

What to Expect When You Break into Web Development Bringing your career into the 21st Century

Chuck Kincaid, Venturi Technology Partners, Kalamazoo, MI

ABSTRACT

Are you a SAS programmer who has wanted to do more web programming, but weren't sure the best way to go about it? Are you confused by all of the things you hear about web programming? This presentation will review the many ways that SAS can be used to take information to the web from the simplest to the most advanced. The goal is to provide programmers with information to help them make career decisions.

INTRODUCTION—WHY WEB APPLICATIONS?

One may be wondering why do people want to build web applications? The first question might really be why build applications at all? We know that we use applications all the time. SAS is an application. This paper was written with multiple applications. However, why would I want to make an application out of my code? It works just fine as it is.

Some of the reasons for building applications are

—*Knowledge Management*: This used to be talked about a lot in the mid-90's, but not so much any more. However, the idea was right even if it's not in vogue. Intellectual Capital is still very important. A corporation might very well want you to convert the work you do into an application. If the steps behind the work you do are "written down" only in your head, then if you were to leave the company that information would be lost. So often it happens that companies lose a lot of money because an employee leaves or, ironically, is fired and they are the only ones that know how to do something. By building an application you are converting Human Intellectual Capital to Corporate Intellectual Capital. Now your company is worth more.

—*Business Process Management*: Maybe the work you do spans multiple computers and/or multiple applications. Maybe it requires cutting and pasting from one to another or building spreadsheets or presentations by hand. We built an application for one company that reduced their time spent creating forecasts from five days to thirty minutes. Now they have the time to delve into alternative business scenarios or refine the forecasts. By automating the manual business process they can perform tasks that provide better value to the company.

—*Business Intelligence*: We hear a lot about business intelligence nowadays. Many vendors want you to believe that to do business intelligence right you have to use their big, fancy system. That's not entirely true. There's a lot you can do to provide your decision makers with the right information using just the simple methods we'll see here today.

—*Analytical Intelligence*: SAS defines Business Intelligence as looking into the past and Analytical Intelligence as looking into the future. I like those definitions and, so, if providing Business Intelligence to your decision makers is a good reason to build an application, then providing Analytical Intelligence is an even better reason. Some people think that predicting, modeling, forecasting, or scoring (whatever you want to call it!) is too complex to convert into an application. I disagree. We have given companies valuable tools by building forecasting models by hand and then incorporating the results into an application that allows them to do "What If?" scenarios. The models can be updated as the data is updated. We have even built automatic model building applications for other clients.

WHY WEB APPLICATIONS

There are advantages and disadvantages to building web applications and they're not for every situation. Some of the advantages are

- *Thin Client*: Easy distribution, Easy maintenance / Upgrade
- *Virtual*: Wide distribution to users in many locations
- *Integration*: Make Legacy Apps available
- *Cool*: Uh, I mean,... "Leverage new technology to strategically impact the bottom line on a competitively strong go forward basis"

Some of the disadvantages are

- *Multiple Target Platforms*: Browsers, OS's
- *Multiple Technologies*: Just grew exponentially!
- *Security*: Multiple places for, but can take more work
- *Speed*: Not always under your control
- *Perceptions*: Everyone wants "web-time"

When would you want a web application vs. a client-server or standalone application? Well, when the advantages significantly outweigh the disadvantages.

TYPES OF APPLICATIONS

There are a variety of types of web applications from the simplest to amazingly complex. For our discussion we will look at the two simplest and break them out as:

1. Web Publishing – Static HTML: ODS
2. Web Publishing – Static HTML: From Scratch
3. Dynamic Content Creation I – CGI
4. Dynamic Content Creation II – xSP, Servlets

PROGRAMMING STYLES

One of the characteristics that we'll describe about these types is the programming style. There are basically two styles. One is the CGI style where *something* is used to create the HTML. That *something* could be SAS, perl, PHP, C, Java or almost any language. The other style is the xSP style, as I call it where x is typically either "J" giving JSP (Java Server Pages) or "A" and ASP (Active Server Pages). The web pages are developed primarily in HTML with calls to Java or VBscript to implement dynamic content.

PROGRAMMING SKILLS

The skills needed to develop programs using the methods discussed range from SAS only to only non-SAS. Like programming in general advancing past a certain level (SAS or non-SAS) requires a paradigm shift to object oriented programming. Because of this SCL programmers (which can be object-oriented) may find it easier to get into Java.

Another point to be aware of is that depending on size of team one person may need multiple skills. The work can be separated, especially when using JSP or ASP, with one person building the web page, another building the business logic and another working with the data. But, of course, not all projects are that big. So one person may need to know all three plus how to work with a web server or java web server.

WEB PUBLISHING

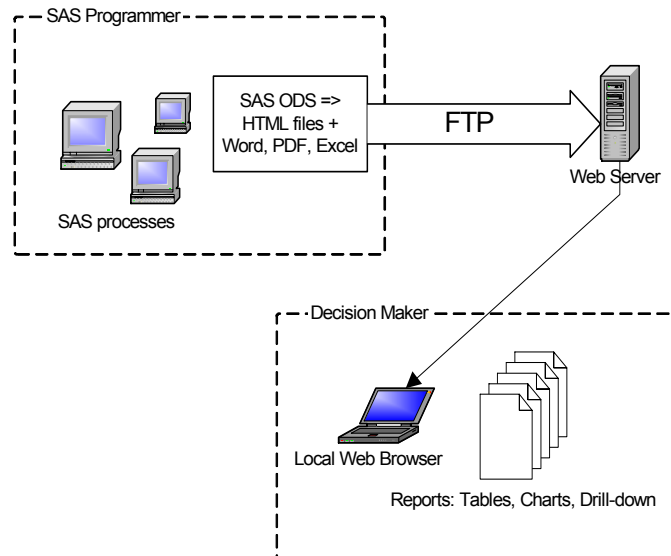


Figure 1: Web Publishing Blueprint

The first type is web publishing. This is the original style of web programming. Static HTML pages are created and placed on a web server. The user browses these pages to get their information. The diagram in Figure 1 shows the relationship among the primary players. If you were building this type of application you would use SAS tools to create HTML pages and then use SAS again to FTP them to the web server. From there the web server would take over.

The skills needed for entry into this level are Base SAS, ODS and whatever is needed to create the content (e.g. Report, Tabulate, SAS/Graph).

An example of code that would generate a web page is

```
options nodate pageno=1 linesize=64 pagesize=60;

ods html body='your-html-file.html';

title 'Leading Grain Producers';
title2 'for 1996';

proc print data=grain noobs label;
var country type kilotons;
run;

ods html close;
```

Very simple. And you can get pretty fancy just with SAS and ODS without using any direct HTML. A good link is <http://support.sas.com/rnd/base/early-access/odsdoc2/sashtml/tw5195/z0471334.htm>

and

<http://support.sas.com/rnd/base/topics/odshtml/index.htm>

for using ODS with Cascading Style Sheets.

If you wanted more control or fancier web pages, then you have to get into the HTML and/or Javascript. An example of using HTML is

```
data _NULL_;
set grain end=last;
file _webout;
if (_N_ = 1) then do;
put "<html><head><title>Leading Grain Producers</title></head><body><table>";
put "<tr><td>Country</td> <td>Type</td> <td>Kilotons</td> </tr>";
end;

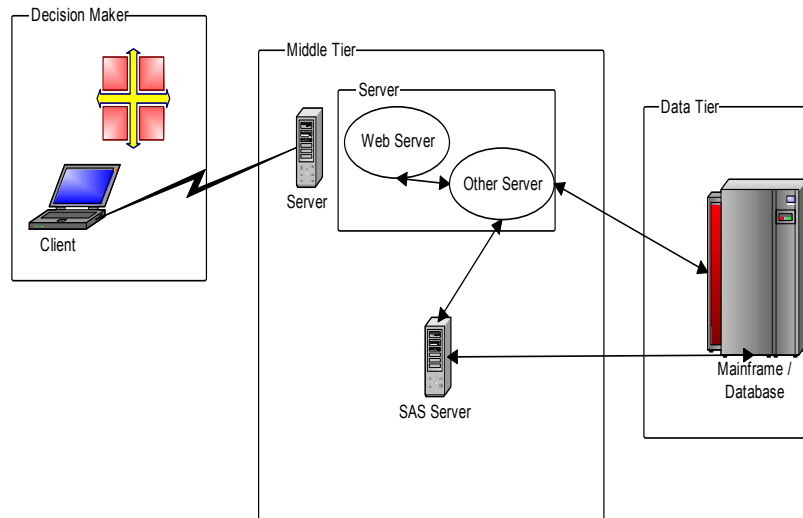
put "<tr><td>" country "</td><td>" type "</td><td>" kilotons "</td></tr>";

if (last) then do;
put "</table></body></html>";
end;
run;
```

As you can see this method intermixes SAS code and HTML code. This example is pretty simple and ODS would be easier, but it works as an example. It gets even harrier when you use quotes as part of the HTML. If you 'macro-tize' your code at this stage, then the code doesn't get too complicated, because you typically won't be using ampersands in the HTML. In a later example we will, though, and then you have to do even more work.

You can add some fancier web functionality with Javascript if you like and there are many web-sites that not only teach you Javascript, but give you working code as examples. You could also use SCL to generate the web pages and it would be a little bit cleaner.

DYNAMIC CONTENT CREATION



Now we get into actually creating content based on input from a user. There is a step change in the skills required to achieve this. In order to build the application the programmer would start with the same knowledge as before, i.e. Base SAS, HTML, Macro, ODS, Javascript, and possibly SCL. Now one would also need to know concepts and syntax regarding HTML and HTTP, which is the protocol used between the web browser and the web server, such as Name/Value pairs, Program Chaining, and Post/Get. Some good references for SAS/Intrnet can be found in the Web Technologies Community

<http://support.sas.com/rnd/web/intrnet/index.html>

The code would look similar to code used in Web Publishing above, though one would probably be building more pages “from scratch” since buttons, text entry, drop down boxes, list boxes, etc. are not created by SAS procedures. An partial example with a login in screen would be

```

data _null_;
  file _webout;
  <pieces cut out here>
  put "<FORM NAME=login>";
  put "<TABLE BORDER=4 CELLPADDING=8 CELLSPACING=2 BORDERCOLOR=#39427B";
  put "BORDERCOLORDARK=#003366 BORDERCOLORLIGHT=#8494C6>";
  put "  <TR ALIGN=center>";
  put "    <TD VALIGN=middle>";
  put "      <B><I>";
  put "        LOGIN";
  put "      </I></B>";
  put "    </TD>";
  put "    <TD>";
  put "      <INPUT NAME=ID>";
  put "    </TD>";
  put "  </TR>";
  put "  <TR ALIGN=center>";
  put "    <TD>";
  put "      <B><I>";
  put "        PASSWORD";
  put "      </I></B>";
  put "    </TD>";
  put "    <TD>";
  put "      <INPUT NAME=pass TYPE=password>";
  put "    </TD>";
  put "  </TR>";
  put "  <TR ALIGN=center>";
  put "    <TD>";
  put "      <INPUT TYPE=button VALUE=Proceed >";

```

```

put "      </TD>";
put "      <TD>";
put "      <INPUT TYPE=reset VALUE=clear >";
put "      </TD>";
put "    </TR>";
put "</TABLE>";
put "</FORM>";

```

Likely the programmer will have to be administering the web server, as well, -- at least in development. In order to do that you'll need to know about HTML error codes, Virtual Directories, Configuring the Broker, the syntax for PROC APPSRV, etc. The HTML information can be found in multiple places. Virtual Directories would be found in the web server documentation. The web server could be IIS or Apache. The administration for each is a little different.

In all of these examples, so far, the communication and work is done as follows (see Figure 2). The web browser makes a request of the web server. The web server passes the request on to the broker (the "Other Server") and the broker passes it on to SAS. SAS receives the request as a call to a SAS program with macro parameters as inputs. SAS runs the program, creates HTML and pipes it back to the broker and on to the web server and the web browser. The work is done by SAS and the broker is a CGI program that just passes the information back and forth. In the next set up, the web server will do more of the work.

If you use Java Server Pages on the front end, then you'll learn about configuring a Java Server such as Tomcat. Of course, you'll also be learning about Java, Custom Tags, Struts and other aspects of JSP. JSP files works inside out from what we've seen. Above SAS was on the outside and HTML was on the inside; now, the HTML is on the outside and the JSP which calls SAS is on the inside.

In Figure 2 the "Other Server" will be a Java Server and it will be the conduit between the web server and SAS. The following partial code (borrowed from http://support.sas.com/rnd/appdev/examples/ServletJSP/ListBoxViewwithCustomRenderer_BLD.htm) creates a listbox based on a dataset. The code sas:Form is a JSP custom tag provided by SAS. Notice how it's in the middle of the HTML code.

```

<%@ taglib uri="http://www.sas.com/taglib/sas" prefix="sas" %>
<%@ page pageEncoding="UTF-8"%>
<html>
  <head>
    <link href="styles/sasComponents.css" rel="stylesheet" type="text/css">
    <script language="JavaScript" src="myJSFile.js"></script>
  </head>
  <body>
    <sas:Form id="form1" name="form1" method="get" action="pushButtonSubmit.jsp" >
      <sas:ListBoxView id="lb1" model="sas_model1"/>
      <sas:PushButton id="pushButton1" text="PushButton"
onClick="lb1.submitSelection(form1);"/>
    </sas:Form>
  </body>
</html>

```

Again, the web browser will process all of the HTML code and then send the custom tags (or java code if we had it) off to the Java Server. The Java Server would then call SAS code if necessary. AppDev Studio is the easiest way to build applications of this type. Because SAS provides so many custom tags (which you can think of as macros for java code) and you can find general custom tags from a lot of sources, it is possible to create a simple JSP application with knowing Java.

HOW TO LEARN MORE

WHITE PAPERS AT SAS

SAS has a lot of good white papers to help programmers get into web development. You can find them, obviously, in the Web Communities section,

<http://support.sas.com/rnd/web/index.html>,

but also in the Base SAS community,

<http://support.sas.com/rnd/base/index.html>

for ODS, and the Enterprise Integration community,

<http://support.sas.com/rnd/eai/index.html>

for Integration Technologies and SAS/Share for JDBC.

SAS ON-LINE DOCS

SAS/Intrnet: <http://support.sas.com/documentation/onlinedoc/91pdf/index.html#intrnet>

ODS: http://support.sas.com/documentation/onlinedoc/91pdf/sasdoc_91/base_ods_7032.pdf

AppDev Studio: <http://support.sas.com/rnd/appdev/doc/index.htm>

JAVA RESOURCES

Sun's Developer's Web site: <http://java.sun.com/>

O'Reilly and Associates: <http://java.oreilly.com/>

Jakarta Struts: <http://struts.apache.org/>

SAS-L ARCHIVES

<http://listserv.uga.edu/cgi-bin/wa?A0=sas-l>

CONTACT INFORMATION

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

Chuck Kincaid

Venturi Technology Partners

5278 Lovers Lane

Kalamazoo, MI 49002

Email: ckincaid@venturipartners.com

Phone: 269-344-4100 ext 536

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.