Trac Reports

Error: Macro TracGuideToc(None) failed

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

The Trac reports module provides a simple, yet powerful reporting facility to present information about tickets in the Trac database.

Rather than have its own report definition format, TracReports relies on standard SQL `SELECT` statements for custom report definition.

> **Note:** *The report module is being phased out in its current form because it seriously limits the ability of the Trac team to make adjustments to the underlying database schema. We believe that the query module is a good replacement that provides more flexibility and better usability. While there are certain reports that cannot yet be handled by the query module, we intend to further enhance it so that at some point the reports module can be completely removed. This also means that there will be no major enhancements to the report module anymore.*
>
> *You can already completely replace the reports module by the query module simply by disabling the former in trac.ini:*
>
> ```
> [components]
> trac.ticket.report.* = disabled
> ```
>
> *This will make the query module the default handler for the ?View Tickets? navigation item. We encourage you to try this configuration and report back what kind of features of reports you are missing, if any.*

A report consists of these basic parts:

- **ID** -- Unique (sequential) identifier
- **Title** -- Descriptive title
- **Description** -- A brief description of the report, in WikiFormatting text.
- **Report Body** -- List of results from report query, formatted according to the methods described below.
- **Footer** -- Links to alternative download formats for this report.

Changing Sort Order

Simple reports - ungrouped reports to be specific - can be changed to be sorted by any column simply by clicking the column header.

If a column header is a hyperlink (red), click the column you would like to sort by. Clicking the same header again reverses the order.

Alternative Download Formats

Aside from the default HTML view, reports can also be exported in a number of alternative formats. At the bottom

of the report page, you will find a list of available data formats. Click the desired link to download the alternative report format.

Comma-delimited - CSV (Comma Separated Values)

Export the report as plain text, each row on its own line, columns separated by a single comma (','). **Note:** Carriage returns, line feeds, and commas are stripped from column data to preserve the CSV structure.

Tab-delimited

Like above, but uses tabs (\t) instead of comma.

RSS - XML Content Syndication

All reports support syndication using XML/RSS 2.0. To subscribe to an RSS feed, click the orange 'XML' icon at the bottom of the page. See TracRss for general information on RSS support in Trac.

Creating Custom Reports

Creating a custom report requires a comfortable knowledge of SQL.

A report is basically a single named SQL query, executed and presented by Trac. Reports can be viewed and created from a custom SQL expression directly in from the web interface.

Typically, a report consists of a SELECT-expression from the 'ticket' table, using the available columns and sorting the way you want it.

Ticket columns

The *ticket* table has the following columns:

- id
- time
- changetime
- component
- severity
- priority
- owner
- reporter
- cc
- version
- milestone
- status
- resolution
- summary
- description

See TracTickets for a detailed description of the column fields.

all active tickets, sorted by priority and time

Example: *All active tickets, sorted by priority and time*

```
SELECT id AS ticket, status, severity, priority, owner,
       time as created, summary FROM ticket
  WHERE status IN ('new', 'assigned', 'reopened')
  ORDER BY priority, time
```

Advanced Reports: Dynamic Variables

For more flexible reports, Trac supports the use of *dynamic variables* in report SQL statements. In short, dynamic variables are *special* strings that are replaced by custom data before query execution.

Using Variables in a Query

The syntax for dynamic variables is simple, any upper case word beginning with '$' is considered a variable.

Example:

```
SELECT id AS ticket,summary FROM ticket WHERE priority='$PRIORITY'
```

To assign a value to $PRIORITY when viewing the report, you must define it as an argument in the report URL, leaving out the the leading '$'.

Example:

```
 http://projects.edgewall.com/trac/reports/14?PRIORITY=high
```

To use multiple variables, separate them with an '&'.

Example:

```
 http://projects.edgewall.com/trac/reports/14?PRIORITY=high&SEVERITY=critical
```

Special/Constant? Variables

There is one *magic* dynamic variable to allow practical reports, its value automatically set without having to change the URL.

- $USER -- Username of logged in user.

Example (*List all tickets assigned to me*):

```
SELECT id AS ticket,summary FROM ticket WHERE owner='$USER'
```

Advanced Reports: Custom Formatting

Trac is also capable of more advanced reports, including custom layouts, result grouping and user-defined CSS styles. To create such reports, we'll use specialized SQL statements to control the output of the Trac report engine.

Special Columns

To format reports, TracReports looks for 'magic' column names in the query result. These 'magic' names are processed and affect the layout and style of the final report.

Automatically formatted columns

- **ticket** -- Ticket ID number. Becomes a hyperlink to that ticket.
- **created, modified, date, time** -- Format cell as a date and/or time.

- **description** -- Ticket description field, parsed through the wiki engine.

Example:

```
SELECT id as ticket, created, status, summary FROM ticket
```

Custom formatting columns

Columns whose names begin and end with 2 underscores (Example: **`__color__`**) are assumed to be *formatting hints*, affecting the appearance of the row.

- **`__group__`** -- Group results based on values in this column. Each group will have its own header and table.
- **`__color__`** -- Should be a numeric value ranging from 1 to 5 to select a pre-defined row color. Typically used to color rows by issue priority.
- **`__style__`** -- A custom CSS style expression to use for the current row.

Example: *List active tickets, grouped by milestone, colored by priority*

```
SELECT p.value AS __color__,
     t.milestone AS __group__,
     (CASE owner WHEN 'daniel' THEN 'font-weight: bold; background: red;' ELSE '' END) AS __style__,
       t.id AS ticket, summary
  FROM ticket t,enum p
  WHERE t.status IN ('new', 'assigned', 'reopened')
    AND p.name=t.priority AND p.type='priority'
  ORDER BY t.milestone, p.value, t.severity, t.time
```

Note: A table join is used to match *ticket* priorities with their numeric representation from the *enum* table.

Changing layout of report rows

By default, all columns on each row are display on a single row in the HTML report, possibly formatted according to the descriptions above. However, it's also possible to create multi-line report entries.

- **`column_`** -- *Break row after this*. By appending an underscore ('_') to the column name, the remaining columns will be be continued on a second line.

- **`_column_`** -- *Full row*. By adding an underscore ('_') both at the beginning and the end of a column name, the data will be shown on a separate row.

- **`_column`** -- *Hide data*. Prepending an underscore ('_') to a column name instructs Trac to hide the contents from the HTML output. This is useful for information to be visible only if downloaded in other formats (like CSV or RSS/XML).

Example: *List active tickets, grouped by milestone, colored by priority, with description and multi-line layout*

```
SELECT p.value AS __color__,
       t.milestone AS __group__,
       (CASE owner
          WHEN 'daniel' THEN 'font-weight: bold; background: red;'
          ELSE '' END) AS __style__,
       t.id AS ticket, summary AS summary_,             -- ## Break line here
       component,version, severity, milestone, status, owner,
       time AS created, changetime AS modified,         -- ## Dates are formatted
       description AS _description_,                    -- ## Uses a full row
       changetime AS _changetime, reporter AS _reporter -- ## Hidden from HTML output
  FROM ticket t,enum p
  WHERE t.status IN ('new', 'assigned', 'reopened')
    AND p.name=t.priority AND p.type='priority'
  ORDER BY t.milestone, p.value, t.severity, t.time
```

Reporting on custom fields

If you have added custom fields to your tickets (experimental feature in v0.8, see TracTicketsCustomFields), you can write a SQL query to cover them. You'll need to make a join on the ticket_custom table, but this isn't especially easy.

If you have tickets in the database *before* you declare the extra fields in trac.ini, there will be no associated data in the ticket_custom table. To get around this, use SQL's "LEFT OUTER JOIN" clauses. See TracIniReportCustomFieldSample? for some examples.

See also: TracTickets, TracQuery, TracGuide