Trac Ticket Queries

Error: Macro TracGuideToc(None) failed

```
'NoneType' object has no attribute 'find'
```

In addition to reports, Trac provides support for *custom ticket queries*, used to display lists of tickets meeting a specified set of criteria.

To configure and execute a custom query, switch to the *View Tickets* module from the navigation bar, and select the *Custom Query* link.

Filters

When you first go to the query page the default filter will display tickets relevant to you:

- If logged in then all open tickets it will display open tickets assigned to you.
- If not logged in but you have specified a name or email address in the preferences then it will display all open tickets where your email (or name if email not defined) is in the CC list.
- If not logged and no name/email defined in the preferences then all open issues are displayed.

Current filters can be removed by clicking the button to the left with the minus sign on the label. New filters are added from the pulldown lists at the bottom corners of the filters box ('And' conditions on the left, 'Or' conditions on the right). Filters with either a text box or a pulldown menu of options can be added multiple times to perform an *or* of the criteria.

You can use the fields just below the filters box to group the results based on a field, or display the full description for each ticket.

Once you've edited your filters click the *Update* button to refresh your results.

Navigating Tickets

Clicking on one of the query results will take you to that ticket. You can navigate through the results by clicking the *Next Ticket* or *Previous Ticket* links just below the main menu bar, or click the *Back to Query* link to return to the query page.

You can safely edit any of the tickets and continue to navigate through the results using the *Next/Previous/Back to Query* links after saving your results. When you return to the query *any tickets which were edited* will be displayed with italicized text. If one of the tickets was edited such that it no longer matches the query criteria the text will also be greyed. Lastly, if **a new ticket matching the query criteria has been created**, it will be shown in bold.

The query results can be refreshed and cleared of these status indicators by clicking the *Update* button again.

Saving Queries

Trac allows you to save the query as a named query accessible from the reports module. To save a query ensure that you have *Updated* the view and then click the *Save query* button displayed beneath the results. You can also save

references to queries in Wiki content, as described below.

Note: one way to easily build queries like the ones below, you can build and test the queries in the Custom report module and when ready - click *Save query*. This will build the query string for you. All you need to do is remove the extra line breaks.

Note: you must have the **REPORT_CREATE** permission in order to save queries to the list of default reports. The *Save query* button will only appear if you are logged in as a user that has been granted this permission. If your account does not have permission to create reports, you can still use the methods below to save a query.

Using TracLinks

You may want to save some queries so that you can come back to them later. You can do this by making a link to the query from any Wiki page.

```
[query:status=new|assigned|reopened&version=1.0 Active tickets against 1.0]
```

Which is displayed as:

> Active tickets against 1.0

This uses a very simple query language to specify the criteria (see Query Language).

Alternatively, you can copy the query string of a query and paste that into the Wiki link, including the leading `?` character:

```
[query:?status=new&status=assigned&status=reopened&group=owner Assigned tickets by owner]
```

Which is displayed as:

> Assigned tickets by owner

Using the `[[TicketQuery]]` Macro

The ?TicketQuery macro lets you display lists of tickets matching certain criteria anywhere you can use WikiFormatting.

Example:

```
[[TicketQuery(version=0.6|0.7&resolution=duplicate)]]
```

This is displayed as:

> No results

Just like the query: wiki links, the parameter of this macro expects a query string formatted according to the rules of the simple ticket query language. This also allows displaying the link and description of a single ticket:

```
[[TicketQuery(id=123)]]
```

This is displayed as:

No results

A more compact representation without the ticket summaries is also available:

```
[[TicketQuery(version=0.6|0.7&resolution=duplicate, compact)]]
```

This is displayed as:

No results

Finally, if you wish to receive only the number of defects that match the query, use the count parameter.

```
[[TicketQuery(version=0.6|0.7&resolution=duplicate, count)]]
```

This is displayed as:

0

Customizing the *table* format

You can also customize the columns displayed in the table format (*format=table*) by using *col=<field>* - you can specify multiple fields and what order they are displayed by placing pipes (|) between the columns like below:

```
[[TicketQuery(max=3,status=closed,order=id,desc=1,format=table,col=resolution|summary|owner|repo
```

This is displayed as:

Results (1 - 3 of 4)

1 2 ?

Ticket	Resolution	Summary	Owner	Reporter
#9	fixed	Review and resolve open tickets that have been implemented	hkaulbersch	phdmakk
#8	fixed	Fix wiki pages and add relevant attachments where applicable	hkaulbersch	phdmakk
#6	fixed	Investigate and realise deployment strategies	hkaulbersch	phdmakk

1 2 ?

Full rows

In *table* format you can also have full rows by using *rows=<field>* like below:

```
[[TicketQuery(max=3,status=closed,order=id,desc=1,format=table,col=resolution|summary|owner|repo
```

This is displayed as:

Results (1 - 3 of 4)

1 2 ?

Ticket	Resolution	Summary	Owner	Reporter
#9	fixed	Review and resolve open tickets that have been implemented	hkaulbersch	phdmakk

Description

Partially implemented features may be closed as well with a note on the state of the implementation and open issues. If there is a need for further work on these tasks, they will be reopened in the future.

Relevant commits may be indicated by appropriate trac-links.

#8	fixed	Fix wiki pages and add relevant attachments where applicable	hkaulbersch	phdmakk

Use proper formatting on the Wiki pages, especially the development documentation.

```
This is code..
   Properly indented
and formatted
```

This is a Level 0 heading

This is a Level 0 heading

Description

- this is a bullet list
  - with nested elements

1. this is a numbered list
   1. with nested elements
   2. in sequence
2. ...

etc.

This allows for using other facilities of Trac that rely on such formatting, such as Table of Contents, PDF export, etc.

Attach and include screenshots where applicable (attachment permissions have been fixed). Attach other documents where applicable as well (although they should preferably be added to SVN and then linked accordingly which makes management bearable).

#6	fixed	Investigate and realise deployment strategies	hkaulbersch	phdmakk

Description

The easiest approach is to export the tooling as a plugin. Investigate how this can be best achieved and possibly automated (e.g. for nightly builds).

Alternatively, a lightweight and environment independent solution would be to deploy the tooling as an RCP application. Investigate how this can be best achieved and what possible drawback it may entail. If the effort is too high, this option could be postponed or discarded.

1 2 ?

Query Language

`query:` TracLinks and the `[[TicketQuery]]` macro both use a mini ?query language? for specifying query filters. Basically, the filters are separated by ampersands (`&`). Each filter then consists of the ticket field name, an operator, and one or more values. More than one value are separated by a pipe (`|`), meaning that the filter matches any of the values. To include a literal `&` or `|` in a value, escape the character with a backslash (`\`).

The available operators are:

- **`=`** the field content exactly matches one of the values
- **`~=`** the field content contains one or more of the values
- **`^=`** the field content starts with one of the values
- **`$=`** the field content ends with one of the values

All of these operators can also be negated:

- **`!=`** the field content matches none of the values
- **`!~=`** the field content does not contain any of the values
- **`!^=`** the field content does not start with any of the values
- **`!$=`** the field content does not end with any of the values

The date fields `created` and `modified` can be constrained by using the `=` operator and specifying a value containing two dates separated by two dots (`..`). Either end of the date range can be left empty, meaning that the corresponding end of the range is open. The date parser understands a few natural date specifications like "3 weeks ago", "last month" and "now", as well as Bugzilla-style date specifications like "1d", "2w", "3m" or "4y" for 1 day, 2 weeks, 3 months and 4 years, respectively. Spaces in date specifications can be left out to avoid having to quote the query string.

`created=2007-01-01..2008-01-01`	query tickets created in 2007
`created=lastmonth..thismonth`	query tickets created during the previous month
`modified=1weekago..`	query tickets that have been modified in the last week
`modified=..30daysago`	query tickets that have been inactive for the last 30 days

---

See also: TracTickets, TracReports, TracGuide