Paper 395-2008

Rich Internet Applications Using SAS/IntrNet® and Microsoft Silverlight

Alan Churchill, Savian, Colorado Springs, CO
Don Henderson, Henderson Consulting Services, Olney, MD

Important note: Please consult:

http://www.sascommunity.org/wiki/Rich_Internet_Applications_Using_SAS/IntrNet_and_Microsoft_Silverlight

for an up-to-date version of this paper. Silverlight 2.0 is a major release and where the focus of this paper lies. That release is expected to be made available 2 weeks before SAS Global Forum 2008. It will be given to the public at the Microsoft MIX08 conference on March 3rd. The content of this paper was constrained by the due date for the Proceedings, and will be out-of-date at the time of SAS Global Forum 2008.

ABSTRACT

As the Web moves into the next phase of its existence, the old static world of Web technology must change. The original concept behind the Web was a static one for displaying documents, yet slowly interactivity crept into Web applications. After a number of years, the classic Web of HTML/JavaScript began to show a lot of signs of stress as those technologies aged and browsers changed.

Users were impacted if they upgraded or changed browsers. Behind the scenes, though, Web developers were significantly impacted by confusing coding guidelines as well as having to test continually on multiple browsers and versions. Additionally, the toolsets for developing web applications were hard to use especially for debugging.

Web 2.0 is different. Web 2.0 will have established frameworks for building applications and will use professional development platforms with robust debugging and testing support. A bright area in the Web 2.0 world is the concept of Rich Internet Applications (RIAs): applications that look and feel like a Windows application but are, in fact, hosted on the Web. These new RIAs will have robust graphics, locally cached storage, fast application response time, and the ability to do a lot of tasks locally rather than on a server. This paper will focus on one of the newest technologies to allow for RIAs known as Microsoft Silverlight.

Many SAS Web developers use either SAS/IntrNet or use other technologies such as the SAS 9 Business Intelligence/Enterprise Business Intelligence (BI/EBI) Architecture (that are built on top of SAS Integration Technologies) to expose SAS information to the Web. This paper will demonstrate how to incorporate existing SAS products into an RIA framework. The examples presented here use the SAS/IntrNet Application Dispatcher but could just as easily use the SAS 9 BI/EBI Technologies.

INTRODUCTION

HTML and JavaScript have outlived their value.

While this is a bold statement, the world of Web programming simply needs more robust technologies to approach the level of interaction desired by most users. Various types of ‘hacks’ have appeared over the last 15 years to try and simulate a better user experience, but none have been robust or well thought out. From older methods such as JavaScript, ActiveX, and DHTML to more modern means such as AJAX, these approaches have been taken to try and bridge the gap between the Windows and the Web experience.

However, while these approaches may appeal to end-users somewhat, the development and maintenance of these applications is extremely difficult. Developers must have complex JavaScript implementations, for example, just to support differences in AJAX calls. AJAX requires large amounts of JavaScript code to address various browsers and versions. Couple this with limited/complex debugging support and web applications quickly escalate in cost and developer frustration.

RIAs are a vision for where the Web should be. It is a vision rooted in the idea that modern, object-oriented languages, debuggable code, and rich development libraries should be the developer norm, and that the end-user experience must be in line with a Windows experience. A number of competing frameworks have come onto the market to support modern RIA development. These include Adobe AIR, Microsoft Silverlight, OpenLazlo, and others.

This paper will focus on Silverlight and the use of SAS to generate Silverlight content. This is done merely to focus the paper and provide a starting point. SAS developers should be able to use the same concepts with the other frameworks if they so desire.

Silverlight can be integrated with SAS in a number of ways. Because Silverlight utilizes an XML language for doing the markup, it is easy to generate that XML using SAS. This provides a bridge for existing SAS developers and opens the door to do more robust work once they familiarize themselves with the basics of the markup language and the deployment process.

WHY ARE RIAS IMPORTANT FOR SAS DEVELOPERS?

1. **Aids in understanding information**

   RIAs are important because they deliver information more interactively. That means faster decision making. The ability to display information to a user and allow for rich interaction means that business information is easier to consume, and that leads to better business intelligence. Advanced graphical capabilities allow for new, creative means of exploring data.

2. **Easier to maintain**

   RIAs are easier to maintain since browser and operating system support is baked in, and the developer does not have to worry about how the information is rendered on various platforms.

3. **Helps you to keep up with the times**

   RIAs are here. Users are experiencing these changes while they surf, and that means a trickle-down effect at the business level. The more users see these types of rich applications elsewhere, the more they will want the corporate IT departments to support rich functionality.

SILVERLIGHT DEFINED

Silverlight is Microsoft's entry into the RIA space and version 1.0 was introduced in May 2007. Silverlight 1.0, while eye-opening, still used JavaScript, which a lot of developers want to move away from. Fortunately, at the same time, Microsoft announced that version 2.0 of Silverlight would allow for all development to be done using .NET[1] languages. This was important since .NET already has millions of developers in place, robust development tools, thousands of 3rd party applications, and is very widely used.

What are Silverlight's strengths?

- Silverlight is cross-platform and runs on Linux, Windows, and Macs.
- It is cross-browser and runs on Internet Explorer, Opera, Firefox, and others.

- Silverlight is a safe choice for Web developers since it runs on the vast majority of platform/browser combinations available today.
- Silverlight applications can also be coded in a variety of languages such as C#, VB.NET, Ruby, Python, C++, and many others.
- Microsoft is also releasing a Dynamic Language Runtime (DLR) that will allow for backend code to be dynamically compiled. This is important because the code that drives events on the Website can be generated by SAS. Dynamic code can be generated by any language and will automatically be compiled as needed. This will allow for fast code execution rather than having to parse the code each time it is called.
- Silverlight applications can be converted into Windows applications and vice-versa [2]
- Silverlight is fast. Very fast. It is ~200x faster than JavaScript.[3]
- Silverlight is free and Microsoft provides developer tools to simplify the creation of Silverlight applications. These tools include Microsoft® Expression Blend and Microsoft® Visual Studio 2008.
- There is a large base of 3rd party tools that greatly simplify development.

SILVERLIGHT COMPONENTS

A Silverlight application consists of 2 types of files: “code-behind” and Extensible Application Markup Language(XAML). Code-behind handles the logic and event execution side and XAML handles the layout and presentation side.

When creating a Silverlight application, it is important to keep the following in mind:

1. If the goal is to only present data, only the XAML is required.
2. If interaction with the page is required, the code-behind files need to be generated.

Microsoft offers tools to help create the XAML and the code-behind, although both can be coded using a text editor. These tools are Microsoft Expression Blend for creating the XAML and graphical layout and Microsoft Visual Studio for creating the code-behind. These tools greatly simplify the creation of the application, and work hand-in-hand with each other. Editing files in one tool will automatically reflect in the other tool. Once a generalized XAML layout is established, SAS can be used to customize that markup code to make it more dynamic.

Code-behind

“Code-behind is a term used to describe the code that is joined with the code that is created by a XAML processor when a XAML page is compiled into an application.”[4]
What does this mean though? XAML lays out how a page looks and handles some behaviors such as animation. Code-behind is processing logic that handles events such as the page load, a button click, a treeview select, and other events. Code-behind is also used for business logic processing such as delivering database content to the screen.

Code-behind can be written in any .NET language including C#, VB.NET, C++, and others.

XAML

XAML is what is used in Silverlight to display a page. Think of it as similar to HTML but much more powerful. XAML was developed by Microsoft as a way to describe graphical elements and their actions. XAML is a common mark-up language that can be used by Windows applications, Web sites, mobile applications, and console applications. This ubiquity of a markup language means that a simple application should run, as-is, across multiple platforms. It also reduces the amount of knowledge a programmer must have when porting from one platform to another.

Here is some example XAML code that displays “Hello World” in a textbox:

```
<Canvas
   xmlns="http://schemas.microsoft.com/client/2007"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   Loaded="Page_Loaded"
   x:Name="parentCanvas"
   Loaded="Page_Loaded"
   x:Class="HelloWorld.Page;assembly=ClientBin/Demo1.dll"
   Width="640"
   Height="480"
   Background="White"
   >
<TextBlock Width=”202” Height=”25” Canvas.Left=”70” Canvas.Top=”74” Text=”Hello
World” TextWrapping=”Wrap” />
```

XAML can be complex and can include code inline similar to HTML and JavaScript. At its basic, XAML operates with controls held within canvases. Silverlight 2.0 comes with a lot of built-in controls including buttons, text boxes, data grids, labels, tabs, and sliders. Additionally, developers can build their own controls that can then be easily integrated into Silverlight.

The main control users will typically use is a canvas. A canvas is simply a holding container for other controls. A canvas can have other canvases, providing a tremendous amount of flexibility. Hence, a developer can create common canvases with specific functionality and then just include that canvas as needed within an application. For a SAS programmer, once they have a basic canvas laid out, perhaps a treeview that gets the datasets available in a library, they can just reuse this canvas as needed from one project to the next.

Since XAML is merely text, SAS can generate the XAML in a number of ways.

SAS COMPONENTS

There are a number of SAS tools and technologies that can be used to generate XAML, e.g.,

- ODS templates and tagsets created using the TEMPLATE procedure
- DATA step using PUT statements
- SAS Server Pages as described in the SAS Press book “Building Web Applications with SAS/IntrNet®: A Guide to the Application Dispatcher”[5]

Since RIAs are, by definition, Web applications, it is important that the generation of the XAML content be handled via a Web interface such as:

- The SAS/IntrNet Application Dispatcher
- htmSQL (part of the SAS/IntrNet product)
- The SAS 9 Stored Process Server (part of the SAS Integration Technologies product)
- A Web Service that packages any of the above under the covers.

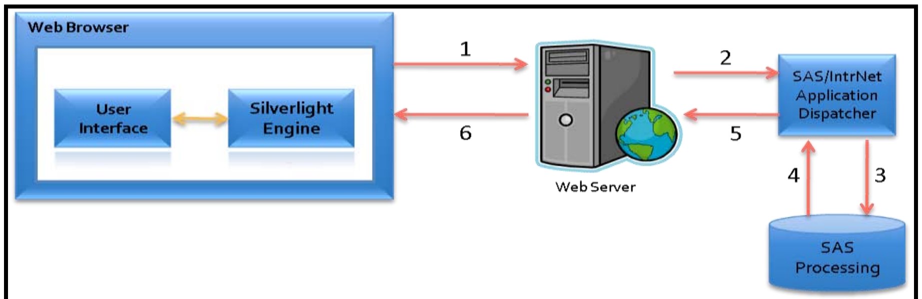
For simplicity, the rest of this paper refers to SAS/IntrNet or the SAS/IntrNet Application Dispatcher. However any of the four options listed above can be used. The choice as to which is the most appropriate will depend on the application and the environment.

SAS/INTRNET

Since XAML is simply XML text, it is easy to use SAS to generate the text. In fact, SAS can be used to generate not just Silverlight applications but any application that can interpret XAML. For example, Windows applications can now be created using XAML and so can a number of other applications including mobile applications.

For Silverlight, the natural choice within SAS for rendering the mark-up can be the SAS/IntrNet Application Dispatcher. Simply put, instead of using SAS/IntrNet to render HTML, use it to render XAML.

Here is a high-level illustration of the overall process:



1. User requests a Web page, interacts, and submits the Web page.
2. Web server passes request to SAS/Intrnet Application Dispatcher.
3. The Application Dispatcher executes a SAS program.
4. The SAS program generates XAML code and sends it back to the Application Dispatcher.
5. The Application Dispatcher passes it back to the Web server for delivery to the requesting client.
6. The Web server delivers it to the requesting client. The Silverlight engine at the client interprets the XAML and displays it to the user.

Figure 1 High-Level Overview of Silverlight to SAS Process

When a user calls a SAS/IntrNet application, a SAS program is executed that renders the Silverlight page. That page is delivered to the user's browser where the Silverlight engine interprets the XAML and displays the content to the user. After a user makes any edits or changes, the information can again be passed to a SAS/IntrNet application where the information can then be acted upon.

HOW TO CREATE A SILVERLIGHT-ENABLED APPLICATION USING SAS

Important note: Please consult:

http://www.sascommunity.org/wiki/Rich_Internet_Applications_Using_SAS/IntrNet_and_Microsoft_Silverlight

for an up-to-date version of this paper. The release of Silverlight 2.0, combined with the availability of third party controls, will have a major impact on the details regarding using SAS as a back-end content generator. The link above has more details regarding the use of SAS as a content generator.

The basic process flow can be summarized in the following steps:

- Design a Silverlight application using Microsoft Expression Blend.

- Use the XAML code produced by Expression Blend as a guide to determine what XMAL the SAS program(s) needs to generate. By simply examining the XAML code produced by the Expression Blend editor, SAS programmers can replicate that code inside of their SAS programs to implement the same functionality dynamically using SAS data
- Code-behind should be created using Microsoft Visual Studio 2008, which works hand-in-hand with Expression Blend.

The SAS process that creates the XAML needed by the Silverlight application may be invoked using any of the products/architectures discussed above (e.g., the SAS/IntrNet Application Dispatcher, htmSQL, the SAS 9 Stored Process Server, a Web Service)

CONCLUSION

RIAs open up the possibility for really rich applications on the Web. These rich applications can open up the business intelligence possibilities by creating very attractive graphical environments. SAS Web developers should consider deploying new RIAs within their environment. These RIAs tend to be easier to code against, easier to maintain, and offer users a much better mechanism for interactivity.

Silverlight is a good choice for SAS developers to get started with. With its use of XAML as the primary display methodology, SAS programs can create the necessary files to build advanced Silverlight applications while allowing SAS programmers to leverage their existing knowledge.

REFERENCES

Henderson, Don. Building Web Applications with SAS/IntrNet®: A Guide to the Application Dispatcher. 1.0. Cary: SAS, 2007. [6]

SAMPLE SILVERLIGHT SITES

Almost all Silverlight sites to date have been referenced here:

http://silverlight.net/showcase/

Cynergy Systems is one of the top RIA firms today and recently won the PhizzPop competition:
http://www.cynergysystems.com/

ACKNOWLEDGMENTS

Thanks to Vince DelGobbo of SAS for the encouragement and advice and Mike Wolf and Dave Wolf of Cynergy Systems for the inspiration.

RECOMMENDED READING

http://Weblogs.asp.net/scottgu/
http://silverlight.net/
http://www.sascommunity.org/wiki/SAS_Server_Pages

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Alan Churchill
Savian
68 W Cheyenne Mtn. Blvd
Colorado Springs, CO 80906

Work Phone: 719-687-5954
E-mail: http://www.savian.net/contact.aspx
Web: http://www.savian.net

Don Henderson
Henderson Consulting Services
3470 Olney Laytonsville Rd., Suite 199
Olney, MD 20832

Work Phone: 301-570-5530
E-mail: http://www.hcsbi.com/contact/contact.aspx
Web: http://www.hcsbi.com

---

[1] .NET is Microsoft's generalized programming framework. Almost all Microsoft development is done in .NET from Windows to Web to mobile applications. .NET supports numerous programming languages including C#, VB.NET, C++, Perl, Python, Ruby, and others.

[2] Silverlight uses a subset of Windows Presentation Foundation (WPF) which is the new method for building Windows applications. Some WPF functionality is not available in Silverlight so not all code can be ported from a Windows application to a Web application.

[3] InfoWorld, *Microsoft Silverlight rivals Flash, AJAX*, Oct 1, 2007.

[4] http://msdn2.microsoft.com/en-us/library/aa970568.aspx

[5] Requires the SAS/IntrNet product.

[6]
http://www.sascommunity.org/wiki/Building_Web_Applications_with_SAS/IntrNet:_A_Guide_to_the_Application_Dispatcher