Self-Service SQL Tool User Guide

The Self-Service SQL Tool allows clients with inhouse SQL expertise to write their own database queries and expose them as custom reports right within Timesheet. Each such custom report has its own title, access settings, preferences, and even runtime parameters. This guide will walk you though the basic functionality of the Self-Service SAL Tool and provide examples that illustrate how to use it to build new custom report.

Functionality

The Self-Service SQL Tool works by scanning the "jwt\cgi" directory (on Windows) or the "pi/apache/serverroot/cgi-bin" directory (on Unix). It looks for files with the .pssql extension and exposes each of them as a custom report on the "Custom Tools" page. The "Custom Tools" page can be accessed from a link in the upper right hand corner of any Timesheet screen.

Clicking the "Custom Tools" link takes the user to a page that lists one custom report for each .pssql file that was found. Clicking the title of a report leads them to the "query screen" for that report, where they will provide values for any runtime parameters defined by the .pssql file (see below). After providing those values, they may preview, export, or email the resulting report data.

If the user has administrative privileges, there will also be "Custom Tool Settings" and "Custom Tool Access" pages available in the navigation after they click the "Custom Tools" link. Visiting those pages will allow the administrator to configure the custom reports, as well as specify which roles authorize users to access them.

About .pssql Files

The Self-Service SQL Tool ships with two sample .pssql files: sample.pssql and sample2.pssql. The contents of sample2.pssql are as follows:

```
# This example SQL shows total hours for each
# user-project combo, as long as those hours
# don't total zero
SQL_TEMPLATE = """
select "p"."pname", sum( "tr"."time_amount" )
from "time_recs" "r", "projects" "p"
where "tr"."record_date"
    between <StartDate> and <EndDate>
and "tr"."id_project" = "p"."id_project"
and "p"."id_project" in <ProjectIDs>
group by "p"."pname"
order by "p"."pname"
"""
COLUMN_HEADERS = \
    [
        """Project Name""",
        """Hours""",
    ]
PRINTABLE_NAME = """Total Project Hours"""
```

You will note that the file consists of three sections:

- SQL_TEMPLATE: defines the SQL query that will be run to generate the report output. May contain special tags, such as <UserIDs>, which become runtime parameters of the custom report (more below). The values provided at runtime will be inserted into the query before it is executed.

- COLUMN_HEADERS: these will be used as headers for the report output, and should consist of one value for each field returned by the query.
- PRINTABLE_NAME: this will be used as the custom report title. It is very important to observe the following formatting rules:
    - Do not, in general, change anything you don't need to.
    - The sql query must have triple quotes both above and below the content. If you just edit the SQL content, you should be safe.
    - The sql query must begin with the word "select" (case is irrelevant) or you will get an "Illegal query" message.
    - The column headers should each be on their own line, surrounded by triple quotes with a trailing comma.
    - The printable name should be surrounded by triple quotes.
    - The use of triple quotes make it legal for you to use single and double quotes inside of values.

Tags

PSSQL files allow for special tags to be used in the SQL content which will be replaced by data from runtime parameters when the report is executed. These tags can be used in place of any static value you would normally place in a SQL query. For instance. the following query which contains a hardcoded list of of user IDs:

SELECT fullname FROM users WHERE id_user IN ('steve', 'dylan', 'andrew')

can instead be based on a runtime user selection mechanism by using the <UserIDs> special tag:

SELECT fullname FROM users WHERE id_user IN <UserIDs>

The <UserIDs> tag tells the product to ask the user to select one or more users from a list of all the users in the system at runtime. The users selections then get substituted in place of the <UserIDs> tag and the command becomes dynamic.

A list of special tags (all tags generate appropriate runtime parameters unless otherwise noted) follows.

Date Widget Tags

The following tags each present the user with a date selection widget that includes a popup calendar. Each is guaranteed to be in YYYYMMDD format, so it can be used for comparison to the dates stored in Timesheet (e.g. time_recs.record_date).

- <StartDate> - Informs the user that the report will contain only data on or after the date they select.
- <EndDate> - Informs the user that the report will contain only data on or before the date they select.
- <EndDateDefaultingToPreviousSaturday> - Same as <EndDate>, but the date selection widget will default to the most recent Saturday.
- <StartDateDefaultingToBeginningOfEndDateWeek> - As <StartDate>, but if the user leaves the date selection widget blank, the day 6 days before the specified EndDate will be used. Must be used in conjunction with <EndDate>.
- <StartDateDefaultingToBeginningOfEndDateMonth> - As <StartDate>, but if the user leaves the date selection widget blank, the first date of the month containing EndDate will be used. Must be used in conjunction with <EndDate>.

Static Date Tags

The following tags will not actually expose runtime parameters, but provide values that may be convenient for use in your SQL.

- <FirstDateOfYear> - Will always be January 1 of the current year
- <LastDateOfYear> - Will always be December 31 of the current year
- <CurrentDate> - Will always be the current date in YYYYMMDD format
- <CurrentDateMinusThreeMonths> - Will always be three months before today. The last day of the month three months ago will be used if that date is invalid (e.g., May 31, which would generate the invalid value February 31, will instead return February 28).

ID Selection Tags

Each of the following tags will present a list of names to the user, but will insert an object ID or IDs into your SQL.

- <UserID> - A list of user IDs from which one may be selected
- <UserIDs> - A list of user IDs from which multiple values may be selected
- <ProjectID> - A list of project IDs from which one may be selected
- <ProjectIDs> - A list of project IDs from which multiple values may be selected
- <ProjectIDsWithDescendantOptions> - A list of project IDs from which multiple values may be selected and options to include the children projects of selected projects.
- <TaskID> - A list of task IDs from which one may be selected
- <TaskIDs> - A list of task IDs from which multiple values may be selected
- <PayTypeID> - A list of pay type IDs from which one may be selected
- <PayTypeIDs> - A list of pay type IDs from which multiple values may be selected
- <BillTypeID> - A list of bill type IDs from which one may be selected
- <BillTypeIDs> - A list of bill type IDs from which multiple values may be selected
- <ExpenseID> - A list of expense IDs from which one may be selected
- <ExpenseIDs> - A list of expense IDs from which multiple values may be selected
- <SourceID> - A list of expense source IDs from which one may be selected
- <SourceIDs> - A list of expense source IDs from which multiple values may be selected
- <CurrencyID> - A list of expense currency IDs from which one may be selected
- <CurrencyIDs> - A list of expense currency IDs from which multiple values may be selected
- <MeasurementID> - A list of travel measurement IDs from which one may be selected
- <MeasurementIDs> - A list of travel measurement IDs from which multiple values may be selected
- <ReasonID> - A list of travel reason IDs from which one may be selected
- <ReasonIDs> - A list of travel reason IDs from which multiple values may be selected
- <VehicleID> - A list of travel vehicle IDs from which one may be selected
- <VehicleIDs> - A list of travel vehicle IDs from which multiple values may be selected