Paper 3604-2019

Lightweight SAS® Web Services

Alan Churchill, Savian, LLC
Don Henderson, Henderson Consulting Services, LLC

ABSTRACT

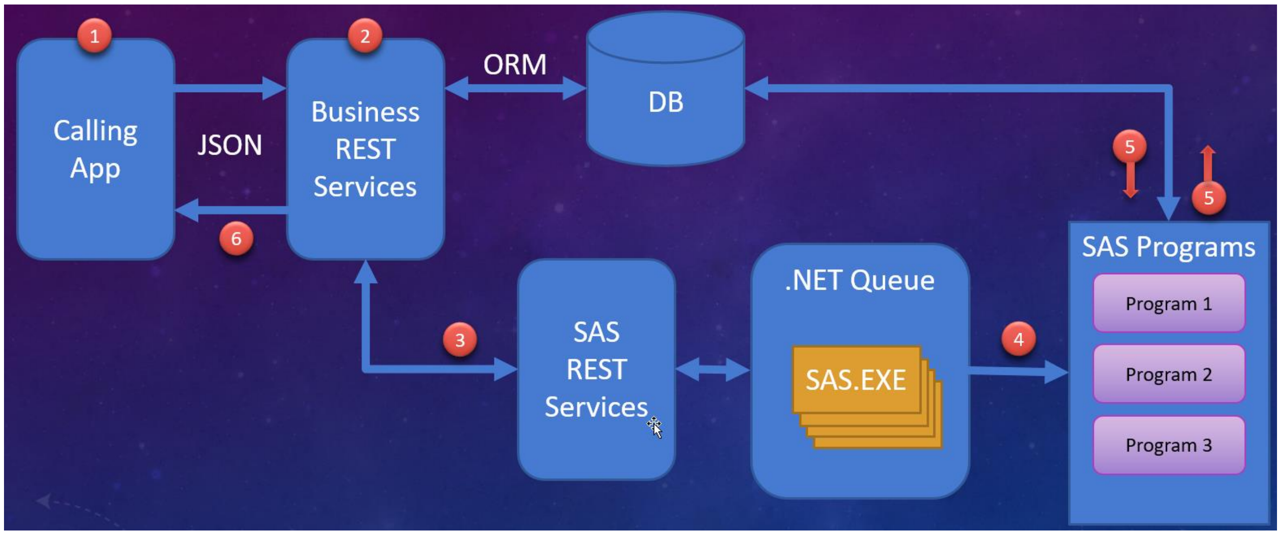
Web services have become very popular over the last decade as enterprises face a need to integrate disparate systems. Additionally, new web server engines and database integration options have arrived that ease that system integration challenge. This paper will illustrate how a lightweight service infrastructure can be put in place that simplifies the process of getting data into and out of SAS® and serving up the output of SAS programs in a very stream-lined fashion. Web services do not need large stacks of software or complex hardware systems to be effective.

INTRODUCTION

Most traditional IT shops doing web services for ETL, modeling, reporting, etc. use a large middleware component to manage the web services and stored process layers. These are Java libraries that integrate with the SAS metadata to determine security, what is available and libraries. Many web services do not need that level of complexity, power, or different components to manage that functionality. For example, simple folder structure security can be handled via groups in a typical Active Directory (AD) world (or LDAP for other variants). Libnames can be managed via macros or direct reference in the SAS code.

Web services do not need to be complex. This paper will illustrate that by using Web API, ORM (Object-Relational Mapping), and a simple directory structure holding SAS code, web services can be delivered with minimal deployment effort on the part of the SAS programmers, and minimal effort on the part of the web services developer.

PROCESSING FLOW



1. A call to a REST (Representational State Transfer)-based web service is made.
2. The business web service backing code binds the JSON (JavaScript Object Notation)) data directly to the database tables
3. The business web service calls the SAS web service
4. The SAS web service gets an instance of SAS from the queue and executes the SAS code
5. The SAS code reads in the data from step 2 tables, does whatever transformations are needed, and then writes it back to the database

6. When sas.exe process ends, the business web service returns the response object back to the calling application.

ASP.NET Core and Web APIs (Business/SAS web services)

Overview

ASP.NET Core MVC is a web framework that provides a powerful, patterns-based way to build dynamic websites and web APIs. ASP.NET Core MVC enables a clean separation of concerns and full control over markup.[1]

Why Use it

Traditionally, ASP.NET has used IIS. IIS is a legacy system that has matured over time but has also lagged. Many things that IIS handles are no longer valid in the modern web era and many things the web does today were just bolted on in IIS. One description of IIS was ‘the developers are so afraid of touching it that they don’t’. ASP.NET Core starts with a very lightweight web server called Kestrel. It is initialized by a Windows service (as an example) and runs in the background passing information into the web services.

ASP.NET Core can still use IIS but the preference is to use Kestrel.

Web APIs

The Web APIs are a mix of controllers and actions that expose web services. Think of the APIs as verbs that dictate what actions can be done with the system. A JSON packet coming into one of the services is an object upon which an action can occur. In return, the APIs return objects (in most cases) that reflect that action. As an example:

Here is an example, in C#, for a very simple API. This is an endpoint as can be seen in the Route below:

```
[HttpGet]
[Route("v1/Admin/GetMachineName")]
public ActionResult<string> GetMachineName()
{
   return Ok(Environment.MachineName);
}
```

GetMachineName returns the machine’s id.

This is another example showing the binding of the incoming payload to the database (see ORM below):

ShoeStore Record ➔ AddToDataBase ➔ return SthoeStoreId on new record ← *Note, the id allows us to link the request with the response*

```
[HttpPost("v1/shoestore/create")]
public IActionResult Create(ShoeStore shoeStore)
{
   db.Add(shoeStore);
   db.SaveChanges();
   return Accepted();
 }
```

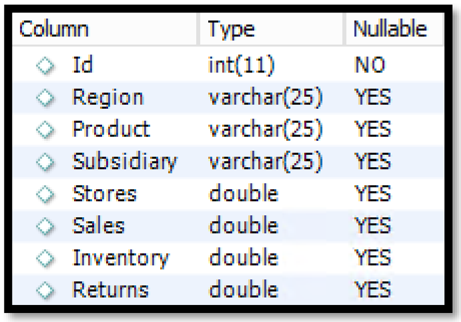
OBJECT RELATIONAL MAPPING (ORM)

OVERVIEW

Object-Relational Mapping (ORM) in computer science is a programming technique for converting data between incompatible type systems using object-oriented programming languages. This creates, in effect, a "virtual object database" that can be used from within the programming language. [2]

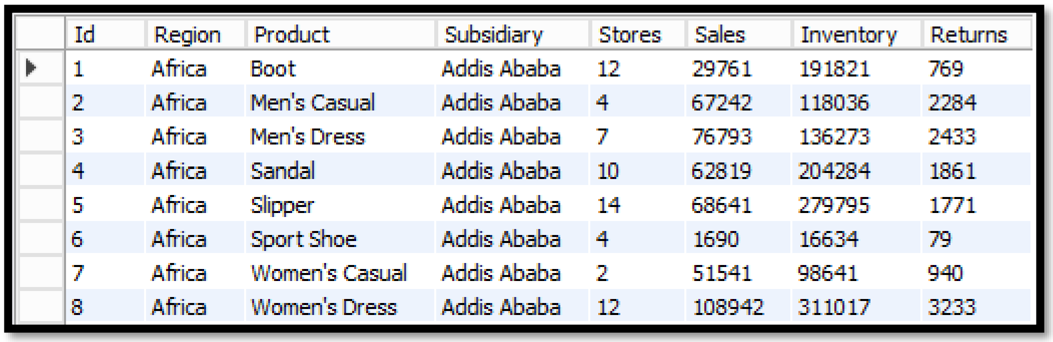
The objects represent the code (C#) which defines the web services. The relational parts are the tables in the database that hold those objects. The mapping between the objects and the relational elements are automatic and represent a contract between the 2 sides. When a new object is defined and saved to the database, the two are matched. Because it is a contract, the 2 sides must always match in type and name. As an example, here is sashelp.shoes expressed as a table in a database (a SQL definition) as well as the corresponding object (the actual data table):

SQL Definition in table shoes



Column	Type	Nullable
Id	int(11)	NO
Region	varchar(25)	YES
Product	varchar(25)	YES
Subsidiary	varchar(25)	YES
Stores	double	YES
Sales	double	YES
Inventory	double	YES
Returns	double	YES

SQL Table



Id	Region	Product	Subsidiary	Stores	Sales	Inventory	Returns
1	Africa	Boot	Addis Ababa	12	29761	191821	769
2	Africa	Men's Casual	Addis Ababa	4	67242	118036	2284
3	Africa	Men's Dress	Addis Ababa	7	76793	136273	2433
4	Africa	Sandal	Addis Ababa	10	62819	204284	1861
5	Africa	Slipper	Addis Ababa	14	68641	279795	1771
6	Africa	Sport Shoe	Addis Ababa	4	1690	16634	79
7	Africa	Women's Casual	Addis Ababa	2	51541	98641	940
8	Africa	Women's Dress	Addis Ababa	12	108942	311017	3233

Classes in C#

1. Define a collection of objects (shoes)

```
public class ShoeCompanyClass
{
    public List<ShoeStore> ShoeStores { get; set; }
}
```

2. Define a ShoeStore object:

```
public class ShoeStore
{
   public int Id { get; set; }
   public string Region { get; set; }
   public string Product { get; set; }
   public string Subsidiary { get; set; }
   public double? Stores { get; set; }
   public double? Sales { get; set; }
   public double? Inventory { get; set; }
   public double? Returns { get; set; }
}
```

Adding to Database

Now that the structures are set up, we can do a databind and work with the data:

```
using (var db = new DbContext())  🡐 Note: The name of the context is arbitrary
{
        var shoeStores = new ShoeCompanyClass();
        shoeStores.Add(new ShoeStore() {Region = “United States”, Product=”Boot”,….};
        shoeStores.Add(new ShoeStore() { Region = “United States”, Product=”Slipper”,….};
        …add more…
        db.ShoeStores.Add(shoeStores);
}
```

Note: An ORM can do full CRUD (Create, Read, Update, Delete) operations and not just adds/creates.

ENTITY FRAMEWORK CORE

The ORM that is being used in our examples is Entity Framework Core which is maintained by Microsoft and is available for free on NuGet.

The SAS Glue

.NET Queue

The SAS services code maintains a queue provided by .NET. This queue is started when the SAS services are initialized, normally when Windows starts since they are typically installed as a Windows service. This queue has a number, say 5, that tells it how many instances of sas.exe to start. As calls come into the services, the queue delivers one of these instances and executes the SAS code. Why is this done? It prevents the startup cost of SAS by Integration Technologies (see beow). This makes the SAS system execute almost instantaneously. Once the SAS code is completed, the instance is cleared of any work-related items and placed back into the queue. The number of instances of SAS running is a variable and is determined by the admin according to anticipated load and simultaneous calls.

Memory is not a limiting factor; the number of instances of SAS that can be running simultaneously on a server is very high (hundreds) for a modern system.

SAS INTEGRATION TECHNOLOGIES

SAS Integration Technologies (SAS/IT) is the mechanism that allows the web services code in .NET to use SAS on the backend. Behind the scenes, SAS/IT relies on a COM object to work with SAS on Windows. For this demo, the server is local so SAS/IT can be called using the local instance of SAS. These dlls support the SAS IOM data provider which is “an OLE DB data provider that supports access to SAS data sets that are managed by SAS Integrated Object Model (IOM) servers.” [3]

```
Interop.SAS.dll
Interop.SASIOMCommon.dll
Interop.SASWorkspaceManager.dll
```

This object, the Workspace Manager, can be called via C#. This example will run the SAS code defined in the first line (sasCode of type string) and then extract the log information:

```
var sasCode = “data test; set sashelp.class”; run;”

SAS.Workspace ws = null;
var lang = ws.LanguageService;                    🡐 Creates a language service to submit SAS code
lang.LineSeparator = "\r\n";                      🡐 Lines in the log will be separated by CRLF
lang.ResetLogLineNumbers();                       🡐 Reset the line numbers back to starting at 1
Common.Log($"Execute SAS code");
lang.Submit(sasCode);                             🡐 Submit the SAS code defined in the string

Array carriage, lineTypes, lines;
var cc = new SAS.LanguageServiceCarriageControl();
var lt = new SAS.LanguageServiceLineType();

lang.FlushLogLines(1000, out carriage, out lineTypes, out lines);    🡐 Get 1000 lines of the log
lang.Reset();                                     🡐 Reset the language service
```

Once that runs, the variable ‘lines’ will contain the log for SAS.

BASE SAS AND SAS ACCESS FOR ODBC

The final part of the solution is the movement of the data to and from the database. For this, the solution links the database to the SAS code using an Access engine. The ODBC access engine is very flexible and can connect to any major database source. Since the systems are web services, the need for specialized Access engines should be minimal but any Access engine should work. The need is to read and write to a database.

In SAS, the data written to the database by the ORM side is read in by SAS. This is the request (records) for services:

Read the data stored by the web service

```
libname MyData ODBC NOPROMPT="server=xxxx;driver=ODBC Driver 13 for SQL Server;
uid=MyUsedId;pwd=MyPwd;Database=SasData;" STRINGDATES=NO
IGNORE_READ_ONLY_COLUMNS=YES SCHEMA=DBO;

data dsShoes;
   set MyData.shoeRequest;
….
run;

…do analysis and transformations…
```

Once processing completes, the response data also needs to be written to the database so that the ORM can pick it up for the response back to the calling application.

Write the data back to the database

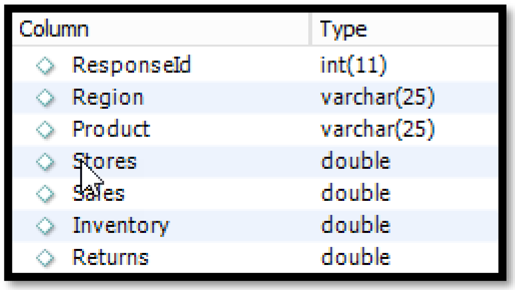
```
%macro AddRecordsToDb(table=,ds=);
   proc append base=MyData.&table. (drop=id) data=&ds. force;
   run;
%mend AddRecordsToDb;

%AddRecordsToDb(table=shoesResponse,ds=dsShoes);
```

Return Objects

Once the sas.exe ends and returns to the queue, the system then knows that the call to SAS services is at an end. The data is read back from the database by the ORM and a response is sent back to the calling application.

The response object is defined as:



Column	Type
ResponseId	int(11)
Region	varchar(25)
Product	varchar(25)
Stores	double
Sales	double
Inventory	double
Returns	double

The columns that are numeric, in our example, would be totals based upon the initial query. So, if the shoesRequest was limited to region and product, the response back would be the sum of the numerics specified here. Notice that the Subsidiary column is missing in the response since the values are aggregated to rge Region and Product level.

The response object will be sent back to the calling app in a JSON format.

CONCLUSION

The world of IT is moving to smaller, more nimble systems. In the past, there were large, monolithic applications that came full-featured even if the application need was light. Today, developers can install only what is needed to power the business application. Additionally, newer technologies such as ORM allow developers to ignore the CRUD operations normally associated with databases and concentrate on the business logic.

An application developer does not need to worry about the SQL commands needed to maintain one or more databases. The ORM handles it and, using SAS/integration Technologies, SAS jobs can be incorporated into the overall flow allowing SAS to interact with the services layer. SAS Access can then be used to handle the database reads/writes.

REFERENCES

[1] https://www.nuget.org/packages/Microsoft.AspNetCore.Mvc/
[2] https://en.wikipedia.org/wiki/Object-relational_mapping
[3]http://support.sas.com/rnd/itech/doc9/dev_guide/dist-obj/winclnt/ext_iom_intro.html

RECOMMENDED READING

- Entity Framework
  https://docs.microsoft.com/en-us/ef/ef6/index
- ORM
  https://en.wikipedia.org/wiki/Object-relational_mapping
- Kestrel
  https://www.tektutorialshub.com/asp-net-core/asp-net-core-kestrel-web-server/
- SAS Integration Technologies and .NET
  https://support.sas.com/rnd/itech/doc9/dev_guide/dist-obj/winclnt/windotnet.html

SUGGESTED SOFTWARE (except for SAS, install using nuget)

- ASP.NET Core
- Entity Framework Core
- SAS/Access to ODBC
- SAS/Integration Technologies
- Swashbuckle ASP.NET Core
- Serilog
  - Files Sink
  - Console Sink

CONTACT INFORMATION

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

Alan Churchill, Savian, LLC
alan.churchill@savian.net

Don Henderson, HCSBI
don.henderson@hcsbi.com