each element using plugin syntax. If you want to use a standard HTML supported attribute, simply use it and it will appear in the resulting HTML. All internal attributes that are meant for the LIST plugin to generate the search results should start with an underscore (_), so that they don't end up in the HTML (which would be invalid). For example, if you want to create a text search field that searches for the content entered:

{input _filter="content" type="text"}

You can add any HTML supported attributes if you want, for e.g.:

{input _filter="content" type="text" class="mysearchbox"}

If you want to specify a specific field/fields to search in as described in PluginList, for example:

{input _filter="content" _field="tracker_field_20" type="text" class="mysearchbox"}
{input _filter="content" _field="tracker_field_28,tracker_field_22" type="text"}

Drop down (with or without multiselect) example:

Note how you can specify labels different from the values if you want

{select _filter="content" _options="1980,1981,1982" _field="tracker_field_104" multiple="multiple"}
{select _filter="content" _field="tracker_field_104" _options="1980,1981,1982" _labels="eighty,eighty-one,eighty-two"}

You can also set a default item to be selected using the _default element. To avoid the blank option from appearing, use the _mandatory attribute.

{select _filter="content" _options="1980,1981,1982" _field="tracker_field_104" multiple="multiple" _default="1980"}
{select _filter="content" _options="1980,1981,1982" _field="tracker_field_104" multiple="multiple" _mandatory="y" _default="1980"}

From Tiki 14.1 onwards custom search will automatically populate the drop-down menu for ItemLink, UserSelector and Text tracker field inputs. You need to specify the _field and _trackerId params only (to be backported to 13 & 12 when tested)
From Tiki 19.0 (and 18.3) you can use the operator attribute to override the site-wide preference for default boolean operator for multiple select fields.

Checkbox example:
The most important thing to keep in mind about checkboxes is that they cannot take a normal "value" parameter like in HTML. Instead, use the _value (with underscore) parameter if you want to set a value. If you do not set a value, the values searched for will by default be y (if checked) or NOT y (if not checked). If you want the checkbox to be checked by default, specify _default="y". For example.

If you want the checkbox to behave in reverse, i.e. search for NOT y when it is checked and y when it is not checked, specify --+value="n"+- .

If you want the checkbox to mean that you want to search for whatever term you want when checked, set _value to something arbitrary, for example:

Radio button example:
The most important thing to remember about radio buttons is that they need a _group attribute. This _group attribute determines which radio buttons are together as a group. For example:

Categories example:
In general, if there are just a few categories to search by, you can use radio buttons or checkboxes (in the following example 24 is the category ID you want to search for when the checkbox is checked):

If you want the category search to not only search the specified category but also all of its sub-categories, specify the _deep attribute:

In addition, to facilitate creating category selection elements for many categories, there is a special way to quickly show multiple checkboxes/radio buttons or a drop-down for multiple categories based on a parent category.

Yield
Note that if you are specifying a _style of checkbox or radio, _group is mandatory.

Other parameters include _categpath which shows the full category path category labels instead of just the category name, and _showdeep which shows not just the immediate children of the parent category specified but all the sub-children as well.

When specifying multiple="multiple" there is not first blank item, you can force a blank or labelled item (has no value) by using the attribute _firstlabel.

Adding a submit button

Adding a submit button

Grouping elements together for the purpose of OR searching instead of AND

It has already been mentioned above that Radio buttons as well as categories (those using a _style attribute of checkbox and radio) need to use the _group attribute mandatory.

However, it is possible to use the _group attribute for other types of fields as well, if you want them to be searched using OR logic instead of AND which is how different filters are combined. So long as the filters you are trying to group together are exactly the same except for the search term, they can be grouped. For example, the following searches for either "apples" or "oranges" if both checkboxes are checked:

Adding a submit button

Adding a submit button

Range searches

Before you start, please be informed that the Unified search is a string based search. In other words, if you are trying to search between 2 numbers, make sure all your numbers are of the same number of digits. Otherwise 2 will be larger than 12. The only way to get around this for now is to make sure all numbers are entered in a padded form, for example, 0012 for 12.

However, from Tik14 if you are using Elasticsearch as the unified search engine, then numeric tracker fields are indexed internally as float and therefore will provide for correct range searches for those fields, even without padding.

The way to generate a search for a value between two values is to use the _daterange attribute to group the 2 items that you want to search in between together. The following example searches a tracker field that contains all 4-digit years.
The following example is an interesting twist, where you automatically set one end of the range through a hidden field.

Show only content after {select _filter="content" _field="tracker_field_104" _default="2000" _mandatory="y" _options="2000,2005,2010" _textrange="2"}{input _filter="content" _field="tracker_field_104" type="hidden" value="9999" _textrange="2"}

To do a range search on things like modification dates, which are "unix timestamps indexed as dates", use _daterange instead of _textrange. This will ensure that the comparison is correct despite the string based search since these timestamps are indexed in YYYYMMDDHHMMSS format.

Show content modified between {select _filter="content" _field="modification_date" _options="946684800,1104537600,1262304000" _labels="2000,2005,2010" _daterange="1"} and {select _filter="content" _field="modification_date" _options="946684800,1104537600,1262304000" _labels="2000,2005,2010" _daterange="1"}

**Date Range Picker**

In Tiki 9 there is also a date range picker you can use like this:

{daterange _field="modification_date" _from="1104537600" _to="1262304000" _showtime="y"}
{daterange _field="modification_date" _from="1104537600" _gap="31536000"}

You can also use "now" as a date. It will give you the current time/date.


{daterange _field="modification_date" _gap="2 weeks" _to="next week" }
{daterange _field="modification_date" _gap="1 month" _to="now" }

You need to set a default filter in the customsearch as follows so that the user is not confused when first arriving at the page.

{filter range="modification_date" from="1104537600" to="1262304000"}
{filter range="modification_date" from="now" gap="31536000"}

Since Tiki 18.3 the end date uses the end of the day, as opposed to midnight at the beginning of that day. This is generally agreed to be more intuitive, but if you need to revert to the previous behaviour add a parameter _toendofday="n". For instance this will find nothing:

{daterange _field="modification_date" _from="today" _to="today" _toendofday="n"}

**Single-ended range searches**

Before Tiki14, the user must enter both ends of the filter for the range search for the search to have any effect. Therefore it is important to make sure the user selects a value, or a _default is set.

From Tiki14 it is possible allow for doing range searches with user only specifying one end of the range and
the other end left blank but still having default value. For example, the following code will generate 2 form fields for searching the tracker field duration. If the user keeps the first one blank and enters 20 in the second field, it will search for the range 0 to 20. If the user enters 50 in the first field but leaves the second field blank, then the range searched for is 50 to 9999. Note that in each of the fields, _emptyto OR _emptyfrom must be used TOGETHER WITH _emptyother for this to work. If you leave out _emptyother, unpredictable results can occur because browsers may not submit empty form fields (and so this is the only way to get the information on the other range endpoint if it's not filled in).

```html
<input class="form-control" type="number" placeholder="Min" _filter="content" _field="tracker_field_duration" _emptyfrom="0" _emptyother="9999" _textrange="1" />
<input class="form-control" type="number" placeholder="Max" _filter="content" _field="tracker_field_duration" _emptyto="9999" _emptyother="0" _textrange="1" />
```

**Distance Searches**

*Since Tiki 16*

This should be considered unfinished and somewhat experimental but basically works with some customisation. The three inputs are:

- distance: Use a number and a unit, e.g. 1000m for one thousand metres
- latitude: A floating point number up between -90 and +90
- longitude: A floating point number up between -180 and +180

```html
{"DIV(class="row")}{distance class="form-control col-xs-4"}{DIV}
```

**Setting default search parameters when first coming to the page**

As already mentioned above, you can use the _default parameter on specific fields. However, in some cases you are coming from another page where you want to pass some input from there via the query string. To do this you can pass in the query string, for example:

`tiki-index.php?page=Search&default[field]=value`

The "field" here is the same as what you have in the _field attribute in your search template.

You can have several search parameters in the url using &.
You may also use an item link/list value (in the example below tracker_field_kind is item link) in some case.

e.g. tiki-index.php?page=Search&default[tracker_field_month]=October&default[tracker_field_kind]=644

If you have categories filters defined, you can set the defaults as follows:

`tiki-index.php?page=Search&defaultcat[parent]=value`  
e.g. tiki-index.php?page=Search&defaultcat[5]=2,7

The Smarty syntax is valid and can be used to solve more complex case.
In the following we use the Smarty syntax to use a real date to filter results by month following:
https://www.smarty.net/docs/en/language.modifier.date.format.tpl

tiki-index.php?page=Search&default[field]="smarty"

- e.g. tiki-index.php?page=Search&default[tracker_field_month]={'this month'|date_format:"%B"}

Note that if you are having problems using square brackets within link syntax in a wiki page, you may want to urlencode the square brackets, for example:

[tiki-index.php?page=Search&defaultcat%5b7%5d=3]

Advanced adding of jquery/javascript on AJAX loading of results

This is relevant only if you are implementing an advanced template for PluginList used in conjunction with Custom Search, and it contains jquery/javascript that affects the search results.

Because in IE prior to version 9 it is not possible (due to crappy DOM handling) to deliver jquery/javascript that affect the content of the AJAX response together with the AJAX response, instead of including such scripts in the template for PluginList, you will need to include such scripts as follows:

Create a wiki page as follows, enclosing your javascript/jquery, and specify the wiki page name as the callbackscript parameter to the CUSTOMSEARCH plugin.

```
{JQ()}
$(document).on("pageSearchReady", function() {
    doYourStuff();
    return true;
});
{JQ}
```

It is important to make sure that this page has Permissions set so that it cannot be edited by non-admins

Example excerpt

1.1.1.1. Wiki page to display results of custom search

```
{CUSTOMSEARCH(wiki="My Custom Search Tpl" recalllastsearch="0" searchonload="1" requireinput="0" searchable_only="1" customsearchjs="1" callbackscript="shortenDescriptions")}
```

(the filters and output of the custom search as usual)

{CUSTOMSEARCH}
1.1.1.1.2. Contents of "My Custom Search Tpl"

The page "My Custom Search Tpl" has this type of code:

```html
<div id="courseOverview">{$row.courseDescription|truncate:30:"...":false}</div> <a class="readMore" >Read more</a>
<div id="courseOverviewFull" style="display: none">{$row.courseDescription}</div> <a class="readLess" style="display: none">Read less</a>
```

1.1.1.1.3. Contents of "shortenDescriptions" (callbackscript)

The page shortenDescriptions:

```javascript
{JQ()}
$(document).on("pageSearchReady", function() {

    //using the .readMore class as a selector for the click event over it.
    $('.readMore').click(function(){
        $('#courseOverview').toggle();
        $('#courseOverviewFull').show();
        $('.readLess').show();
        $('.readMore').hide();
    });
    $('.readLess').click(function(){
        $('#courseOverview').toggle();
        $('#courseOverviewFull').hide();
        $('.readLess').hide();
        $('.readMore').show();
    });

    return true;
});
{JQ}
```

Advanced Pagination in Smarty Template

Advanced pagination not using pagination="y" but instead directly creating the pagination within the Smarty template, by specifying the customsearch ID manually as follows:

```smarty
{assign var='customsearchid' value="xyz"}
{pagination_links offset_jsvar="customsearch_`$customsearchid`.offset" _onclick="$('#customsearch_`$customsearchid`').submit();return false;" resultset=$results}{/pagination_links}
```

Setting maxRecords, sort_mode, and offset

Overriding defaults that apply on arriving at search page

The defaults can be overridden by passing into the query string these variables. For example, if you want the number of search results to show to be set to 10, you can do:
(or with SEFURL on) http://yoursite.tiki.org/MySearch?maxRecords=10

For sort_mode, you can also set it inside your search template. Please be warned that sorting is based on string. If you are trying to sort by numbers, make sure all your numbers are of the same number of digits. Otherwise 2 will be larger than 12. The only way to get around this for now is to make sure all numbers are entered in a padded form, for example, 0012 for 12.

{sort mode="tracker_field_18_desc"}

Changing the settings after arriving at the search page

You will need a little Jquery/javascript for this. It is possible to set the javascript variables "customsearch_<id>.maxRecords", "customsearch_<id>.sort_mode", "customsearch_<id>.offset" programmatically and then resubmit the form by calling the submit() event handler on the form using the jquery. For example:

```
( "#datesort" ).click(function()
    customsearch_<id>.sort_mode = "modification_date_desc" ;
    $('#customsearch_<id>').submit();
    return false;
});
```

Customizing the interface with CSS

The final step is to customize the search interface with CSS. The search results is in a DIV with the ID customsearch_<id>_form. The search form is in a DIV with the ID customsearch_<id>_results. Because you can assign your own CSS classes to each of the form elements above, you can customize it anyway you like.

Using Facets

Facets are dynamic search filters provided by the search engines. They allow to refine the user’s search. Not all search engines support this feature. At this time, it is only supported by the Elasticsearch implementation.

Using facets with CustomSearch requires that hidden fields be added to the search form. These fields will be filled as the user selects filters. To simplify binding, make sure you set the ID explicitly.

```
{input _filter="content" type="hidden" _field="object_type" id="my_object_filter"}
{input _filter="content" type="hidden" _field="deep_categories" id="my_deep_categories"}
```

The CustomSearch plugin must then define which facet it wishes to obtain. When configuring the facets, you get to choose how to handle multiple selection. By default, OR filters will be used.

```
{CUSTOMSEARCH()}
...
{facet name=object_type operator=or}
{facet name=deep_categories operator=and}
```
Then, you need to modify the results template to display the facets selector tools (see the sample below)

**Facets from Categories**

To use categories as facet generators, go to the search control panel, and on the "search results" tab add the top category or categories you will need to the "Generate custom facets from categories" preference, apply that and rebuild the search index.

Add the hidden fields to your search form as above, but using this special format for the id:

```
deep_categories_under_XXX
```

replacing the XXX with the top category id

```
{input _filter="content" type="hidden" _field="tracker_field_types" id="deep_categories_under_28"}
{input _filter="content" type="hidden" _field="tracker_field_styles" id="deep_categories_under_42"}
```

and then add the facet commands to your custom search page like this:

```
{CUSTOMSEARCH()}
...
{facet name="deep_categories_under_28"}
{facet name="deep_categories_under_42"}
...
{CUSTOMSEARCH}
```

Then, you need to modify the results template to display the facets selector tools (see the sample below)

**Facet Parameters**

**Operator**

Sets the input above to use "AND" or "OR" (default is "OR")

**Count**

*Since 12.4*

You can specify the maximum number of facets returned. This defaults to 10, but can be changed globally using the "Facet result count" preference on the search results control panel, or per facet in the plugin like this:

```
{CUSTOMSEARCH()}
...
{facet name=deep_categories operator=or count=50}
...
{CUSTOMSEARCH}
```
Order

*Since 20.1*
Sets the order the facets are returned in the results, use either *_count* (default) or *_term* for ES5 (or *_key* for ES6+). Append *_asc* or *_desc* for ascending and descending.

Useful for charts where you need to set the colours according to the order the facets/aggregations are returned.


```json
{CUSTOMSEARCH()}
...
{facet name="tracker_status" order="_term_asc"}
...
{CUSTOMSEARCH}
```

Min

*Since 20.1*
Set the min_doc_count - useful for displaying empty aggregations

```json
{CUSTOMSEARCH()}
...
{facet name=deep_categories min="0"}
...
{CUSTOMSEARCH}
```

Date Based Facets

*Since 20.1*

Ranges

Used for *type=date_range* each range is separated by a pipe | character, and within that use commas to separate from, to, label

More info on date formats accepted here on elastic.co

```json
{CUSTOMSEARCH()}
...
{facet name="date" type="date_range" id="date_range" ranges="2018-01-01,2018-04-01,2018 Q1|2018-04-01,2018-07-01,2018 Q2|2018-07-01,2018-10-01,2018 Q3|2018-10-01,2019-01-01,2018 Q4"}
...
{CUSTOMSEARCH}
```

Interval

For *type=date_histogram* (more info here)

```json
{CUSTOMSEARCH()}
...
{facet name="date" type="date_histogram" id="date_histogram" interval="year"}
...
```
Facets selector tools in your results template

Finally, you need to modify the results template. The $facets variable is available and contains all the facets returned by the search engine. It is important to verify that the results have been provided.

Test the facets array is populated properly
{$facets|var_dump}

In this example, Chosen is used to render the facet elements nicely and each change to a facet causes the results to reload. The registerFacet() jQuery function binds the select element to the hidden field annotated by data-for.

```html
<div>
  <div class="facets">
    {if $facets.deep_categories}
      <h6>{$facets.deep_categories.label|escape}</h6>
      <select multiple name="{$facets.deep_categories.name|escape}" data-for="#my_deep_categories" data-join="{$facets.deep_categories.operator|escape}">
        {foreach from=$facets.deep_categories.options key=value item=label}
          <option value="{$value|escape}">{$label|escape}</option>
        {/foreach}
      </select>
    {/if}
    {if $facets.object_type}
      <h6>{$facet.label|escape}</h6>
      <select multiple name="{$facets.object_type.name|escape}" data-for="#my_object_filter" data-join="{$facets.object_type.operator|escape}">
        {foreach from=$facets.object_type.options key=value item=label}
          <option value="{$value|escape}">{$label|escape}</option>
        {/foreach}
      </select>
    {/if}
  </div>
  <div style="float: left; width: 66%;">
    {foreach from=$results item=row}
      <li>{object_link type=$row.object_type id=$row.object_id}
        {$row.highlight|escape}</li>
    {/foreach}
  </div>
</div>

{jq}$applyChosen();$('.facets select').registerFacet();$('.facets select').change(function () {
  $('#customsearch_0').submit();
});
```

Generating Charts from Facets

Since Tiki 19 (r68500) example templates can be found in templates/examples/search/. Some examples of this can be found on PluginCustomSearch Chart Examples

Related profiles

**Shopping Cart profile**
The [https://profiles.tiki.org/Shopping_Cart](https://profiles.tiki.org/Shopping_Cart) profile sets up a more complicated custom search using smarty templates etc.

Feel free to apply it to your tiki site and learn how it work, adapt it to your own needs, etc.

Aliases

[Custom search](#) | [Plugin Custom Search](#) | [CustomSearch](#)