Tech Tip

Capturing the SQL Text for Dynamic Queries

By Chris Geiss
chris_geiss@yahoo.com
http://www.chrisgeiss.com

Tech Tip -- Capturing the SQL Text for Dynamic Queries

By Chris Geiss
Revised 3/19/2002

Table of Contents

About the Author

Chris Geiss is an independent consultant and software developer with over 16 years of software development experience. He has been working with the Actuate reporting products for 4 years and specializes in consulting, development, training and mentoring services for the Actuate e.Reporting Suite. Chris can be contacted at chris_geiss@yahoo.com. See his web site at http://www.chrisgeiss.com.

Introduction

The Actuate e.Reporting Suite has many features that enhance developer productivity. One of these is the Actuate Foundation Classes' ability to dynamically modify the query that is sent to the database based on adhoc or static parameters provided by the user and/or settings made by the developer in Actuate e.Report Designer or e.Report Designer Professional (ERDPro). For example, when a user provides values for adhoc parameters, the AFC modifies the Where clause of the query before it is sent to the database, so that the data can be filtered according to the user's requirements. In addition, by default, when you add group sections to a report design, Actuate automatically modifies the Order By clause of your query so that the sort order of the data set matches the nesting order of the group sections in your report design. Sometimes you need to know exactly how Actuate is modifying your query in order to help troubleshoot problems you may be experiencing. Actuate provides several functions that make it easy to capture the text of the modified SQL query that is being sent to the database. Once you have captured the query text, you can paste it into your favorite SQL tool to debug the query.

Capturing SQL Queries

The GetAppContext() function allows your report to determine in which environment it is running (developer tool, end user tool, Report Server, etc.) SetClipboardText() places the specified text onto the Windows clipboard and is appropriate to use when your report is running in a client side tool such as ERDPro. Although the call will work, it does not really make sense to call SetClipboardText() when the report is running on the Report Server. Instead, you can call ShowFactoryStatus() to output a text string to the completion notice that gets generated when your report runs. You can then use ERDPro Navigator or Administrator Desktop to copy the SQL text from the completion notice text (from the Status tab). The code shown below should be placed in the ObtainSelectStatement() function for your data source. The Actuate Report Factory calls ObtainSelectStatement() when it needs the text of the query that should be sent to the database server. Note, this technique only works with queries that have been created using the Graphical Query Editor (and thus derive from AcSQLQuerySource).

```
Function ObtainSelectStatement( ) As String

      ' Capture the SQL query built by Actuate

      ObtainSelectStatement = Super::ObtainSelectStatement( )

      ' If the report is running in ERDPro, put the text on
      ' the Windows clipboard
      If GetAppContext() = DWBContext Then
            SetClipboardText(ObtainSelectStatement)
      End If

      ' If the report is running on the server, output the text
      ' to the completion notice
      If GetAppContext() = ServerContext Then
            ShowFactoryStatus(ObtainSelectStatement)
      End If

End Function
```

The code above is just a simple example of how you can use these functions. With a little extra work, you can also have the query text output to a text based log file. I often build code like this into a library component that inherits from AcSQLQuerySource, and then use that component in my DataStream slots. In this way, I get this functionality without having to recode it each time I need it. Keep in mind, that if you use this technique, and your report has more than one query source, the call to SetClipboardText() will overwrite any query already on the clipboard. This problem could be overcome using code like:

```
SetClipboardText(GetClipboardText()
+     & Chr$(13) & Chr$(10) & ObtainSelectStatement)
```

Note, that if your report is deployed to a Report Server, you should only enable the ShowFactoryStatus() code when you need to troubleshoot a particular problem. For normal production use, this code should be disabled to avoid the processing and storage overhead of saving the SQL text to the completion notice every time the report runs. This is especially important with very large queries. If you want to get fancy, you could modify the code to examine a registry setting or environment variable on the server. You could thus enable and disable this functionality via the registry key or environment variable you define without having to recompile and redeploy your reports.

How to Report Errors and Omissions

If you find any errors or omissions in this document, or have any suggestions, please email them to chris_geiss@yahoo.com so I can update this document accordingly.