Camshaft MDX Query Tool

Jason Jones

Table of Contents

Introduction

Camshaft is a simple Java utility that executes a MDX query against a given Essbase cube and outputs the results.

There are several configuration options available to achieve the desired formatting.

Runs MDX queries against an Essbase server and outputs the results to a file. You might then use that file to load with a load rule.

This tool is meant to be an alternative to running MDX queries from MaxL and then having to strip a bunch of stuff off of them in order to try and get to a parseable format.

Example Usage

This example requires Java to be on the `PATH` (i.e., you can type `java` at the terminal and it runs fine) This example should be entered as one line but has been put into multiple lines for readability here

```
java -jar camshaft.jar \
  --server=epm11123 \
  --user=admin \
  --password=password \
  --app=ASOSamp \
  --database=Sample \
  --query="SELECT CROSSJOIN({[Jan], [Feb], [Mar]}, {[Curr Year], [Prev Year]}) \
           ON COLUMNS, {[Measures].Levels(0).members} ON ROWS"
  --outfile=mdxpull.txt
```

The output from the preceding example might look like the following:

```
                        Jan, Curr Year          Jan, Prev Year
Original Price          #Missing                #Missing
Price Paid              #Missing                #Missing
Returns                 #Missing                #Missing
Units                   #Missing                #Missing
```

Notice that the combination (tuple) of members from the time and year dimensions are merged into a single entry, separated by a comma. Also notice that the top row has been indented so that it lines up with the data below it. All of the behavior is customizable.

Options

There are numerous options available that control Camshaft's behavior and formatting. You can always run the program with no arguments in order to get a list of all current flags and alternative names you can pass.

`--query=`

You can specify either an MDX query, or a path to a specific file containing just a single MDX query in it. If the given query turns out to be a file, then it'll be laoded and used.

`--no-header`

This suppresses the header rows in case you don't need them. This would cause the following output:

```
Original Price          #Missing                #Missing
Price Paid              #Missing                #Missing
Returns                 #Missing                #Missing
Units                   #Missing                #Missing
```

`--outfile=filename.txt`

You can instruct Camshaft to send the MDX output to a particular file. If no file is specified, then the output is just printed out to the console. Printing to the console might not be too useful as there are numerous diagnostic messages that also print out.

`--no-access-text=#Nope`

Specifies what the text should be for cells witha value of #NoAccess. Default is `#NoAccess`.

`--missing-text=#Mi`

Specifies what the text should be for cells with a value of #Missing. The default is `#Missing`. Here's an example with `--missing-text=-`:

```
                            Jan                         Jan
                            Curr Year                   Prev Year
Original Price              -                           -
Price Paid                  -                           -
Returns                     -                           -
Units                       -                           -
```

`--output-engine=engine`

At present, there are just two values for this, `plain` and `html`. The default is plain. The HTML output engine is experimental, but might generate the following output:

```
<table border='1'>
    <tr>
        <td>Original Price</td>
        <td>#Missing</td>
        <td>#Missing</td>
    </tr>
    <tr>
        <td>Price Paid</td>
        <td>#Missing</td>
        <td>#Missing</td>
    </tr>
    <tr>
        <td>Returns</td>
        <td>#Missing</td>
        <td>#Missing</td>
    </tr>
    <tr>
        <td>Units</td>
        <td>#Missing</td>
        <td>#Missing</td>
    </tr>
</table>
```

`--line-per-header`

By default, tuple members in the column headers will be consolidated down to a single cell value and separated by the separator character (default is a comma and space). You can turn on thie "line per header" option so that the tuple members will print stacked, in multiple lines. For example, turning this on could cause output to look like this:

```
                        Jan                     Jan
                        Curr Year               Prev Year
Original Price          #Missing                #Missing
Price Paid              #Missing                #Missing
Returns                 #Missing                #Missing
Units                   #Missing                #Missing
```

See that there are two lines for the tuple headers (Jan, Curr Year) whereas before they were squished onto a single line.

`--no-indent-header`

Suppresses indenting the header tuples (they are indented by default to look nicer but could cause data load issues for certain tools). Turning this on would do this:

```
Jan                         Jan
Curr Year                   Prev Year
Original Price              #Missing                    #Missing
Price Paid                  #Missing                    #Missing
Returns                     #Missing                    #Missing
Units                       #Missing                    #Missing
```

`--tuple-separator=/`

If you aren’t printing the header tuples in multiple lines, then you can choose what text should be put between differen items. The default is a comma followed by a space. You might want to have something else. For example, a forward slash:

```
                            Jan/Curr Year               Jan/Prev Year
Original Price              #Missing                    #Missing
Price Paid                  #Missing                    #Missing
Returns                     #Missing                    #Missing
Units                       #Missing                    #Missing
```

You can specify spaces on this commaond argument by putting the whole argument in double quotes, such as `--tuple-separator=" / "`. You may need to escape the quotes with a backslash.

`--maximum-fraction-digits=X`

You can specify the maximum number of digits that should be displayed after a decimal. By default this value is 6. You may wish to specify 2 if all of the values in the cube are currency or otherwise do not need more detail.

Dynamic Parameters

Some output engines have other options that can be specified as follows:

`-Dkey=value`

That is, it’s a dash, a capital D (as in Define option), the key name, an equals sign, and then the value. For example, a "format string" can be passed to the plain text output engine, like this:

`-DformatString=%-24s`

The format string is a Java format string and in this case causes the value of a given cell to be printed using exactly 24 characters (no matter what), and to left align it.

`-Ddelimiter=,`

You may want a delimiter such as a comma instead of the default, a tab character.

```
,Jan,Jan,
,Curr Year,Prev Year,
Original Price,Oops,Oops,
Price Paid,Oops,Oops,
Returns,Oops,Oops,
Units,Oops,Oops,
```

Author

Written by Jason Jones.

http://www.jasonwjones.com

http://www.jasonwjones.com/projects/camshaft

License

Distribution binaries are free to use with no warranty implied or given. Currently closed source but the binaries may be considered useable under the Apache Software License, verion 2.0.

Version History

1.0.3

- Option to print properties
- Option to print property names
- Option to specify alias table
- Option to specify revert to member name if no alias in specified alias table

1.0.2

- Fix option to read query from file
- Add option to affect printing of numbers (scientific notation/other): --maximum-fraction-digits
- Fix printing so that large/detailed numbers won't print in scientific notation

1.0.1

- Allow to specify custom field separator
- Lots more options for output such as field separtors, line per header, and more
- HTML output option

1.0.0

- Initial release
- Supports outputting to file and standard output