Intellimerce

Snowflake Metric Developer Guide

Snowflake Metric Developer Guide

This publication was produced by Intellimerce Inc.

We welcome all comments. Please send your comments as an email to support@intellimerce.com.

Revision History

Date of Issue	Document Id	Document Version	Software Version
Month Year	Filename	Draft or version	Version & Rev.
Mar. 2004	Snowflake Metric Developer Guide.doc	1.0	1.0.0

Snowflake Metric Developer Guide 1

Introduction 4
Tutorial 5
Creating New Project 5
Adding the Snowflake Metric control to the toolbox 7
Adding Snowflake Metric Control to the WebForm 9
Setting Component Properties 10
Setting the XML For Analysis Provider 11
Data Source name and Catalog 13
StartUpQuery 15
Enabling the ShowValue Property 18
Setting the Snowflake Metric control comparison boundaries 20
Run Application 23
Changing the MDX Statement Dynamically 24
Handling the DataFetched Event 29
Appendix A - Requirements 33
Appendix B - Authentication & authorizations scenarios 34
Appendix C - List of Snowflake Metric Properties and Events 35
Properties 35
Events 36

Introduction

Snowflake Metric is an asp.net control that can be used in business intelligence portals and dashboards to display metrics. Metrics for the purposes of this control are single scalar values that are returned from an XMLA query. The metric value can be compared to bounds and different image can be displayed depending on the bound. If the query returns more than one value only the first value is examined against the bounds. One can also display the numeric value as well as programmatically access the value through the DataFetched event.

The rest of this developer guide is dedicated to tutorial that will help you implement Snowflake Metric within an asp.net project. We hope that you enjoy the tutorials and that we save you significant time in your next BI portal project. Note that you can browse the code in the completed SnowflakeMetricTest web application sample that is part of the installation package. The installation package will install sample web application that uses Snowflake Metric control. You can find the SnowflakeMetric.dll in the bin subdirectory of the installed web application.

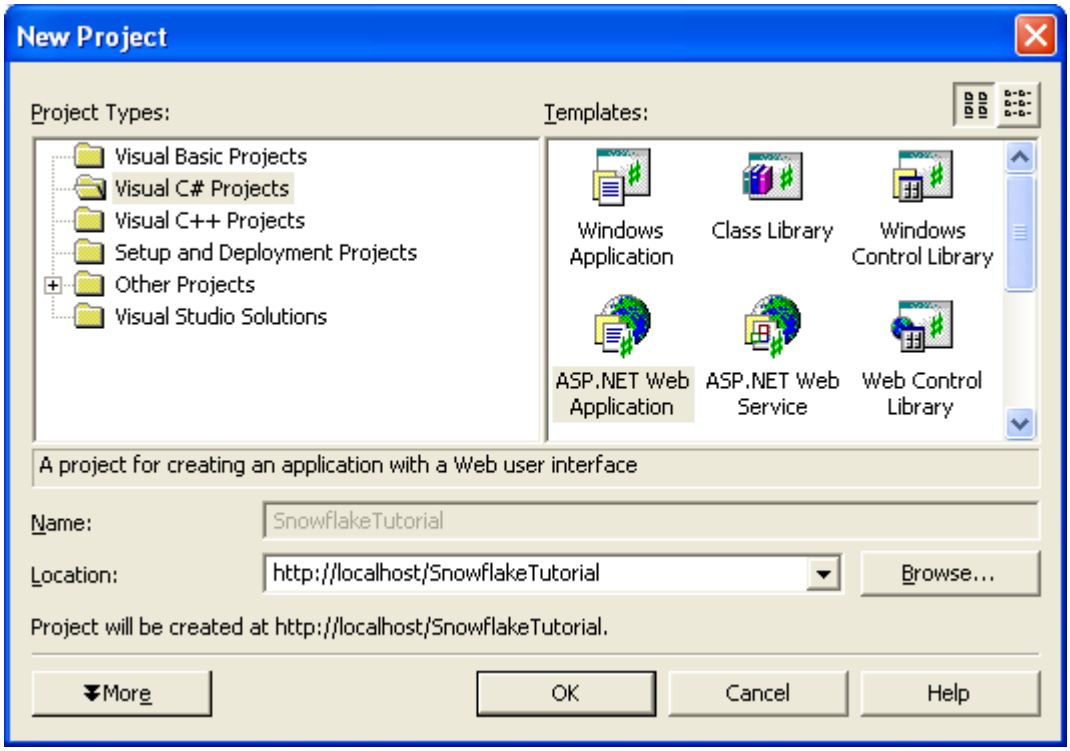
Please check out the Snowflake Metrics older brother - Snowflake.net the leading XMLA asp.net server control – on our site at http://www.intellimerce.com.

Tutorial

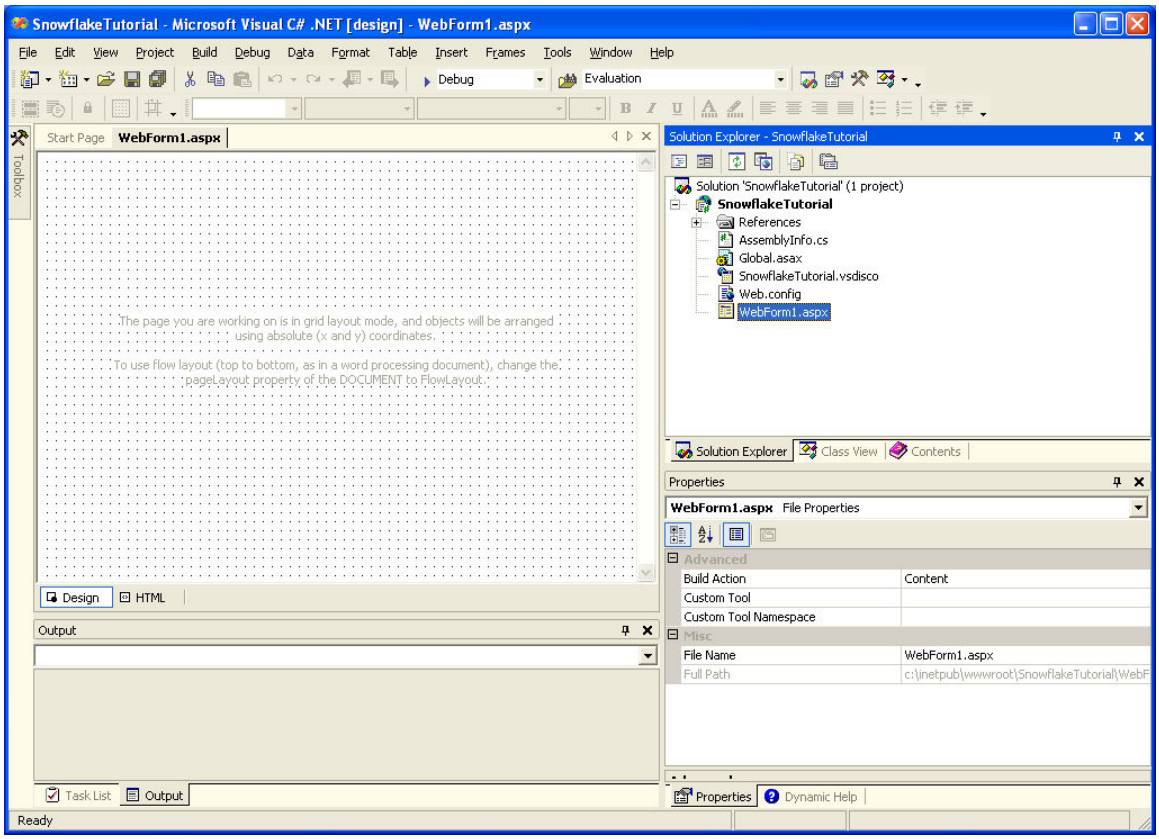
In this tutorial you will create a simple web page that will utilize the Snowflake Metric asp.net control to display XML for Analysis results. This tutorial takes about 10 minutes.

Creating New Project

Create a new ASP.net Web Application by clicking on File->New Project in the Visual Studio Developer and selecting ASP.NET Web Application. You can name your application whatever you wish but for the simplicity of following the tutorial use the "SnoflakeTutorial" name.



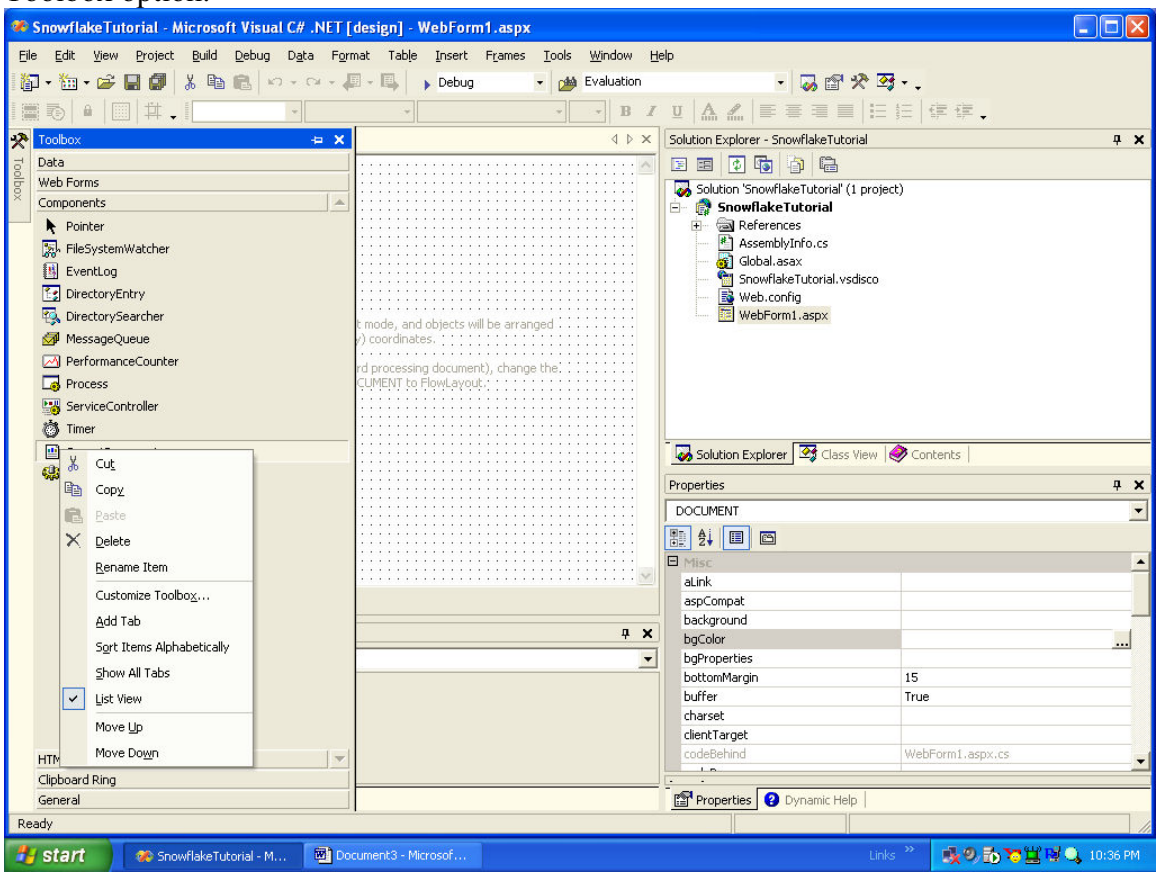
Once the project is created the project explorer will look as follows:



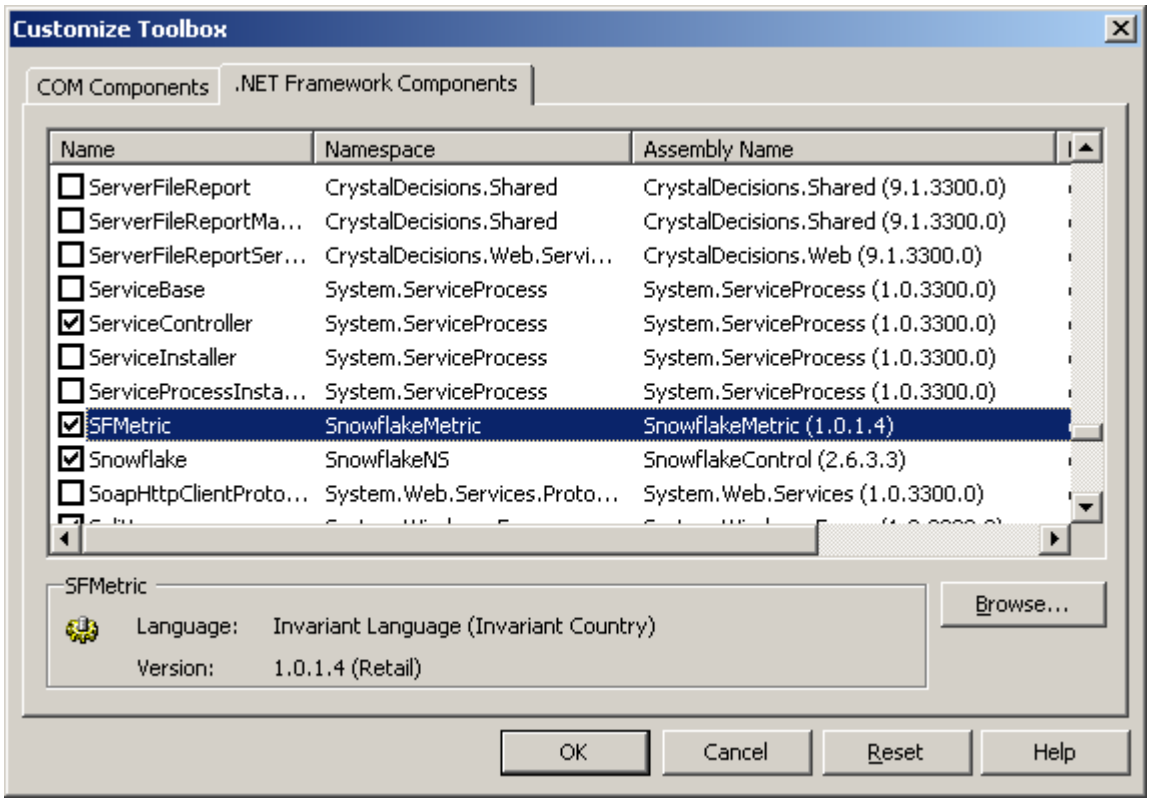
You will be adding the control to the WebForm1.asp web form (by the way the language of our choosing in this example was C#; you can use whatever language you wish without much difference to the process).

Adding the Snowflake Metric control to the toolbox

One of the easiest ways to gain easy access to the Snowflake Metric control is to add it to the toolbox in the components section. To do this expand the toolbox by hovering the mouse over it, right mouse button click in the toolbox panel and select the Customize Toolbox option.



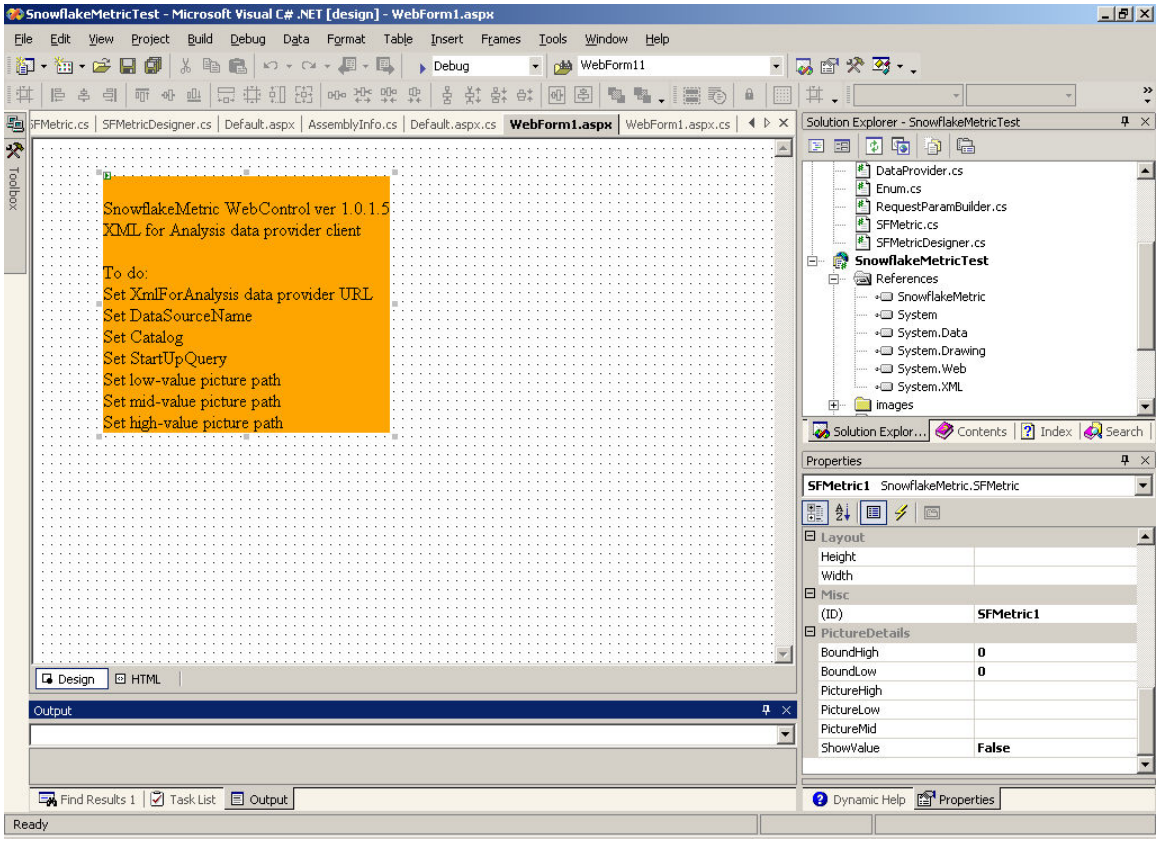
You can add references to com and .net components in the toolbox. Snowflake Metric is a .net component. Click on the browse button in the .net frameworks tab and select the SnowflakeControl.dll from the location where you have installed the Snowflake Metric control.



The control will now show up in your Toolbox->Components section. Visual Studio will automatically add the appropriate references to the project whenever you use the Snowflake Metric control within your web pages.

Adding Snowflake Metric Control to the WebForm

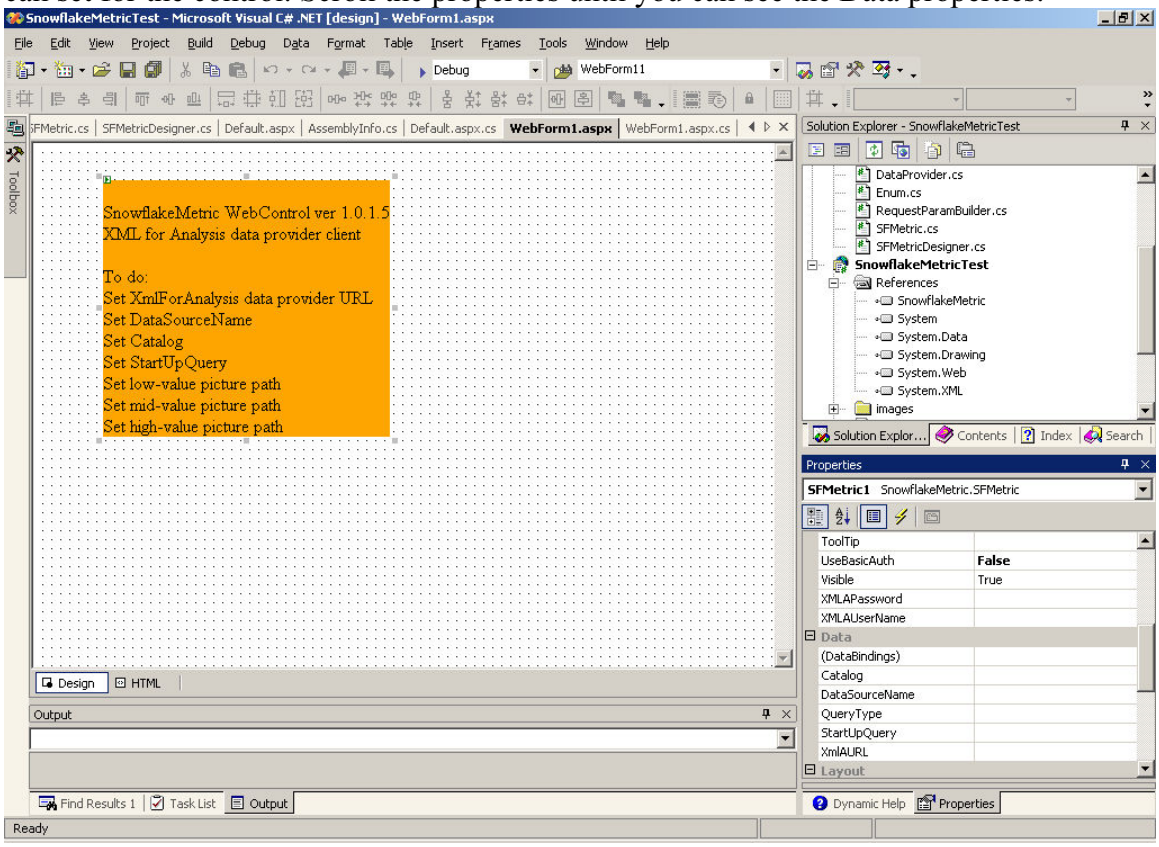
Adding the Snowflake Metric control to the WebForm is a simple matter of dragging and dropping the Snowflake Metric control from the Toolbox->Controls to the WebForm1.aspx page in the design mode. Resulting WebForm1.aspx looks as follows:



As you can see the project has the appropriate reference to the SnowflakeMetric control. Snowflake Metric control design component displays the list of outstanding tasks that you have to complete before the page is ready for use. We will go through each one of the tasks.

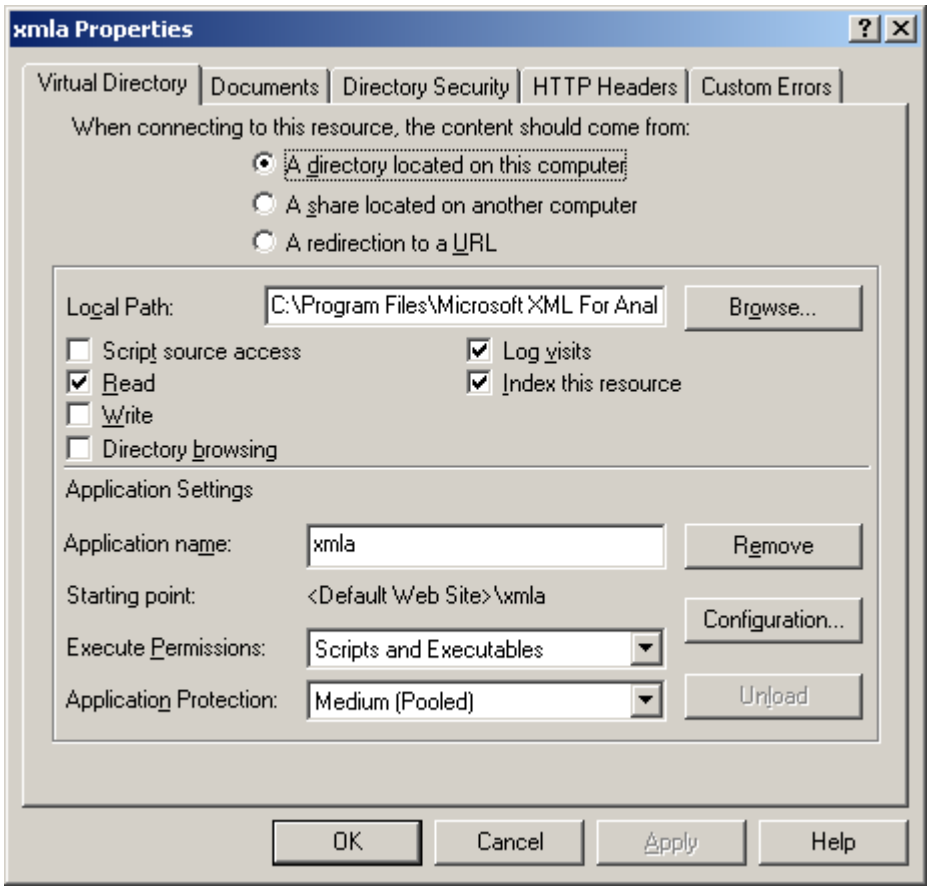
Setting Component Properties

Select the control by clicking on it. The Property pages will display properties that you can set for the control. Scroll the properties until you can see the Data properties.

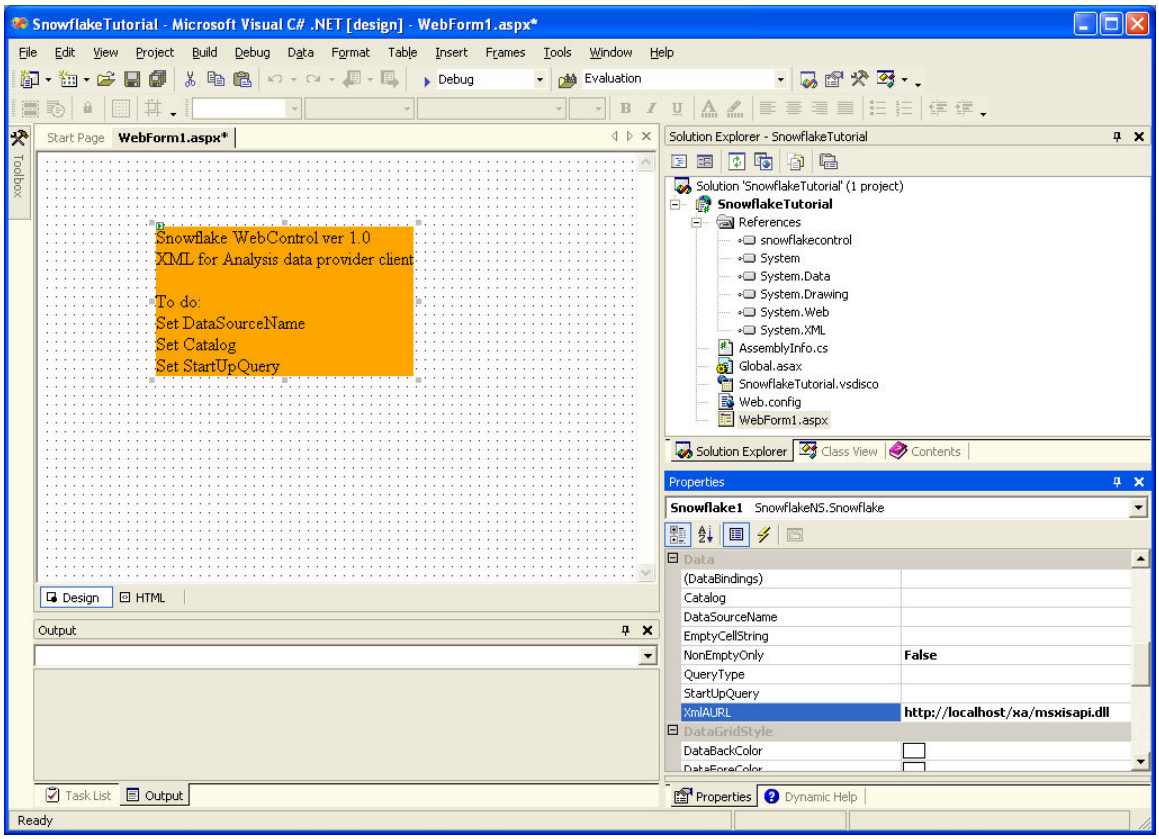


Setting the XML For Analysis Provider

In this section we are assuming that you have installed XML for Analysis provider available from Microsoft. The provider will by default install in you Program Files directory under Microsoft XML for Analysis SDK. The provider is activated as and isapi extension. You have to add the reference to the msxisapi.dll to the XmlAURL property of the control. To locate the appropriate XmlAURL make sure you follow the XML for Analysis SDK instructions and create a web site that refers to the correct directory. This is how the configuration looks on our machine:



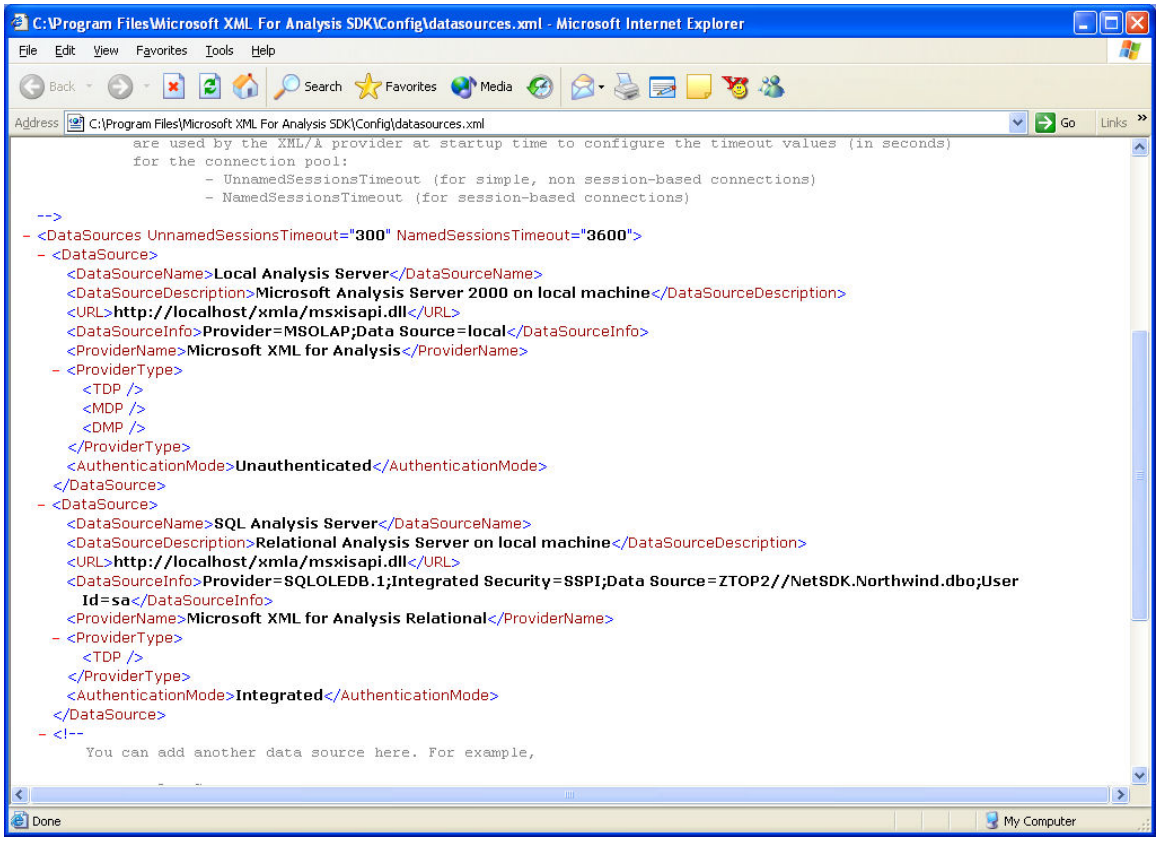
We have used the default name for the virtual directory for the XML for Analysis – xmla and our XMLA provider is located on a machine with name “moria” and therefore our XmlAURL is http://moria/xmla/msxisapi.dll. If you have installed the XMLA SDK on your local machine using default settings then the XmlAURL would be http://localhost/xmla/msxisapi.dll



Note that the provider could be located anywhere from your local machine to a machine anywhere on the Internet.

Data Source name and Catalog

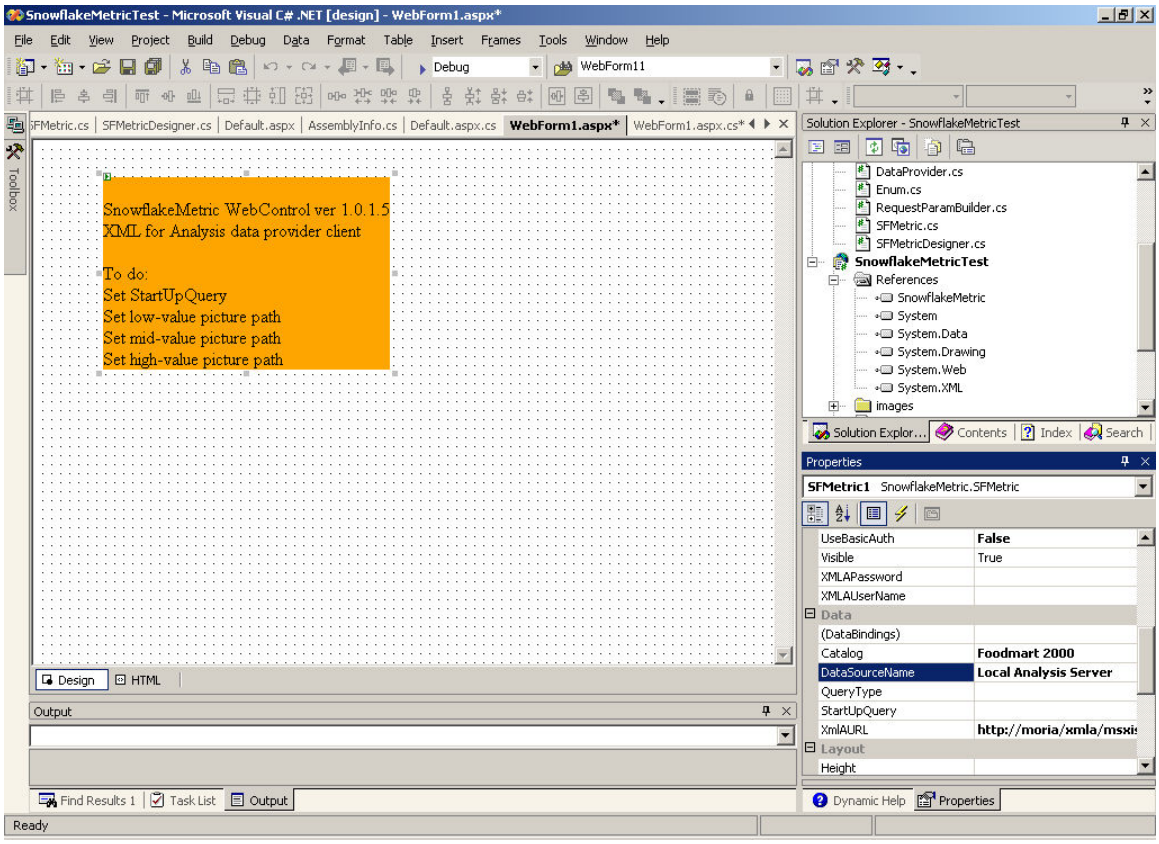
Both the Data Source name and the Catalog properties depend on your XML for Analysis provider configuration. You configure the provider by modifying the datasources.xml file in the configuration directory of the XML for Analysis SDK (in our case located in the C:/Program Files/Microsoft XML for Analysis SDK/Config directory). Here is how our datasources.xml file looks like:



In our case we have defined both a multidimensional and relational data source. Data source that we will use in the tutorial is the Local Analysis Server data source. This data source is using the sample OLAP cube : Foodmart (or Foodmart 2000) that comes with the SQL Server 7.0 and 2000. The provider is MSOLAP and the data source is local. Set the DataSourceName to the data source name that you defined in the datasources.xml (in our case - “Local Analysis Server”) and the Catalogue to the catalogue of your choosing on your Analysis Server (in our case Foodmart).

Note that for the XMLA 1.1 SDK you will need to specify an additional property if you wish to use http as opposed to https through the AllowInsecureTransport element such as this:

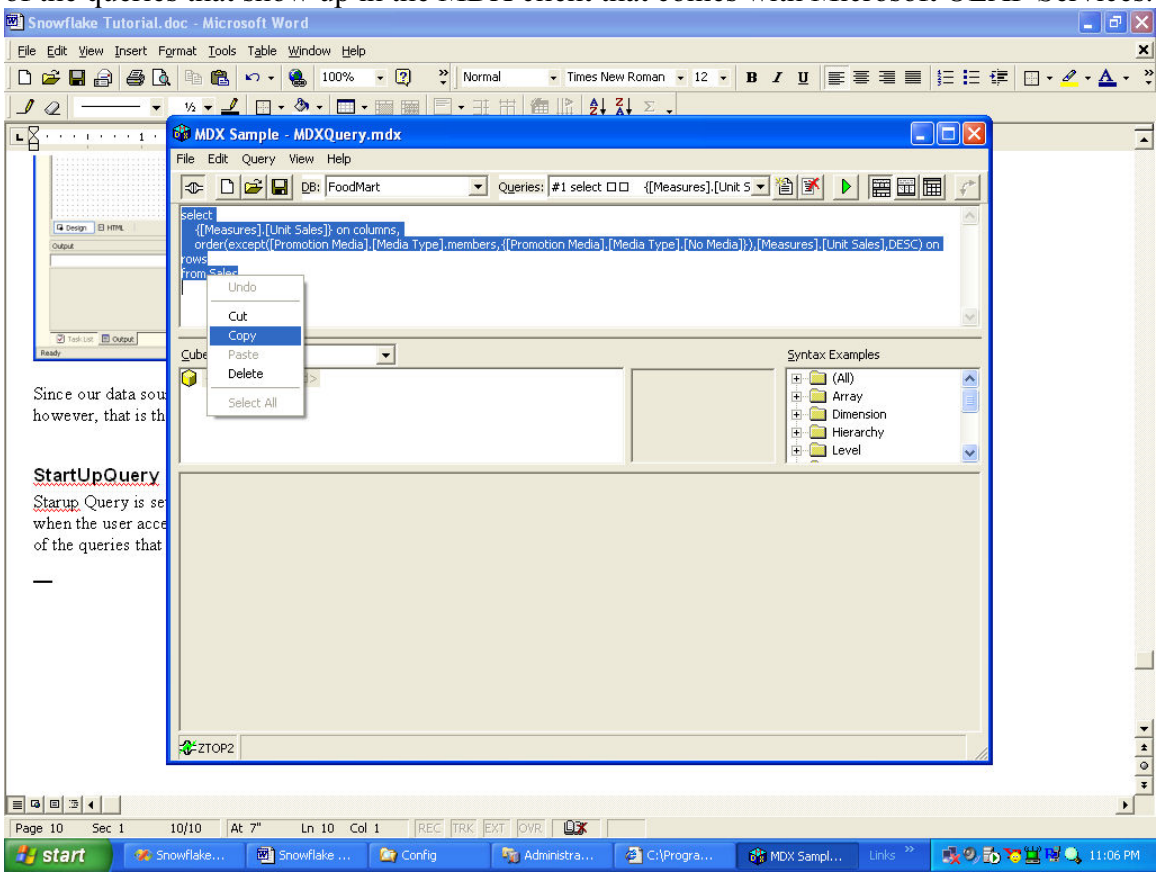
```
- <DataSources UnnamedSessionsTimeout="300"
    NamedSessionsTimeout="3600" AllowInsecureTransport="1">
```



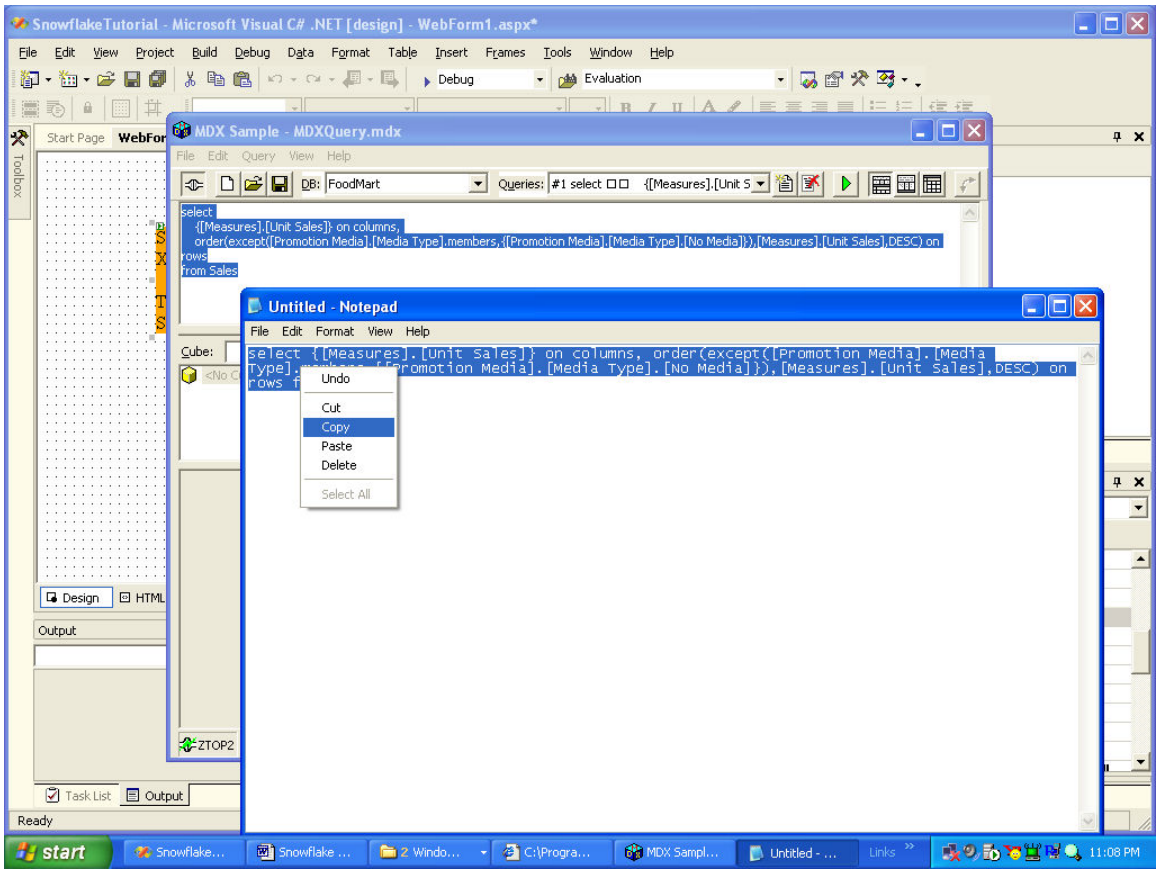
Since our data source is an OLAP data source we can set the Query Type to MDX, however, that is the default anyhow so you do not have to change anything.

StartUpQuery

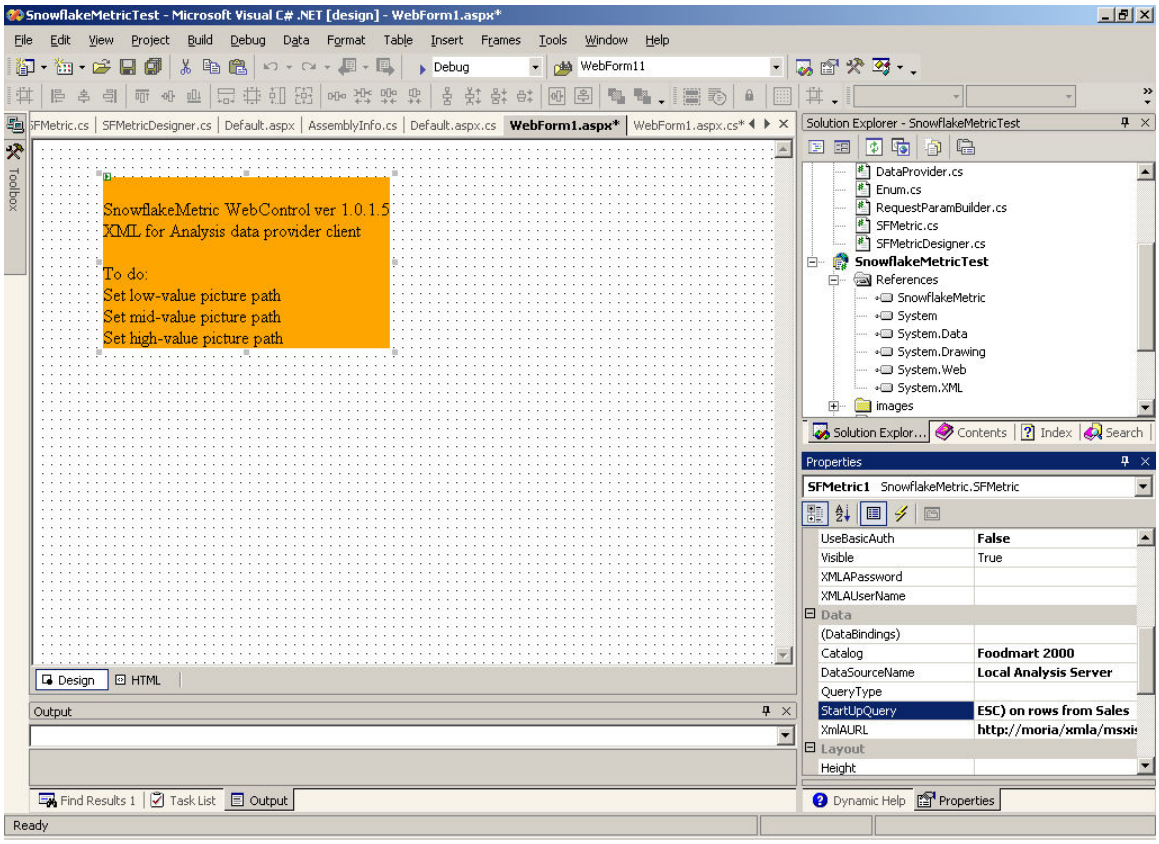
Starup Query is set through the StartUpQuery property. Startup Query will be executed when the user accesses the page. You can pick any valid mdx query. We have chosen one of the queries that show up in the MDX client that comes with Microsoft OLAP Services.



We have copied the query from the MDX Query designer into notepad so that we can remove the carriage returns



and pasted the query into the StartupQuery property

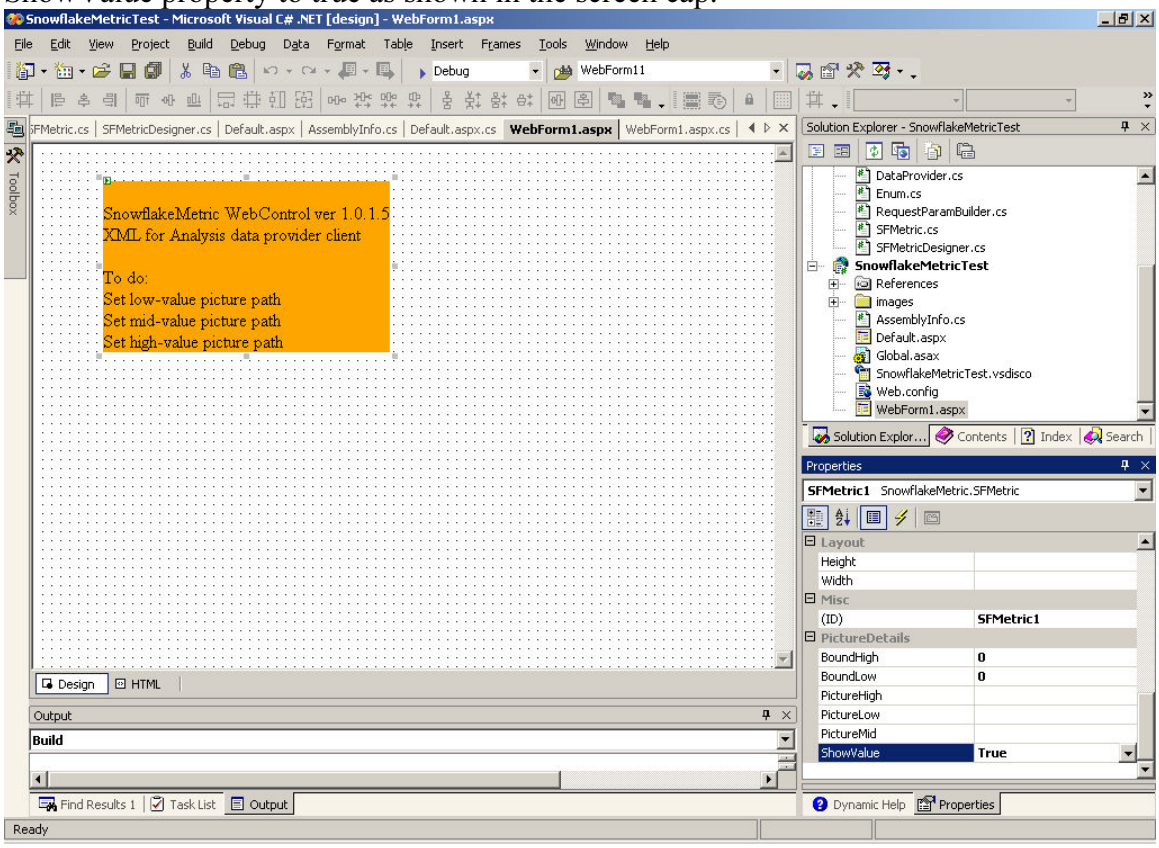


Note that while the query returns a result set that might have any number of values the Snowflake Metric control only looks at the first value in the set to perform the bound comparison and/or return data value.

You are now ready to execute the application and see your first analysis web page.

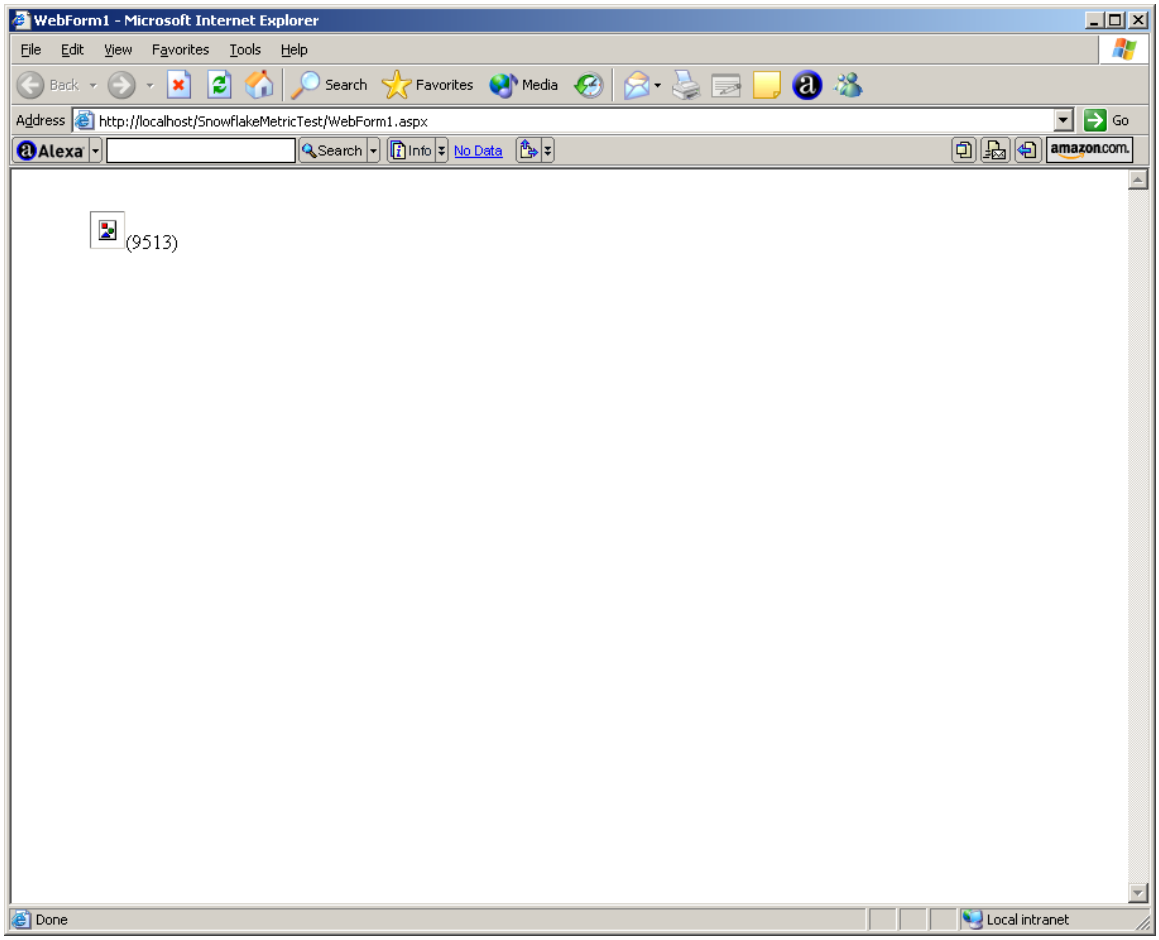
Enabling the ShowValue Property

In order for us to see results of our query we will enable the ShowValue property in PictureDetails property group. Select the Snowflake Metric control in the web page and scroll down in the list of properties until you see the ShowValue property. Set the ShowValue property to true as shown in the screen cap.



Running the App

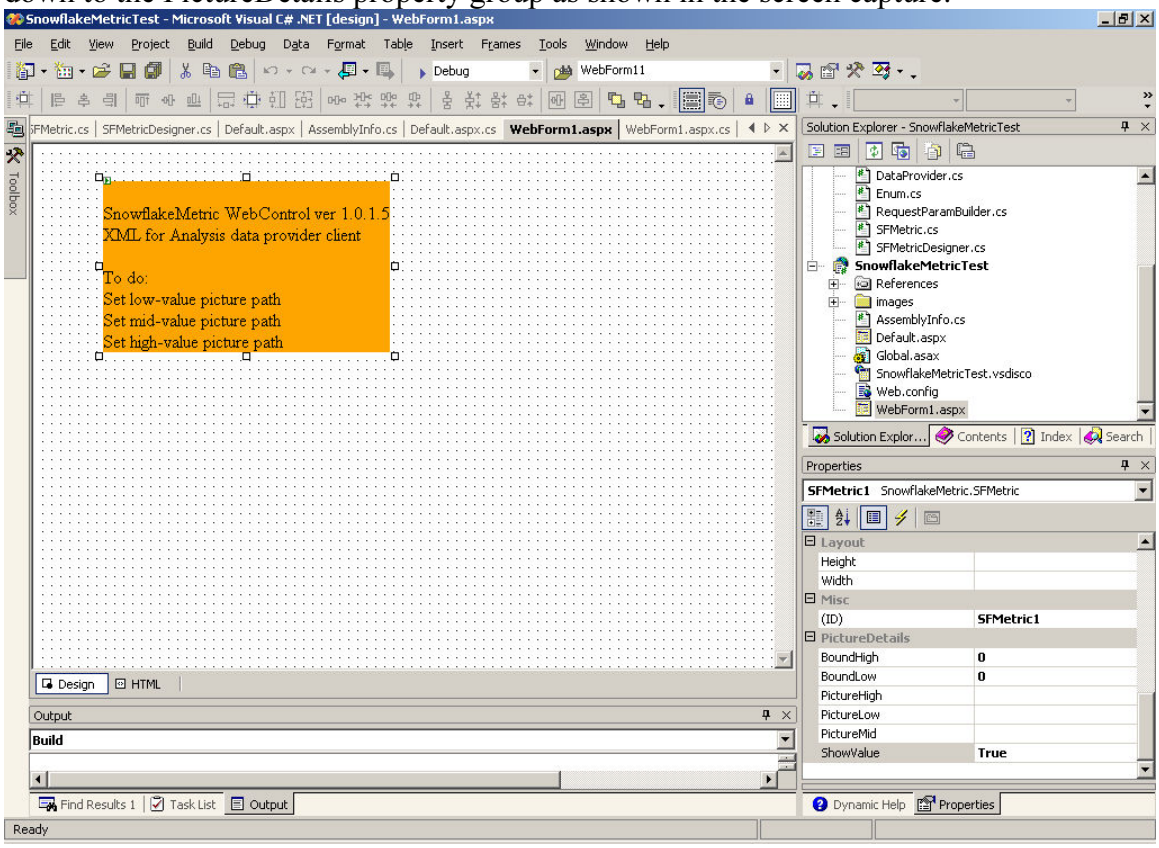
To actually see the results of your work activate the application by either debugging it (F5) or by starting it without debugging (CTRL+F5). A browser window will open and you should see the results similar to what is shown in the following figure.



Granted not much is displayed but you have managed to execute an MDX query against the data source and retrieve the first value in the result-set. The reason why you see a broken image is because we have not set the bounds and image paths for the boundary comparison. We will do that next.

Setting the Snowflake Metric control comparison boundaries

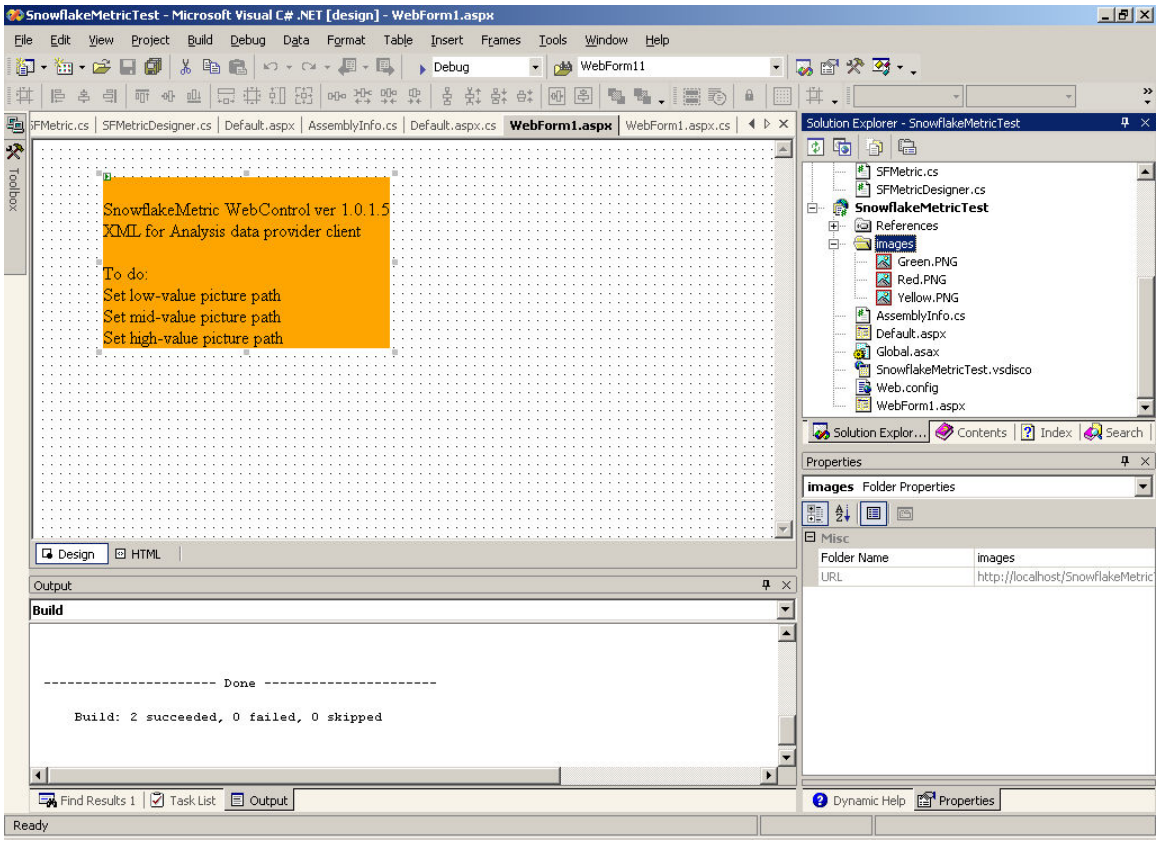
In order for us to receive more interesting results lets set the comparison boundaries for the control. To do this select the Snowflake Metric control in the web page and scroll down to the PictureDetails property group as shown in the screen capture.



We need to set the BoundHigh and BoundLow properties as well as the image paths for the PictureHigh, PictureLow and PictureMid properties.

Lets first add the images to the project.

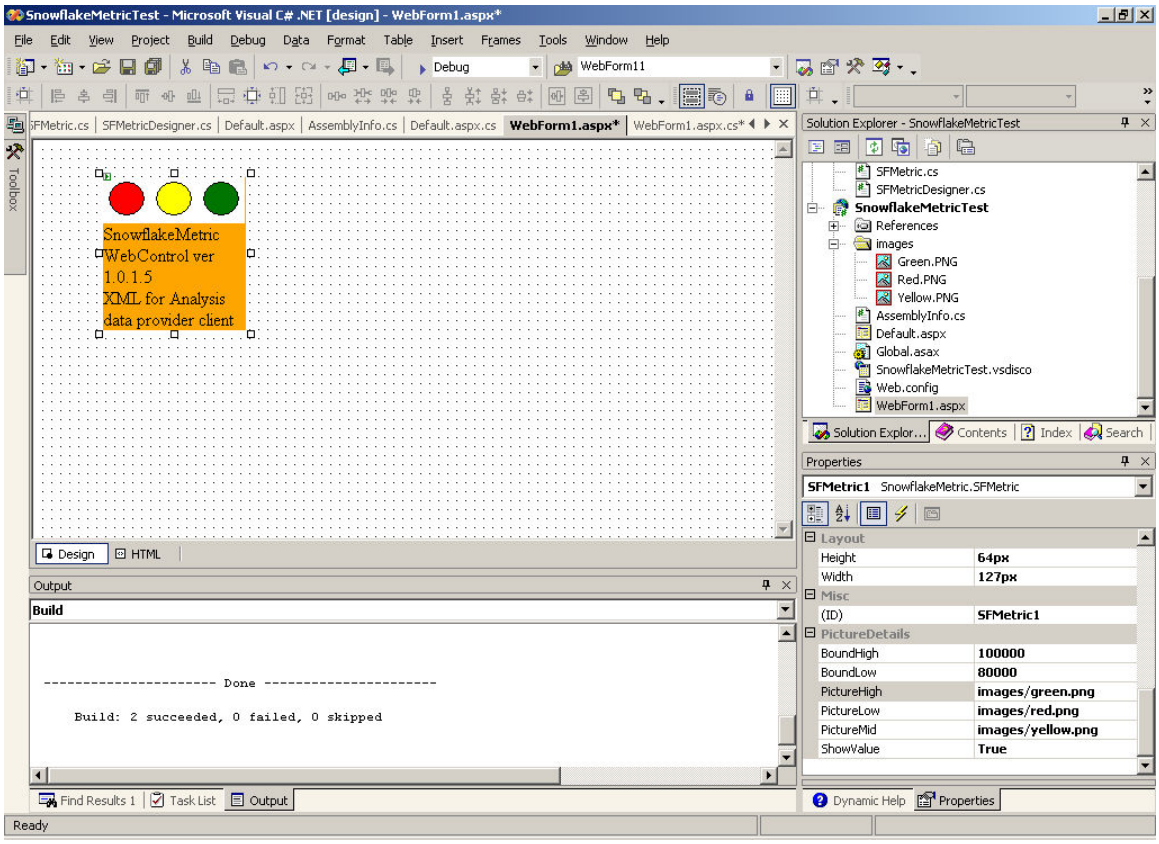
To add images to the project create a subdirectory called images and add 3 images of your choice to that directory by right mouse button clicking on the directory and selecting Add existing item to folder. We have added 3 images: Green.png, Red.png and Yellow.png to the images folder as shown in the screen capture.



With the images added and available to the rest of the project we can now set the bounds and the images in the PictureDetails property group. Set the properties to following values:

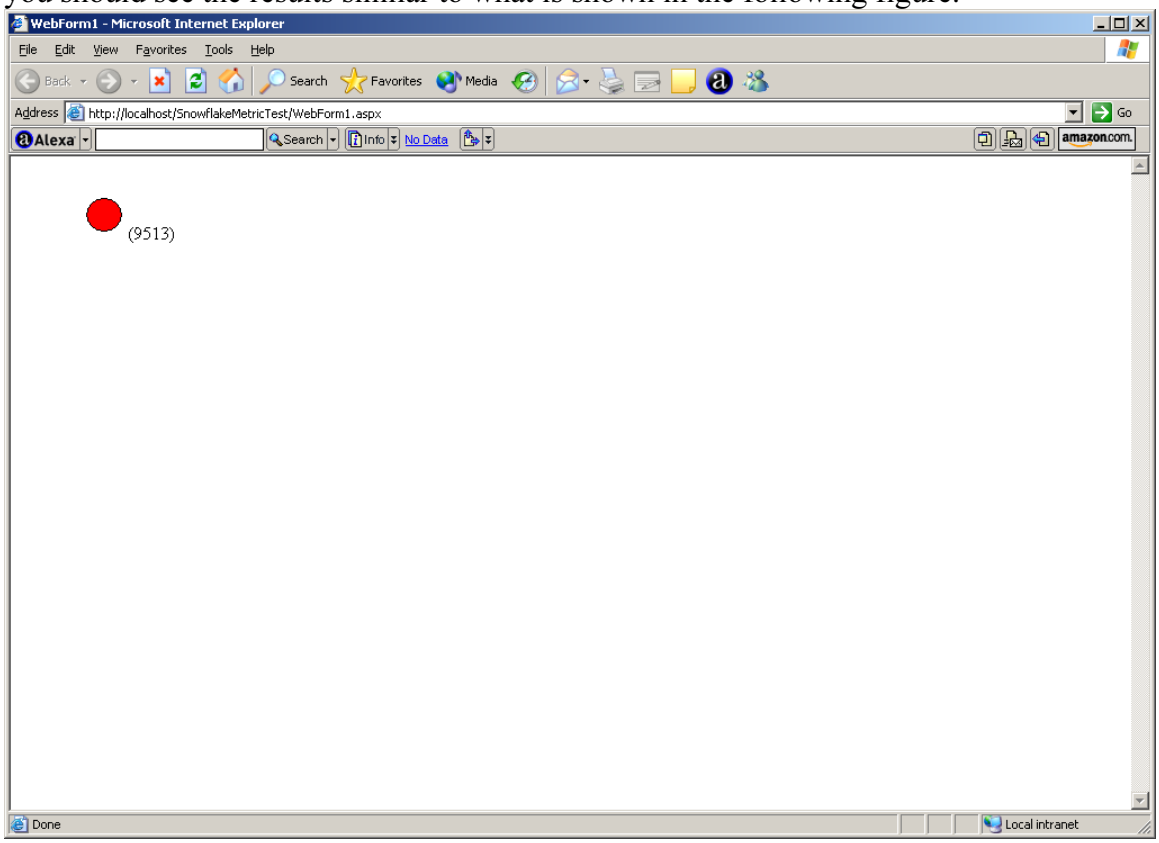
Property	Value
BoundHigh	100000
BoundLow	70000
PictureHigh	Images/Green.png
PictureLow	Images/Red.png
PictureMid	Images/Yellow.png

This is how the values when set look like in VS.net.



Run Application

To actually see the results of your work activate the application by either debugging it (F5) or by starting it without debugging (CTRL+F5). A browser window will open and you should see the results similar to what is shown in the following figure.

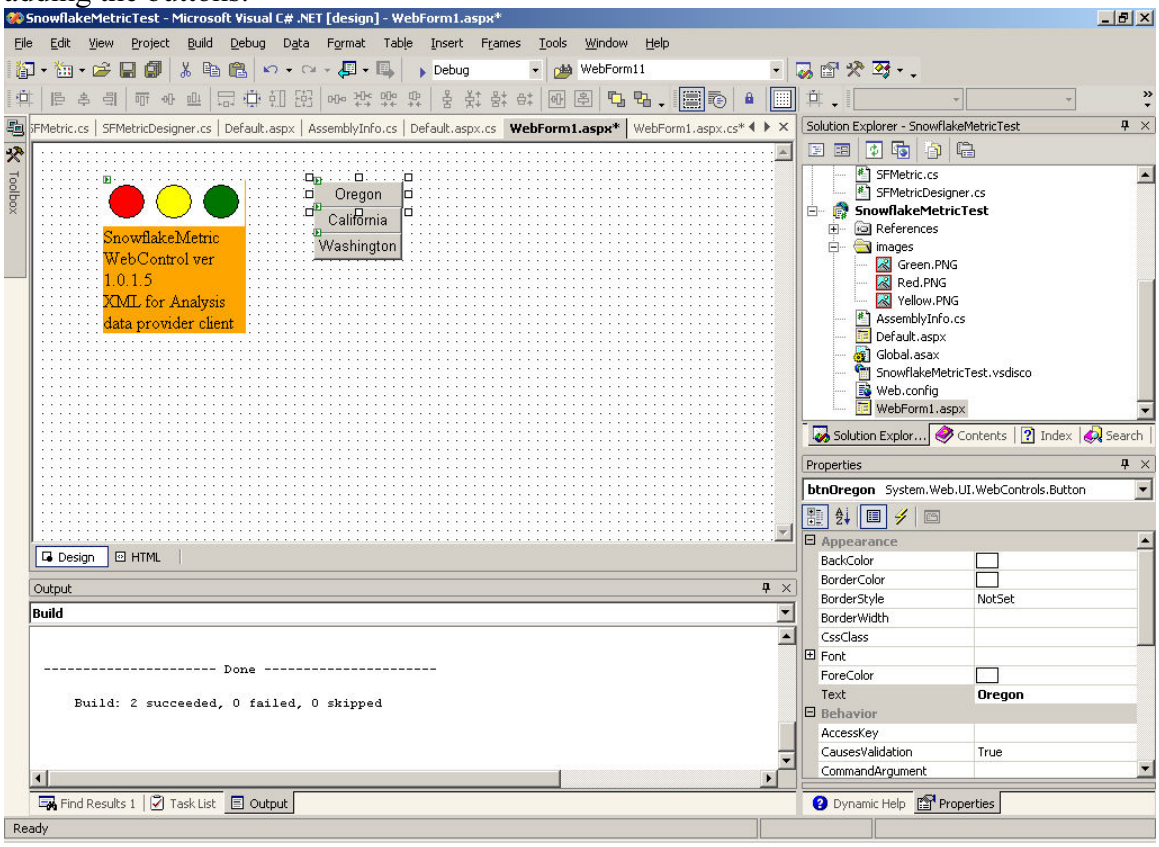


Now, this is slightly better as we have some color showing. As a matter of fact you might be able to do a whole lot in your BI portal by just setting the properties that we have shown.

Next we will see how you can make the form more dynamic by adding some buttons and executing different MDX statements and thereby seeing different color displayed.

Changing the MDX Statement Dynamically

In order to dynamically change the query we will add 3 asp buttons to the page. For each one of them give an appropriate ID. Our buttons are called btnOregon, btnCalifornia and btnWashington and have corresponding text on them. Here is how the form looks after adding the buttons.

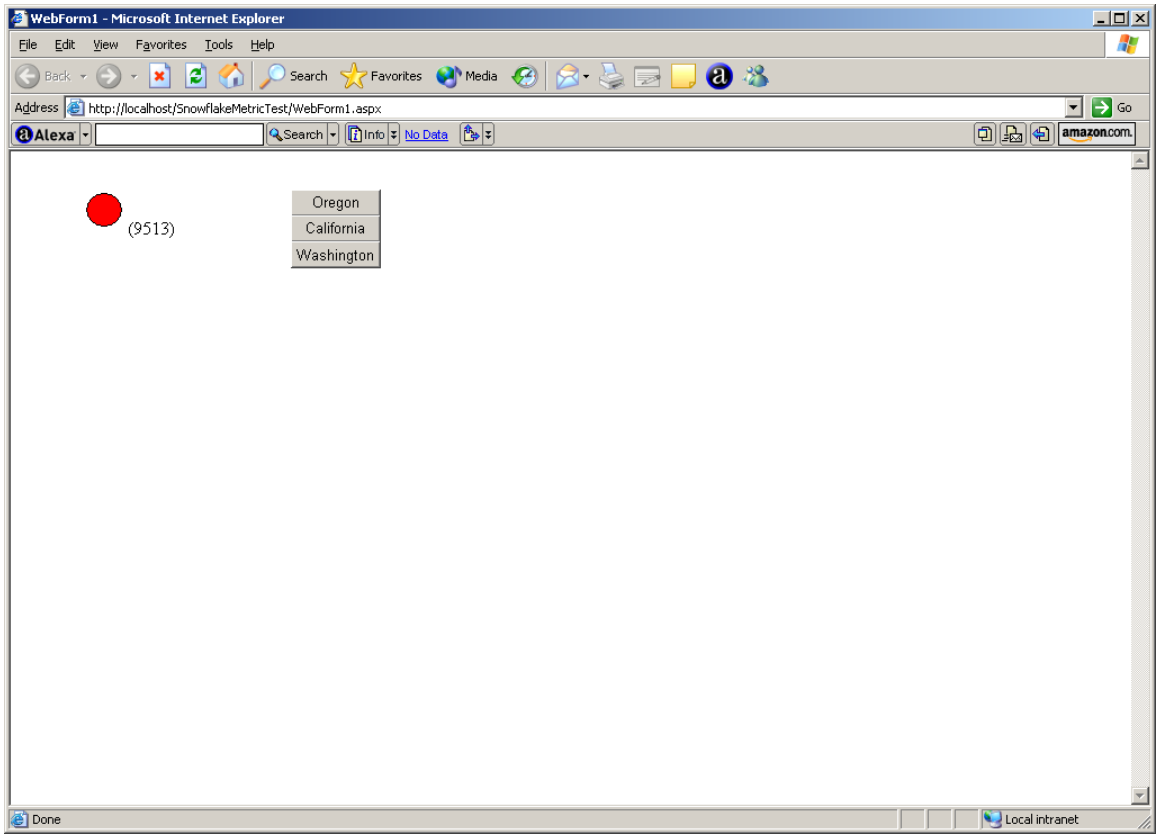


Now lets add event handlers for the clicks on the button. Double click each of the buttons and add the event handling procedures so that the code looks like this:

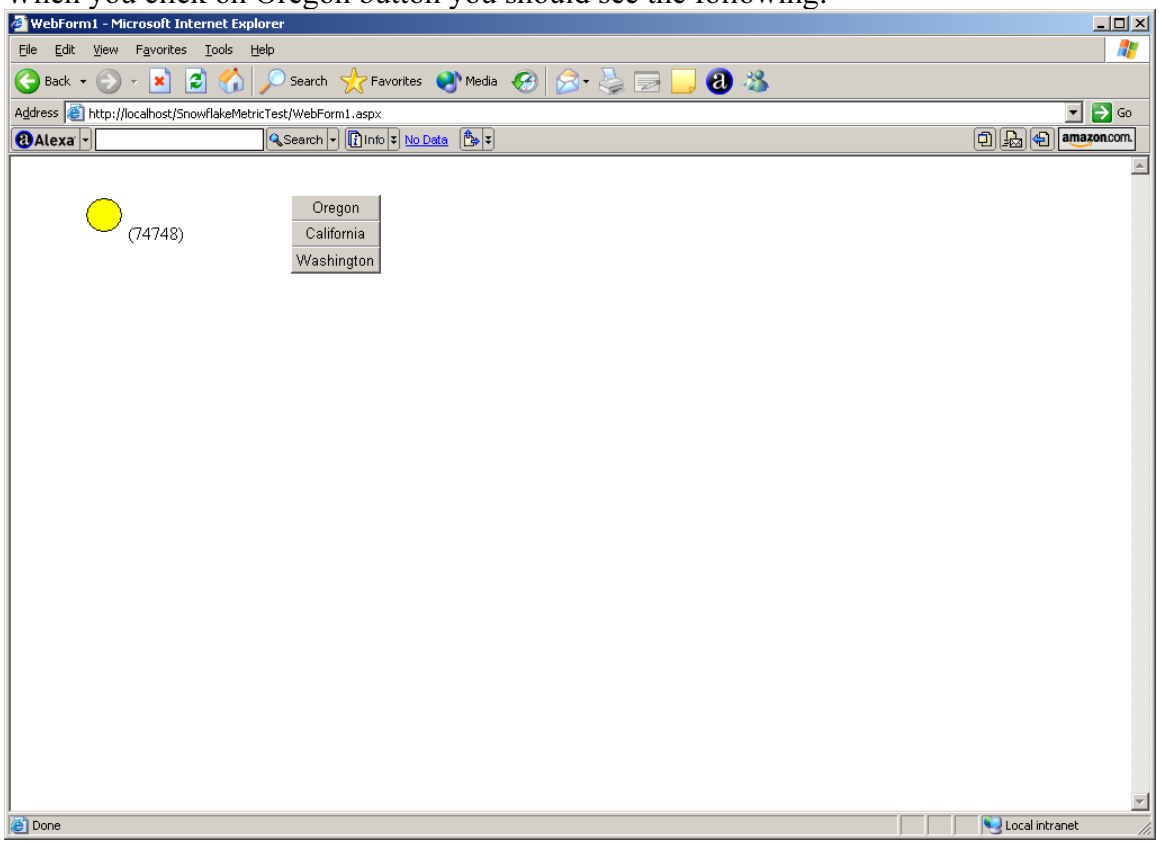
```
private void btnOregon_Click(object sender, System.EventArgs e)
{
      SFMetric1.StartUpQuery = "SELECT FROM [SALES] WHERE
([Customers].[All Customers].[USA].[CA])";
}
private void btnCalifornia_Click(object sender, System.EventArgs e)
{
      SFMetric1.StartUpQuery = "SELECT FROM [SALES] WHERE
([Customers].[All Customers].[USA].[OR])";
}
private void btnWashington_Click(object sender, System.EventArgs e)
{
      SFMetric1.StartUpQuery = "SELECT FROM [SALES] WHERE
([Customers].[All Customers].[USA].[WA])";
}
```

The code simply changes the StartUpQuery property on control instance. Note that we assumed in this example that the Snowflake Metric control ID on the web page is SFMetric1.

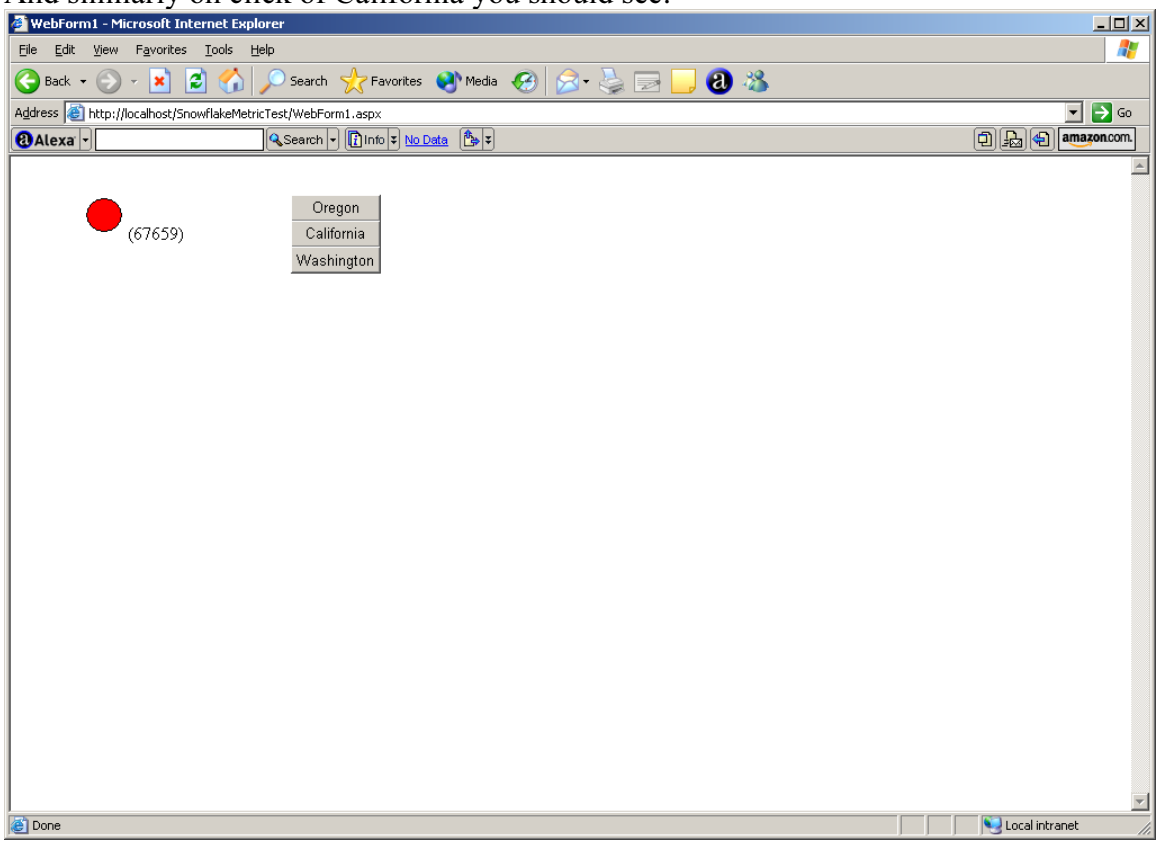
Now run the application by pressing F5 or CTRL+F5. You should see something like this:



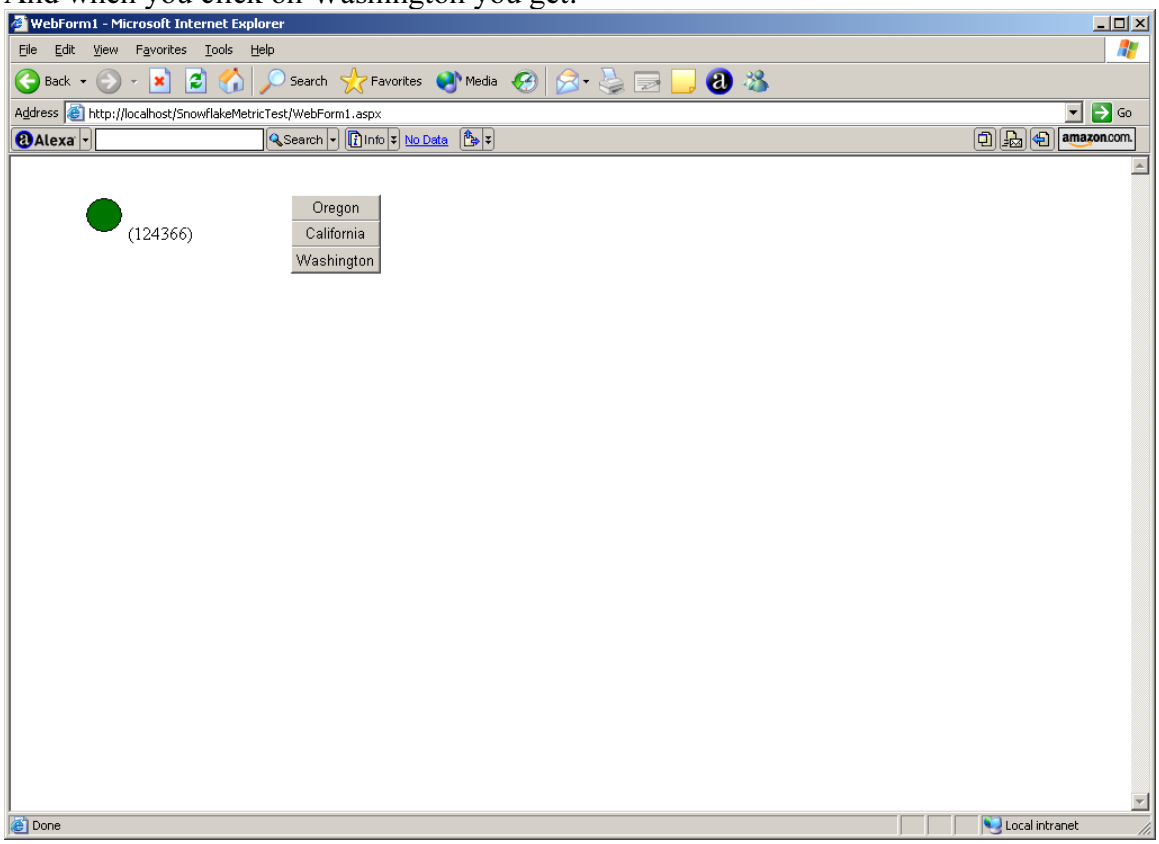
When you click on Oregon button you should see the following:



And similarly on click of California you should see:



And when you click on Washington you get:

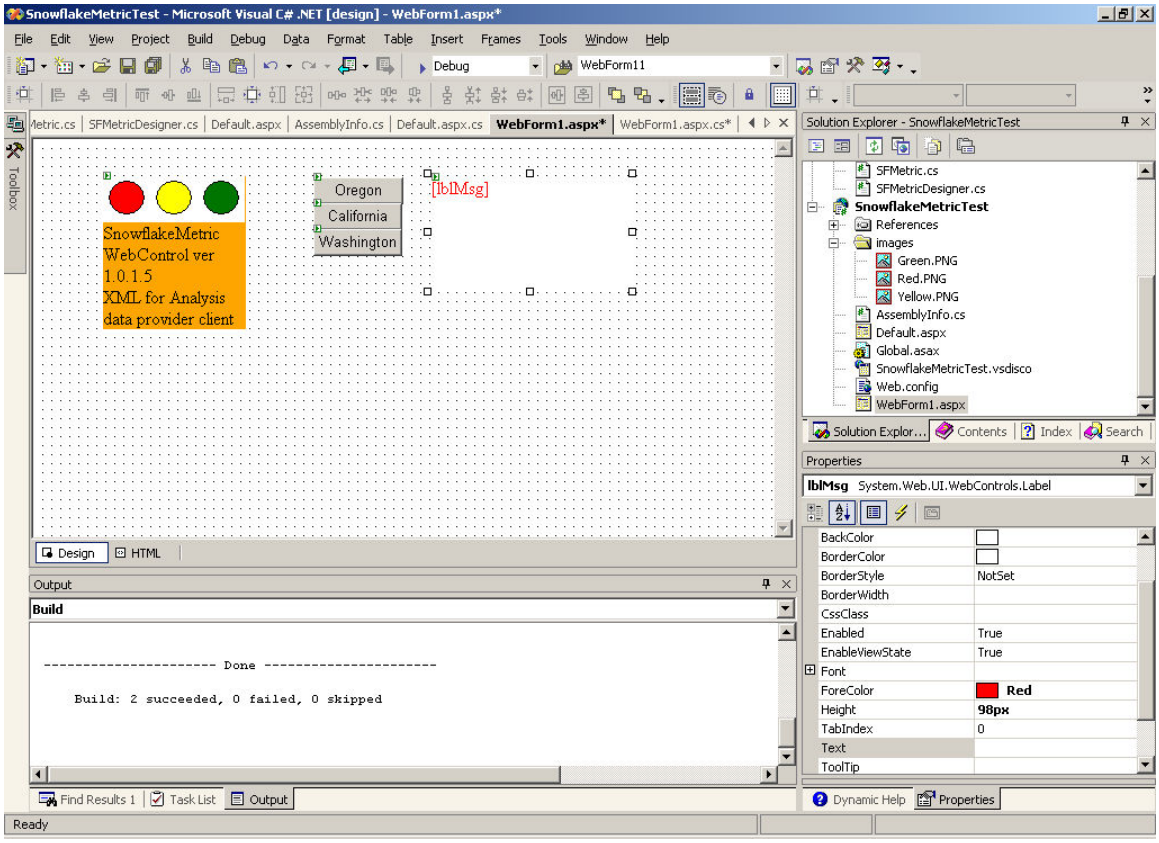


Therefore we have achieved the goal of dynamically changing the image depending on the MDX statement values. In next section we will see alternate means of modifying elements on the form depending on the values returned from the MDX queries activated by the button click.

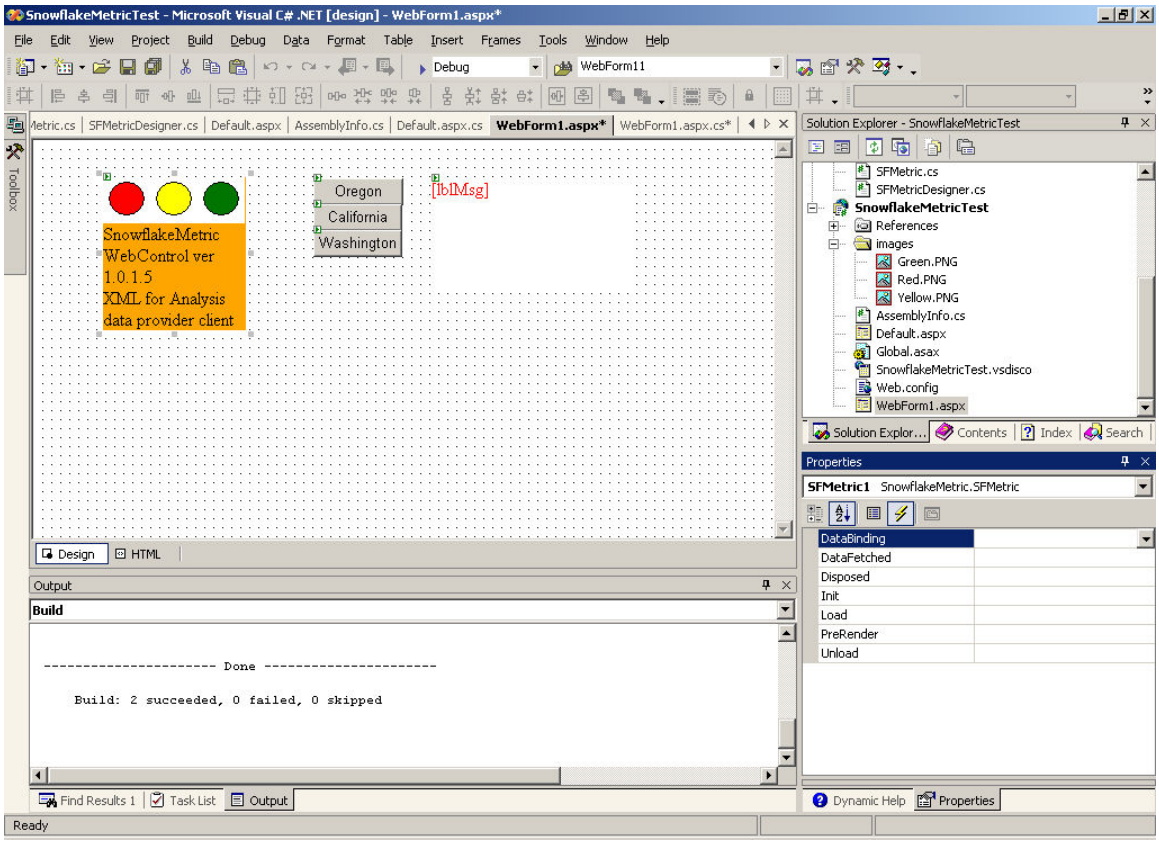
Handling the DataFetched Event

In this section we will see how to handle data handling events of Snowflake Metric control and how to modify form elements based on the query results.

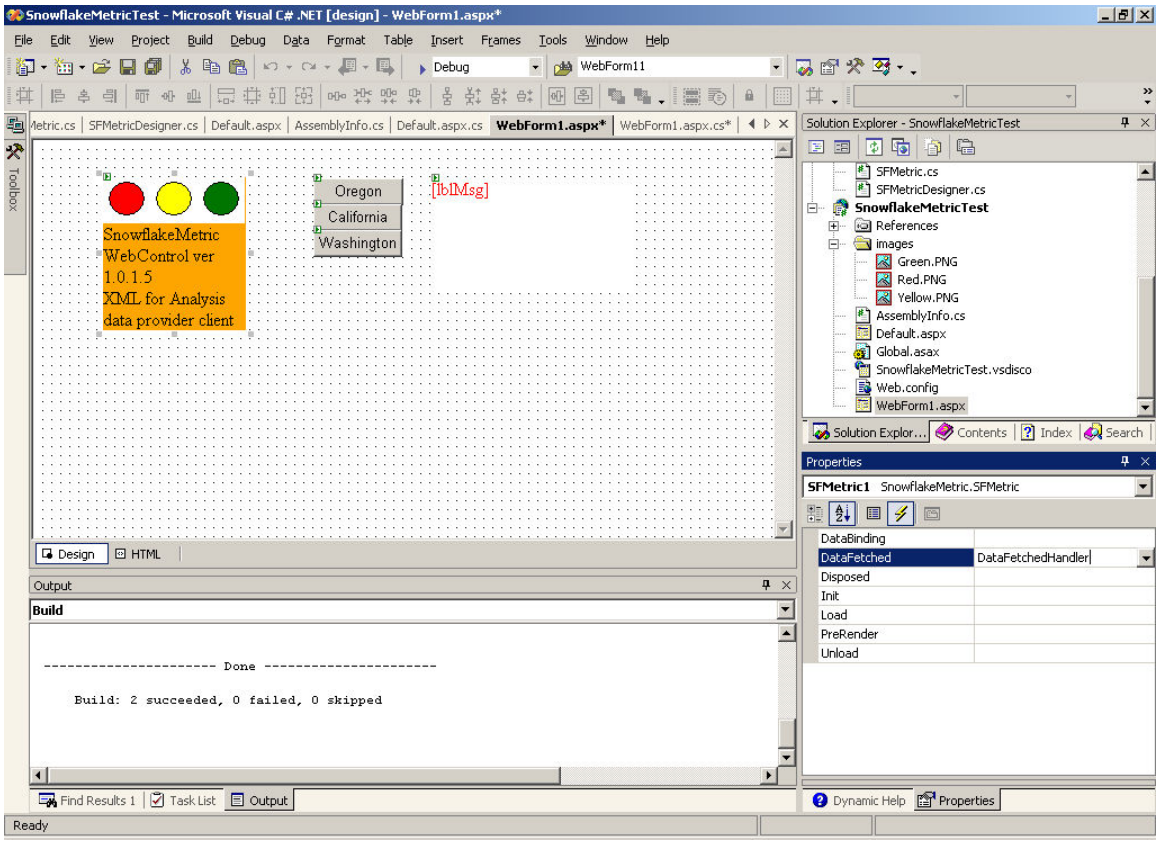
First we will add a label to the page so that we can display messages that will be set in the event handler. Add an asp.net label to the Web form and name it lblMsg. Make the font color red and clear the Text property. You should have a form that looks something like this:



We will now wire a function to handle the DataFetched event. DataFetched event is triggered prior to the rendering of the control. You could even set the control visibility to false so that the user does not see anything from the control in which case you are using the Snowflake Metric control as a pure back-end control that provides execution of the MDX statement. To wire the DataFetched event select the instance of the Snowflake Metric control in the web page and select the events (lightning sign) in the properties tab. You should see something like this:



Type in DataFetchedHandler in the DataFetched event slot as shown in the screen cap:

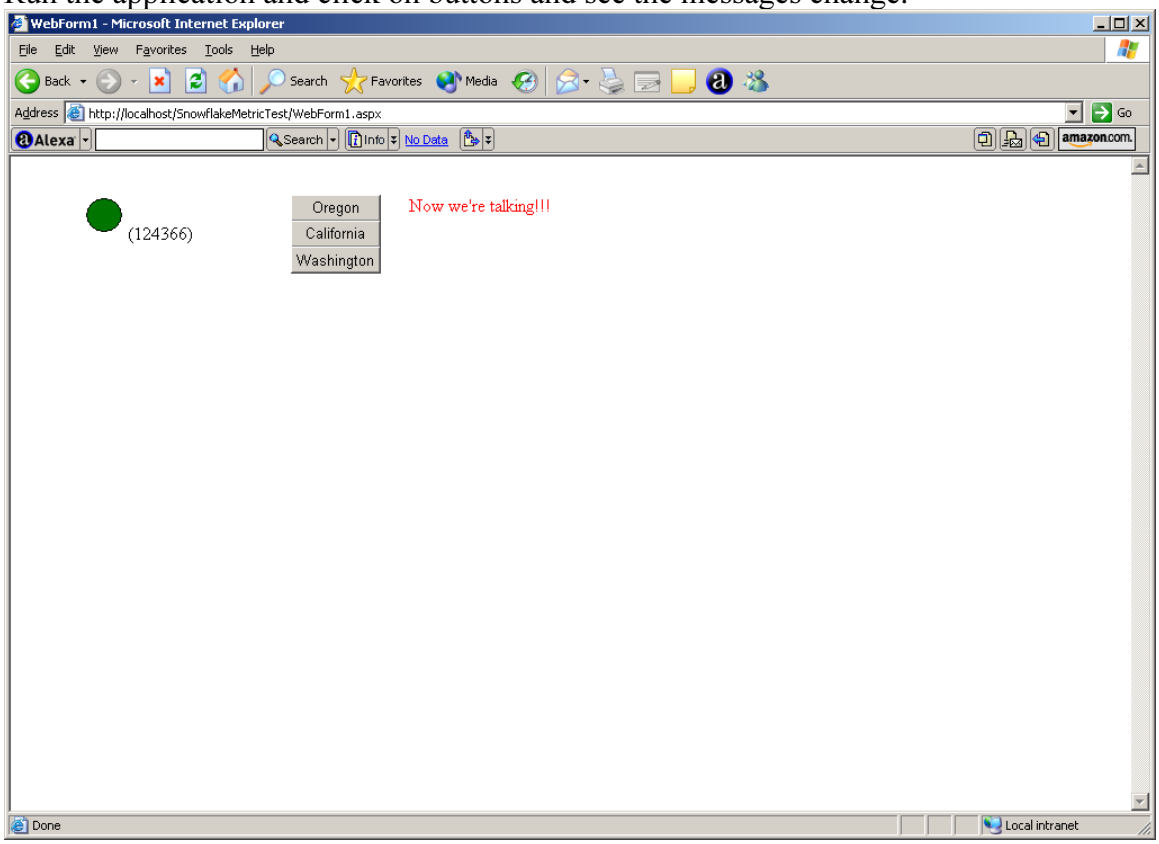


Visual Studio will wire the event and create a function entry in the code called DataFetchedHandler. Key in the following code in the function:

```
private void DataFetchedHandler(object sender,
SnowflakeMetric.SFMetric.DataFetchedEventArgs e)
{
      // this is where you can get the data prior to the display
      // you can even not display anything at all by setting the
      control to invisible
      if (e.dataValue > 100000)
            lblMsg.Text = "Now we're talking!!!";
      else if (e.dataValue > 70000)
            lblMsg.Text = "Not so bad!";
      else
            lblMsg.Text = "That is really low!!!";
}
```

The handler will therefore display different messages depending on the values that we receive from the dataValue member of the event argument.

Run the application and click on buttons and see the messages change.



You can extend this concept to do anything that you might wish inside the web form as now you have means of programmatically modifying form elements depending on the value returned from the MDX query.

Appendix A - Requirements

1. Server requirements:
    -Windows 2000 Server, Windows XP, Windows NT Server with SP6
    -IIS installed
    -.Net Framework installed
    -Access to web service implementing XML for Analysis data provider
     (Snowflake control is tested with Microsoft XML/A provider 1.0)

2. Client browser requirements:
    -IE >= 5.0, Netscape Navigator >=6

3. Web application requirements:
    -ViewState enabled

4. Installation note:
    -Set up a XML for Analysis provider properly.
     If you use Micorsoft XML/A provider, test it with demo applications from XML/A SDK.

Appendix B - Authentication & authorizations scenarios

UNAUTHENTICATED

- Set XML/A virtual directory security to Anonymous access
- Data sources (OLAP cubes, SQL databases...) must have public access
- web.config should have impersonate="false";

INTEGRATED WINDOWS with users credentials

- Set XML/A virtual directory security to Integrated Windows (dissable Anonymous access)
- Set XML/A datasources (XAfolder\config\dataSources.xml) for integrated security. For example:
  <DataSourceInfo>"Provider=SQLOLEDB.1;Integrated Security=SSPI;Data Source=MyServer"</DataSourceInfo>
- Set your application virtual directory security to Integrated Windows (dissable Anonymous access)
- Insert in your application's Web.config file:
  <identity impersonate="true"/>

INTEGRATED WINDOWS with specific credentials

- Set XML/A virtual directory security to Integrated Windows (dissable Anonymous access)
- Set XML/A datasources for integrated security
- Insert in your application's Web.config file:
  <identity impersonate="true" userName="MyDomain\MyUser" password="MyPass" />

Appendix C - List of Snowflake Metric Properties and Events

Properties

Appearance:

BackColor	- background color for the whole control
BorderColor	- border color for the whole control
BorderStyle	- border style for the whole control. One of: • Not Set • None • Dotted • Dashed • Solid • Double • Groove • Ridge • Inset • Outset
BorderWidth	- in pixels
CssClass	- inherited, not used
Font	- general font applicable to the control
ForeColor	- general font color applicable to the control

Behavior

AccessKey	- not used
Enabled	- enables the control. True by default.
EnableViewState	- enables the view state tracking. True by default.
TabIndex	- tab index on the web page
ToolTip	- tool tip for the whole control
UseBasicAuth	- should the basic authentication be used
Visible	- sets the control visibility
XMLAPassword	- fixed password for use with Basic Authentication
XMLAUserName	- fixed user id for use with Basic Authentication

Data

DataBindings	- leave empty
Catalog	- Name of catalog/database for query execution. Example: Foodmart 2000
DataSourceName	- Set to an existent XA data source name, as defined in {Xafolder}\config\dataSources.xml. Example: Local Analysis Server
QueryType	- Set to MDX or SQL.Default is MDX
StartUpQuery	- MDX or SQL Select sentence to be executed on startup

XmlAURL	- URL to Xml for Analysis provider. Default is http://localhost/XmlA/msxisapi.dll

Layout

Height	- height of the control
HorizontalAlign	- alignment of the control with respect to the page
Width	- width of the control
Wrap	- wrap if there is not enough space

Misc

Id	- id of the control on the web page
CurrentQuery	- set the XMLA query at design or run-time

PictureDetails

BoundHigh	- threshold value that determines what is considered high
BoundLow	- threshold value that determines what is considered low
PictureHigh	- image to be used for high value
PictureMid	- image to be used for medium value
PictureLow	- image to be used for low value

Events

DataFetched	- data fetched event provides the value that was retrieved upon execution prior to the control being rendered. Event argument has dataValue property that is used to retrieve the value.