ps-plugin-ExcelTools

This plug-in allows you to export

1. Export SQL data to Excel
2. Export CDT data to Excel
3. Export Portal Report to Excel

- Smart Service
- Servlet

4. Export SQL data to CSV
5. Parse Excel to CDT
6. Merge Excel Documents
7. Export Process Metrics to Excel
8. Export Process Model Details to Excel
9. Function queryappianlogs

Modules

Export Portal Report to Excel Smart Service

This exports an Appian report to Excel workbook. You can define the Sheet number where the data needs to be populated.

Export Sql to Excel Smart Service

This takes an SQL statement and popluates the data into the excel workbook. You can specify which tab to write the data to. Tabs are zero based index with first tab =0. You need to have a JNDI defined and the value of the JNDI needs to be passed to this plugin.

In addition, you can write data to any arbitrary cell by defining an array of cell numbers e.g. {"A1","B6","Al4"} and passing values to these cells. This is a one-to-one cell data population.

Currently tested on MySql and Oracle database

Known Issues: If you are using this with Oracle where the columns are created with lower case and require double quotes to be appended to the SQL, it doesnt work.

Export Report Servlet

A servlet that can be used inside an a!safeLink to simulate export to excel of Tempo reports that rely on getportalreportdatasubset().

Tips

You need multiple of these nodes to write to different sheets/tabs of the excel report. The excel report template MUST have these sheets defined otherwise the node will fail

You can configure graphical chart on an excel report for a defined set of rows/columns and when the data is populated in these cells, the data should automatically get translated into the graph (as per your configuration on the template)

Use multiple of these to combine Appian and RDBMS data where required. For e.g. You can write the data from Appian report followed by SQL data in the next tab.

Usage guide

Export SQL to Excel Smart Service

Cell_Keys : Array of cell keys where info needs to be written to e.g: {"A1","B1","C1"}

Cell_values Array of cell values where info provided will be written (for the corresponding values in Cell_Keys) e.g: {"Apple",100,"10/Oct/2012"}

Document_name_to_create Name of the document to be generated.

Document_save_directory Folder to save the generated document

Document_to_overwrite If a document generated should overwrite another existing document, use this. Useful if you are invoking this to write to different sheets of an excel document.

Excel_base_template The excel template on which the output will be written to

include_header_row Boolean true() or false() allowing header row to be written

Jndi Name Name of the JNDI (Quick help: Look at your database node to get the JNDI name.) This will be something like java:/jdbc/AppianDS

Sheet_number Excel workbook Sheet number to write to. First sheet is 0. If your excel has only one sheet, this should be 0 (required)

SQL SQL to use or query. You can use an SQL that contains Select * here. If using Stored Procedures, avoid output parameters. If an output parameter cannot be avoid, make sure the CALL statement names all @ values (i.e. do not pass in only @)

Starting_cell Cell to start writing the report output. For e.g. if you want the output of sql to start from cell "A4" onwards, specify "A4" here.

Export Portal Report to Excel Smart Service

Cell_Keys : Array of cell keys where info needs to be written to e.g: {"A1","B1","C1"}

Cell_values Array of cell values where info provided will be written (for the corresponding values in Cell_Keys) e.g: {"Apple",100,"10/Oct/2012"}

Document_to_overwrite If a document generated should overwrite another existing document, use this. Useful if you are invoking this to write to different sheets of an excel document.

Excel_Base_Template Excel template to be used

Filter_Column Column name where a filter needs to be applied

Filter_Value Value of the filter that needs to be applied.

Include_header_row Boolean true() or false() allowing header row to be written

New_Document_Folder Folder to save the generated document

New_Document_Name Name of the document to be generated.

Report Name of the Appian report (select) that will be written to the excel file.

Report_context Reporting context (if applicable)

Starting_cell Cell to start writing the report output. For e.g. if you want the output of sql to start from cell "A4" onwards, specify "A4" here.

Sheet_number Excel workbook Sheet number to write to. First sheet is 0. If your excel has only one sheet, this should be 0 (required)

Export Portal Report to Excel Servlet

Use SAIL to create an a!safeLink to the following
URL: `<host>/plugins/servet/excelReport?reportId=...`
Pass in the following parameters to the above url like `<paramName>=<paramValue>`. Use `&` to separate additional param/value pairs after `reportId`

- reportId (required)
- filename - if not specified, filename will be "Appian_Data_Export"
- context - a semicolon-delimited list of context IDs
- startIndex - if not specified, will be 0
- batchSize - if not specified, will be -1 (meaning all rows)

Example: `http://localhost:8080/suite/plugins/servlet/excelReport?reportId=23&filename="this_is_awesome"`

Export Portal Report to Excel Servlet v2

This servlet adds WYSIWYG functionality to the original Export Portal Report to Excel Servlet. The values displayed in the Portal Report are exported to Excel, e.g. correctly formatted User and Group names (rather than IDs).

Use SAIL to create an a!safeLink to the following
URL: `<host>/plugins/servet/excelReportv2?reportId=...`
Otherwise usage is the same as Export Portal Report to Excel Servlet.

Example: `http://localhost:8080/suite/plugins/servlet/excelReportv2?reportId=23&filename="this_is_awesome"`

Export SQL to Flat File Smart Service

Delimiter The separator between data fields in the file.

Document_name_to_create Name of the document to be generated.

Document_save_directory Folder to save the generated document

Document_to_overwrite If a document generated should overwrite another existing document, use this. Useful if you are invoking this to write to different sheets of an excel document.

End_of_line The character to be used as a line break. For a carriage return use char(13), for line feed use char(10) and for CR/LF use char(13)&char(10).

Extension The file extension of the file generated.

include_header_row Boolean true() or false() allowing header row to be written

Jndi Name Name of the JNDI (Quick help: Look at your database node to get the JNDI name.) This will be something like java:/jdbc/AppianDS

SQL SQL to use or query. You can use an SQL that contains Select * here.

Parse Excel Spreadsheet to CDT, originally written by Lizzie.

Parse Excel Spreadsheet to CDT Smart Service

- Blank rows are ignored
- Limited to first 100,000 rows
- Stops processing if 500 consecutive blank rows are found
- To keep leading zeros in an integer (for example zip codes), add a leading apostrophe to the column data and it will be treated as a string.

Data Tab

Input	Data Type	Required	Multiple	Descrip
Excel Doc	Document	Yes	No	The Excel document to
Sheet Number	Integer	No	No	The worksheet number

Input	Data Type	Required	Multiple	Descrip
CDT	Any Type	No	-	The CDT type to popula

Output	Data Type	Multiple	Descripti
Return CDT	Any Type	-	The populated CDT
Error Occurred	Boolean	No	Whether an error occurred
Error Text	Text	No	The error message

queryappianlogs() Custom Function

This Custom Function allows to perform an SQL statement against an Appian CSV log file. Only the following SELECT statement is supported:

SELECT [DISTINCT] [table-alias.]column [[AS] alias], ...

FROM table [[AS] table-alias]

WHERE [NOT] condition [AND | OR condition] ...

GROUP BY column ... [HAVING condition ...]

ORDER BY column [ASC | DESC] ...

LIMIT n [OFFSET n] The name of the table must correspond to the name of the CSV log to query. The function is limited to the first 1000 rows.

Input Parameters

Input	Data Type	Required	Multiple	Description
sqlStatement	Text	Yes	No	SELECT SQL Statement to perform

Input	Data Type	Required	Multiple	Description
subFolder	Text	No	No	Specify with a value if the CSV log file is in a log s values: "audit", "data-metrics", "perflogs"
columnTypes	Text	No	Yes	Ordered list (based on the column order in the SC the column will be formatted. Accepted values: "I "Date", "Time", "Timestamp", "String", "Boolean". the same amount of values as the number of colu Statement

Output Parameter

The returned parameter is a CDT made of the following fields:

{

success: Boolean value, based on the success of the SQL Statement execution

errorMessage: Text value, describing an SQL error (if any) returned during the SQL Statement execution

data: List of CDT, where each field corresponds to the column name (or alias assigned). Field type corresponds to the ordered values > set in che input "columnTypes", or Text for all of them otherwise

}