Creating Advanced Functions

Yellowfin custom advanced functions extend **com.hof.mi.interfaces.AnalyticalFunction**. In order for the advanced function to be used by Yellowfin, you will need to implement the following methods:

Method	Description
public String getName()	Define the name of the advanced function.
public String getDescription()	Define the text description for the advanced function.
public String getCategory()	Specify or define which category the advanced function will appear under. For example, “Statisitical”, “Analysis”, ” Text”. If the specified category does not exist, a new one will be created.
public int getReturnType()	Specify the data type that will be returned to the report. See advanced function data types.
public String getColumnHeading (String colName)	Set the column name in the report header. The colName parameter is the original name of the selected column.
public boolean acceptsNativeType(int type)	Specify which type of column this function can be run on. Return true for each parameter type that is acceptable. See advanced function data types.
public Object applyAnalyticFunction(int index, Object value) throws Exception	Returns the value to be displayed in each row of the column. This function is run for every row in the data set, where *value* is the object contained at the current *index* within the selected column. If an exception is encountered, the row will be displayed as blank.

The following methods are optional, but are often necessary to perform more complex processes:

Method	Description
public void preAnalyticFunction (Object[] selectedCol)	This function is run prior to applying the advanced function and is used to perform operations across the entire data set. Store data into instance variables to be used in applyAnalyticFunction.
protected void setupParameters()	This function is used to create user input parameters. Create parameter objects and then call addParameter() for each.

Example

Following is an example of a simple advanced function.

```
import com.hof.mi.interfaces.AnalyticalFunction;

public class AccumulativeTotal
           extends AnalyticalFunction{

           private Double total = 0.0;
           public String getName()
           {
                   return "Accumulative Total";
           }
           public String getDescription()
           {
                   return "Calculates the accumulative total for the selected field";
           }

           public String getCategory()
           {
                   return "Analysis";
           }
           public String getColumnHeading(String colName)
           {
                   return "Accumulative Total of " + colName;
           }
           public int getReturnType()
           {
                  return TYPE_NUMERIC;
           }
           public boolean acceptsNativeType(int type)
           {
                   return type == TYPE_NUMERIC;
           }
           public Object applyAnalyticFunction(int index, Object value) throws Exception
           {
                  if (value == null) return null;
                  this.total += Double.valueOf(value.toString());
                  return this.total;
           }

}
```

Previous topic: Basic procedure
Next topic: Parameter set up