PluginList advanced output control block

This section provides details on the use of wiki and Smarty templates that can be invoked with the OUTPUT control block.

Wiki Formatting (In Separate Wiki Page)

(Available from Tiki8) This is effectively the same as Wiki Formatting Within Plugin - see PluginList OUTPUT control block - but with the difference that the formatting of each item in the search/filter result set can be defined in a separate wiki (template) page.

First configure a LIST plugin using an OUTPUT control block to specify a wiki page where the wiki 'templating' will be defined, for example:

Separate Wiki Page Output Sample

```
{LIST()} {output wiki="ResultsTemplatePageName"} {FORMAT(name="label")}{display name="title" format="objectlink"}{FORMAT} {FORMAT(name="desc")}{display name="description"}{FORMAT} {LIST}
```

The simplified lower case output plugin-like format can be used, i.e. without separate opening and closing OUTPUT tags since any body content would be ignored when a separate wiki page template is used. Also the use of the FORMAT control blocks allows individual objects to be 'formatted' and then referenced in the various OUTPUT methods - see PluginList FORMAT control block for more details.

After that, create the wiki page template "ResultsTemplatePageName" in which you will put the following

Separate Wiki Page Formatting Sample

```
{literal} This is the label: {display name="label"} This is the description: {display name="desc"} {/literal}
```

Note that the "literal" Smarty tags are required to prevent `{display name=label}` and `{display name=description}` from being parsed as usual smarty variables.

If you want to use HTML in your template, the following is the correct way:

Separate Wiki Page Formatting Sample

```
{literal} {HTML(wiki="1")} <div>This is the label: {display name="label"}</div> <div>This is the description: {display name="desc"}</div> {HTML} {/literal}
```

Results link to display an item in a template using wiki variables

In some cases you want your plugin list to display a list of trackeritems (customsearch, results, etc) and have the link to each item to display the tracker fields selected from within a template.

You'll have to:

- Enable "Wiki argument variables"
  Go to tiki-admin.php?page=textarea => Wiki Syntax and check that "Wiki argument variables" is enabled
- Create a template - let's assume it's called "fiche" - to display the single item. Please note trackerId 1 is set for demonstrate, change to your tracker Id. Add { {itemId} } variable without space.

```
{LIST()} {pagination max="1"} {filter content="1" field="tracker_id"} {filter field="object_id" content="{ {itemId} }"} {LIST}
```

- Create your primary list as you normally would and add an output tag to redirect the item to the proper template (fiche in this case)

```
{LIST()} {filter content="1" field="tracker_id"} {OUTPUT(template="table")} {column sort="title" label="Title" field="title_link"} {OUTPUT} {FORMAT(name="title_link")}[fiche?itemId={display name="object_id"}|{display name="title"}]{FORMAT} {LIST}
```

Results link to display an item in a template using alias

Another way to have the link to each item to display the tracker fields selected from within a template is to use Wiki Aliases

You'll have to:

- Go to tiki-admin.php?page=wiki => Features and check all the following
  - Enable "Backlinks"
  - Enable "Semantic links"
  - Enable "Similar pages"
  - Enable "Redirect to similar wiki page"
  - Insert the word (for instance `project-`) you want to use as Alias page (Redirect pages using these prefix-alias semantic links)
- Create a display page named *project* (or what you like) to display the single item (change project- for whatever alias you want). I added a CSS class to hide the alias.

```
{LIST()} {pagination max="1"} {filter field="object_id" content="{ {itemId} }"} {LIST} {DIV(class="hidden")}(prefixalias(project-)){DIV}
```

  Add detailed OUTPUT or a template for displaying more information that the item title.

- Create your primary list as you normally would and add an output tag to redirect the item to the proper display page (project- in this case)

```
{LIST()} {filter content="1" field="tracker_id"} {OUTPUT(template="table")} {column sort="title" label="Title" field="title_link"} {OUTPUT} {FORMAT(name="title_link")}[project-{display name="object_id"}-{display name="title"}|{display name="title"}]{FORMAT} {LIST}
```

Advanced Smarty Formatting

For more advanced formatting the plugin can use Smarty templates stored on the server (parameter template) or on a Wiki page (parameter tplwiki).

Whilst this is generally more complicated it is also much more flexible since the template is sent the search/filter result set as input and the template can contain logic to loop through the result set multiple times outputting the results in various ways and applying additional filtering logic if needed.

The content of the output plugin control block is ignored when the smarty template parameter is set.

Using a smarty template file

```
{LIST()} {output template="/path/to/file.tpl"} {FORMAT(name="label")}{display name="title"
```

format="objectlink"} {display name="description"}{FORMAT} {LIST}

Using a Wiki page as smarty template

{LIST()} {output tplwiki="mypage_tpl"} {FORMAT(name="label")}{display name="title" format="objectlink"} {display name="description"}{FORMAT} {LIST}

As with the wiki page template option, the simplified lower case output plugin-like format can be used, i.e., without separate opening and closing OUTPUT tags since any body content would be ignored when a separate server stored smarty template is used.

However, you may also pass a variable to the smarty template in the body content of the OUTPUT tag. In the below example, you could access orientation using {$options.orientation} in your file.tpl smarty template.

Passing a variable to a smarty template

{LIST()} {OUTPUT(template="file.tpl")} {options orientation="vertical"} {OUTPUT} {LIST}

Passing a value from a plugin List or CustomSearch into a smarty template

There are cases were you need to pass a value through a variable into your smarty template.
For this you can use the format block with a value to pass from page to template or template to template.

Passing a tab value

{FORMAT(name="tabValue")}1{FORMAT}

Accessible variables:

- `$results` - containing the result set. Each results contain all values provided by the search query along with those requested manually.
- `$count` - the total result count
- `$maxRecords` - the amount of results per page
- `$offset` - the result offset
- `$options` - the options passed as described just above

In Tiki8, the following have been added:

- `$offsetplusone` - basically `$offset + 1` , so that you can say "Showing results 1 to ...."
- `$offsetplusmaxRecords` - basically `$maxRecords + $offset` , so you can say "Showing results 1 to 25"
- `$results->getEstimate()` - which is the estimate of the total number of results possible, which could exceed `$count` , which is limited by the max Lucene search results to return set in Admin...Search

Other general smarty variables, e.g., `$user` , etc., can only be invoked if they have been 'set up' using the FORMAT control block - see PluginList FORMAT control block for more details.

Including Pagination

To include pagination below the template, `pagination=1` can be used, just like when using the wiki formatter. However, if you want to locate the pagination elsewhere, you can do it using Smarty. Skip the `pagination=1` part.

Tiki8

{pagination_links resultset="$results"}{/pagination_links}

WARNING when using smarty IF:

Please note that if you are using smarty if or otherwise testing for the values of variables returned using

the plain formatter, that the value of such variables is not just the string but has the `noparse` tags as well. These no parse tags are needed in order to prevent extra line breaks and other wiki parsing artifacts to appear. It is recommended to simply test values of raw unformatted variables instead.

For example: ~np~This is the string~/np~

From Tiki11, you can add `mode=raw` on the FORMAT plugin to avoid considering the output will be wiki-parsed.

You can also remove the ~np~ tags by using the `nonp` smarty modifier, e.g.

nonp

```
{if not empty($row.fieldname|nonp)}thing...
```

Example server stored Smarty Templates (TPL files)

Example showing an add to cart plugin

```
<table> {foreach from=$results item=row} <tr> <td> <h3>{$row.title}</h3> <p><em>Category: {$row.category}</em> <strong>Price:</strong> {$row.price}</p> {wikiplugin _name="addtocart" code=$row.tracker_field_20 description=$row.tracker_field_21|escape price=$row.tracker_field_24}{/wikiplugin} </td> </tr> {/foreach} </table> {pagination_links resultset="$results"}{/pagination_links}
```

This example shows how a foreach loop must be used to loop through the search/list results and very detailed/customised formatting can be produced with appropriate html to display the data from each item (row) in the search list, referenced by the `$row . object_ref_name` syntax.

This example also shows how further complex formatting can be achieved by using wiki plugins within the template itself - but as can see a different syntax/style must be used where the use of a plugin is flagged by the opening/closing 'wikiplugin' tags, the plugin name is indicated by the _name parameter and the individual plugin parameters are added in the normal way and can also be set from the data itself e.g. in the example above the code parameter for the 'addtocart' wiki plugin is set from the item/row data with: `code=$row.tracker_field_20`

Example showing various other possible uses

```
Total number of unapproved items: {$count}{if $count >100} - but only the first 100 are shown{/if}</br> <div class="unapproved_header"><table cellpadding="5" cellspacing="0" > <tr> <td style="background-color: rgb(247, 222, 222); font-size: 14px;" width="100px"> <div style="text-align: center;"><strong>monitoring date</strong></div></td> <td style="background-color: rgb(247, 222, 222); font-size: 14px;" width="320px"> <div style="text-align: center;"><strong>monitoring location</strong></div></td> <td style="background-color: rgb(247, 222, 222); font-size: 14px;" width="200px"> <div style="text-align: center;"><strong>monitor name</strong></div></td> <td style="background-color: rgb(247, 222, 222); font-size: 14px;" width="80px"> <div style="text-align: center;"><strong>submitted by</strong></div></td> <td style="background-color: rgb(247, 222, 222); font-size: 14px;" width="80px"> <div style="text-align: center;"><strong>current user</strong></div></td> </tr> {foreach from=$results item=row} <tr> <td style="text-align: center;" width="100px">{$row.tracker_field_ArkRRDSdataMeasurementDate|date_format:"%d %b %Y"}</td> <td style="text-align: center;" width="320px">{$row.location}</td> <td style="text-align: center;" width="200px">{$row.monitorname}</td> <td style="text-align: center;" width="80px">{if $row.tracker_field_ArkRRDSdataCollectionuserId ne ''}{$row.submitter}{else}***submitter not set***{/if}</td> <td style="text-align: center;" width="80px">{$row.theuser}</td> </tr> {/foreach}
```

</table>

In the example above, in a similar way to the previous example, a table of results are produced with each table row generated within the foreach loop but more customised html is used. In addition this example shows:

- how smarty variable modifiers can be used ie when an item/row element is a date the piped date_format modifier can be used to produced a specific date output/format
- how IF logic can be used to check if an item/row element has a value and can display an alternative if not - this is necessary since the ALTERNATE control block cannot be used as this would normally be placed in thge body of the OUTPUT control block, but body content is ignored when a smarty or wiki template is used.
- the current userId can be displayed by using the referenced object $row.theuser - where theuser has been previously speceified using a FORMAT control block - see the PluginList FORMAT control block for details.

Example showing how you can loop through the data multiple times

<table> {foreach from=$results item=row} <tr>
<td>{$row.tracker_field_ArkRRDSdataMeasurementDate|date_format:"%d %b %Y"}</td>
<td>{$row.tracker_field_ArkRRDSdataMonitorName_text}</td>
<td>{$row.tracker_field_ArkRRDSdataCouldNotBeCollected}</td> <td style="width:
10px">{$row.tracker_field_ArkRRDSdataRiverflyScoreTotal}</td> <td style="width:
10px">{$row.tracker_field_ArkRRDSdataCasedCaddisNumber}</td> <td style="width:
10px">{$row.tracker_field_ArkRRDSdataCaselessCaddisNumber}</td> <td style="width:
10px">{$row.tracker_field_ArkRRDSdataMayflyEphemeridaeNumber}</td> <td style="width:
10px">{$row.tracker_field_ArkRRDSdataBWOephemerellidaeNumber}</td> <td style="width:
10px">{$row.tracker_field_ArkRRDSdataFlatBodiedHeptageniidaeNumber}</td> <td style="width:
10px">{$row.tracker_field_ArkRRDSdataOlivesBaetidaeNumber}</td> <td style="width:
10px">{$row.tracker_field_ArkRRDSdataStonefliesNumber}</td> <td style="width:
10px">{$row.tracker_field_ArkRRDSdataFreshwaterShrimpNumber}</td> </tr> {/foreach} </table>
</br> <h4 style="text-align: center;">Riverfly total score</h4> {wikiplugin _name="gdgraph"
type="barvert" bg="#fff" height="400" width="930"} {foreach from=$results item=row}
{$row.tracker_field_ArkRRDSdataMeasurementDate|date_format:"%b%y"},{$row.tracker_field_ArkRR
DSdataRiverflyScoreTotal}</br> {/foreach} {/wikiplugin}

The example above shows how you can loop through the data more than once with multiple foreach loops, in this case to first produce a table of results and then a bar chart of the results.

Note: In Tiki 9.0 (at least) `$row` *values are wrapped in reversed no-parse tags, so if you want to test for a value within the TPL (e.g. where price == 0 for instance) you would need to use:*

{if $row.stock neq '~ /np~0~np~'}
(but without the space between the `~` first and the `/`)
All the LIST Plugin control blocks

- PluginList pagination or list control block
- PluginList filter control block
- PluginList output control block
- PluginList format control block
- PluginList display control block
- PluginList sort control block
- PluginList advanced output control block

- PluginList multisearch output control block
- PluginList aggregate control block
- PluginList overview about control blocks parameters and values
- Troubleshooting
- GUI
- Hacks and Fun
- PluginList and Metatags - SEO