

Editing data sets via the web using SAS/IntrNet or SAS AppDev Studio in Version 9

Minh Duong and Kevin Davidson

ABSTRACT

Visually editing a data set is often a more straightforward method than writing and submitting code. This is especially true when frequent editing is required or when many end-users are doing the editing. Traditional SAS offers visual editing options with ViewTable, SAS/FSP , and SAS/AF. This paper will introduce ways to accomplish data set editing including modifying, adding, and deleting records via the web. The methods presented will be categorized in terms of ease of use, flexibility, and extendability. SAS/IntrNet and SAS AppDev Studio will be the software components discussed.

INTRODUCTION

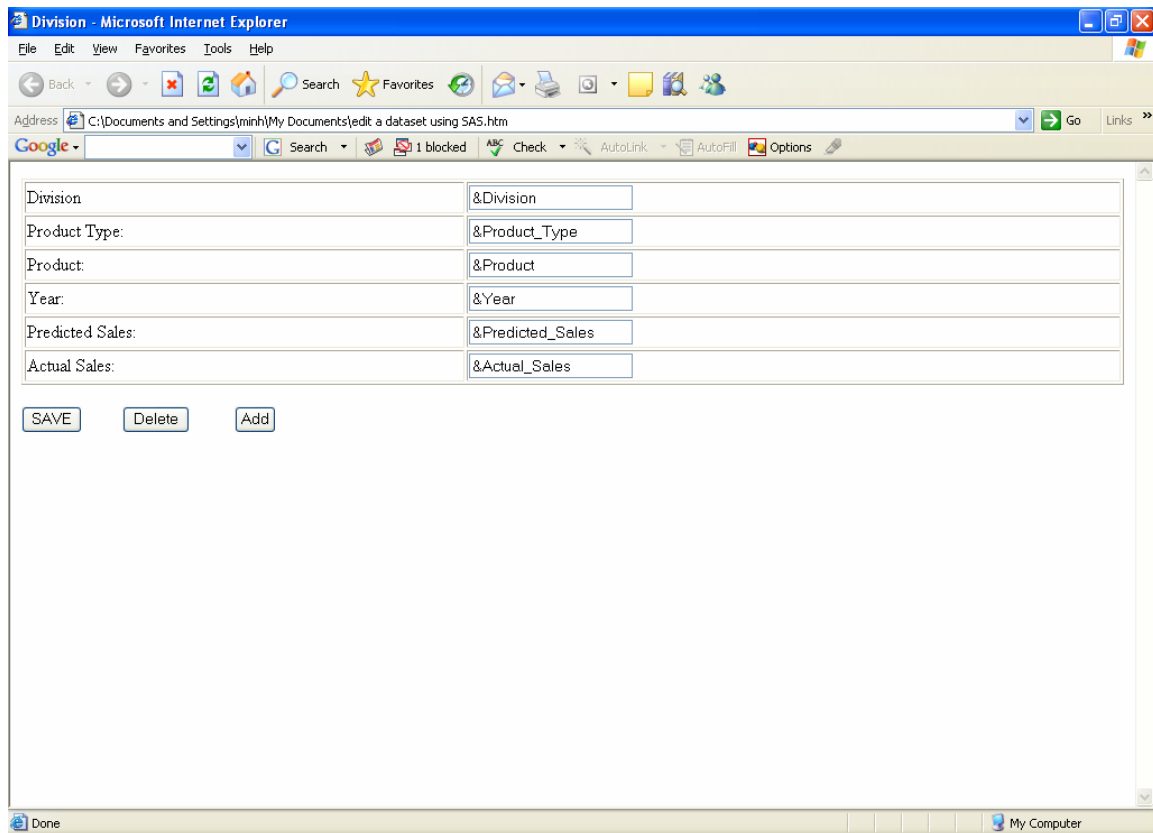
Writing SAS code to edit datasets can be a time consuming process and require some SAS knowledge. This paper will discuss two ways to create a web based interface to edit SAS data sets.

When creating an interface with SAS/IntrNet, the main two components are the HTML code and the SAS program. HTML is used to create the form for the user's input. This usually contains radio buttons, drop down list, and text boxes. The SAS program is the backend application that edits the data set.

AppDev Studio can be also be used to create an interface to edit a dataset. The webAF DataBean wizard allows you to create a DataBean to access and update data in a data set. The DataBean contains the property of each column in a data set and displays the data set one row at a time.

SAS/IntrNet

The first step in creating an interface for editing a data set is creating a HTML page. By using a HTML editor like Frontpage or Dream Weaver, creating an interface can be done very quickly. The HTML page can contain any column within the dataset you wish to display and edit. Embedded within the HTML page are macro variables and javascript. The macro variables will be resolved and display the values of the macro variable.



This program opens the data set to a specific record and creates a macro variable for every variable in the data set.

```
data _null_;
length name $ 32 type $ 1 value $2000;
drop dsid i num rc;
dsid=open(('s_data.data(where=(unique_id="" || upcase("&id") || "")));
rc=fetch(dsid);
num=attrn(dsid,"nvars");
do i=1 to num;
  name=varname(dsid,i);
  type=vartype(dsid,i);
  if type = 'C' then do;
    value=getvarc(dsid,varnum(dsid,name));
  end;
  else value=getvarn(dsid,varnum(dsid,name));
  call symput(trim(left(name)),trim(left(value)));
end;
rc=close(dsid);
run;
```

This program then webouts the HTML and resolves the embedded macro variables within the HTML file. The resolve function is used to resolve the embedded macro variables.

```

filename screen ( "c:\html_file\interface.html"
                );

data _null_;
  length _fname $300;
  infile screen trunccover pad filename=_fname lrecl=2500;
  input r $ 1-2500;
  file _webout;
  r = resolve(r);
  put r;
run;

```

A Proc Append is used to add a new record to the data set. When the user clicks the add button, a new blank record is added to the data set. After the new data is inputted the data can be saved by clicking the save button.

```

data new_screen;
  output;
  set s_data.data(obs=0);
run;

proc append base= s_data.data data=new_screen appendver=v6 force;
run ;

```

The save button executes a simple modify data step that saves the new data. Javascript is used to build a string with all the form element names. The list_of_quesnos is a hidden variable that is passed from the HTML page. The value of list_of_quesnos is the string built by javascript. The purpose of using list_of_quesnos is to only update the items from the HTML form and not all the variables in the data set. Below is an excerpt of the SAS program.

```

data s_data.data;
  modify s_data.data (where=(unique_id= upcase("&id"))) keep= &LIST_OF_QUESNOS);
  length _vname $32;
  array qn(*) &list_of_quesnos;
  do _i=1 to dim(qn);
    call vname(qn(_i),_vname);
    qn(_i)=symget(_vname);
  end;
  provider_notified = "";
  _error_=0;
run;

```

To delete a record the data step below is ran when the user clicks delete.

```

data s_data.data;
  modify s_data.data (where=(unique_id= upcase("&id")))
  remove;
run;

```

SAS AppDev Studio

APPDev Studio can be used to develop Java web applications through a visual development environment. This enables you to rapidly develop applications by using an object-oriented interface that reduces the amount of programming needed. This example below is from the SAS AppDev Studio Developer's Site.

The first step in creating a web based interface with SAS AppDev Studio to create a project in webAF.

1. Create a new project named DataBeanForm.
2. Select Web Application from the webAF Projects list.
3. Accept the defaults as you go through the Web App wizard until you have reached step #3. Select the SAS Taglib and TBeans (Version 2) option in the Options list. Leave the SAS Runtime Classes and SAS Taglib and TBeans (Version 3) selected. Proceed to step #4. At this step you will need to select Examples in the radio box titled Display list for. Choose Model 2 Servlet from the list box titled Type of initial content.
4. Continue accepting defaults as you complete the Web App wizard.

The next step is to setup the data source. This is done by creating a databean with the databean wizard.

1. Start the IOM Spawner by selecting Start Menu SAS AppDev Studio Services SAS V9.1 Start SAS V9.1 IOM Spawner.
2. Start the DataBean Wizard by selecting Tools Wizards DataBean Wizard from the webAF menu bar.
3. Wizard Step 1: Leave the default JDBC data source selected and click Next.
4. Wizard Step 2: Select the SAS IOM JDBC driver. Enter select * from sashelp.prdsale for the query statement and click the Next button.
5. Accept the defaults for Steps 3 and 4. On Step 5 enter databeans.PrdsaleDataBean as the class name and click the Next button.
6. Accept the defaults for Step 6, and click the Finish button on Step 7.

Next add the import statements to the Model2ExampleControllerServlet.java source file.

```
import com.sas.actionprovider.HttpActionProvider;  
import com.sas.storage.jdbc.JDBCConnection;  
import databeans.PrdsaleDataBean;
```

Replace the code in the doGet method with the following code in the Model2ExampleControllerServlet.java source file.

```

// Note: Add User DO_GET code here
HttpSession session = request.getSession();

// Setup the ActionProvider
HttpActionProvider actionProvider = null;
synchronized (session)
{
    if (session != null)
    {
        actionProvider = (HttpActionProvider)session.getAttribute("actionProvider");
    }

    //if ActionProvider is null, create one and put it on the session
    if (actionProvider == null)
    {
        actionProvider = new com.sas.actionprovider.HttpActionProvider();
        actionProvider.setLocale(request.getLocale());
        actionProvider.setControllerURL(request.getContextPath() + "/Model2Example");
        actionProvider.setName("actionProvider");
        // store object in its scope
        if (session != null)
        {
            session.setAttribute("actionProvider", actionProvider);
        }
    }
    //else execute the ActionProvider command
    else
    {
        actionProvider.executeCommand(request, response, response.getWriter());
    }
}

synchronized (session)
{
    try
    {
        //Setup the JDBC connection
        JDBCConnection sas_JDBCCConnection = null;
        PrdsaleDataBean dataBean = null;
        if (session != null)
        {
            sas_JDBCCConnection = (JDBCConnection)session.getAttribute("sas_JDBCCConnection");
            dataBean = (PrdsaleDataBean)session.getAttribute("dataBean");
        }
        if (sas_JDBCCConnection == null)
        {
            sas_JDBCCConnection = new JDBCConnection();
            sas_JDBCCConnection.setDriverName("com.sas.rio.MVADriver");
            sas_JDBCCConnection.setDatabaseURL("jdbc:sasom://localhost:5310");
            session.setAttribute("sas_JDBCCConnection", sas_JDBCCConnection);
        }
        //Setup the DataBean
        if (dataBean == null)
        {
            dataBean = new PrdsaleDataBean();
            dataBean.setDataSource(sas_JDBCCConnection);
            session.setAttribute("dataBean", dataBean);
        }
    }
    catch(Exception e)
    {
        throw new RuntimeException(e);
    }
}

RequestDispatcher rd = getServletContext().getRequestDispatcher("/Model2ExampleViewer.jsp");
rd.forward(request, response);

```

Next add imports to the Model2ExampleViewer.jsp below the taglib directives:

```
<%@page import="com.sas.servlet.tbeans.util.validators.NumericInputValidator" %>
<%@page import="com.sas.util.transforms.BaseFormatTransform" %>
<%@page import="com.sas.util.transforms.FormatTransform" %>
<%@page import="com.sas.servlet.tbeans.navigationbar.html.NavigationBarEditingElement" %>
<%@page import="com.sas.servlet.tbeans.navigationbar.html.NavigationBarRowScrollingElement" %>
```

Next add the style sheet and title to the Model2ExampleViewer.jsp. Place the code between the <head> tags.

```
<link href="styles/sasComponents.css" rel="STYLESHEET" type="text/css">
<title>Product Sales Data</title>
```

Next add this code between the <body> tags in the Model2ExampleViewer.jsp

```
<Center><B>Product Sales Data</B>
<hr>
<sasads:DataBeanForm id="dataBeanForm1" name="myForm"
model="dataBean" actionProvider="actionProvider">
<tr>
    <td>Division:</td>
    <td><sas:TextEntry name="division"/></td>
</tr>
<tr>
    <td>Product Type:</td>
    <td><sas:TextEntry name="prodtype" /></td>
</tr>
<tr>
    <td>Product:</td>
    <td><sas:TextEntry name="product"/></td>
</tr>
<tr>
    <td>Year:</td>
    <td><sas:TextEntry name="year"/></td>
</tr>
<tr>
    <td>Predicted Sales:</td>
    <td><sas:TextEntry name="predict"/>
</td>
</tr>
<tr>
    <td>Actual Sales:</td>
    <td><sas:TextEntry name="actual"/>
</td>
</tr>
</sasads:DataBeanForm>
</Center>
```

Next modify the PrdsaleDataBean's resultSetConcurrency property, Add this code to the doGet method after the dataBean has been created.

```
dataBean.setResultSetConcurrency(java.sql.ResultSet.CONCUR_UPDATABLE);
```

Next replace the code below in Model2ExampleViewer.jsp

```
<sasads:DataBeanForm id="dataBeanForm1" name="myForm" model="dataBean" actionProvider="actionProvider">
```

With this code

```
<sasads:DataBeanForm id="dataBeanForm1" name="myForm" model="dataBean" actionProvider="actionProvider"
render="false" />

<%
//Get the editing element from the DataBeanForm navigation bar
NavigationBarEditingElement editElement =
(NavigationBarEditingElement)dataBeanForm1.getNavigationBar().getComponent("NAVIGATIONBAR_EDIT_ELEMENT");
//Hide the insert and delete buttons
editElement.setInsertVisible(false);
editElement.setDeleteVisible(false);

//Get the scrolling element from the DataBeanForm navigation bar
NavigationBarRowScrollingElement scrollingElement =
(NavigationBarRowScrollingElement)dataBeanForm1.getNavigationBar().getComponent("NAVIGATIONBAR_ROW_ELEMENT");
//Hide the page scrolling buttons
scrollingElement.setPageBackwardVisible(false);
scrollingElement.setPageForwardVisible(false);
%>

<sasads:DataBeanForm ref="dataBeanForm1" render="true" >
```

Now Start the Java Web Server from the webAF menu. Select Build ➤Execute. Your browser should display the image below. You can scroll through the dataset one row at a time and make any edits you want. You can also add and remove a record from the data set.

Product Sales Data

⬆ ⬆ ⬆ Row of 1440 ⬇ ⬇ ⬇ ✓ ✗ ✕ +

Division:	EDUCATION
Product Type:	FURNITURE
Product:	SOFA
Year:	1993
Predicted Sales:	850.0
Actual Sales:	925.0

CONCLUSION

Since most users would rather use an interface to edit a data set instead writing SAS code and submitting it. Creating a web based interface to edit a data set can save time and allow non SAS users to edit a data set.

SAS/IntrNet and AppDev Studio allows you to create your own interface for modifying data sets.

REFERENCES

SAS APP Dev Studio Developer's Site

http://support.sas.com/rnd/appdev/examples/ServletJSP/DataBeanForm_abt.htm

CONTACT INFORMATION

Minh Duong
FSD Data Services
2020 Southwest Fwy
Suite 206
Houston, TX 77098
Phone(713) 942-8436
Email: minhd@fsddatasvc.com

Kevin Davidson
FSD Data Services
2020 Southwest Fwy
Suite 206
Houston, TX 77098
Phone(713) 942-8436
Email: kevind@fsddatasvc.com