Customizable Mobile Report Viewer

BBJasperViewer Becomes Highly Customizable and Goes Mobile

If you are displaying a report in a BBj® application, then you have probably taken advantage of the BBJasperViewer functionality. The BBJasperViewer is a utility that allows end users of BBj applications to view and interact with the JasperReports engine, an open source Java reporting library. Reports designed with the WYSIWYG iReport open source, cross-platform report designer rely on the JasperReports engine to generate the report. BASIS built the original BBJasperViewer, found in earlier versions of BBj, around a Jasper Java class. This became problematic when BBj introduced the browser user interface (BUI) because it could not utilize Java client objects in this environment. To overcome this limitation, BASIS rewrote the BBJasperViewer utility in pure BBj code and released it in BBj 12.

Can BUI use Java objects? On the server, yes, on the client, no! Java code requires a Java Virtual Machine (JVM) so without it running on the client, Java cannot run. The client only needs a browser to run BBj code in BUI since one of the great benefits of BBj is that it translates code into the browser-rendered HTML, JavaScript, and CSS. All smartphones, tablets, and other mobile devices include a browser, thus making BBj a truly multi-platform, "write once run anywhere" language. It was therefore only logical to use BBj to create a BUI compatible viewer for JasperReports. Since BBj is an object-oriented language, it was easy to translate much of the Jasper Java Viewer object-oriented code into the BASIS language. Much of this code deals with displaying the report in the viewer. The BBJasperViewer obtains a PNG image for the report from the Jasper API, translates it into a BBjImage via the BBjImageManager and then subsequently displays it using a BBjImageCtrl. The end result is a perfectly displayed report in your browser that is fully interactive, as shown in **Figure 1**.

Figure 1. BBJasperViewer running in a web browser

By Brian Hipple
Quality Assurance Supervisor

Not only does this new BBJasperViewer utility version retain all existing functionality, BASIS added new features and functions as well as an enhanced user interface. More control over the viewer is now possible with new methods such as being able to set the initial zoom rate and page number. A new tool button saves the current report page as a PNG image to better share information from the report with others. For example, a department manager reviews a sales report and sees a transaction that is important for the owner of the company to review. Now what? The manager can simply create an image of that particular page and send it by email as an attachment.

To improve the user experience, BASIS also updated the user interface, shown in **Figure 2**, to include a new child window that houses the various tool buttons - updated with new graphics - that manipulate the report.

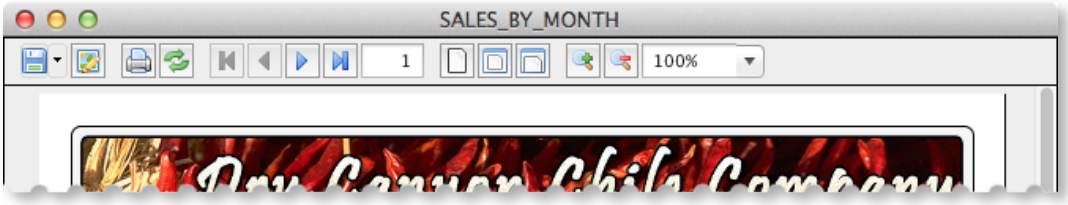

Figure 2. New user interface

Programmatic access to these controls is now possible, making the viewer highly customizable by BBj applications. One particular BASIS customer has customized their viewer by removing the ability to save the report to Google Docs from the [Save] BBjMenuButton. In addition, developers can add any BBj control to the toolbar child window, further customizing the viewer to meet the needs of their application. For a really useful example, have a look at **Figure 3**, which shows the code that adds custom email and fax toolbuttons to the BBJasperViewer toolbar and sets the appropriate callback routines. The developer then adds the backing code to send the current report to one or more email addresses selected from a list and the modification is complete!

```
rem Create the viewer window to display the report
viewerWindow! = new BBJasperViewerWindow(report!)
viewerControl! = viewerWindow!.getViewerControl()

rem Add tool buttons to the viewer control
viewerControl!.addToolbarSpacer()
emailToolButton! = viewerControl!.addToolbarButton("email.png","Send Email")
faxToolButton! = viewerControl!.addToolbarButton("fax.png","Send Fax")

rem Register callbacks
emailToolButton!.setCallback(BBjToolButton.ON_TOOL_BUTTON_PUSH,"OnEmail")
faxToolButton!.setCallback(BBjToolButton.ON_TOOL_BUTTON_PUSH,"OnFax")

rem Show the viewer window
viewerWindow!.setReleaseOnClose(1)
viewerWindow!.show(0)

rem Process the events
process_events
```

Figure 3. Adding the custom email and fax toolbar buttons to the BBJasperViewer window

In addition to these enhancements to the BBJasperViewer, BASIS enhanced the BBJasper utility as well. Beginning in BBj 13 and higher, the utility adds the ability to fill a report with data from a ResultSet, making it possible to use and display Jasper reports in applications without using SQL or SPROCs.

The code in **Figure 4** is an example of this new feature.

```
rem Create a BBJasperReport with result set

rem Use statements
use ::bbjasper.bbj::BBJasperViewerWindow
use ::bbjasper.bbj::BBJasperViewerControl
use ::bbjasper.bbj::BBJasperReport
use java.sql.ResultSet

rem Set the report file
reportFile$ = System.getProperty("basis.BBjHome") + "/utils/reporting/bbjasper/chileco_customers.jrxml"

rem Create the RecordSet
template$ = "CUST_NUM:C(6):LABEL=CUST_NUM:,FIRST_NAME:C(20):LABEL=FIRST_NAME:,LAST_NAME:C(30):LABEL=Last:,"
template$ = template$ + "COMPANY:C(30):LABEL=Company:,BILL_ADDR1:C(30):LABEL=BILL_ADDR1:,"
template$ = template$ + "BILL_ADDR2:C(30):LABEL=BILL_ADDR2:,CITY:C(20):LABEL=CITY:,"
template$ = template$ + "STATE:C(2):LABEL=STATE:,COUNTRY:C(20):LABEL=COUNTRY"
recordSet! = BBJAPI().createMemoryRecordSet(template$)

rem Add records to the record set
data! = recordSet!.getEmptyRecordData()
data!.setFieldValue("CUST_NUM","000001")
data!.setFieldValue("FIRST_NAME","Brian")
data!.setFieldValue("LAST_NAME","Hipple")
data!.setFieldValue("COMPANY","BASIS")
data!.setFieldValue("BILL_ADDR1","5901 Jefferson")
data!.setFieldValue("BILL_ADDR2","")
data!.setFieldValue("CITY","Albuquerque")
data!.setFieldValue("STATE","NM")
data!.setFieldValue("COUNTRY","USA")
recordSet!.insert(data!)
data! = recordSet!.getEmptyRecordData()
data!.setFieldValue("CUST_NUM","000002")
data!.setFieldValue("FIRST_NAME","Dr. Kevin")
data!.setFieldValue("LAST_NAME","King")
data!.setFieldValue("COMPANY","BASIS")
data!.setFieldValue("BILL_ADDR1","5901 Jefferson")
data!.setFieldValue("BILL_ADDR2","")
data!.setFieldValue("CITY","Albuquerque")
data!.setFieldValue("STATE","NM")
data!.setFieldValue("COUNTRY","USA")
recordSet!.insert(data!)

rem Get the ResultSet from the BBjRecordSet
resultSet! = recordSet!.getJDBCResultSet()
resultSet!.beforeFirst()

rem Params
params! = new java.util.HashMap()

rem Create and fill the report with the result set
report! = new BBJasperReport(reportFile$,resultSet!,params!)
report!.fill()

bbjasperViewer! = new BBJasperViewerWindow(report!)
bbjasperViewer!.show(1)
end
```

Figure 4. Using a ResultSet to construct a JasperReport

Summary

What started out as a rewrite to get the report viewer to work in the BUI environment has turned into so much more. BASIS now makes it easier than ever to incorporate your application's custom reporting needs, no matter what environment your application runs in. What are you waiting for? Realize the power of BASIS' BBJasper application building block utility today, available in preview beginning with BBj 12.10 and in full release in BBj 13! ■

- Review these features in the online docs at links.basis.com/basishelp
- Read these BASIS Advantage articles
  - *Jazz up Your Applications - Seamlessly Embed JasperReports* at links.basis.com/09jasperreports
  - *New BBJasper Output Types, Including the Cloud!* at links.basis.com/11bbjasper
- Download the sample code referenced in **Figure 3** at links.basis.com/12reportviewer-code