DocOutWrapper

The DocoutWrapper custom class greatly simplifies and minimizes the amount of code needed to convert legacy report print programs to use the Barista Docout class. The primary gain comes from adding entire lines to the document rather than loading individual columns.

A program at the beginning of the DocoutWrapper.bbj file illustrates how to use the DocoutWrapper custom class. Run the DocoutWrapper.bbj program to execute and create a report.

The DocoutWrapper custom class has two constructors; one uses the Barista guest account, and one uses a specific Barista user account.

List the firm id, document id, and the report title arguments as follows:

```
  report! = new DocoutWrapper("01", "DocOutTest", "DocOut Test Report")
```

List the firm id, document id, and the report title, user id and password as follows:

```
report! = new DocoutWrapper("01", "DocOutTest", "DocOut Test
Report", "username", "password")
```

The addColumn() method is just like the addColumn() method on the docout object with the exception that it omits the Control Type argument, which the Docout Object documentation says is for future use. For details on the addColumn() method argument list, refer to the Document Output Object document.

Once you have added all the columns to the DocoutWrapper, the program can get a template definition from the DocoutWrapper with the getRowTemplate() method. The program can also get a column count from the DocoutWrapper with the getColumnCount(). The contained program illustrates this well.

```
  dim row$:report!.getRowTemplate()
```

Use a templated string to pass rows to the DocoutWrapper. This is often the best method if only a few columns appear in the print row. For example:

```
  5040 REM PRINT (F)@(139),TOTYTD:M7$
  5041 dim row$:fattr(row$); row.column10$ = str(TOTYTD)
  5042 report!.addRow(row$)
```

Line 5040 is the original print line that printed the total on the report in the tenth column. Lines 5041 and 5042 create the string template, load the data, and call the addRow() method. Using

the template method allows the program to only load the single column needed on this print line. The other 9 columns are already in the string template terminated by line feeds.

To build lines that print to all ten columns, simply create a string with the each column separated from the next column by a line feed character. For example:

```
3030 REM PRINT(F)
@(1),E0$,@(9),NAM$(1,26),@(37),E2$(1,20),@(57),E2$(21,20),@(78),E2$(41,15),
@(94),E2$(56,2),@(97),E2$(58,10),@(109),E4$(1,11),@(125),SCRMTD:M7$,@(139),
SCRYTD:M7$,@(155),SCR$(35,6)
3032 xRow$ = E0$ + $0A$ + NAM$(1,26) + $0A$ + E2$(1,20) + $0A$ + E2$(21,20)
+ $0A$ + E2$(41,15) + $0A$ + E2$(56,2) + $0A$ + E2$(58,10) + $0A$ +
E4$(1,11) + $0A$ + str(SCRMTD) + $0A$ + str(SCRYTD) + $0A$ + SCR$(35,6) +
$0A$
3034 report!.addRow(xRow$)
```

The addRow() method handles both a string template and a normal string with the column data separated by line feeds.

When the report is complete, execute the renderReport() method to present the report in the docout window as shown in **Figure 1**.

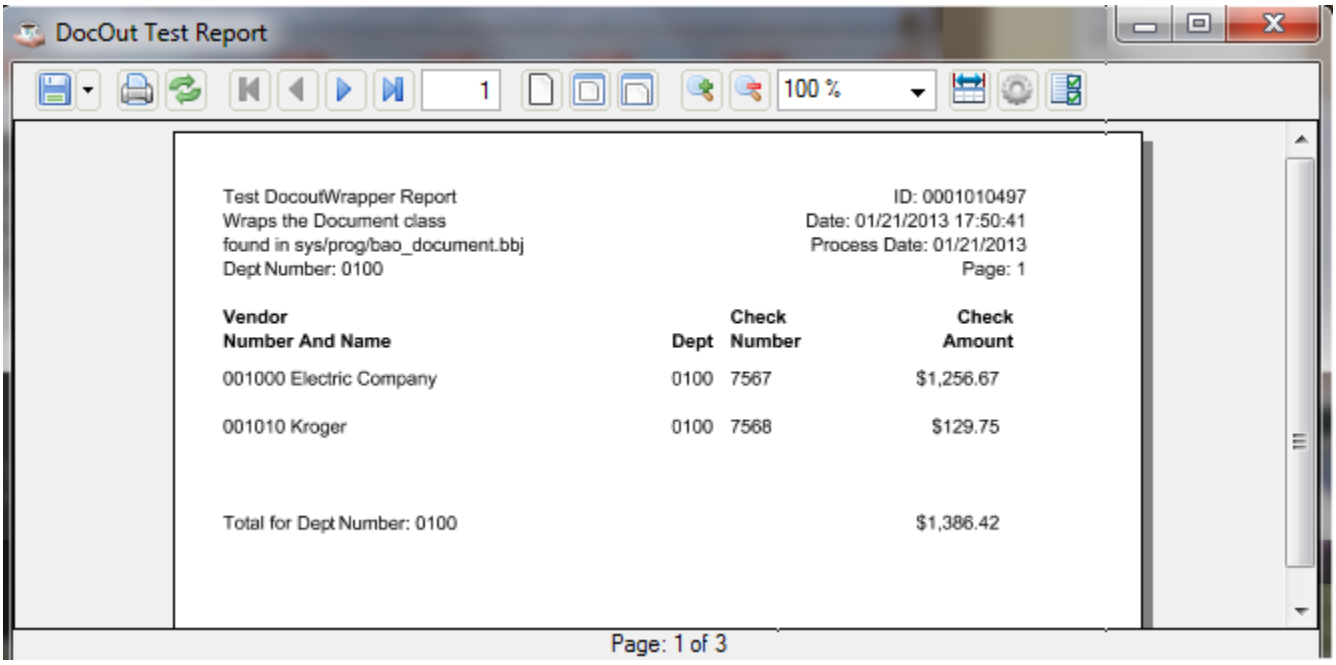

Figure 1. report!.renderReport()

DocoutWrapper is intended for converting legacy report programs to use the Docout object. Often, you will need to run the converted program outside the Barista framework in a session started with a config file that is different than the barista.cfg. The documents render much more effectively if you render them in the Barista environment. DocoutWrapper determines if it is not

running in a Barista environment and if so, will SCALL a new BBj session with the proper config and pass the docout object to the new session using a namespace and running the program renderDocument.bbj.

The DocoutWrapper class traps any errors and passes them up to be handled by the application program by throwing an error. For example:

```
throw errmes(-1) + " : DocOutWrapper renderReport method at " + str(tcb(5)
), err
```

Features and Benefits

- Minimizes the amount of code that must be added to the legacy report program
- Provides multiple approaches to adding report lines to the report
- Uses an add row technique rather than adding individual columns to the output vector
- Hides the handling of the output vector from the application program
- Allows the application program to add a blank line with a simple addBlankRow() method call rather than adding individual empty entries in the output vector, up to the the number of columns in the row
- Renders, when needed, the report by SCALLing a new BBj session with the Barista configuration file, which in most cases will be different than the applications normal config
- Enables developer to convert legacy report programs more quickly than working directly with the Docout object