Le BIRT Expert

The original blog posting is available at:

http://lebirtexpert.com/wordpress/2010/03/02/using-a-popup-debug-window/

Description

It's easy to become frustrated when debugging JavaScript code in BIRT report designs. One little typo can mess up your whole function or event handler. Luckily, the exception messages have steadily improved in BIRT, giving the developers a better clue as to where to look for the problem.

Recently, a JavaScript debugger was included with BIRT, and that can help in some situations. However, I found it a little too buggy to work with sometimes. It doesn't really work well with external js files and doesn't always recognize global variables.

Logging continues to be the best way to output variable values and other messages during runtime. However, sometimes, it's a little tiresome to switch back and forth between the designer and your text editor to view the log file. Not to mention having to search the log file for the messages you outputted for that particular time of running. Also, depending on how you implemented your logging, a new log file might be created each time you run the report, which will require you to open a new file every time.

(Don't get me wrong, logging should be a part of every report design. Perhaps I will cover this in more detail on a future blog posting.)

I discovered a way to create a "Debug Window" in Eclipse that the report can "output" to while running. This is a quick and easy way to display messages inside the Designer using a JavaScript function call. This function will open a "JFrame" inside of Eclipse and print a message to it.

How it works

I have isolated the JavaScript function into its own file "debugWindow.js". Let's take a look at how it works.

```
importPackage(Packages.java.util.logging);
importPackage(Packages.javax.swing);


/****************************************************
 * Set "blnDisplayDebugWindow" to true/false
 * to enable/disable the debug window
 ***************************************************/
blnDisplayDebugWindow = true;
```

```
/*****************************************************
 * Name:          logToDebugWindow
 * Description: Quick and dirty way of displaying a debug
 *                      messages to a Window in eclipse using Swing.
 *                      Uses global variable so all function calls
 *                      print to same window
 * Parameters:  strMessage - message to display
 * Returns:       void
 *****************************************************/
function logToDebugWindow(strMessage) {

      // can turn on/off using this variable
      if(!blnDisplayDebugWindow) { return; }

      // current JTextArea - global parameter
      var currJTextArea = reportContext.getGlobalVariable("jfDebugWindow");

      // if null, then create and store as global variable
      if(currJTextArea == null) {

            // Create JFrame
            var jFrame = new JFrame("Debug Window");

            // Create JTextArea - append message and new-line
            var jTextArea = new JTextArea(20, 100);
            jTextArea.append(strMessage);
            jTextArea.append("\n");

            //Create a JPanel
            var jPanel = new JPanel();
            // add text area to scroll pane
            jPanel.add(jTextArea);

            //Create a vert and horiz scroll bar using JScrollPane
            var jScrollPane = new JScrollPane(jPanel,
JScrollPane.VERTICAL_SCROLLBAR_ALWAYS, JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
            jScrollPane.getHorizontalScrollBar().setUnitIncrement(5);
            jScrollPane.getVerticalScrollBar().setUnitIncrement(5);

            //Add JScrollPane into JFrame
            jFrame.add(jScrollPane);
            jFrame.setSize(400, 400);
            jFrame.show();

            // set JTextArea as a Global Parameter for future reference
            reportContext.setGlobalVariable("jfDebugWindow", jTextArea);
      } else {
            // Print to Global Reference of the JTextArea
            currJTextArea.append(strMessage);
            currJTextArea.append("\n");
      }
}
```

Using the importPackage call, I can bring in and utilize Java Swing classes.

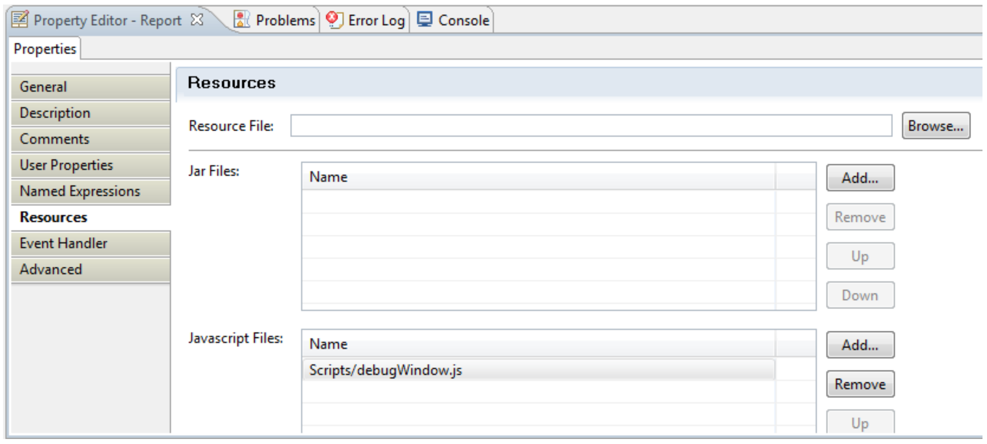
First I check to see if the debug window has been enabled.

If so, check to see if a global variable called “jfDebugWindow” has been created. This variable holds the reference to my “JTextArea”. This is where the messages will be appended to.

If this variable does not exist, then create it. This involves creating the appropriate JFrame, JPanel and JScrollPane. Once they are created, the show() method on the JFrame is called, which actually displays the window.

How to use it

In your report design, reference the “debugWindow.js” file as an external resource. In the report design general properties, navigate to the “Resources” tab. Add the reference to the JavaScript file.



Scripts/debugWindow.js

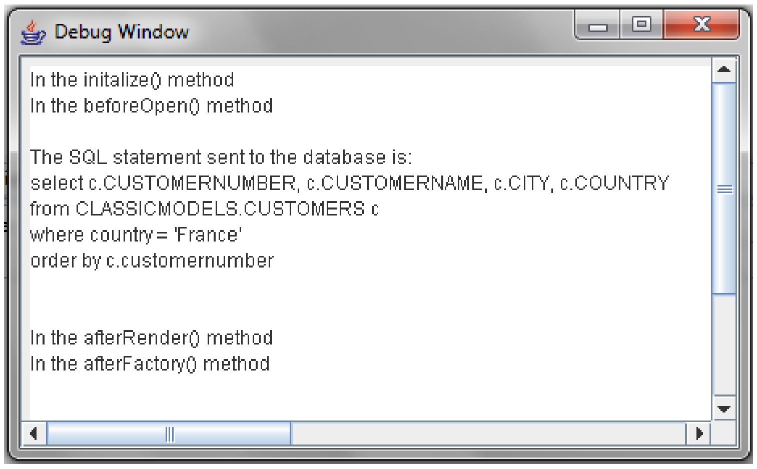
In the “debugWindow.js” file, confirm the “blnDisplayDebugWindow” is set to “true”.

```
// Enable/Disable Debug Window
blnDisplayDebugWindow = true;
```

Use the “logToDebugWindow(msg)” function call to display a message to the Debug Window.

```
logToDebugWindow("In the initalize() method");
```

When the report is run in Preview mode (or in any other format such as PDF, DOC, etc), the Debug Window will popup and display the messages.

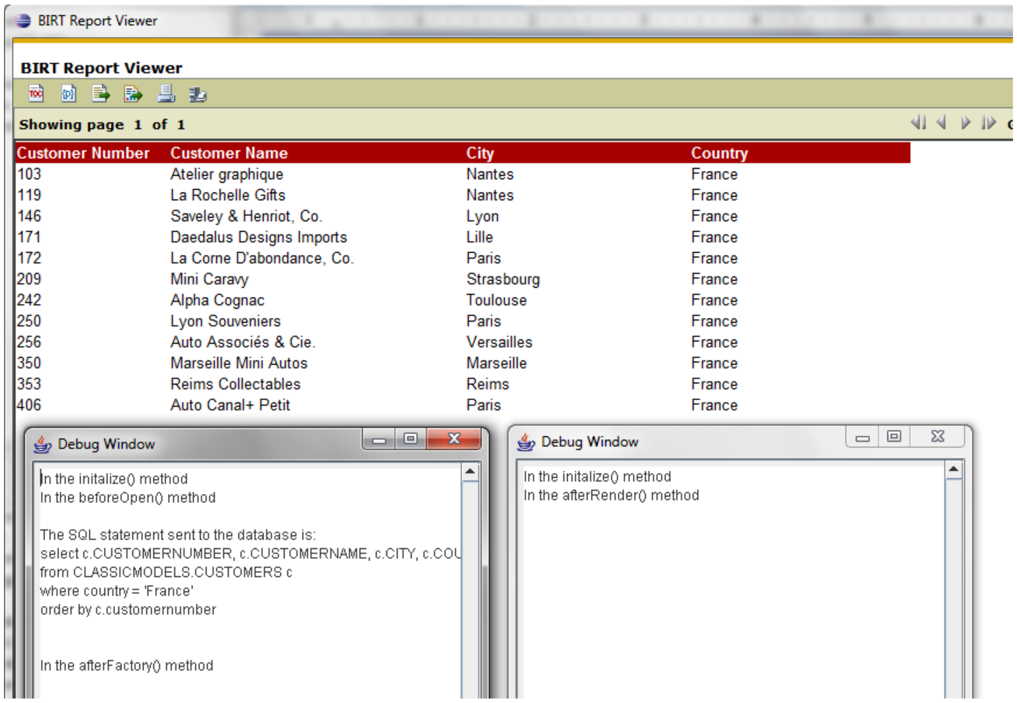


```
In the initalize() method
In the beforeOpen() method

The SQL statement sent to the database is:
select c.CUSTOMERNUMBER, c.CUSTOMERNAME, c.CITY, c.COUNTRY
from CLASSICMODELS.CUSTOMERS c
where country = 'France'
order by c.customernumber


In the afterRender() method
In the afterFactory() method
```

When viewing in the Web Viewer, there will be two windows popup. One is for the execution task and the other is for the rendering task.



Customer Number	Customer Name	City	Country
103	Atelier graphique	Nantes	France
119	La Rochelle Gifts	Nantes	France
146	Saveley & Henriot, Co.	Lyon	France
171	Daedalus Designs Imports	Lille	France
172	La Corne D'abondance, Co.	Paris	France
209	Mini Caravy	Strasbourg	France
242	Alpha Cognac	Toulouse	France
250	Lyon Souveniers	Paris	France
256	Auto Associés & Cie.	Versailles	France
350	Marseille Mini Autos	Marseille	France
353	Reims Collectables	Reims	France
406	Auto Canal+ Petit	Paris	France

Debug Window

```
In the initalize() method
In the beforeOpen() method

The SQL statement sent to the database is:
select c.CUSTOMERNUMBER, c.CUSTOMERNAME, c.CITY, c.COU
from CLASSICMODELS.CUSTOMERS c
where country = 'France'
order by c.customernumber


In the afterFactory() method
```

Debug Window

```
In the initalize() method
In the afterRender() method
```

The Debug Window will be created and displayed *each* time the report is run. If the report is run 5 times, 5 debug windows will be created. You will need to close the window after running the report.

NOTE: The Debug Popup Window will only appear when the report is running inside the Designer. It is recommended that the debug window messages be removed or the debug feature be disabled (using the "blnDisplayDebugWindow" variable) before the report is deployed.

Good luck debugging!!

Le BIRT Expert
David Mehi
expert@lebirtexpert.com
www.lebirtexpert.com