Paper 234-30

ASP.Net and SAS® - A New Model for Developing Web-based SAS System Applications.

Graham Murray, Independent Consultant - WebMiner.co.uk, Dunfermline, Scotland, UK.

ABSTRACT

Microsoft has invested heavily in the .NET architecture as its vision for the future of the Windows platform. One of the primary components of this 'big idea' is the ASP.Net Framework. SAS System features can be utilised from within ASP.Net applications through the SAS Integration Technologies product and this paper sets out how to make full use of the power of the SAS System using ASP.Net and related technologies such as ADO.NET, XML and 'Web Services'. Enabling technologies, development environment options and system requirements for building web-based SAS applications will be described and a possible 'development architecture' will be outlined. The paper will build on the a Reports Quick Start sample provided at http://www.asp.net and describe how the freely available Web Matrix ASP.Net development tool can be used to quickly build SAS System applications. Finally, the presentation will briefly describe the emerging standard that is known as 'Web Services' and how this exciting new methodology can provide a framework for making use of legacy SAS code in web-based enterprise applications.

INTRODUCTION

These days nearly all of the reference material you'll ever need for building systems and solving problems is available online. In this paper I will focus on the key code elements to implement the hooks between the SAS System and Microsoft's .NET technology. A fuller description of .NET and it's associated languages and interfaces can be found in the internet links and references at the end of the paper.

.NET AND ASP.NET – WHY?

'One of the things people are looking for is a one-sentence definition of .NET. What is it? Why should I care? .NET is Microsoft's strategy for software that empowers people any time, any place, and on any device.'

(as long as it's Windows ! Ed.) - Microsoft TechNet Article based on a presentation by Alan Le Marquand, Program Manager of Field Content for TechNet and MSDN.

Other agendas notwithstanding, the primary reason Microsoft have developed the .NET architecture is to support XML Web Services. Microsoft (as well as most of the other major software vendors) see a future world where software components, and latterly business transactions, are served - and therefore can be charged for - on a 'need to use' basis. Later in this paper I will return to Web Services but in the meantime it is sufficient to understand that Web Services, at a simplistic level, are just program modules or objects available to be served over the internet, which send and receive messages in XML. The .NET architecture consists of two main components, the common language runtime (CLR) and the .NET framework class library. The CLR handles the low-level facilities such as memory management, thread execution and other system services. Whereas the .NET Framework is an object-oriented class library supporting common programming tasks like string management, database connectivity and other similar functionality.

Probably the single most important enhancement .NET provides over earlier Windows releases is the end of the '.dll hell' issue. The .NET Framework supports separate assemblies per application thereby allowing 'side by side deployment'. NO MORE SHARED .DLLs !

Another reason 'why' for .NET is almost certainly Microsoft attempting to ensure it doesn't lose ground in the application/web server world dominated by Java and the Apache web server.

SAS/INTEGRATION TECHNOLOGIES – WHY?

Traditionally, the SAS System has been used strategically in many organisations, with the number of real users limited to maybe a few dozen highly skilled 'Power Users'. Many IT departments, in my experience, seem a bit daunted by the product, perceiving it as 'difficult' or hard to use. This is largely because the SAS System is very comprehensive as well as being cohesive and self-contained. Windows and Java developers don't expect the kind of integration between product modules to be as good as it is in SAS. On the other-hand, traditional methods of deploying SAS applications required a fairly heavy 'footprint' on the desktop for response times suitable for an end-user. The introduction of SAS/Intrnet and Integration Technologies changes all that as well as the new server-based architecture of SAS 9.

SAS/Intrnet is discussed in many SUGI papers and the SAS user community is well aware of it's function set and what it provides. Integration Technologies is aimed more at software developers and is not yet so widely used or

known in the SAS community.

One potential benefit of using SAS Integration Technologies is that parts of it are bundled with Base SAS. With ASP.Net applications this means that you can have full access to SAS System facilities from an ASP.Net written web page with a Base SAS license, **as long as you have Base SAS installed on the same physical machine as your web server.** To serve ASP.Net applications from a web server running on a different machine to the SAS server install would still be possible but requires a SAS/Integration Technologies license.

In version 8.2, the Integration Technologies components bundled with Base SAS, allow read-write access to SAS data. In version 9, Base SAS Integration Technologies only provides a read-only engine for SAS data access without a SAS/IT license.

This method of accessing SAS functionality is made possible by interaction with an API that SAS call their 'Integrated Object Model' or IOM. In SAS 8.2 running on the Windows platform this is implemented using support for the Microsoft Component Object Model or COM.

ASP.NET WEB APPLICATION DEVELOPMENT TOOLS

The .NET Framework is the main component for developing ASP.Net web applications irrespective of SAS and is available as a free download for Windows 2000 (Professional and Server versions) as well as Windows XP. It is also an integral part of Windows Server 2003. Although the framework in itself is really just the sub-routine library - can I call it that or am I giving away my age ? - to really get going with .NET you need an IDE (Integrated Development Environment). Visual Studio.Net (VS.Net) is the primary enterprise-level IDE but Microsoft have also released a tool called 'ASP.Net Web Matrix', which is available as a free download. This looks and feels like VS.Net but is essentially a program file editor rather than being project-based like Visual Studio. For simple web page development ASP.Net Web Matrix is a straightforward and effective tool to use. At the time of writing this paper Microsoft are now providing a download of a beta release of Visual Studio Express, which is also currently free for non-commercial development.

COMPONENTS OF SAS/IT

The introduction of Integration Technologies with version 8.2 opens up the SAS System toolset in a way not possible in earlier versions. There are a number of components of Integration Technologies, each with different purposes. The components applicable to .NET application programming are the 'Workspace Manager' and the 'IOM - Integrated Object Model - Data Provider'. These components open up the SAS programming language interpreter and the SAS data library architecture to the external programming language interfaces provided by ASP.Net - in this paper we shall see examples using Visual Basic.NET and C#. It is important to be aware that although this paper focuses on Microsoft's ASP.Net Framework, these same facilities can be utilised Windows applications, console applications and also from Java using the IOM Bridge.

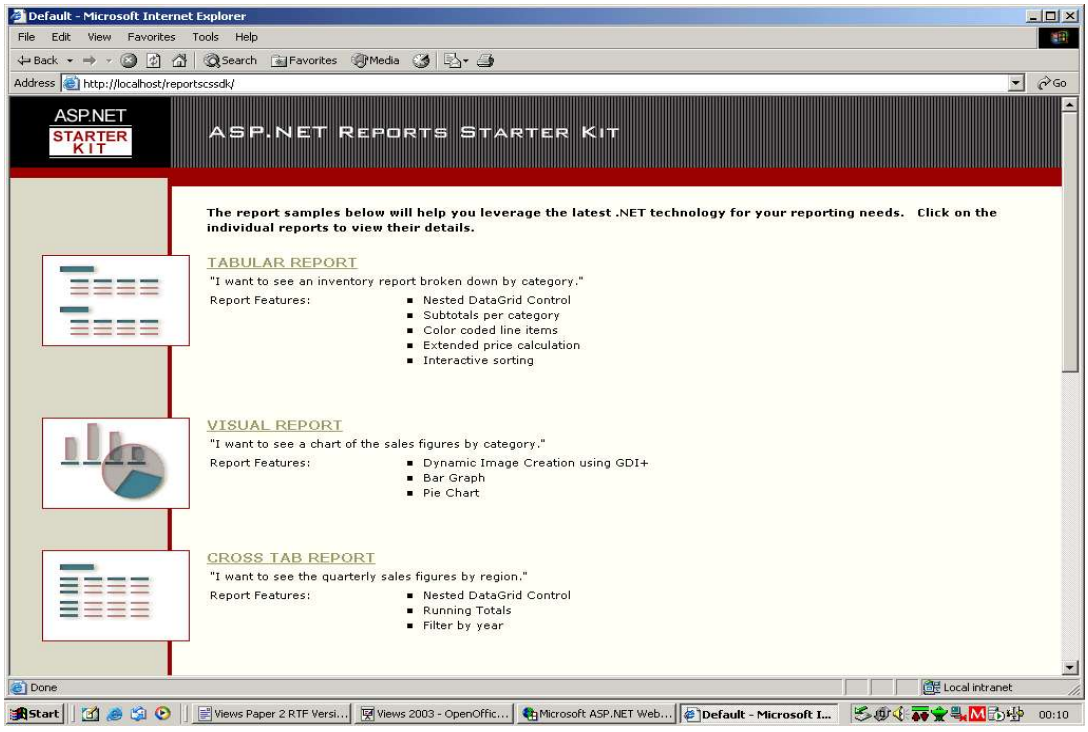

Figure 1: ASP.Net Reports Starter Kit.

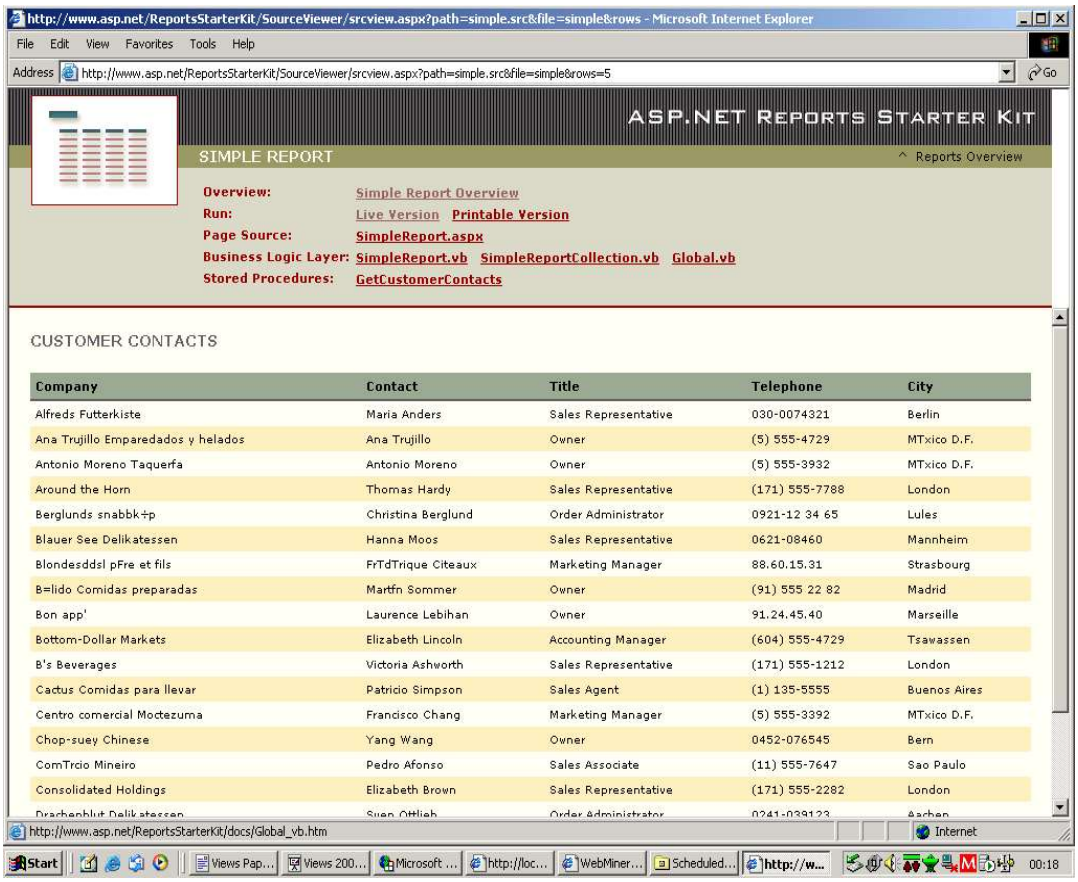

Figure 2: ASP.Net Starter Kit Report Sample: Simple Report.

WALKTHROUGH ASP.NET STARTER KIT – REPORTS SAMPLE.

Microsoft have made a number of 'Quick Start' sample applications available on the www.asp.net website. Each sample is available in a C# or Visual Basic.Net flavours and comes with enough documentation for a developer familiar in these types of languages to get started. The 'Starter Kits' are also well supported by the assorted developers and gurus who frequent the ASP.Net Forums.

The Reports Starter Kit (see Figure 1, above) was chosen to illustrate how SAS can integrate with .NET, as reporting data from disparate data sources is where SAS is traditionally used most widely. This also allows us to compare and contrast the performance trade-offs between .NET client data manipulation and using the SAS data manipulation tools and passing the results back to .NET. As a starting point I'm going to use the 'Simple Report' (see Figure 2) option from the ASP.Net Starter Kit shown above. In SAS terminology we would know this as a listing report or 'PROC PRINT'. The first draft of this paper was written using the first SAS supplied ASP.Net sample. The most recent SAS sample demonstrating ASP.Net and SAS integration illustrates streaming HTML from a SAS procedure using ODS (Output Delivery System) as well as SAS data library access.

The client web application page shown was initially developed to call SQL Server stored procedures; this architecture is useful for us insomuch as SAS provide a 'RunStoredProcedure' method as a component of their sample 'ASP.Net Web Service'. I'll modify this method to be called by the Simple Report application and illustrate some of the object-oriented programming techniques supported by .Net at the same time.

BUILDING AN ASP.NET APPLICATION.

ASP.Net applications can consist of a number of different possible components: ASP.Net Web Forms (files with . aspx extension), Server Controls (.ascx) and class files written in a language that supports ASP.Net. I've opted to

illustrate this paper with examples written in C# (pronounced C – sharp).

The report shown above is displayed using the ASP.Net Web Form defined in the file 'SimpleReport.aspx'. While I don't intend to go through this line by line there are couple of key statements and architecture issues that it is important to be aware of.

```
<%@ Page language="c#" %>
<%@ Import Namespace="System" %>
<%@ Import Namespace="System.Web.UI.WebControls" %>
<%@ Import Namespace="ASPNET.StarterKit.Reports.Components" %>
<script runat="server">
.....
```

The 'Namespace' defined by ASPNET.StarterKit.Reports.Components is essentially a number of classes defined to separate the database layer from the presentation layer. These classes are written in C# and compiled and stored in an assembly (.dll) in the /bin directory of the Reports web application. (A namespace is a logical naming convention for classes and custom variable types defined by the developer.) The SimpleReports.aspx web form uses the 'inline' code model to implement the web application, that is the page itself is split into two logical elements: the first element containing the code controlling the page and second element, the HTML and scripts calling the ASP.Net code and controls defined in the first part. ASP.Net compiles this the first time it is called from the web server and stores a compiled version of the page. With Visual Studio.Net, the default is for a page to be built using the alternative method, the 'Code Behind' model where the source controlling the page is stored in a physically separate file to the page itself. The examples in this paper use the 'inline code' approach.

THE DATAGRID WEB CONTROL.

ASP.Net comes with a number of useful Web Controls, items that display on a web form, one of which is the DataGrid control. This tool is commonly used to display tabular data in an ASP.Net web page. The SimpleReport display uses a DataGrid to display the results of a SQL Server query. The stub below shows the source code:

Inline Code Component

.....

```
//**************************************
// The BindGrid method retrieves a collection of
//simple report  items and databinds it to the
//CustomerGrid
//**************************************

 private void BindGrid()
{
SimpleReportCollection customerList = SimpleReport.GetCustomerContacts();

SortGridData(customerList, SortField, SortAscending);

CustomerGrid.DataSource = customerList; CustomerGrid.DataBind();
}
```

Web Form Component

```
<asp:datagrid id="CustomerGrid"  .... runat="server" AutoGenerateColumns="False" >
.....
</asp:datagrid>
```

The **BindGrid**() method instantiates a SimpleReportCollection class by running the **GetCustomerContacts**() method and then binds the data returned to the CustomerGrid instance of the DataGrid class. If you are interested in how this works take a look at the SimpleReport and Simple ReportCollection class definitions online at: http://www.asp.net/ReportsStarterKit/SourceViewer/srcview.aspx?path=simple.src&file=simple&rows=5.

BRINGING THE POWER OF SAS INTO THE EQUATION

There are a couple of ways in which the facilities provided by SAS Integration Technologies or more specifically the Integrated Object Model or IOM, could provide access to SAS System facilities. The method I've chosen to illustrate here makes use of the SAS Institute supplied 'ASP.Net Web Service sample'. I'll come back to the Web Services idea later in this paper, but at this stage I want to concentrate on the facilities the SAS web service provides from within an ASP.Net application.

The SAS Institute supplied sample provides two public web methods which could be consumed as web services. The **GetDatafromSAS** method is a test which runs a static piece of SAS code to check that the SAS Server is available. **RunStoredProcedure** is more useful as it allows us to run any native SAS code.

IMPLEMENTATION OF THE SIMPLE REPORT DRIVEN BY A SAS DATABASE.

To implement the SAS web service we will modify the code in the BindGrid() method of the Simple Report example shown above, as follows:

Inline Code Stub:

```
private void BindGrid()

{

//Create an ADO.NET DataSet to pass into RunStoredProcedure()

System.Data.DataSet sourceDataSet = new System.Data.DataSet("sasdata");
sourceDataSet.ReadXml(Server.MapPath("dummy.xml"));

// DataSet to contain the results of RunStoredProcedure()
 System.Data.DataSet resultDataSet;
// Create a new instance of our Web Service, Service1
 SASService svc1 = new SASService();

// Call SASStoredProcedure web method passing in the dataset
resultDataSet = svc1.RunStoredProcedure("GetCustomers", sourceDataSet);

//Bind the resultDataSet to the Web Form Control
CustomerGrid.DataSource = resultDataSet;
CustomerGrid.DataBind();
}
```

Web Form Component Stub.

```
<asp:datagrid id="CustomerGrid"  .... runat="server" AutoGenerateColumns="False" >
.....
</asp:datagrid>
```

The **RunStoredProcedure()** SAS web service takes two parameters; a text string naming the SAS source to call, and an input XML dataset. In my example, I've created a dummy input DataSet as our application doesn't require an input dataset to run the SAS Stored Procedure. The SAS stored procedure **GetCustomers**, shown below, is a simple PROC SQL; reading from the SAS Dataset, SASHELP.COMPANY and writing to an ADO.NET DataSet, as follows:

```
%LET outData = work.out;
%LET inData = _null_;

*ProcessBody;

proc sql;

create table &outData as
select level5 as Contact, level3 as Department, level2 as City, job1 as Job
from sashelp.company as company
order by department;

quit;
```

The **RunStoredProcedure()** web service method sets up a connection to a SAS server, and defines a directory as a Repository for the source code of the stored procedure. Also, the SAS web service source definition has been written in Visual Basic which nicely illustrates the multi-language support of ASP.Net.

```
<WebMethod()>
```

```
Public Function RunStoredProcedure(ByVal storedProcedureName As String,
ByVal inputDataset As DataSet) As DataSet

Dim obSAS As SAS.Workspace = GetSAS()

' Send the data to SAS; create the WEBSVC libname

SendData(obSAS, inputDataset)

' Run the requested stored process
obSASEvents = obSAS.LanguageService
obSASEvents.Async = True
obSASEvents.StoredProcessService.Repository = "file:c:\SASRepository"

'Parameters to a StoredProcess are name=value 'space-separated pairs

Dim params As String
' "inData=WEBSVC.SASDATA outData=WORK.OUT" params = "inData=WEBSVC." &
inputDataset.Tables(0).TableName
        params = params & " outData=WORK.OUT"
        obSASEvents.StoredProcessService.Execute(storedProcedureName, params)

' This method copies the named SAS data set into a .NET DataSet

RunStoredProcedure = GetData(obSAS, "WORK.OUT")

DoneWithSAS(obSAS)
End Function
```

IMPLEMENTING THE SAS/IT ASP.NET APPLICATION.

Once the code is written there is one final piece required to implement our ASP.Net web application. The SAS Integration Technologies API's were written to the Microsoft COM standard, whereas the .NET Framework has superceded this. Therefore, a couple of utilities need to be run to make the COM API's accessible from .NET. The command is called **'tlbimp'** and the command line required to implement our example is shown here.

```
tlbimp /sysarray "c:\Program Files\SAS Institute\Shared Files\Integration
Technologies\SAS.tlb"

tlbimp /sysarray "c:\Program Files\SAS Institute\Shared Files\Integration
Technologies\SASWman.dll"
```

These command lines create two COM interop assemblies, essentially a bridge between the COM definition and ASP.Net, SAS.dll and SASWorkspaceManager.dll. Last of all, to make the Web Service available to the ASP.Net client application you need to run the WSDL command (Web Services Description Language) to create a Web Service proxy file and compile the proxy. I've chosen to generate the proxy in Visual Basic but I've included the syntax to create a C# proxy in **bold**.

```
wsdl /out:webServiceProxy.vb (.cs) /language:VB (CS)
http://localhost/WebService/sasws.asmx?WSDL

vbc (csc) /out:webServiceProxy.dll /t:library  /
r:System.XML.dll,System.Web.Services.dll,System.Data.dll,System.dll
webServiceProxy.vb (cs)
```

Figure 3 shows the output from the SimpleReport.aspx driven by the data in the SAS dataset SASHELP.CUSTOMER.

A POSSIBLE ARCHITECTURE FOR FUTURE SAS SYSTEM DEVELOPMENT.

This paper has only really scratched the surface of how the SAS System could be used in conjunction with ASP.Net to provide browser-based applications using SAS server technology. The purpose of this paper was to show how quickly and easily it can be done rather than exploring the myriad user interface controls which are built into the . NET framework. Within the SAS application space, developers can now choose from the following development tools: SAS/Intrnet, SAS/AF, AppDev Studio, or raw Java and ASP.Net using Integration Technologies. Basic local

facilities of Integration Technologies are bundled with Base SAS including read/write data access in version 8.2 and read-only data access with SAS 9. This means that a dedicated Windows 2003 server with IIS 6.0 web server and the SAS System installed, offers a cost effective and powerful SAS application development platform. The potential uses of this configuration are pretty far-reaching for existing SAS sites wanting to make legacy SAS applications and datastores available over the web.

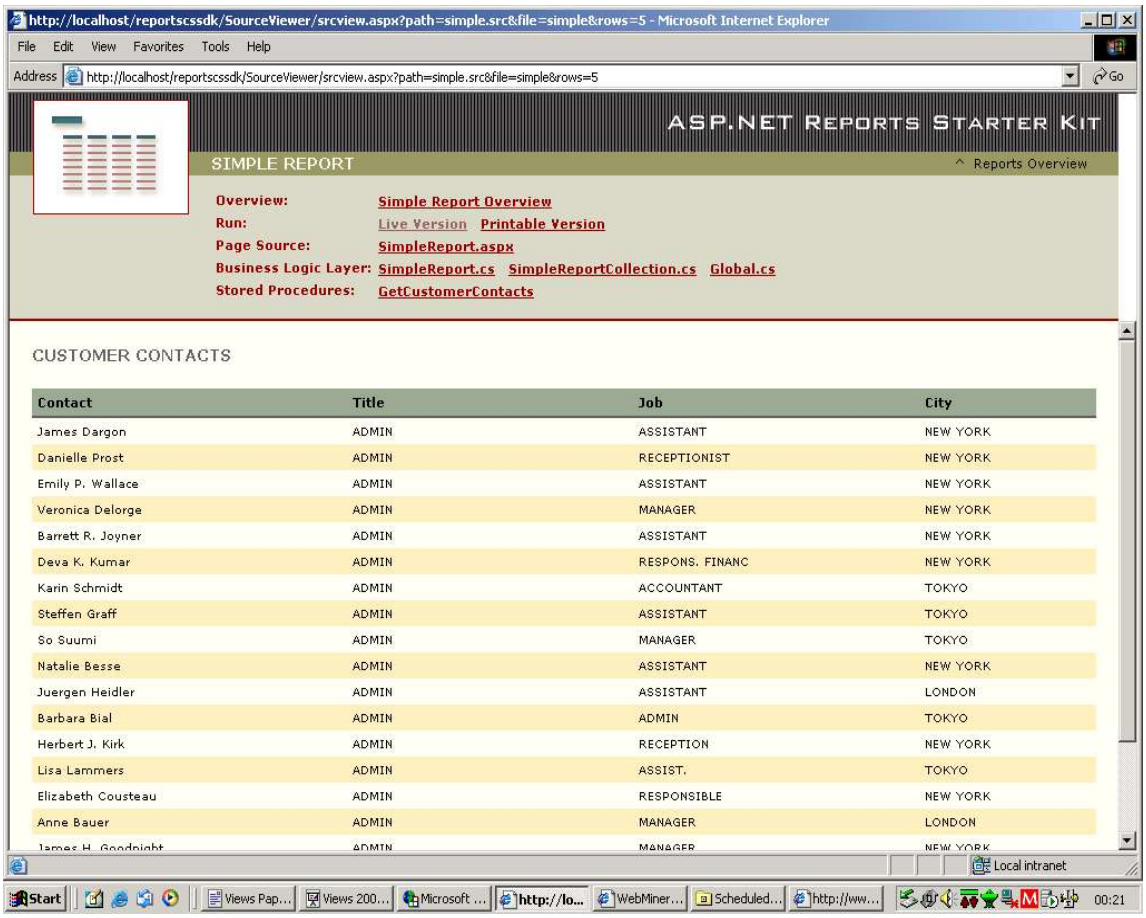

Figure 3: SimpleReport.aspx with SAS System Data source.

FUTURE DEVELOPMENTS

In SAS 9, specifically with Enterprise Guide release 2.0 and above, 'Web Services' support is being built-in, so code produced in Enterprise Guide will have ready made mechanisms to support the implementation of web services.

Another area to watch for is XMLA or XML for Analysis. Microsoft, Hyperion and SAS Institute have produced a specification of XML for Analysis. For more information visit http://www.XMLA.org.

It is also worth noting that Microsoft's IIS server as well as Apache 2.0 supports ASP.Net.

SUMMARY

The combination of the .NET Framework, XML Web Services and SAS Integration Technologies offers a cost effective and powerful development platform for thin-client, browser-based SAS application development. This paper has set out an example ASP.Net application based on freely available sample code from Microsoft and SAS Institute, as well as making use of the 'free' developer IDE, ASP.Net Web Matrix.

REFERENCES

SAS Institute's ASP.Net Web Service Sample:
http://support.sas.com/rnd/eai/samples/DotNetWebService/index.html

SAS Institute's latest ASP.Net Sample:
http://support.sas.com/rnd/eai/samples/asl.net/index.html

Microsoft's ASP.Net Site: http://www.asp.net.

Microsoft TechNet Article entitled : Microsoft .NET for IT Professionals - www.microsoft.com/technet/treeview/default.asp?url=/technet/itsolutions/net/evaluate/itpronet.asp?frame=true.

SAS Integration Technologies online Documentation : http://support.sas.com/rnd/itech/doc/dist-obj/winclnt/index.html

Jim Metcalf - Director of Foundation Technology Strategy at SAS Institute - online presentation video from the Institute website - SAS 9 - The Revolution continues.: http://www.sas.com/apps/cm/index.jsp?code=s9v

Andrew Ratcliffe's NoteColon Newsletter article 'Integration Technologies is free' from Note: Issue 7. (http://www.NoteColon.info.)

Builder.com Article: ASP.Net Web Matrix: Keeping it Simple - http://builder.com.com/article.jhtml?id=u00220020731sem01.htm

Inside ASP.Net Web Matrix: Alex Homer and Dave Sussman -http:// www.asp.net/Tools/redir.aspx?path=webmatrixbook

XML for Analysis : http://www.XMLA.org.

A Viable IIS Alternative ? Apache 2.0 on Windows 2000:http://www.serverwatch.com/tutorials/article.php/1474251

ACKNOWLEDGMENTS

Thanks to Dan Jahn @ SAS Institute for inspiration and whetting my appetite by pointing me to his ASP.Net Sample (link above).

CONTACT INFORMATION

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

Graham G. Murray
Independent Consultant
8 Christie Street
Dunfermline, Fife.
Scotland.
Work Phone: +44 (0)781 806 4072
Email: Graham.Murray@WebMiner.co.uk
Web: http://www.WebMiner.co.uk