Defining Data Sets

Data Set Function List

A connector data set is an implementation of abstract java class AbstractDataSet. When defining a data set the following functions require implementation:

- public abstract String getDataSetName();
- public abstract List<ColumnMetaData> getColumns();
- public abstract List<FilterMetaData> getFilters();
- public abstract Object[][] execute(List<ColumnMetaData> columns, List<FilterData> filters);
- public abstract boolean getAllowsDuplicateColumns();
- public abstract boolean getAllowsAggregateColumns();

Other functions that can be overridden:

- public boolean isFilterValueEnabled(String filter);
- public List<Object> getFilterValues(String filter, HashMap<String, FilterData> appliedFilters);

Data Set Function Definitions

public abstract String getDataSetName();

Return a DataSetName as a string. This should not be internationalized.

top

---

public abstract List<ColumnMetaData> getColumns();

Return a collection of ColumnMetaData objects that define the columns that are available in the data set. A column can also be defined to be used as a filter.

ColumnMetaData objects require the following metadata to be defined:

Attribute	Description
columnName	Name of the column. This should be unique and should not be internationalized. User friendly names and internationalization can be applied at the Yellowfin metadata level.
columnType	DataType of the column. See DataType in the appendix for more information.
fieldType	FieldType of the column. See FieldType in the appendix for more information.
availableAggregations	Array of AggregationType. This defines the aggregations that can be applied to this column. See AggregationType in the appendix for more information.
availableFilterOperators	Array of FilterOperator. This defines the operators that can be applied to this column if it is used as a filter. If this column cannot be used as a filter is should be set as **null**. See FilterOperator in the appendix for more information.

There are multiple constructors for ColumnMetaData, that allow for defining this object with a single line of Java code.

top

public abstract List<FilterMetaData> getFilters();

Return a collection of **FilterMetaData** objects that define the filters that are available in the data set. A filter in this context is a parameter that can be used for a report, it does not return data like a column does.

FilterMetaData objects require the following metadata to be defined:

Attribute	Description
filterName	Name of filter. This should be unique and should not be internationalized. User friendly names and internationalization can be applied at the Yellowfin metadata level.
filterType	Filter type of the column. See FieldType in the appendix for more information.
Mandatory	Define whether this filter is mandatory or not. Data cannot be returned from this data set without this filter being set.
availableAggregations	Array of AggregationType. This defines the aggregations that can be applied to this filter. Currently unsupported. See AggregationType in the appendix for more information.
availableOperators	Array of FilterOperator. This defines the operators that can be applied to this filter. See FilterOperator in the appendix for more information.

There are multiple constructors for FilterMetaData that allow for defining this object with a single line of Java code.

top

public abstract Object[][] execute(List<ColumnMetaData> columns, List<FilterData> filters);

Returns the result set from this data set. This is the main function for processing the query. The parameter column specifies the columns that have been chosen for the query from Yellowfin. The parameter filters specifies the filters that have been assigned to the query from Yellowfin. The function must return a 2-dimensional object array with the contents of the columns requested.

The parameter column is of the type ColumnMetaData. This has information about the columns selected and any aggregation modifiers that have been made to those columns.

The ColumnMetaData function getSelectedAggregation() returns the selected aggregation applied to a column. This is of type AggregationType.

The parameter filters is of the type FilterData. This has information about the filters selected and the values that have been assigned to them.

FilterData has the following attributes:

Attribute	Description
filterName	Name of the filter or column that this filter represents.
metaData	FilterMetaData of filter. This will be a link to the FilterMetaData object that was defined in getFilters() or FilterMetaData object that is created automatically to wrap a column that supports filtering.
filterValue	The value of the filter. This is a Java object that holds the data for the datatype of the filter. Filters that hold multiple values will be returned in a Java List. This is for filters using Between, Not Between and In List and Not In List.
filterOperator	Instance of type FilterOperator. This defines the operator that has been applied to this filter.
aggregationType	Instance of type AggregationOperator. This defines the operator that has been applied to this filter.

A custom error message can be shown if an error occurs by throwing a ThirdPartyException(). See **Custom Error Messages**.

top

---

public abstract boolean getAllowsDuplicateColumns();

Return whether or not this data set supports the same column being selected more than once.

If this returns true, then the data set will be sent duplicate columns via the execute function, if they are required.

If this is set to false, then only a single instance of each column will be sent to the execute function and Yellowfin will duplicate repeated columns after receiving the data.

top

---

public abstract boolean getAllowsAggregateColumns();

Return whether or not this data set supports native aggregations.

If this is false, then Yellowfin will enable application level aggregations. This will allow for Yellowfin to aggregate data once it has been returned from the data set.

If this is true, then the data set must return aggregated data when requested. Columns can be defined with what aggregations can be applied to them during report creation. This is defined in getColumns().

top

---

public boolean isFilterValueEnabled(String filter);

Return whether filter values can be returned for a particular filter (or column filter). Returning true will mean that calling getFilterValues() will return a list of values for this filter.

top

---

public List<Object> getFilterValues(String filter, HashMap<String, FilterData> appliedFilters);

If isFilterValueEnabled() returns true for a given filter name, then this function will be called to return a Java List of available filter options for user selection. The parameter appliedFilters will hold a Map (keyed by filtername) that holds the values of other filters that are currently set. This information can be used to further restrict the values returned by this function.

top

Previous topic: Define data source
Next topic: Package connector