Paper 037-29

Exploiting the SAS Business Intelligence Architecture Using SAS AppDev Studio™ 3.0

Rich Main, SAS Institute Inc., Cary, NC

ABSTRACT

This paper gives an overview of the new features provided in SAS AppDev Studio 3.0 that enable enterprise application developers to exploit the powerful capabilities of the new SAS 9.1 Business Intelligence Architecture. Topics covered include new components that support the use of Information Maps and the scalable OLAP server, enhancements to the development environment to support the creation of content for the SAS Information Delivery Portal, and new project templates that speed the development of web applications that use the SAS Business Intelligence Platform.

INTRODUCTION

You have just received your new copies of SAS AppDev Studio 3.0 and SAS 9.1. Now what do you do? The goal of this paper is to provide you with a vision of how we designed SAS AppDev Studio 3.0 to support development of enterprise applications that harness the power of the SAS Business Intelligence Architecture. We will accomplish this not just through words, but also by example – allowing you to see the product in action as we designed it to be used.

WHAT'S NEW IN SAS APPDEV STUDIO 3.0

Several key market-driven goals guided the development of SAS AppDev Studio 3.0:

- Implement enhanced development capabilities that take advantage of the latest Java technology
- Deliver new components and graphs to support customer requirements
- Enhance support for web application development given the popularity of server-based Java solutions
- Provide customizable content templates to enable rapid application development
- Support data access using the new SAS Information Maps
- Create and deploy content for the SAS Information Delivery Portal

These goals resulted in the addition of specific new and enhanced features within the SAS AppDev Studio product bundle. We discuss these features below along with several examples of how to use these new capabilities to exploit the power of the SAS Business Intelligence Architecture from Java programs.

GENERAL JAVA DEVELOPMENT ENHANCEMENTS

SAS AppDev Studio 3.0 supports the latest Java technologies, including J2SE 1.4 and J2EE 1.3. SAS AppDev Studio 3.0 has been certified as J2EE 1.3 compatible. Some of the key new Java development features within webAF include:

- WebAF uses the popular open source Apache Ant[1] build tool to manage project builds. Users can add custom build tasks, which webAF then automatically integrates into the build menu.
- The webAF source editor provides enhanced intelligent editing assistance, including auto-completion support for JavaServer Pages syntax and JSP custom tag attributes.
- WebAF includes an integrated Apache Tomcat[2] servlet container to support embedded testing and debugging of both applets and web applications.
- The webAF debugger includes integration with the source editor to provide data-tips during debug sessions along with an enhanced variables view that highlights all variables modified since the last debugger command. The debugger also includes new web application debugging support and enhanced JSP and servlet debugging capabilities.
- WebAF supports team development through integration with source code control systems that use the Microsoft SCC interface.

[1] See http://support.sas.com/rnd/appdev/webAF/server/WebAFCustomBuild.htm for more information on how webAF 3.0 integrates with the Apache Ant build tool.
[2] See http://jakarta.apache.org/tomcat/ for more information on the Apache Tomcat servlet container.

For more details about these features and all of the additional new capabilities that SAS AppDev Studio 3.0 provides, visit the SAS AppDev Studio 3.0 Developers Site[3].

NEW COMPONENTS AND GRAPH STYLES

SAS AppDev Studio 3.0 builds upon the rich component library from previous releases with a significant collection of new components, many of them included "by popular demand". Here is just a sampling of the new components that are available:

- New adapters that support attaching the SAS AppDev Studio visual components (both Swing and JSP based) to JDBC data sources. This supports access to any data source for which there is a JDBC driver.
- New connection factory component that allows you to manage the SAS Business Intelligence Platform connection service, including administration of pools of back-end SAS workspace servers.
- New RemoteFileSelector component, which allows users to choose files from a variety of remote data stores (filesystems, SAS libraries, etc.).
- New server-side viewers that support web data entry and form based data navigation, new JSP custom tag viewers such as TreeView, ComboBoxView, ListBoxView, a variety of dual selector components, menu bar, and navigation bar.
- New Swing-based graphics for the client and associated server-side graphics generation components, including BarChart, BarLineChart, LineChart, LinePlot, PieChart, RadarChart, and ScatterPlot. All of the new graph components support the SAS 9.1 Graph Styles.

[3] The AppDev Studio 3.0 Developers Site is located at http://support.sas.com/rnd/appdev/.

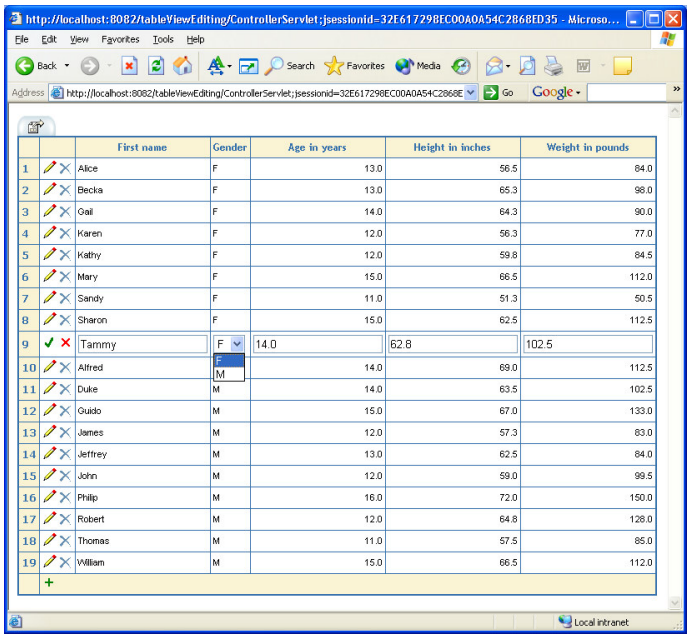


Figure 1 - Web-based data entry using the new JSP TableView component

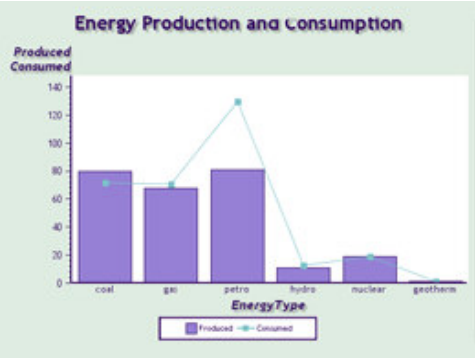


Figure 2 - New BarLineChart Component

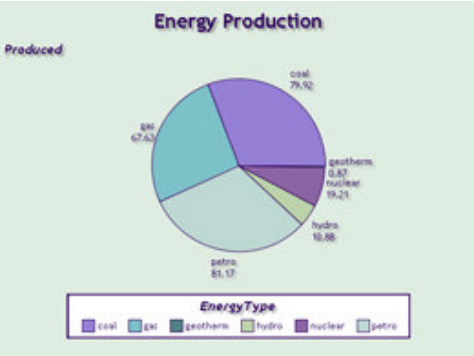


Figure 3 - New PieChart component

WEB APPLICATION DEVELOPMENT AND DEPLOYMENT

WebAF 3.0 includes significantly enhanced support for server-side Java development. This includes comprehensive support for development of standard J2EE web applications via the following features[4]:

- A new Web Application project wizard that guides creation of standard J2EE web applications using pre-packaged content templates
- A new set of JSP custom tags that support development of web applications using the JSP Standard Tag Library (JSTL)[5]
- A new set of JSP custom tags that support development of web applications that use the popular open source Struts[6] web application framework
- Integrated web application test and debug support[7]
- Web application packaging and deployment support via the Package Wizard

We will see some of these features in action later as we walk through some examples.

WEB APPLICATION TEMPLATES

WebAF 3.0 includes a set of pre-packaged web application templates that greatly speed the process of creating J2EE web applications for business intelligence. In addition to these pre-packaged web application templates, webAF 3.0 also supports the creation and use of custom web application templates. This allows developers to tailor their web application content to suit their particular environment. This paper presents an example showing how to create and use a custom web application template below.

ACCESSING DATA USING THE NEW SAS INFORMATION MAPS

SAS AppDev Studio 3.0 includes data modeling components that support access to both relational and OLAP data using Information Maps. The new Visual Data Explorer component (provided in an update to SAS AppDev Studio that will be available with SAS 9.1.2) provides a highly functional composite viewer that can navigate any relational or OLAP Information Map.

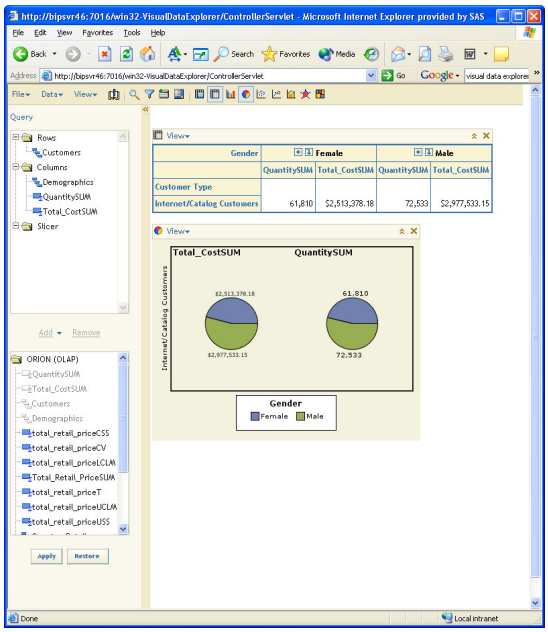


Figure 4 - Visual Data Explorer

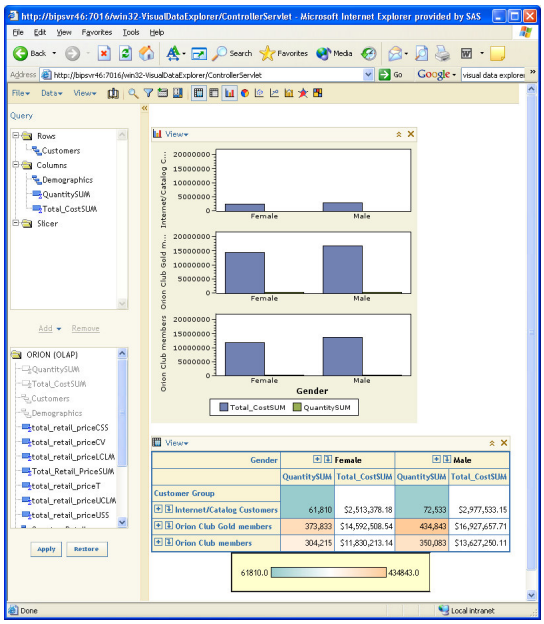


Figure 5 - Visual Data Explorer

Developers can use the Visual Data Explorer for rapid development of web-based data exploration applications, as we will see shortly.

---

[4] See http://support.sas.com/rnd/appdev/webAF/server/WebAFWebAppBenefits.htm for more details.
[5] See http://support.sas.com/rnd/appdev/webAF/server/JSTL_intro.htm for more information on webAF 3.0 and JSTL.
[6] See http://support.sas.com/rnd/appdev/webAF/server/Struts_Intro.htm for more information on how webAF 3.0 supports development of web applications that use the Struts framework.
[7] See http://support.sas.com/rnd/appdev/webAF/server/WebAppDebugging.htm for more information.

SAS INFORMATION DELIVERY PORTAL INTEGRATION

SAS AppDev Studio 3.0 supports rapid development and packaging of portlets, which provide "pluggable" content for deployment to the SAS Information Delivery Portal. In addition, developers can easily deploy web applications created using SAS AppDev Studio into the portal environment.

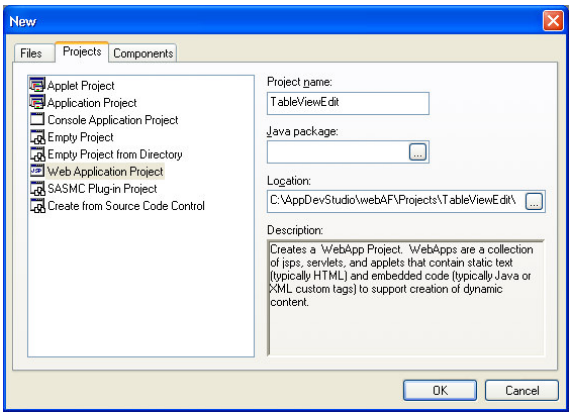
As we will see in a subsequent example, these capabilities result in a powerful combination of content creation with content presentation.

NEW WEB APPLICATION DEVELOPMENT FEATURES

As mentioned previously, web application development capabilities are greatly enhanced in SAS AppDev Studio 3.0. WebAF now includes support for J2EE web application projects and provides web application templates to speed the creation of web-based applications. In this first example, we will show how these templates, in conjunction with the powerful set of Java components provided with SAS AppDev Studio, make it easy to build web applications that interoperate with the SAS server.

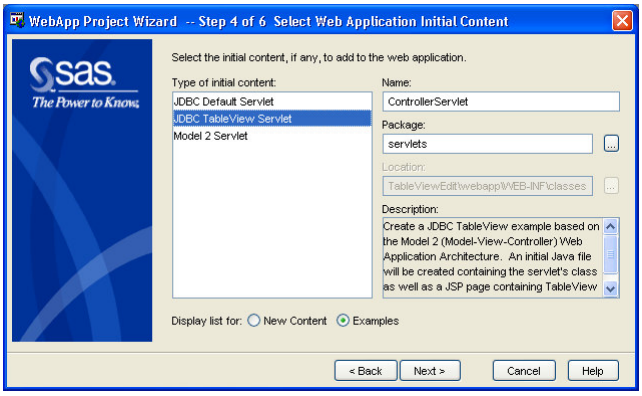
This example is a modified version of the TableView Editing example found on the SAS AppDev Studio developer site.[8] It uses a JDBC connection to access a data set on the SAS server, providing the ability to add, edit, or delete records in the dataset from a web-based data entry screen. First, we use the Project Wizard's new web application project support to create a project called **TableViewEdit**, as follows:

- Select **File > New** from the webAF main menu. On the **Projects** tab of the New dialog, choose **Web Application Project** as the project type and enter **TableViewEdit** as the project name:



- Click on the **OK** button and then accept the default values on the first three screens of the resulting web application project wizard dialog.

- On step 4 of the web application project wizard, select the **Examples** radio button to display the list of example template content and then choose **JDBC TableView Servlet** as the initial content, as shown below:



[8] See http://support.sas.com/rnd/appdev/examples/ServletJSP/RecordLevelEditing_abt.htm.

- Continue through the web application project wizard, accepting the remainder of the default settings. WebAF will then generate the web application project and initial content using the information from the selected template. The webAF status bar and Component Log output window will show status information as the operation proceeds.

Once the web application project generation completes, you will have a basic web application shell that requires some minor modifications to specify the data source of interest and to support web-based data entry[9]. The web application contains several key files. `ControllerServlet.java` contains the source code for the "fronting servlet" that manages the web application resources. This is where we will set up the JDBC connection to the SAS data set and define whether or not the result set can be edited. The `index.jsp` file is the main "view" for the web application. This is where we will define the data presentation and how the user can interact with the data.

First, let's define the query for the JDBC connection so that we can access some SAS data:

- Open the `ControllerServlet.java` file from the Files tab of the webAF Project Navigator.
- Search for the placeholder string ENTER_QUERY_STRING_HERE and replace it with an appropriate JDBC query. The resulting line of code should look something like this:

```
String jdbcQuery = "select * from SASUSER.CLASS";
```

- Just below this statement is a block of code that creates the JDBC model adapter. This code must be modified to allow the user to edit the data:

```
if (adapter == null){
    try{
        //Create the model adapter and set it on the session
        adapter = new JDBCToTableModelAdapter(sas_JDBCConnection,
                                              jdbcQuery);
        adapter.setReadOnly(false);
        if (session != null){
            session.setAttribute(SAS_MODEL, adapter);
        }
    }
    catch(Exception e){
        throw new RuntimeException(e);
    }
}
```

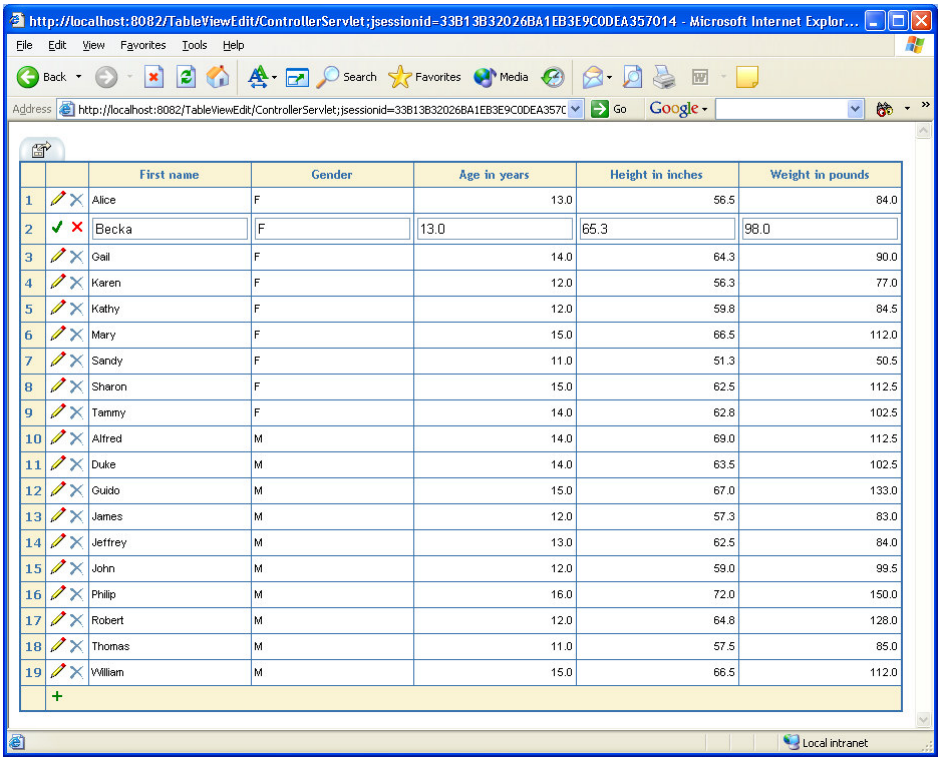
If we execute the application at this point, we will simply see a view of the data set without any editing capabilities. This is because the default view for this template only supports data display. So, the next thing that we need to do is to modify the view to include the necessary data editing support:

- Open the `index.jsp` file from the Files tab of the webAF Project Navigator.
- Add `<sas:TableView>` and `<sas:Edit>` JSP custom tags to support member level editing of the data, as shown below:

```
<sas:TableViewComposite id="sas_TableView1" model="sas_model"
                        actionProvider="sas_actionProvider"
                        scope="session">
    <sas:RelationalMenuBar />
    <sas:TableView>
        <sas:Edit enabled="true" />
     </sas:TableView>
</sas:TableViewComposite>
```

Now, when we execute the application, we get a web-based data view with editing support:

[9] See the paper *Understanding the Directory Structure for webAF Web Applications* on the SAS AppDev Studio Developers Site (http://support.sas.com/rnd/appdev/webAF/server/WebAFWebAppDirectories.htm) for details concerning the various directories and files that webAF generates when creating a new web application project.



		First name	Gender	Age in years	Height in inches	Weight in pounds
1		Alice	F	13.0	56.5	84.0
2		Becka	F	13.0	65.3	98.0
3		Gail	F	14.0	64.3	90.0
4		Karen	F	12.0	56.3	77.0
5		Kathy	F	12.0	59.8	84.5
6		Mary	F	15.0	66.5	112.0
7		Sandy	F	11.0	51.3	50.5
8		Sharon	F	15.0	62.5	112.5
9		Tammy	F	14.0	62.8	102.5
10		Alfred	M	14.0	69.0	112.5
11		Duke	M	14.0	63.5	102.5
12		Guido	M	15.0	67.0	133.0
13		James	M	12.0	57.3	83.0
14		Jeffrey	M	13.0	62.5	84.0
15		John	M	12.0	59.0	99.5
16		Philip	M	16.0	72.0	150.0
17		Robert	M	12.0	64.8	128.0
18		Thomas	M	11.0	57.5	85.0
19		William	M	15.0	66.5	112.0

At this point, it would be nice to add a cell editor for the *Gender* field so that users could only select valid values from a drop-down list of choices. The SAS AppDev Studio model and associated JSP custom tags easily support such capabilities. To take advantage of them, simply follow these few steps:

- First, we must add some code to `ControllerServlet.java` to create the data model that supports the proper selections. We need an import statement at the top of the file to include the package where our data model is defined:

```
import javax.swing.DefaultComboBoxModel;
```

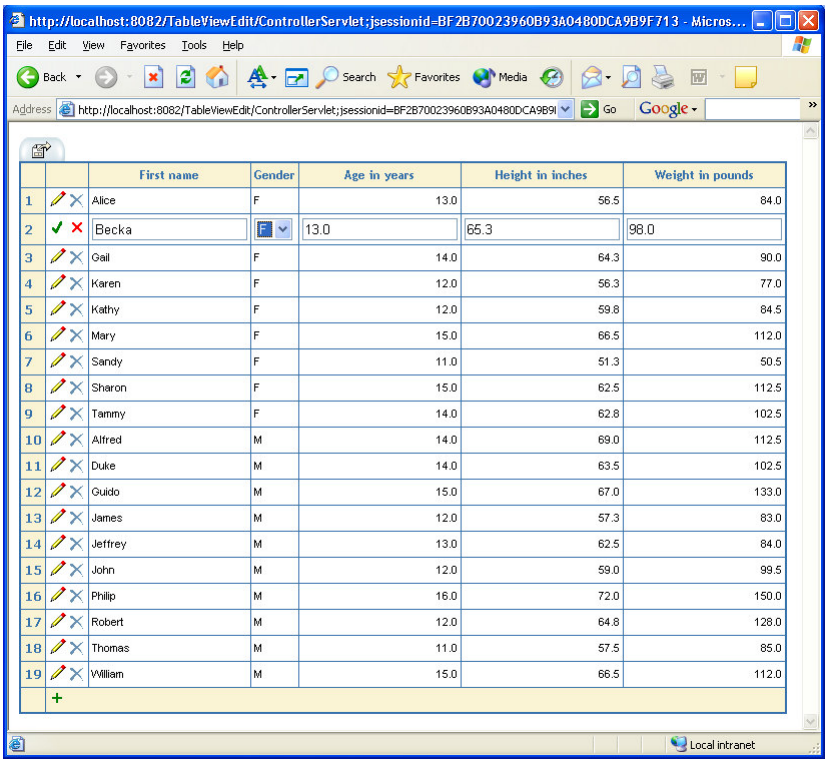
- Next, we need to create the data model and make it available in the session context so that it can be used by the view:

```
if (session != null){
    sas_actionProvider =
        (HttpActionProvider)session.getAttribute(ACTION_PROVIDER);
    String items[] = {"F", "M"};
    DefaultComboBoxModel comboBoxModel =
        new DefaultComboBoxModel(items);
    session.setAttribute("sas_ComboBoxModel",comboBoxModel);
}
```

- Finally, we need to update the view to use the newly defined model to support cell level editing of the *Gender* column:

```
<sas:TableView>
      <sas:Edit enabled="true"/>
      <sas:CellEditor startRow="1" endRow="-1"
                      startColumn="2" endColumn="2">
            <sas:CellContentsChoiceBoxEditor model="sas_ComboBoxModel"/>
      </sas:CellEditor>
</sas:TableView>
```

Now when we execute the application, we get a data entry form with a drop-down selection list for setting the gender:



	First name	Gender	Age in years	Height in inches	Weight in pounds
1	Alice	F	13.0	56.5	84.0
2	Becka	F	13.0	65.3	98.0
3	Gail	F	14.0	64.3	90.0
4	Karen	F	12.0	56.3	77.0
5	Kathy	F	12.0	59.8	84.5
6	Mary	F	15.0	66.5	112.0
7	Sandy	F	11.0	51.3	50.5
8	Sharon	F	15.0	62.5	112.5
9	Tammy	F	14.0	62.8	102.5
10	Alfred	M	14.0	69.0	112.5
11	Duke	M	14.0	63.5	102.5
12	Guido	M	15.0	67.0	133.0
13	James	M	12.0	57.3	83.0
14	Jeffrey	M	13.0	62.5	84.0
15	John	M	12.0	59.0	99.5
16	Philip	M	16.0	72.0	150.0
17	Robert	M	12.0	64.8	128.0
18	Thomas	M	11.0	57.5	85.0
19	William	M	15.0	66.5	112.0

CUSTOM WEB APPLICATION TEMPLATES

In addition to providing pre-packaged web application templates, webAF allows developers to create and share their own templates. For example, the SAS AppDev Studio Developer Site contains a set of custom example templates.[10]

In this example, we will show how to modify the template used above so that webAF generates the table editing support code automatically.

After downloading the example templates from the SAS AppDev Studio Developers Site and extracting them into your SAS AppDev Studio installation directory, the JDBC TableView template is located in the following directory:

```
<appdev-install-dir>\webAF\Master\webapps\options\SASTaglib\examples
```

The templates.xml file deflines the base set of templates that are available. In this file, we need to add a content type the refers to our custom template. This is done by creating an entry in the content types DOCTYPE mapping and then referring to that entry in the <content-types> block.

```
<!DOCTYPE content-types [
    .
    .
    .
    <!ENTITY jdbc-edit SYSTEM "JDBC-Edit-Templates.xml">
>]

<content-types resources="resources\Resources.properties" keyPostfix="@res"
removePostfix="true">
    .
    .
    .
    &jdbc-edit;
</content-types>
```

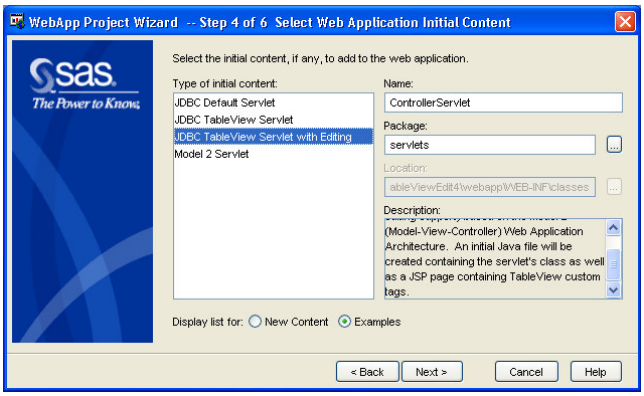
[10] Go to http://support.sas.com/rnd/appdev/webAF/server/template_updates.htm to download these example templates.

Next, we need to create a copy of the existing `JDBCTemplates.xml` file (which defines the JDBC TableView template) using the file name that we defined in the DOCTYPE (`JDBC-Edit-Templates.xml` in this example). In this file, the `http.servlet.code` property defines the content template to use for generating the controller servlet code and the `http.servlet.jsp` property defines the content template to use for generating the viewer JSP code. We need to modify these properties to point to new code generation templates that add the editing support, as follows:

```
<property name="http.servlet.code"
stringkey="http.servlet.JDBCTableViewEditCode"/>
<property name="http.servlet.jsp" stringkey="http.servlet.JDBCTableViewEditJsp"/>
```

To finish things off, we need to modify the global content templates file (for these example templates the file is called `BQGlobals.xml`) to include the new content templates with the editing support included. Copy the old content template blocks (`<string key="http.servlet.JDBCTableViewCode">` and `<string key="http.servlet.JDBCTableViewJSP">` and change the keys to match the new keys defined above. Then, modify the servlet code template to include the `setReadOnly(false)` call and modify the JSP code to include the `<sas:TableView>` and `<sas:Edit>` tags, as shown in the prior example.

Finally, save all of the changes to the files. Now, when we run through the Web Application Project Wizard, we see the "JDBC TableView with Editing" template choice listed under the set of initial content selections:



Choosing that selection for initial content tells webAF to generate the controller servlet and viewer JSP code with the table editing support code built right in. All the developer needs to do at this point is to define the correct JDBC query to retrieve the data for display / edit.

ACCESSING INFORMATION MAPS AND OLAP CUBES

Since the Visual Data Explorer is a new component available in the SAS 9.1.2 timeframe, the Visual Data Explorer examples are not available in time for the SUGI paper deadline. We will present examples of the Visual Data Explorer in action at SUGI 29. The SAS AppDev Studio Developer Site will include an updated copy of this paper with these examples included shortly thereafter.[11]

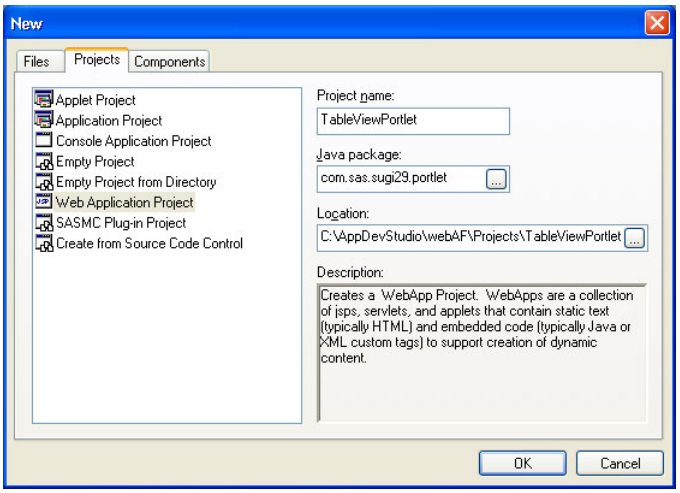
SAS INFORMATION DELIVERY PORTAL INTEGRATION

In this example, we will re-create our previous table editing example as a portlet that can be deployed to the SAS Information Delivery Portal. In doing so, we will learn how to map the controller servlet behavior from the web application example into a portlet action class that sets up the database connection and then delegates the display work to a JavaServer Page that is defined as the portlet view.

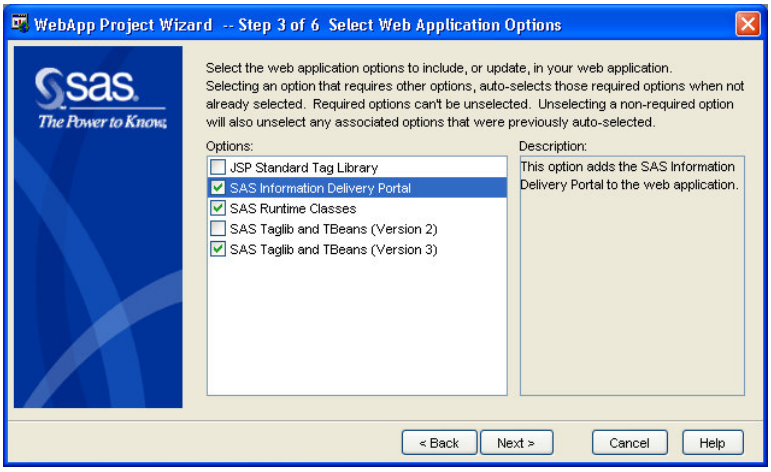
WebAF 3.0 supports portlet creation within the context of a web application project. This is parallel to the SAS Information Delivery Portal itself, since the portlet container is a web application. So, the first thing that we need to do is to create a web application project with a portlet selected as the initial content:

- Select **File > New** from the webAF main menu. On the **Projects** tab of the New dialog, choose **Web Application Project** as the project type, enter **TableViewPortlet** as the project name, and **com.sas.sugi29.portlet** as the package name:

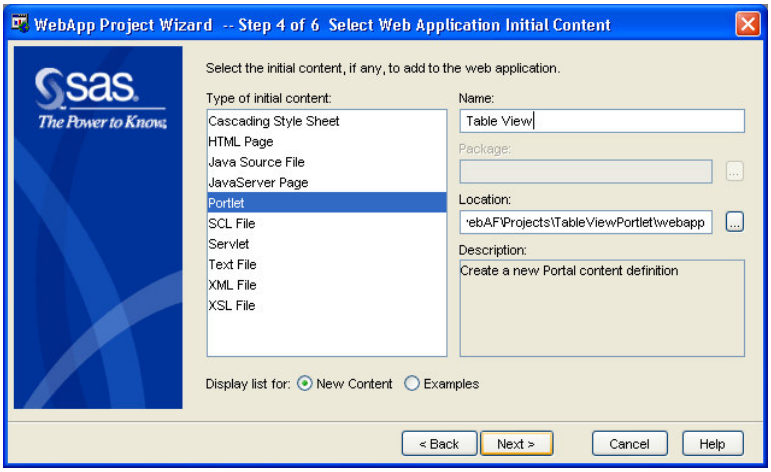
[11] The final version of this paper will be available at http://support.sas.com/rnd/appdev/doc/index.htm#techpapers.



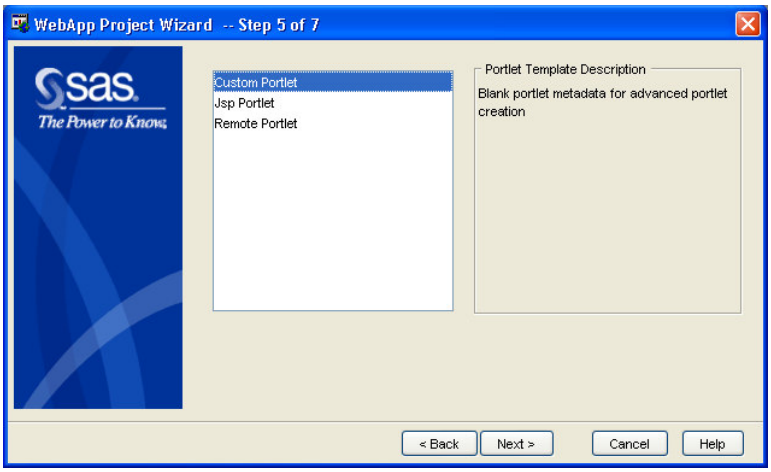
- Click **OK** and then accept the default settings on the next two wizard screens. On the **Web Application Options** screen, check the **SAS Information Deliver Portal** option in addition to the default selections:



- Click **Next** and then choose **Portlet** as the initial content type. Enter **"Table View"** in the portlet name field:

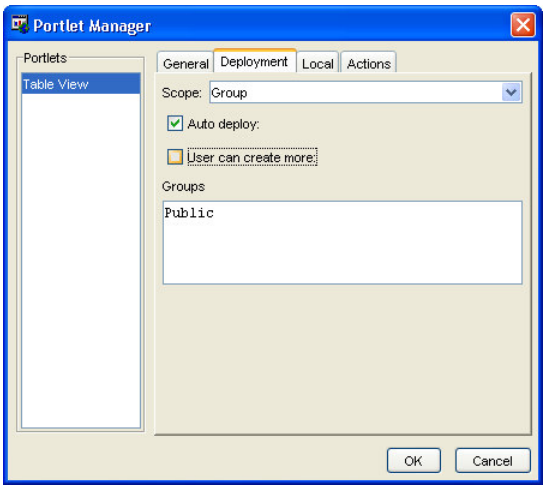


- Click **Next** and choose **Custom Portlet** for the portlet type:

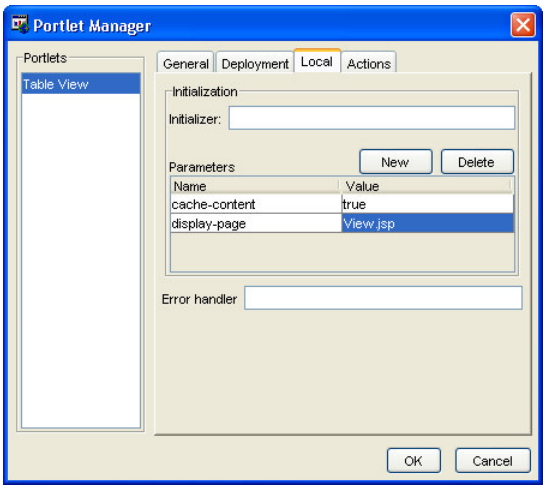


Accept the rest of the defaults in the wizard. Once you click the **Finish** button, webAF will create the base portlet content for you. Now, we need to modify the default portlet to specify how to handle display requests. We will do this using webAF's Portlet Manager.

Select **Tools -> Portlet Manager…** from the webAF main menu. Switch to the **Deployment** tab and check the **Auto deploy** checkbox. Clear the **User can create more** checkbox:

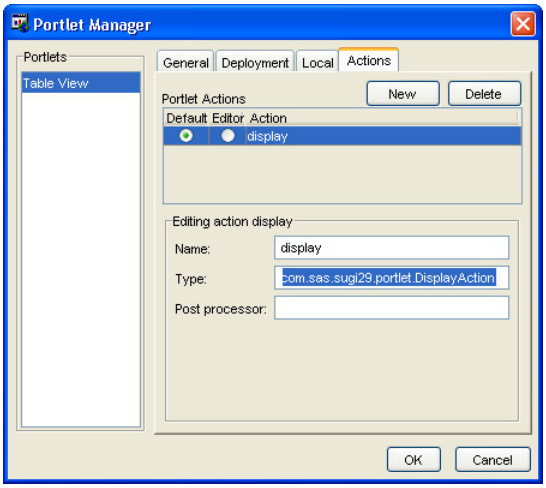


Switch to the **Local** tab. Here we can define parameters that tell the portal how to display or edit our portal as well as define an error handler and an initialization routine. To keep this example simple, we are merely going to define the display page for the portlet here. To do that, click on the **New** button and create a parameter called **display-page** with a value of **View.jsp**:



Now we need to define an action handler for the portlet display. The portal will invoke the display action handler prior to calling the defined portlet display page. In this example, we will use the display action handler in a manner similar to how we used the controller servlet in our earlier web application example. The display action handler will set up the JDBC connection to our data source for use by the viewer.

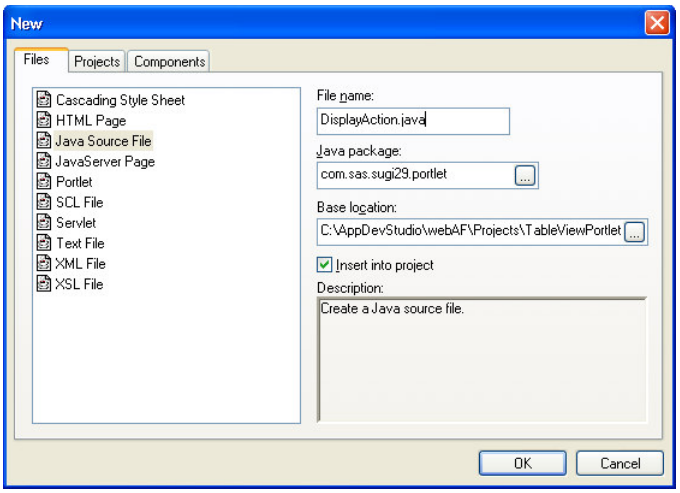
To configure the display action handler, switch to the **Actions** tab of the Portlet Manager. Click on the existing "Display" action in the **Portlet Actions** list. In the **Type** field, change the value to **com.sas.sugi29.portlet.DisplayAction**, which will be the name of our display action handler class:



Click **OK** to exit the Portlet Manager. Notice that webAF updates the portlet descriptor file (portlet.xml) with the new information.

Next, we need to create the display action handler using the following steps:

- Select **File > New** from the webAF main menu. On the **Files** tab, choose **Java Source File** and change the file name to **DisplayAction.java**. The package name should already be set to our default project package name of **com.sas.sugi29.portlet**.

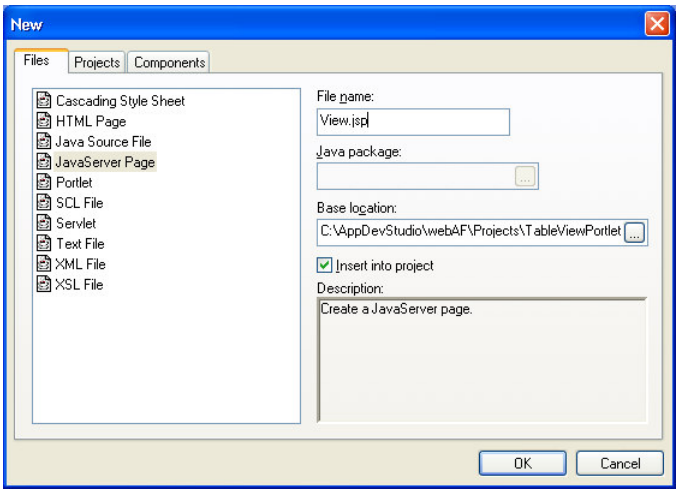


- Click **OK** and accept the default file type of **Simple Java File** on the resulting wizard screen. Click **Next** and then **Finish** to exit the wizard. WebAF will generate the boilerplate code for the new Java source file and open the resulting file in the source editor.

Replace the contents of the file with the source code for **DisplayAction.java** found on the SAS AppDev Studio Developers Site.[12]

Now that we have defined the display hander for the portlet, our next step is to create the view JSP that defines the portlet display. Create this file using the following steps:

- Select **File > New** from the webAF main menu. On the **Files** tab, choose **JavaServer Page** and change the file name to **View.jsp**.



- Click **OK**. Click **Next** to accept the default content type of **SAS Tag Library JavaServer Page** and then click **Finish** to exit the wizard and create the file.

Replace the contents of the file with the source code for **View.jsp** found on the SAS AppDev Studio Developers Site. Then, execute **Build > Build Project** from the webAF main menu to build the project. To package the portlet into a portlet archive (.par) file, execute **Build > Project Tasks > package-par** from the webAF main menu. This will create a file called **TableViewPortlet.par** in the project main directory. This file can be deployed for execution in the SAS Infromation Delivery Portal.

[12] See http://support.sas.com/rnd/appdev/doc/sugi29-portlet-example.htm for the source code for this example.

CONCLUSION

As we have seen through these few brief examples, SAS AppDev Studio 3.0 includes a host of valuable new features that support rapid development of custom applications that tap in to the unparalleled power of the SAS Information Delivery Platform.

CONTACT INFORMATION

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

Rich Main
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
Email: Rich.Main@sas.com
Web: http://www.sas.com