

Using Excel As a Web Front-end for SAS

Tom Taylor
Palladian Analysis & Consulting, LLC
ttaylor@computer.org
Office 713-526-6727 ext 1#
August, 2002

Abstract

We have developed a framework for using Excel running within a Web browser as a user interface for SAS. This is very attractive when users want to see Excel but the application is in SAS on a server. SAS IOM and Microsoft Office Web objects are the enabling technologies. In this presentation, we present the basic terminology and programming requirements. An example of Excel and graphics running in a Web browser and ultimately powered by SAS will be presented.

Introduction

Excel is a tool that many analysts are very comfortable with. I have found over the years there to be some customer resistance when I say their data are to be presented in a SAS format. When I switch over and say their data will be presented in Excel on their desktop machine, there is always immediate acceptance of the point.

A true SAS-ite typically views Excel as slow and cumbersome and the reaction generally is, "Why would anyone bother with Excel when they have SAS?" Most experienced SAS consultants know the answer to this. It is the vast majority of analysts "grew up" in an Excel world and only reluctantly switched to SAS when the problem became too big or complex for Excel.

As an independent consultant, I always seek the tradeoff between meeting design objectives and keeping users engaged in the final solution. Excel has proved to be a big asset in engaging users in the final solution and silencing the mental chant that goes like "Oh please, please, please, not one more piece of software to learn." When they see SAS produced data in Excel sheets on their browser, they see an old acquaintance, one they can communicate with, know which key does what, and immediately get information from. They feel they can do something today with fewer concerns of being hung up for days-on-end learning a new system. This is comfort. They are always ready to give it a try.

The second problem I have had is the project-scuttling cost of SAS on clients machines. Now I believe that every true analyst should have SAS installed on their local machine. The problem is with people who are users of an application written in SAS. These are people who typically use the SAS based application to deal with a very specific process. They use it for a couple of hours a week and past that a spreadsheet with a few hundreds of records is all their world needs. To continue on with the point, I have had two nice well functioning SAS applications completely trashed by clients because the cost of licenses for the client machines was just overwhelming. You know there is not even scrap metal value when people trash a SAS application. It is just exiled to some obscure backup system to be ultimately sanitized from the world with the next backup system upgrade.

The Solution

Based on completion of the most recent semester in life's school of hard knocks (summarized above), my group has settled on using Excel, running in a web browser, as a front-end for SAS. Now this does require paying tribute to the Redmond, Washington based empire in the form of licenses for Excel and some form of Windows. But are we not already doing that? Yes, in fact I believe every reader of this paper is already doing that. So if one can do just this, use Excel in a web browser to interface a user to a SAS application, does this solve the issues of user resistance to a new application? Yes. Does it solve the problem of

expensive and underused SAS clients? Yes. Does it allow users to use practically any Windows based machine to run the application? Yes. Then life is good and maybe there is still a market for custom SAS development after all.

Developments that allow Excel and SAS to work together across the Web

What we have proposed is only possible due to the relatively recent software developments of :

- Microsoft
 - Office 2000 Web Components, consisting of:
 - a. Spreadsheets (has most of Excel's functionality)
 - b. Charts
 - c. PivotTables
 - d. Data Sources
- SAS
 - OLE Automation Interface for SAS - Released with version 6.12
 - SAS Integration Technologies - Released with version 8.x

Review of Microsoft Excel Data Access Techniques

Microsoft has improved many times their data access techniques. All the techniques listed below are still in use today.

- CSV files -- Comma Separated Variables -- mid 1980's
 - No formatting information.
- DDE -- Direct Data Exchange -- circa 1990
 - Literally types directly into a spreadsheet, very temperamental
- ODBC -- Open Data Base Connectivity -- early 1990's
 - Runs in background but requires extensive machine specific setups
- OLE DB -- Object Linking and Embedding for DB -- late 1990's
 - Extension of ODBC to more data types
 - Pure application code setup, more flexible, easier install & maintain
- XML -- Extensible Markup Language -- circa 2000
 - Generalized form for data exchange, modeled on HTML
 - Marketed to be the cost effective EDI (Electronic Data Interchange)
 - Just now becoming mature enough to economically use

Software vendors are rapidly making XML relatively easy, automatic in some cases, to use. Most new projects should be based on XML technology.

Confusing Microsoft Lingo

Microsoft is infamous for renaming and repackaging as time goes on, I think it is just their test to see if one is current on their latest and greatest. At any rate, the following is a recent interpretation of mine of Microsoft newspeak.

- ADO (ActiveX Data Objects)
 - A language-neutral object model that exposes data raised by an underlying OLE DB provider.
 - Most common use is the OLE DB Provider for ODBC drives, which exposes ODBC data sources to ADO.
- Activex is newspeak for
 - COM -- Component Object Model (Microsoft's answer to Java-RMI & CORBA)
 - DCOM -- Distributed COM (COM across machines)

- DAO (Data Access Objects)
 - Superseded by ADO, but still works -- used in Office 97
 - Deprecated -- as the Java developers would say --

Software Needed

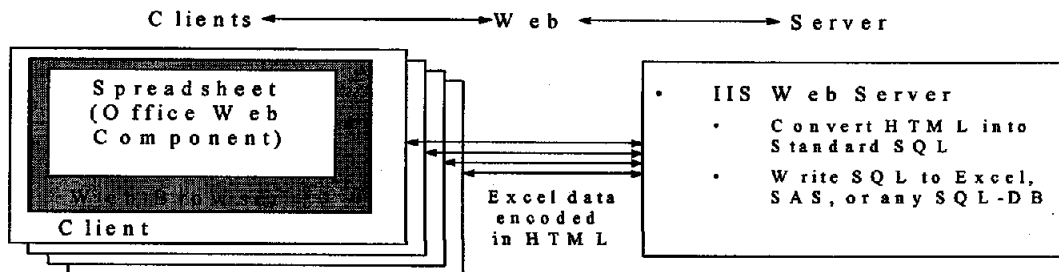
We have used the following software in our development processes. The Microsoft list appears long, but it is mostly development tools. For SAS most any of the math extensions could be added and integrated if there was a need for the product.

What is disappearing from the SAS picture is SAS Connect, the standard telnet based connection tool that allowed SAS clients to talk to a SAS server. The web server effectively replaces Connect.

- SAS
 - Base
 - SAS Integration Technologies (improved Microsoft interface)
 - SAS Access to PC file formats (optional)
- Microsoft
 - Visual Studio 6.0 or later – (ASP Development Environment)
 - Office 2000 Web Components (Generates Excel within Browser)
 - FrontPage (Optional Development Tool)
 - FrontPage extensions on server
 - Office 2000 (only on server)
 - Excel (actually optional)
 - MS Access (optional)
 - IIS 5.0 or later (Web Server)
 - IE 4.01 or later Web browser, we used 5.5
 - Netscape Navigator will work in most cases, also.

What We are Doing

This diagram sums up our basic system. On the server there is SAS and the Microsoft IIS Web Server. All Excel is sent across the web as encoded XML. The spreadsheet runs in the web browser.



Another Look at Data Flow

- Data is entered into spreadsheet in browser
- Browser creates a file of spreadsheet format + data in HTML

- HTML is sent via HTTP or “parsed across the web”
 - Note: HTTP requires text-only data strings, no non-printing characters
- We store the HTML in a MS Access Memo field (optional)
 - Very handy for indexing multiple variations of the same base sheet
- Create a memory-only spreadsheet component on server
 - Write the HTML into the spreadsheet.
 - Read the spreadsheet cell by cell
 - Write the data to any OLE-DB / SQL compatible system

Excel vs. Spreadsheet Component

The Excel we are using is correctly called a spreadsheet component. Nearly identical, there are a few differences.

- Spreadsheet component is almost Excel
 - Designed by the same developers who built the recalculation engine for Excel.
 - Supports nearly all the functions in Excel 2000
 - Excluded functions include FREQUENCY, TRANSPOSE, DATEDIF, LEFTB, plus about 15 others
 - These are all more easily accomplished in SAS anyway
- Comparison of sizes
 - Spreadsheet component -- supports 65,536 rows by 702 columns
 - Excel -- supports 65,536 rows by 256 columns.

FAQ's

Some common questions and answers.

- Can users cut and paste data from browser spreadsheet to a local Excel file? -- Yes, they can even bring up Excel pre-filled with their spreadsheet data, with just one click.
- How fast does it run?
 - On our tests, speed has been fine, not a problem.
- Why not use the SAS IntrDev products?
 - It is not Excel
 - IntrDev is a Java-oriented product
 - Ongoing cost
- What about charts?
 - Complex charts are very quick across this system
- Why not just FTP spreadsheets across the web?
 - For individual sheets this will work, but as part of an application it would not be appropriate.
- Can system access multiple sheets within one Excel workbook? -- Yes

How Microsoft & SAS Communicate

Microsoft overall controls the application. SAS becomes a batch process to be called as needed. We have been very successful in construction .sas files on the fly and then submitting them via a batch call from ASP code.

- Program Control
 - Microsoft ASP controls the overall flow
 - SAS runs as batch jobs kicked off by ASP
- Data
 - Data is communicated by having SAS read & write Excel files, MS Access and small text files.
 - Microsoft ASP should be able to write SAS files directly via OLE-DB, but we have not tried this.

SAS Integration Technologies

Although a limited version of this can actually be accomplished without integration technologies. It works much better with integration technologies and we recommend using it on any project such as this.

- Integrated Object Model (IOM)
 - Provides a much more flexible interface than OLE automation
 - Enables SAS programs to run asynchronously.
 - VB can continue to run while SAS is performing analyses.
 - Allows VB to call SAS even when SAS is running on Unix, OS/390, etc.
- We have not used IOM
 - It would be our first choice on any new project

Code!

This is a small example of a VBScript call to SAS.

```
Dim OleSAS As Object
Set OleSAS = CreateObject ("SAS.Application")
****This is ADO working in creating this object ****
OleSAS.Submit ("libname sample 'c:\sas\sample';
               proc insight data=sample.drug;
               run;")

OleSAS.Quit
Set OleSAS = Nothing
```

Biggest Problems

Most problems are in the interfaces between the products. Occasionally a lot of time can be consumed identifying the problem. Error messages are sometimes unavailable and the developer is left with a careful trial and error game.

- SAS Ole Automation is a little flaky
 - OleSAS.Busy never worked right in our early tries. Integration technologies cures this problem.
- Browser error messages may give irrelevant line numbers.
 - Disinformation can send one down many erroneous paths.
- Parts of the SAS-ASP-IIS-Browser-HTML system provide no error messages,
 - Recognizing these failure patterns is time consuming.. Microsoft .Net should help.
- Microsoft Web security can be mysterious
 - Handling DCOM permissions ate time on one server install

A Little about ASP

ASP is Microsoft. It is a tool and code structure that blends Visual Basic, VB script, JavaScript, HTML, running on both the client and server to produce the desired interface. ASP is Microsoft's scheme to allow VB skills to be used on web development.

- The ASP environment allows for a mixture of
 - Client side VBScript and Java Script
 - Server side VBScript and others languages
 - HTML code can be co-mingled
- In ASP, server script runs between the <% -- %> symbols

Summary

Users are very comfortable with Excel. The fear-of-the-unknown evaporates when we offer to use Excel as the user interface for SAS applications.

The web-based Microsoft front-end to SAS allows for across-the-web use of, spreadsheets, pivot tables, and charts.

SAS for this application is required only on the server. SAS Analysts would still want SAS on their workstations. This software development system presents a big learning curve for developers.

References

- OLEDB for the ODBC Programmer by Michael Pizzo and Jeff Cochran (Visual Studio 6.0 CD, a C++ Article)
- SAMPLE: ExcelADO Demonstrates How to Use ADO to Read and Write Data in Excel Workbooks (Q278973) – MSDN Library
- HOWTO: Query and Update Excel Data Using ADO From ASP (Q195951) -- MSDN Library
- Manipulate Excel Spreadsheet Data in ASP using ADO, Jason Fisher (ActiveServer Development Journal, Sept. 2001, MSDN Library)
- Embedding Your Spreadsheet Data into Web Pages by Ulrich Schwanitz www.ASPToday.com subscription
- Exporting ASP Reports to Excel Format by David Schultz www.ASPToday.com subscription
- Developing an Open Client in Visual Basic David Jan, SAS Institute, Inc. SAS Website
- How to Use DAO to Link an Excel Spreadsheet Article ID: Q173748 -- MSDN Library
- Chapter 5: Choosing the Right Data Access Technology (Visual Studio 6.0 CD)
- Microsoft Universal Data Access Web Site (microsoft.com/data/ado)
- Introducing the Office Web Components --MSDN Library
- Opening Other Microsoft Database Formats by Using the Jet 4.0 OLE DB Provider -- MSDN Library
- Microsoft Office 2000/Visual Basic Programmer's Guide Chapter 14: Microsoft Office Data Access Technologies: OLE DB and ADO
- Web-Based Excel Reports, by Chandra Raghawendra, 11/21/2000, www.ASPToday.com subscription

Presentation Posted at Website
<http://www.palladian.ac>
Select the Excel and SAS Presentation button