em7 Portal Template OCX

Portal Template Samples

Guide to the sample projects

Table of Contents

Introduction

The samples installation contains three working examples of the Portal Template OCX. The first one shows how Portal may be used in a VB application to connect to a database through DAO and dynamically create a hierarchical drill down report. The second example demonstrates how the Portal may be used inside server side Active Server Page scripts to create VRML graphs on the fly. The last example shows how Portal may be used to create Excel 97 macros that construct graphs from entire worksheets, or just selected parts of a worksheet.

The core of this document will involve how to set up these examples. It is assumed that the developer can then step through the code to understand the detailed workings of the Portal Template library. More detailed discussion of the specific graph objects properties and methods may be found in the document that came with the Portal Template OCX.

Samples

Visual Basic with DAO database access

Setup

To begin working with this sample, start Visual Basic and load the project DAOReport.vbp, which is located in the directory \DAOReport off of the directory where the samples were installed. The contents of this project should be a form and two class modules.

The only condition for on running this application is that the database DB401k.mdb is located in the application directory (which is where it is installed by default). Since DAO connects to the database directly in this example, it is not necessary to set up an ODBC data source to the database.

Finally, make sure that the RelativeProtoPath property is correctly set. See the Portal Template documentation for information on setting this property through the property pages. The project is now ready to be run or built into an executable.

Results

This example produces a hierarchical report from the DB401k database. This database contains information concerning the employees in a company and how they invest in the funds that make up their 401k plan.

The first result from the application is a main graph. This graph shows all of the employees for the company in a pie graph. The size of the slice represents, percentage wise, how much they earn compared to the other employees. All of the employees in the graph are included in the legend next to the graph. Clicking on a slice will allow it to be dragged from the graph and will also display the actual salary of that particular employee.

In this company, some employees are participants in the 401k plan and some are not. If an employee is not a participant, then their slice is very transparent. If they are a participant, then their slice is completely opaque and it will include a drill down cone.

Clicking on the drill down will take you to another pie graph which illustrates (percentage wise) how the employee allocated their 401k contributions to the different funds in the plan. Only funds to which the employee contributes are in this graph. In most cases, each fund slice will have another drill down cone.

When this drill down is followed, the viewer gets to see a column graph of the transactions that the employee has performed on that particular fund. Each column represents one transaction and their heights represent how much money was contributed at that particular transaction. This chart includes pop up information at the top of each column to let the viewer see exactly how much contribution the column represents. The viewer may then drill back up when they are done viewing.

This set of files represents how a developer can create an entire graph drill down hierarchy from a single source of data. The hierarchy includes three levels of information and also contains full navigation support for drilling up and down.

Application flow

The following describes the general flow that the application takes for creating the report.

- The application connects to the DB401k database. If the connection was successful, it continues.
- Next, the overview graph is created from the EmployeeInfo table. All of the employee information is retrieved and a PieGraph object is created. Then for each employee, a slice is detailed. If the employee is a participant, then a drill down is added to the slice and the next layer of graph is created.
- For the next pie graph, we need information from both the Allocation and Fund tables. For each fund to which the employee has contributed, a slice is added to the pie and has it's attributes set. For each slice, a drill down is added and a column graph is created for transaction information about that employee's contributions to that particular fund. **(If no fund has been contributed to by this employee, the function reports False and the drill down is removed from the original employee slice.)**
- For the final graph, a column graph, we query the Transactions table and get all transaction information for the particular employee and fund. We use two different PortalScaler objects here, because shares and amounts are scaling to different final ranges. The share values will scale to the range [1..3] and the amount values will scale to the range [4..7] (the graph is 8 VRML units tall). This guarantees that all amount columns will be at least 1 VRML unit taller than the share columns. Because we are using mixed scale, the Y axis tick labels are not included in this graph. Pop up text is also added to each column so that it's actual value may be viewed by brushing with the mouse. **(If no transactions have been performed on this fund by this employee, the function reports False and the drill down is removed from the original fund slice.)**
- Finally, the application reports with a message box that the report has completed. The report may be viewed by loading the Overview.wrl file into Cosmo Player.

Active Server Pages with ADO database access

Setup

The Active Server Pages example is more difficult to set up. This is because Visual Interdev projects are so machine dependent that it is almost impossible to distribute them. This text will guide you step by step through setting up the project in VI.

The first step is to create a data source name for the DB401k.mdb database in the same directory as the Visual Basic example project. If you have VI installed (with the demos) chances are you already have this (DB401k.mdb is one of the sample databases that comes with VI). To create the data source, go to the Control Panel and run 32bit ODBC. Select the SystemDSN tab and click the Add button. This is an Access database, so choose that and click Finish. Set the Data Source Name to "DB401k" and click select to find the DB401k.mdb file. Once you have done this, click OK and exit out of ODBC setup and Control Panel. Your ODBC data source is now set up.

Start Visual Interdev and select File | New and begin a new Web Project Wizard called "401kReport". Click Next and select a server, then click next and choose to create a new web. Click Finish to create the project.

Next, select Project | Add to project | Data connection from the main menu. Choose the "Machine Data Source" tab and choose the DB401k data source (the one you just created). You should now hava a "DataConn" field as a child to the global.asa file in the

project explorer. Now select Project | Add to project | Files from the main menu. Select the following files from the \ASPReport directory off of the Portal samples directory :

- EmployeeProfile.asp
- Participation.asp
- ParticipationGraph.asp
- VRMLEmployeeProfile.asp
- VRMLOverview.asp

Now, right click on the bold faced web project name in the Project explorer. In the pop up menu, you will see an item for "New folder" to the project. Select this and name it "protos". Right click on the new folder and select "Add files". Add the following files, which will be in the install directory for the em7 Protos (this should be "C:\Program Files\em7\VRAssist\" by default). Add the following files :

- VR_COMMON.wrl
- VR_GRAPH.wrl
- VR_NAV.wrl
- VR_UI.wrl
- VR_DB.wrl
- EM7HUD.gif

Right click on the "images" folder and select "Add files". Add the following files from the \ ASPReport directory off of the Portal samples directory :

- OVERVIEW.gif
- PARTICIPATION.gif
- RETURN.gif
- STATUS.gif
- VIEW.gif

We are now finally done with the file setup. It would be a good idea to save the project now. Now we just have to make sure that the Portal Template knows the correct proto path. Double click on the VRMLEmployeeProfile.asp file. Right click inside the declaration for the Portal Template OCX and choose "Edit ActiveX Control". This will bring up the property pages for Portal. Make sure that you have the control registered correctly on the "Portal Registration" tab. Next, choose the "Proto Path" tab. Select the "Internet URL of proto files" radio button. In the URL field, type the following :

http:// <your server name> /401kReport/protos/

Once this is complete, click OK and close out the property page and the control form to return to the ASP file. The example is now ready to run. The file Participation.asp is the root file, so right click on it and run it in your internet browser.

Results

This example is similar to the VB example, but only provides two levels of data. This database contains information concerning the employees in a company and how they invest in the funds that make up their 401k plan.

The file Participation.asp displays a table layout of all of the employees in the database. If they participate in the 401k plan, then a link image is placed to the right of their table entry. Clicking on this link will take you to another pie graph which illustrates (percentage wise) how the employee allocated their 401k contributions to the different funds in the plan. Only funds to which the employee contributes are in this graph. This page also has a link back to the main participation page.

At the bottom of the table is a link image to view the employee graph. This graph shows all of the employees for the company in a pie graph. The size of the slice represents, percentage wise, how much they earn compared to the other employees. All of the employees in the graph are included in the legend next to the graph. Clicking on a slice

will allow it to be dragged from the graph and will also display the actual salary of that particular employee.

In this company, some employees are participants in the 401k plan and some are not. If an employee is not a participant, then their slice is very transparent. If they are a participant, then their slice is completely opaque and it will include a drill down cone.

Clicking on the drill down will take you to the same graph that the link image next to the employee's entry in the participation table took you to.

This set of ASP files represents how a developer can create an entire graph drill down hierarchy from a single source of data using ADO. The hierarchy includes two levels of information, a mixture of HTML with embedded VRML, and also contains full navigation support for drilling up and down.

Application flow

The following describes the general flow that the application takes for creating the web pages.

- When the Participation.asp page is requested, it connects to the DB401k.mdb database. It then queries the EmployeeInfo table to get all of the employee information. This is set up in table form with a loop that traverses the ADO result set. The link images are added for employees participating in the 401k plan, and their employee IDs and names are appended as parameters to the link url.
- If an employee link image is clicked, the EmployeeProfile.asp page is requested. The request also contains the parameters of the employee's ID and their name. The name is displayed and the VRMLEmployeeProfile.asp file is embedded into the page. This ASP file is the VRML file for the employee allocation graph. Again, it takes to url parameters; employee ID and name. That information is appended to the embedded url.
- The VRMLEmployeeProfile.asp page is the first VRML ASP page we have seen. It is extremely important that when we are creating a VRML file with ASP, that the first line of the file be :
  <%@ LANGUAGE="VBSCRIPT" %><% response.contenttype = "model/vrml" %>#VRML V2.0 utf8
  This tells the requesting browser what the MIME type of the file is and it also declares the VRML header, which **must** be the very first line in the VRML file. When this page is requested, it connects to the database and queries the Allocation and Fund tables. It then creates the pie graph with Portal Template. Before we set pie graph's attributes, we traverse the allocation percentages to get the total allocated sum. Each value which went into calculating this sum will later be divided by the sum to find that value's percentage in the graph. When that has been done, we set the pie graph's properties and add a viewpoint which centers the pie. Finally, we export the graph to a string and send it using Response.Write.
- If the Overview graph link image is clicked from the main participation page, the ParticipationGraph.asp page is requested. The VRMLEmployeeProfile.asp file is also embedded into the page. This ASP file is the VRML file for the employee overview graph. It works very similarly to the VRMLEmployeeProfile.asp page described previously.

Excel 97 spreadsheets

Setup

To begin working with this sample, start Excel 97 and load the project PopulationStats.xls, which is located in the directory \ExcelReport off of the directory where the samples were installed. Excel will prompt whether or not you want to enable the macros... respond Yes, that you do want to enable them. The only dependency in running this sample is that the graph is set by default to export to the following folder :

C:\Program Files\em7\VRAssist\PortalSamples\ExcelReport\

This is where the Excel example installs by default, but if you want to change that, just look in the "PopulationGraph" function/macro in the VB editor and change the last line in the procedure. To view the macro code, select Tools | Macro | Visual Basic Editor. Also, if the output directory is changed, make sure that the RelativeProtoPath property is changed to reflect it. To change this property, double click on the frmMain entry in the Project explorer, single click on the Portal Template control and set the property in the Properties window.

Results

This example produces different graph from the other two examples. In this example, the data relates to populations of various groups by state. When the user selects one or many different rows in the spreadsheet, the macro will create a column graph of the historical, current and projected populations of men and women in the selected states.

The graph is a column graph with three columns per cell; historical, current and projected value. There is one cell on the X axis for each state in the graph and three cells on the Z axis; one for men, one for women and one for totals. To run the macro, simply select a range of rows in the spreadsheet (optimal is about 8 – 12), and press Ctrl-G. Or, you can select Tools | Macro | Macros and run the PopulationGraph macro from the list. When run, the file ExcelGraph.wrl will be created for you, ready to view in your VRML browser.

Application flow

The following describes the general flow that the application takes for creating the graph from Excel.

First, the PopulationGraph macro makes sure that at least one rows are selected. If not, it exits, because there's nothing to graph. The next step is to determine the min and max of the selected data in columns 3-11 (the data columns for this particular graph). Then, a PortalScaler object is created. Notice that the min and max are slightly reduced before being assigned to the scaler. This gives a little buffer on the top and bottom of the graph so that the columns don't press against the floor or ceiling. It makes the graph look nicer.

Next we find the number of selected rows and create the column graph. Inside the CreateGraph call, we create the graph object, set it's properties and then set the non-data oriented properties on all of the column series in the graph. We will set the data values later. We also show only one instance of each column type in the legend, so that we don't repeat duplicates. Finally, a viewpoint is added to center the graph.

We then take every selected row in the spreadsheet and take the data from it to populate the column's height values. The call PopulateRowData takes a single selected row and assigns cells 3-11 to the appropriate column heights. Notice that their original values are

assigned to the pop up text for the specific column, then they are run through the PortalScaler object to get the column height in VRML units.

Finally, the graph is exported to a file and the macro returns. There is much more data in this spreadsheet which could be graphed. For instance, pie charts could be made to visualize the age distribution in the population between males and females (columns O-T).