Using ASP.NET to leverage data and processes in SAS Servers

Anton Fuchs
EMEA Business Intelligence Technology Strategist
Objectives

- The presentation should give an overview on
  - How SAS supports the .Net framework today
  - How SAS supports the .Net framework in future
- Examples based on ASP.Net
Agenda

- The importance of integration
- .Net framework and SAS – The Present
- Conclusion
The importance of integration

- Information and data overload

- Information coming from various systems and architectures.
  - Heterogenous environments the rule rather than exception.
  - Multiple application platforms the rule rather than exception

- It becomes more and more complex to integrate those.
The integration imperative

Complex Hardware Environments

Explosion of Data

Disparate Applications

More users

IT asked to do more with less

More users

IT asked to do more with less
The importance of Integration

- One aspect of integration is the ability to **interoperate** between different software systems and architectures.
  - Microsoft infrastructure
  - Java infrastructure
  - Mainframe, Windows, Unix, Linux, OpenVMS, etc…
  - 64-bit vs. 32-bit
  - Etc….

- SAS delivers a highly interoperable, hardware-agile and future-proof platform for IT…that integrates!
The importance of Integration

Cornerstones to SAS Intelligence Architecture

- Usability
- Scalability
- Manageability
- Interoperability
The importance of Integration

- COM/DCOM support and COM bridge
- SAS drivers for ODBC and OLE DB
- SAS driver for OLE DB for OLAP
- XML interchange engine, application, APIs
- ADO/OLE DB to SAS for Windows clients
- SAS BI Web services support
- LDAP support (Microsoft Active Directory)
- Message-oriented Middleware (MOM) adapters
  - Microsoft Message Queue (MSMQ)
- SAS analytics deployable in C++
- SAS/ACCESS to PC File formats (DBF, DIF, WKx, XLS)
- Access to SQL Server Analysis Services from Enterprise Guide 2.0
- Integrate into .Net framework (ADO.Net, ASP.Net, VB.Net...)
- more, more, more...
The importance of Integration

Our goal as a company: to enhance the value of our customer’s investment in SAS, no matter what the technology exploitation tools are.
Interoperability with the .Net framework today

- The following slides will give you an overview on some of the various possibilities to integrate with the Microsoft .Net framework

- Integration can be on
  - Data level
  - Process level
SAS and the .NET framework

- SAS already supports the .NET framework.
  - Data access via ADO.NET
  - SAS COM objects can be references via COM interop
  - Support for .NET Web services
  - Access to .Net Dataset via:
    - SAS XML Engine
    - SAS OLEDB data provider

- Further support will come with future versions (see slides ahead for more information)
Data access via ADO.Net

- There are 3 data providers from SAS that support the OLE DB specifications
  - SAS Local Data Provider
    - Requires only BASE SAS
    - Access to SAS datasets in a COM environment
  - SAS/SHARE Data Provider
    - Requires BASE SAS and SAS/SHARE
    - Access to SAS datasets via a SAS/SHARE server
  - SAS IOM Data Provider
    - Requires BASE SAS and SAS Integration Technologies
    - Access to SAS datasets via a SAS Object Server
Data access via ADO.Net

Overview of the Reading SAS Data in .Net using ADO and ADO.NET Sample

This sample demonstrates how you can use C# and VB.NET with both ActiveX Data Objects (ADO) and ADO.NET to retrieve data from a SAS Integrated Object Model (IOM) Server using the SAS IOM Object Linking and Embedding Databases (OLE DB) Data Provider. Each server file included in this sample can be compiled into an executable program. Each program takes one command line argument: a data set name in the form \{Databases\}\{username\}. The programs then try to connect to a local instance of the SAS IOM Server and read the given data set with the SAS IOM OLE DB Data Provider.

Specifically, this sample demonstrates how to do the following in both C# and VB.NET:

- Use the OleDbConnection ADO.NET class to connect to a SAS IOM Server.
- Use the OleDbCommand ADO.NET class to specify a data set.
- Use the DataTable ADO.NET class to read schema information describing a data set.
- Use the OleDbDataAdapter ADO.NET object to read rows from a data set.
- Use the Connection ADO class to connect to a SAS IOM Server.
- Use the Recordset ADO class to read a data set.

Note: Update access is not supported with the SAS OLE DB Providers or with ADO.NET. If you want update access, you must use ADO.

For more information about the SAS OLE DB Providers, see the ADO/OLE DB Cookbook.

http://support.sas.com/rnd/eai/samples/ADOnet/index.html
Example 1: ASP.Net and ADO.Net

Display Dataset

Library name: 
Dataset name: 

Get dataset

<form id="Form1" method="post" runat="server">
  <h3>Display Dataset</h3>
  <asp:TextBox id="libName" runat="server" />
  <asp:RequiredFieldValidator RunAt="server"
    ControlToValidate="libName"
    ErrorMessage=" *** Required Field ****"
    Display="static" />

  <asp:TextBox id="dataSet" runat="server" />
  <asp:RequiredFieldValidator RunAt="server"
    ControlToValidate="dataSet"
    ErrorMessage=" *** Required Field ****"
    Display="static" />

  <asp:button id="Button1" runat="server"
    onclick="Button1_Click"
    Text="Get dataset" />
</form>
Example 1: ASP.Net and ADO.Net

```csharp
<%@ Page language="c#" %>
<%@ import namespace="System.Data" %>
<%@ import namespace="System.Data.OleDb" %>
<script runat="server" language="c#">
private void Button1_Click(object sender, System.EventArgs e) {
    // See next slide for code
}
</script>

System.Data.OleDb: Namespace that contains all OLE DB relevant classes to access 3rd party DBMS.

Button1_click: The c# method that is called when clicking on the “Get dataset” button. See next slide for code that will be executed.
Example 1: ASP.Net and ADO.Net

```csharp
try {
    OleDbConnection cn = new OleDbConnection();
    cn.ConnectionString = "Provider=sas.ShareProvider.1;
    Data Source=shr1;
    Location=localhost;
    User ID=xxxxxx;Password=xyz";
    cn.Open();
    OleDbCommand cmd = cn.CreateCommand();
    cmd.CommandType = CommandType.TableDirect;
    cmd.CommandText = libName.Text + "." + dataSet.Text;
    OleDbDataReader reader = cmd.ExecuteReader();
    resultDataGrid.DataSource = reader;
    resultDataGrid.DataBind();
    reader.Close();
}
catch (Exception ex)
{
    Label1.Text = "An exception was thrown. Error: " + ex.Message;
}
if ( ( cn != null) && ( cn.State != ConnectionState.Closed ) ) cn.Close();
cn = null;
```

The highlighted code snippet defines an OLE DB connection to a SAS/Share server, retrieves the values from SAS and stores them in the reader object.
Example 1: ASP.Net and ADO.Net

```csharp
try {
    OleDbConnection cn = new OleDbConnection();
    cn.ConnectionString = "Provider=sas.ShareProvider.1;Data Source=shr1;Location=localhost;User ID=xxxxxx;Password=xyz";
    cn.Open();
    OleDbCommand cmd = cn.CreateCommand();
    cmd.CommandType = CommandType.TableDirect;
    cmd.CommandText = libName.Text + "." + dataSet.Text;
    OleDbDataReader reader = cmd.ExecuteReader();
    resultDataGrid.DataSource = reader;
    resultDataGrid.DataBind();
    reader.Close();
}
catch (Exception ex) {
    Label1.Text = "An exception was thrown. Error: ": ex.Message;
}
if ((cn != null) && (cn.State != ConnectionState.Closed)) cn.Close();
cn = null;
```

resultDataGrid is an ASP.Net web control component that displays a data table. By binding the reader object to the table, the table will be filled with the data.
### Example 1: ASP.Net and ADO.Net

#### Display Dataset

Library name: sashelp  
Dataset name: class

<table>
<thead>
<tr>
<th>NAME</th>
<th>SEX</th>
<th>AGE</th>
<th>HEIGHT</th>
<th>WEIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>F</td>
<td>13</td>
<td>56.5</td>
<td>84</td>
</tr>
<tr>
<td>Becka</td>
<td>F</td>
<td>12</td>
<td>60</td>
<td>100</td>
</tr>
<tr>
<td>葛</td>
<td>F</td>
<td>14</td>
<td>64.3</td>
<td>90</td>
</tr>
<tr>
<td>Karen</td>
<td>F</td>
<td>12</td>
<td>56.3</td>
<td>77</td>
</tr>
<tr>
<td>Kathy</td>
<td>F</td>
<td>12</td>
<td>59.8</td>
<td>84.5</td>
</tr>
<tr>
<td>Mary</td>
<td>F</td>
<td>15</td>
<td>66.5</td>
<td>112</td>
</tr>
<tr>
<td>Sandy</td>
<td>F</td>
<td>11</td>
<td>51.3</td>
<td>50.5</td>
</tr>
<tr>
<td>Sharon</td>
<td>F</td>
<td>15</td>
<td>62.5</td>
<td>112.5</td>
</tr>
<tr>
<td>Tammy</td>
<td>F</td>
<td>14</td>
<td>62.8</td>
<td>102.5</td>
</tr>
<tr>
<td>Alfred</td>
<td>M</td>
<td>14</td>
<td>69</td>
<td>112.5</td>
</tr>
<tr>
<td>Duke</td>
<td>M</td>
<td>14</td>
<td>63.5</td>
<td>102.5</td>
</tr>
<tr>
<td>Guido</td>
<td>M</td>
<td>15</td>
<td>67</td>
<td>135</td>
</tr>
<tr>
<td>James</td>
<td>M</td>
<td>12</td>
<td>57.3</td>
<td>83</td>
</tr>
<tr>
<td>Jeffrey</td>
<td>M</td>
<td>13</td>
<td>62.5</td>
<td>84</td>
</tr>
</tbody>
</table>
SAS and COM interop

- SAS COM objects can be referenced via COM interop
  - Although .NET assemblies are the best choice for new applications, you may at times need to employ COM objects
  - The ability to easily and naturally call COM components in a .NET framework is called **COM interop**
  - You can create such interoperability objects for the **SAS type libraries** (see next slides)
SAS Integrated Object Model (IOM)

- SAS Workspace
  - DataService
    - Libref
  - FileService
    - Fileref
  - Utilities
    - FormatService
    - ResultPackageService
    - OptionService
    - HostSystem
  - LanguageService
    - StoredProcessService

- ADO/OLE DB or JDBC
SAS and COM interop

- SAS COM objects can be references via COM interop
  - Visual Studio .NET or the .Net SDK contain a utility, tlbimp, that lets you import information from a type library into a .NET application.

```bash
 C:\Program Files\SAS Institute\Shared Files\Integration Technologies\SAS.tlb
```

```bash
 C:\Program Files\SAS Institute\Shared Files\Integration Technologies\SASWman.dll
```

Run the following two commands to create the COM interop assemblies. You will need to modify the path to your shared files folder if you did not install SAS in the default location.
Example 2: SAS and COM interop

```vbnet
<%@ Page language="VB" Debug="true" %>
<HTML>
<BODY>
<%@ import namespace="SASWorkspaceManager" %>
<%@ import namespace="SAS" %>
<%
    Dim wsStoredProcessService, wsFref, wsBinaryStream, wsStreamHelper, VisibilityProcess
    VisibilityProcess = 1
    Dim wsm as WorkspaceManager = New WorkspaceManager()
    Dim ws as Workspace = wsm.Workspaces.CreateWorkspaceByServer("server", VisibilityProcess, nothing, ",", ",", ",")
    wsstoredProcessService.Repository = "file:C:\seugi\2003\samples"
    wsStoredProcessService.Execute ("aspsample.sas", "sex=M")
    wsFref = ws.FileService.UseFileref("b")
    wsBinaryStream = wsFref.OpenBinaryStream(0)
    wsStreamHelper = CreateObject("SASScripto.StreamHelper")
    response.binarywrite (wsStreamHelper.ReadBinaryArray(wsBinaryStream, 0))
    wsBinaryStream.Close
    wsm.Workspaces.RemoveWorkspace (ws)
    ws.Close
%>
</BODY>
</HTML>
```
## Example 2: SAS and COM interop

### The SAS System

<table>
<thead>
<tr>
<th>Obs</th>
<th>NAME</th>
<th>SEX</th>
<th>AGE</th>
<th>HEIGHT</th>
<th>WEIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Alfred</td>
<td>M</td>
<td>14</td>
<td>69.0</td>
<td>112.5</td>
</tr>
<tr>
<td>11</td>
<td>Duke</td>
<td>M</td>
<td>14</td>
<td>63.5</td>
<td>102.5</td>
</tr>
<tr>
<td>12</td>
<td>Guido</td>
<td>M</td>
<td>15</td>
<td>67.0</td>
<td>133.0</td>
</tr>
<tr>
<td>13</td>
<td>James</td>
<td>M</td>
<td>12</td>
<td>57.3</td>
<td>83.0</td>
</tr>
<tr>
<td>14</td>
<td>Jeffrey</td>
<td>M</td>
<td>13</td>
<td>62.5</td>
<td>84.0</td>
</tr>
<tr>
<td>15</td>
<td>John</td>
<td>M</td>
<td>12</td>
<td>59.0</td>
<td>99.5</td>
</tr>
<tr>
<td>16</td>
<td>Philip</td>
<td>M</td>
<td>16</td>
<td>72.0</td>
<td>150.0</td>
</tr>
<tr>
<td>17</td>
<td>Robert</td>
<td>M</td>
<td>12</td>
<td>64.8</td>
<td>128.0</td>
</tr>
<tr>
<td>18</td>
<td>Thomas</td>
<td>M</td>
<td>11</td>
<td>57.5</td>
<td>85.0</td>
</tr>
<tr>
<td>19</td>
<td>William</td>
<td>M</td>
<td>15</td>
<td>66.5</td>
<td>112.0</td>
</tr>
</tbody>
</table>
SAS and .NET Web services

- Support for .NET Web services
  - SAS co-sponsors the XML for Analysis organization together with Microsoft
  - Contributing member of Web services Interoperability organization (WS-I)
  - White paper on how to implement a SAS Web service
    http://www.sas.com/apps/whitepapers/whitepaper.jsp
SAS and .NET Web services
Sample application

Overview

This sample demonstrates how you can build an XML Web service using the Microsoft .NET Framework and SAS Integration Technologies. The .NET Framework makes it very easy to develop a new Web service and write clients to access the Web service. SAS Integration Technologies opens up the power of SAS analytics so it can be accessed through the Web service. This sample presents the necessary code and methodology to make a sample XML Web service run. It also presents the steps necessary to develop an ASP.NET client that accesses the Web service.

An XML Web service enables you to reuse code in a variety of applications. The functionality of the service is typically exposed to Web clients via the Simple Object Access Protocol (SOAP). Information about the functionality that is provided by the service is described in a Web Service Description Language (WSDL) document. The WSDL document enables new users to quickly start using the XML Web service.

When your Web service supports these open standards for XML Web services, you broaden integration options and enable many different applications and programming languages to call your Web service. Any client that supports making method calls using SOAP can call methods on your Web service. So a Java client, ASP.NET client, desktop Windows application, or even Microsoft Word can call the methods in your Web service.

This sample demonstrates how to:

- write an ASP.NET XML Web service that uses SAS Integration Technologies
- call a SAS stored process through a Web service and get back results

SAS and .NET Web services
Example: Noel-Levitz using SAS 8.2

Data Process Solution
Base SAS
SAS/ACCESS to ODBC

SAS Enterprise Miner models
Integration Technologies
Web services

Admissions
College
Student

Enrollment Management
Recruitment
Retention
Assistance

SQL
Scoring Models
SAS and .NET Web services
Example: Noel-Levitz using SAS 8.2

- Reduced time to deliver information to customers
  - “First in the mailbox”
  - Students more likely to apply

- Reduced cost to implement
  - $200K instead of $600K
  - Paradigm shift of implementation to provide instantaneous answers to customers

- Improved customer satisfaction by allowing unlimited scoring

- Increased sales
SAS 9.1 and .NET Web services

- SAS Integration Technologies 9.1 includes support for creating Web services.
  - SAS BI web services for .Net
  - SAS BI web services for Java

See Presentation The SAS® XML/A Web Services Framework at 16:05.
SAS 9.1 and the SASObjectManager

IIS

ASPX

ObjectManager

Sassps.dll

SAS Stored Process Server

SAS Metadata Server
SAS 9.1 and the .Net framework

Common Infrastructure

- Shared Metadata
- Shared Data
- Business Metadata

- .Net Web services
- ASP.NET etc
- SAS® Web Report Studio
- SAS® Enterprise Guide
- SAS® Add-In for Microsoft Office
- SAS® Information Delivery Portal
Conclusion

- SAS is an open platform supporting all important technologies and industry standards
- Microsoft .Net framework is an important part and we can easily interoperate.
- Support for MS tools and the .Net framework will even be extended in version 9.1
References

- Microsoft asp.net  
  http://www.asp.net

- SAS Integration Technologies  
  http://support.sas.com/rnd/itech

- Enterprise Integration Community  
  http://support.sas.com/rnd/eai

- Anton.Fuchs@eur.sas.com