

Building Web Applications with SAS[®] AppDev Studio[™] 3.0

Frederick Pratter, Eastern Oregon University, La Grande OR

ABSTRACT

The SAS/IntrNet[®] Software product is now nearly 10 years old and uses the obsolete CGI interface. SAS has provided several newer products, included in AppDev Studio 3. Using SAS Integration Technologies or Java custom tag libraries, it is possible to create dynamic Web pages that provide access to SAS datasets.

AppDev Studio[™] 3.0 includes a graphical user interface for developing and deploying Web applications called *webAF*. This workshop covers building a simple interactive JavaServer Page using *webAF*. The only prerequisites are knowledge of HTML and an interest in Web programming in SAS. Java programming experience is helpful but not absolutely essential.

INTRODUCTION

Version 9 SAS Web Technologies include four products for building Web-based applications: SAS/IntrNet, Integration Technologies, the SAS Information Delivery Portal and AppDev Studio (<http://support.sas.com/rnd/web>). Together, these offer a rich and sometimes bewildering array of options for integrating the capabilities of SAS with the Web. This workshop focuses specifically on the last of these, *AppDev Studio*, which provides developers with the resources to create Java applications that access SAS data and procedures.

A *Web application* is the term used by Sun to describe a collection of documents that may contain Java Servlets, JavaServer Pages, style sheets and/or other supporting XML. Servlets are the server-side version of applets, Java programs that run in a Web browser. JavaServer Pages are an extension of HTML that includes embedded Java code. (For more information about Web applications, see Bodoff, 2003.)

AppDev Studio is a standalone development environment that includes a Java IDE (*webAF*) and a set of prewritten components (Information Beans) available as part of a custom tag library (<http://www.sas.com/technologies/bi/appdev/appdev>). When SAS says that AppDev Studio is a standalone environment, they mean that you do not need to have SAS installed on the machine where you are building your application. Of course, you do need to have access to a SAS server in order to provide data and procedures to the programs along with a Web server to host the application. But you can create your Web pages on one machine and easily port them to another environment, using the Wizards SAS has provided. Note however, that *webAF* is only available for the MS Windows operating system.

Although *webAF* can do many things, its primary function is building JavaServer Pages. JSP is Sun's technology for serving dynamic Web content (<http://java.sun.com/products/jsp>). According to SAS's Robert Girardin:

JSP technology enables the user interface view to be separated from the business logic of the application. This is known as Model-View-Controller or MVC Type II, which relies on the ability of a servlet to delegate, or "dispatch," a request to another resource (Girardin 2003).

In other words, the recommended way to build a Web application is first to write a Java servlet, the front-end controller, which constructs a model and then initializes one or more JavaServer Pages that act as the view component of the architecture.

This can be a daunting procedure. Fortunately, SAS has gone to a lot of trouble to write Web Application example templates using the MVC II architecture (<http://support.sas.com/rnd/appdev/webAF/server/WebAppExampleTemplates.htm>). This workshop will cover using one of these, the JDBC TableView Template. This handy application allows you relatively easily

to build a dynamic Web page that uses Java Database Connectivity (JDBC) to access a SAS data set. The application supports cell, row and member level editing; records can be inserted into and deleted from the dataset via the Web. For this workshop, we are going to follow the directions in the online documentation at http://support.sas.com/rnd/appdev/examples/ServletJSP/RecordLevelEditing_abt.htm, with a few customized wrinkles to make it more interesting.

THE WEBAF DEVELOPMENT ENVIRONMENT

Register Connections

The webAF IDE provides a complete set of services for building and testing your Web applications. The first thing you should probably do is check to make sure that you have a connection to the SAS server. This can either be a SAS/CONNECT or a SAS IOM server. By default, webAF creates an IOM connection to a server running your local host. If SAS is installed on your local machine you can start the server by going to the main Windows menu and selecting **Start Menu ➤ SAS AppDev Studio ➤ Services ➤ SAS V9.1 ➤ Start SAS V9.1 IOM Spawner**. If you are using another server on the network, you need to register a connection to it using the tool provided (see below).

In either case, start by selecting **Tools ➤ Register Connections**. You should see a list of “Persisted Connections”, that is, server links that webAF knows about. If you need to create a new one, select the **New** button and fill in the requested information (see Pratter, 2003 for more information about customizing connections). Once you have a connection, either to the local host or to a remote server, you can check it by selecting the connection name and clicking the **Edit** button. Select the **Test** tab and click on **Check Connection**. If the connection is live you should see the following message at the bottom of the window:

Connection Success!!

You can now create a Web page that accesses the data on that server.

Creating a Web Application

SAS uses the concept of a project to manage Web applications. To create a new project open webAF and select **File ➤ New**. In AppDev Studio 3.0 you should see the following list of choices:

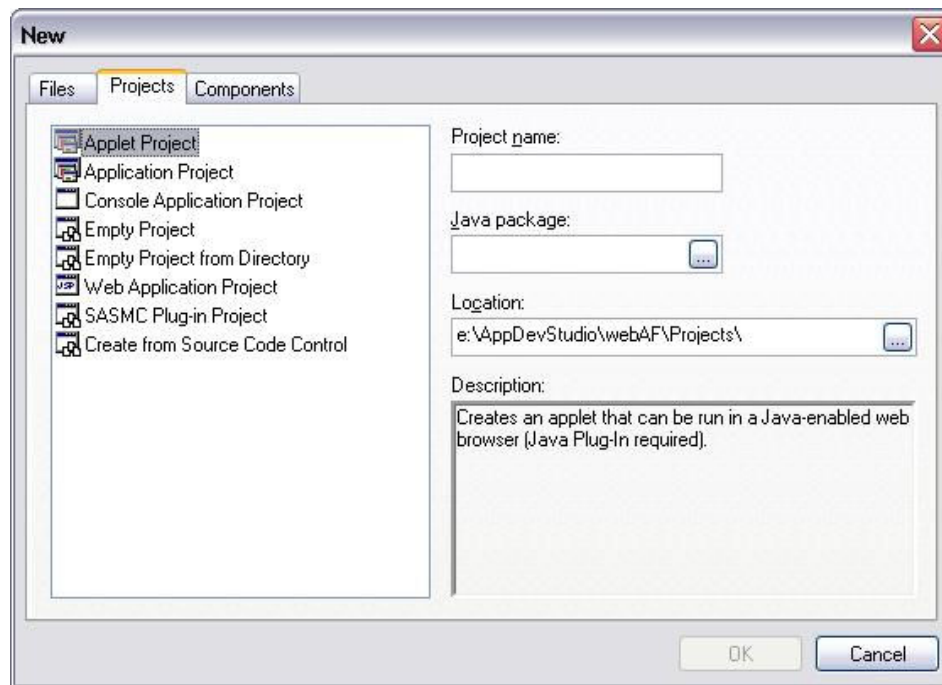


Figure 1. Create New Project

Select `Web Application Project` from the left hand menu, and type in the project name; for this workshop use `NESUGDemo` (no spaces). Click **OK**. This will start the WebApp Project Wizard. Click **Next** on Steps 1, 2 and 3 to accept the defaults.

Step 4 is "Select Web Application Initial Content" as shown in **Figure 2** below:

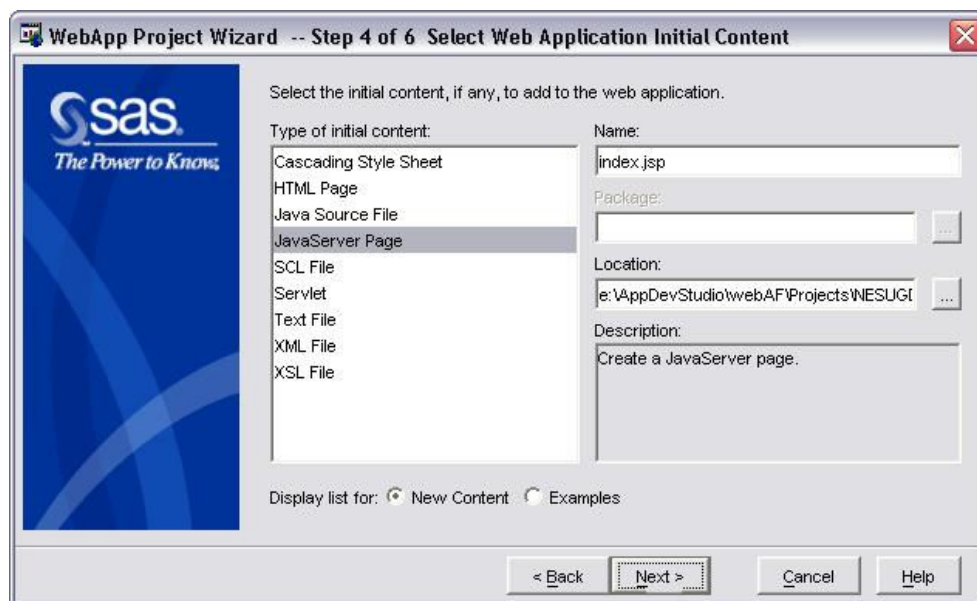


Figure 2. Web Application Initial Content

In order to use the pre-built template, you need to select `Examples` in `Display list for`. On the screen that appears next (see **Figure 3**) select `JDBC TableView Servlet`, and then click **Next**.

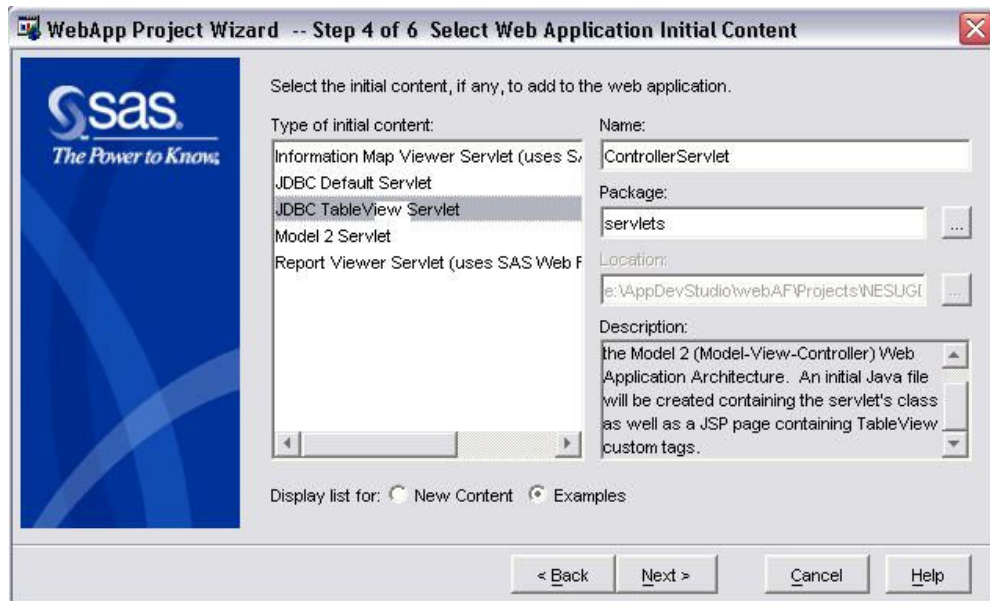


Figure 3. Create JDBC TableView Servlet

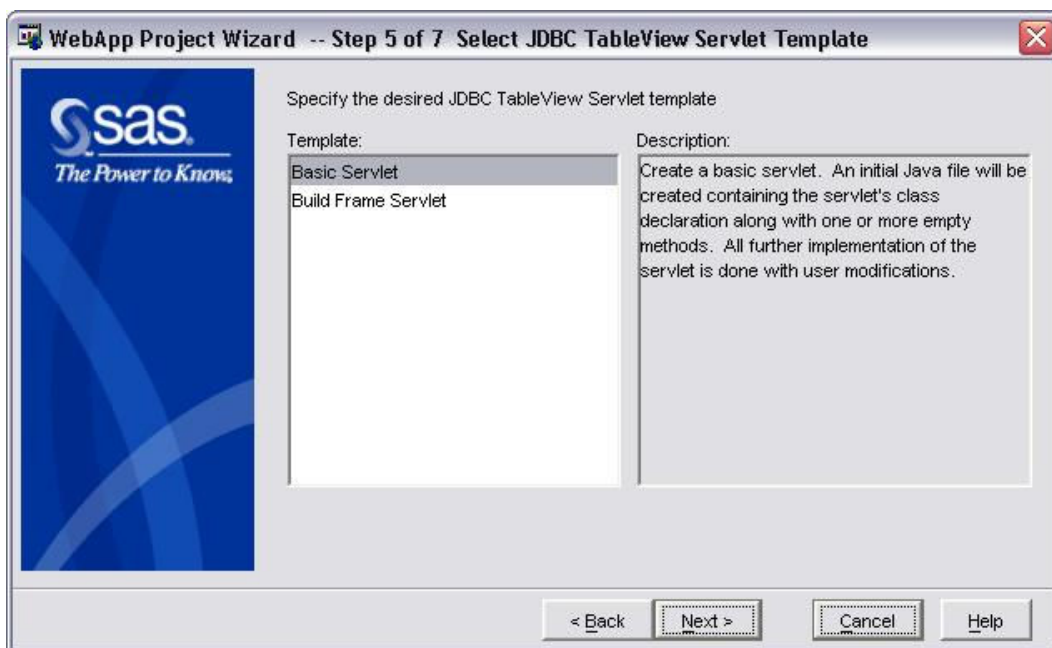


Figure 4. Select JDBC TableView Servlet Template

Select `Basic Servlet` as shown, and click **Next**. Click **Next** on Step 6, `Specify Basic Servlet Options`. Click **Next** on Step 7, `Specify WebApp Project Options`. Click **Finish** on Step 8 to build the application. This concludes the WebApp project Wizard.

Modify the Controller Servlet

To view the generated servlet, click on the second tab “Files” in the webAF Project Navigator Window and then select `<webappbase>\WEB-INF\classes\servlets\ControllerServlet.java` as seen in **Figure 5:**

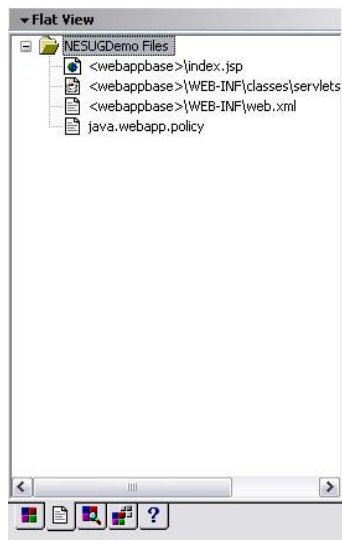


Figure 5. webAF Project Navigator

You need to make three changes to the generated code in order to point to a specific SAS dataset and to allow table editing:

1. Locate the static variable `JDBC_DATABASE_URL`. Change the value to point to your server.
2. Change the value of the variable `jdbcQuery` to read the desired data:

```
String jdbcQuery = "select * from sasuser.shoes";
```
3. Locate the initializer of the variable `adapter`. Add the following line to allow editing of the data:

```
adapter.setReadOnly(false);
```

This completes the necessary changes to the Java servlet code!

Note that if you need more information on any of the Java components in the servlet or JSP, highlight the term you want to search and press F1 for help. SAS has provided context sensitive help and links to the API to make it somewhat easier to learn about the components and their methods.

Modify the JavaServer Page

Select `<webappbase>/index.jsp` in the webAF Project Navigator. This page contains the custom tags required for the application. In order to enable Row level editing, add the following code to the `TableViewComposite` tag:

```
<sas:TableView>  
    <sas:Edit enabled="true" singleRowEditing="false"/>  
</sas:TableView>
```

That should do it!

Test the Application

To build the project using the Java Ant tool, select **Build ➤ Build Project** from the main webAF menu or just press F7.

The sample templates all assume you are using a local SAS server. If your data reside on a remote host, you need to change the properties of the project by selecting **File ➤ Project Properties**. Select **Startup** in the left-hand navigator window. In the box labeled *For this project, pass additional arguments to the Java interpreter:* you need to change the name of the local host to that of the remote server.

AppDev Studio has a built-in Tomcat 4 servlet engine. You can start it from within webAF by going to **Tools ➤ Services ➤ Start Java Web Server**. You should see a command window with some cryptic messages in it. When the server finally starts, the last two lines will say something like:

```
Aug 31, 2004 8:27:24 PM org.apache.coyote.http11.Http11Protocol start
INFO: Starting Coyote HTTP/1.1 on port 8082
```

Now run the application by selecting **Build ➤ Execute in browser** or by clicking on the exclamation point on the Build toolbar.

Depending on the speed of your processor, it can take a very long time to display the page for the first time. This is because the JSP must be parsed, validated, translated into a servlet and finally executed. After the first time the page is displayed, however, it should load much more quickly.

If you have followed all of these directions, the following page should appear in your browser window:

	REGION	PRODUCT	SUBSIDIARY	Number of Stores	Total Sales	Total Inventory
☐	Africa	Booty	Addis Ababa	12.0	29761.0	191821.0
☐	Africa	Men's Casual	Addis Ababa	4.0	67242.0	118036.0
☐	Africa	Men's Dress	Addis Ababa	7.0	76793.0	136273.0
☐	Africa	Sandal	Addis Ababa	10.0	62819.0	204284.0
☐	Africa	Slipper	Addis Ababa	14.0	68641.0	279795.0
☐	Africa	Sport Shoe	Addis Ababa	4.0	1690.0	16634.0
☐	Africa	Women's Casual	Addis Ababa	2.0	51541.0	98641.0
☐	Africa	Women's Dress	Addis Ababa	12.0	108942.0	311017.0
☐	Africa	Boot	Algiers	21.0	21297.0	73737.0
☐	Africa	Men's Casual	Algiers	4.0	63206.0	100982.0
☐	Africa	Men's Dress	Algiers	13.0	123743.0	428575.0
☐	Africa	Sandal	Algiers	25.0	29198.0	84447.0
☐	Africa	Slipper	Algiers	17.0	64891.0	248198.0
☐	Africa	Sport Shoe	Algiers	9.0	2617.0	9372.0
☐	Africa	Women's Dress	Algiers	12.0	90648.0	266805.0
☐	Africa	Boot	Cairo	20.0	4846.0	18965.0
☐	Africa	Men's Casual	Cairo	25.0	360209.0	1063251.0
☐	Africa	Men's Dress	Cairo	5.0	4051.0	45962.0

Figure 6. SASHELP.SHOES

Note that the URL for this page points to the `ControllerServlet`, not to `index.jsp`. That is because the servlet is delegating the HTTP request to the page.

The format of this page is determined by a Cascading Style Sheet (CSS) file supplied as part of the default application. In this case, the file is located under the project directory in `webapp\styles\sasComponents.css`. You can change the appearance of the page by editing this style sheet; scroll down about 900 lines to the comment where it says `TABLEVIEW STYLES` (applies to `com.sas.servlet.tbeans.tableview.html.TableView`). Changing the font, color or weight of the visual components is just a matter of locating the correct element tag and making the desired changes.

SAS has also provided a template for using what they call “Renderers” to display rows and columns in alternating colors and fonts (see http://support.sas.com/rnd/appdev/examples/ServletJSP/DisplayingTableusingJDBC_BLD.htm). This template can be loaded in the same way as the `TableView` example we have been using.

Figure 7 shows the bottom of the screen. Note the navigation arrows: these are used to scroll up and down over the records in the dataset. The three icons in the lower left (✓, X and +) run JavaScripts. If the checkbox in the first column of the page is checked, clicking the checkmark symbol at the bottom of the form deletes the row; otherwise, this action commits any changes made to the record. The X symbol cancels any edits, while the + symbol is used to add a new record to the dataset. Clicking on a column heading brings up a menu for sorting the data or moving the column. The small symbol in the upper left of **Figure 6** can be used to export the entire dataset to MS Excel.

10	☐	Africa	Men's Casual	Algiers	4.0	63206.0
11	☐	Africa	Men's Dress	Algiers	13.0	123743.0
12	☐	Africa	Sandal	Algiers	25.0	29198.0
13	☐	Africa	Slipper	Algiers	17.0	64891.0
14	☐	Africa	Sport Shoe	Algiers	9.0	2617.0
15	☐	Africa	Women's Dress	Algiers	12.0	90648.0
16	☐	Africa	Boot	Cairo	20.0	4846.0
17	☐	Africa	Men's Casual	Cairo	25.0	360209.0
18	☐	Africa	Men's Dress	Cairo	5.0	4051.0
19	☐	Africa	Sandal	Cairo	9.0	10532.0
20	☐	Africa	Slipper	Cairo	9.0	13732.0

Rows 1 - 20 of 395

Figure 7. SASHELP.SHOES (continued)

DEPLOYING THE WEB APPLICATION

Sending the page to another Web server is made simple by the webAF Package Wizard. Note that the only version of Tomcat formally supported by SAS is Version 4.18; anything newer (Tomcat 5 for instance) is not guaranteed to work; the example below was run in Tomcat 4.30.

Note also, the application has to have access to the SAS server. For that reason, the connection to the localhost should use the SAS server name, not a local address like 127.0.0.1.

To create a Web archive (war) file of the application, click on **Tools ➤ Wizards ➤ Package Wizard**. There are only two steps; you probably want to accept the defaults, so just click on **Next** each time. The war should contain all of the components needed to deploy the file.

Now open the Tomcat Manager application on the Web server. Go to **Upload a WAR file to install**. You should be able to browse to a file called `NESUGDemo.war` in the project directory. Click **Install**. The war file (which can be quite large) will be copied to the server and deployed. If all goes well, you should get the message `OK - Installed application at context path /NESUGDemo`.

Now enter the URL for the servlet, in this case

```
http://hostname:8080/NESUGDemo/ControllerServlet
```

Again, it may take some time for the contents of the war file to be extracted. You should see the same page as shown in **Figure 6** above. Congratulations. You have now created and deployed a Java Web application.

CONCLUSION

This workshop is intended simply as an introduction to the webAF sample templates. We have closely followed the examples in the SAS documentation but you are encouraged to explore and try new combinations of components. The topic of Java Web applications is covered in literally hundreds of books and presentations from Sun and others. The references at the end of the paper offer some additional help and guidance, and of course, the SAS Web Tools community web site should always be consulted for the most up-to-date information.

CONTACT

Frederick Pratter, Ph.D.
Computer Science/Multimedia Studies Program
Division of Science, Math and Technology
Eastern Oregon University
One University Boulevard
La Grande OR 97850
541-962-3065 (phone); 541-962-962-3596 (fax)
fpratter@eou.edu
<http://www.eou.edu/~fpratter>

TRADEMARK CITATION

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 registered trademarks or trademarks of their respective companies.

REFERENCES

Bodoff, Stephanie. 2004. Web Applications. *The Java Web Service Tutorial*.
<<http://java.sun.com/webservices/docs/1.0/tutorial/doc/WebApp.html>>. Copyright © 2004 Sun Microsystems, Inc., Santa Clara, California.

Ferguson, Chad and Sandra Brey. 2003. Developing Data-Driven Applications Using JDBC and Java Servlet/JSP Technologies. *Proceedings of the 28th Annual SAS Users Group International Conference*. Cary, NC: SAS Institute Inc.

Girardin, Robert. 2003. Introduction to the SAS Custom Tag Library. *Proceedings of the 28th Annual SAS Users Group International Conference*. Cary, NC: SAS Institute Inc.

Pratter, Frederick. 2003. *Web Development with SAS by Example*. SAS Books by Users. Cary NC: SAS Institute Inc.

Pratter, Frederick. 2004a. Building an Online Entry Form with WebAF (and a Little Java). *Proceedings of the 29th Annual SAS Users Group International Conference*. Cary, NC: SAS Institute Inc.

Pratter, Frederick. 2004b. Java Server Pages. *The Internet Encyclopedia*. Hossein Bidgoli, ed. John Wiley & Sons, Inc.

SAS Institute Inc. 2003. *SAS AppDev Studio 3.0 Migration Guide*. Cary, NC: SAS Institute Inc.