Database Example to SQL Server Express

Introduction

This exercise shows some of the benefits of tiered architecture while showing how to programmatically connect to a SQLServer database without using database controls.

The exercise assumes you have installed SQL Server Express. My installation of SQL Server Express was included in Visual Studio 2005. What we are going to do in this exercise is change a working but badly programmed example into an improved example with respect to programming practices.

First, we are going to write this code badly, tightly coupled with too much logic in the DBclass (Model). This is the habit of new programmers. The badly programmed solution will be a working solution; however, the DB class is only good inside of this one example. It will not function correctly if instantiated in any other class.

Second, we will take the class and refactor it into a better example of a two tiered architecture.

You will see the code for how to use SQLClient class to operate with SQL Server Express. The items demonstrated are:

1. **Connecting to SQL Server Express**
2. **Creating a table in an existing database called cosc444**
3. **Inserting rows into the Sample Table**
4. **Selecting from the newly created table**
5. **Deleting the table**
6. **closing the connection**

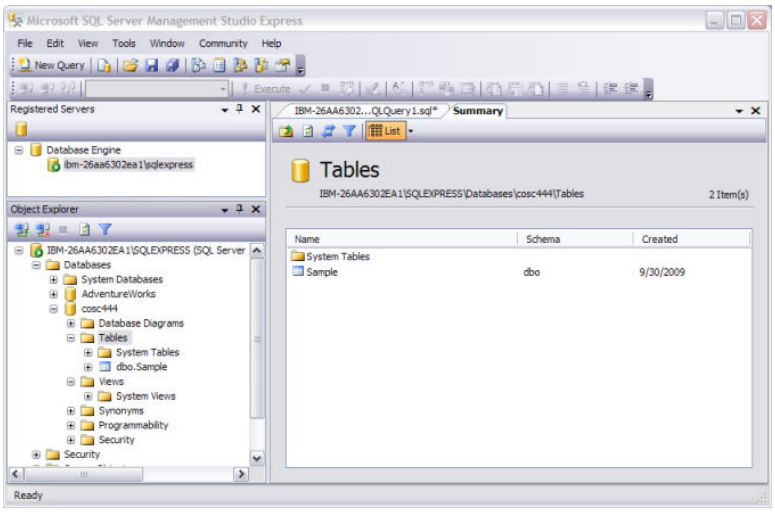
SQL Server Management Studio Express

First you will need to install SQL Server Management Studio Express from the following URL: http://www.microsoft.com/downloadS/details.aspx?familyid=C243A5AE-4BD1-4E3D-94B8-5A0F62BF7796&displaylang=en

Once installed and connected to your SQL Server Express, the management tool will appear similar to this:

Database Example without any database controls to SQLServer Express

COSC 444



Right click on databases in the left pain and choose → New
Name your new database cosc444 and you are ready to continue with visual basic.

There are 3 Panes:

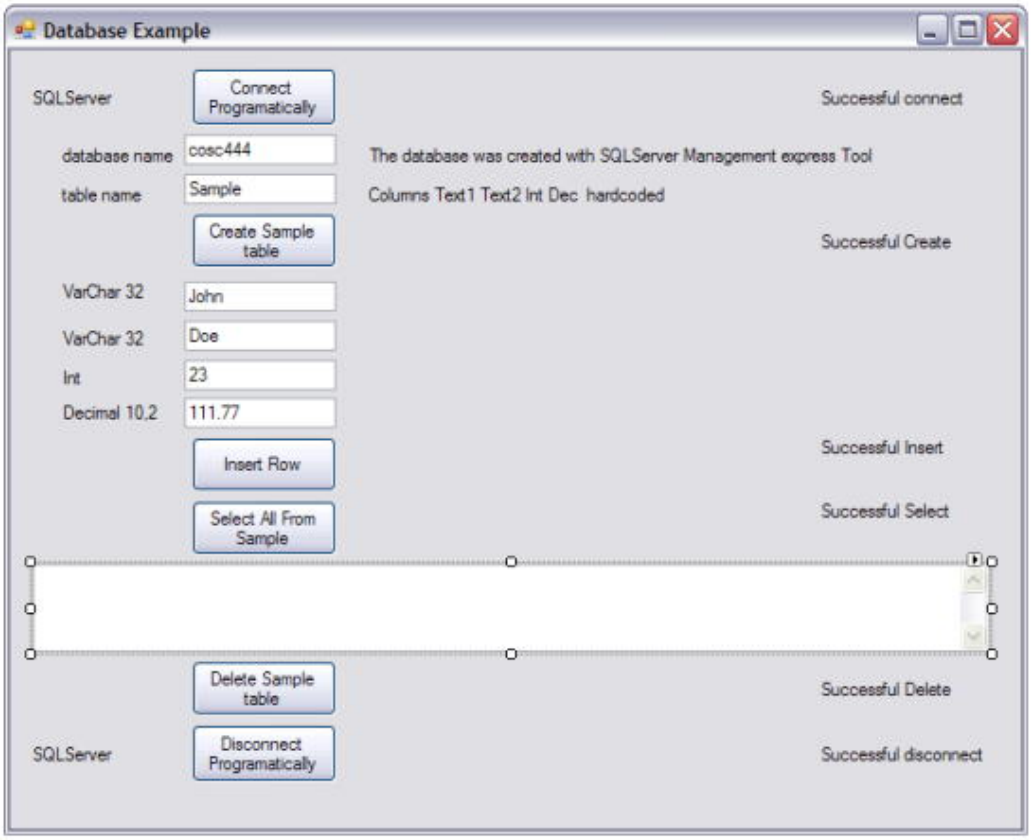
- **Left Top is the server name for SQLServer Express**
- **Left Bottom lists the artifacts in you DBMS**
- **Right gives details on the highlighted artifact**

Back to Visual Studio 2005 design your form

We are well into the semester so I am not going to go into every detail of putting all the controls on the example form below.

Start Visual Studio 2005 and create a new windows project, I called mine databaseExample.

Make the default form look like the following. Pay attention to the names of the controls in the code that follows the screen shot.



Database Example without any database controls to SQLServer Express

COSC 444

You will need:
7 Text Boxes / 16 labels / 6 buttons If you choose to use all of the labels that are not affected by the programming.

Control Name	Text Property
txtDatabase	cosc444
txtTable	sample
txtVarChar1	john
txtVarChar2	doe
txtInt1	23
txtDecimal1	111.77
txtSelectResult	
lblSuccessfulSelect	Successful Select
lblFromCode	Successful Connect
lblDisFromCode	Successful Disconnect
lblSuccessfulCreate	Successful Create
lblSuccessfulDelete	Successful Delete
lblSuccessfulInsert	Successful Insert
lblSQLServerConnect	#other labels not changed at runtime#
lblDatabase	
lblTable	
lblVarChar1	
lblVarChar2	
lblInt1	
lblDecimal1	
lblSQLServerDisConnect	
lblDBDescription	
lblTableDescription	
btnConnFromCode	Connect Programmatically
btnDisConnFromCode	Disconnect Programmatically
btnCreateTable	Create Table
btnInsertRow	Insert Row
btnSelectAll	Select All
btnDeleteTable	Delete table

Steps to get this first example to run:

1. Determine the server name for your SQLServer Express from the management tool pictured on page 1
2. Change line 7 below to match your server name
3. Create the database cosc444 in your SQLServer express
4. Design the form according to the table and image above
5. copy/paste the frmDBExample class into your forms code editor

The class frmDBExample is on the next page of this document lines numbered 1-36.

6. In visual studio choose Project → add class. Name your class DB
7. copy/pase lines 37-180 into the DB class in the code editor.

8. look for errors in the display at the bottom of visual studio and fix any naming problems from when you were designing your form
9. Execute the program by pressing the start debugging button

DISCLAIMER
THERE ISN'T MUCH ERROR CHECKING IN THIS PROGRAM, THIS PROGRAM IS TO SHOW DATABASE OPERATIONS. IF YOU TRY TO DELETE THE TABLE WITH HAVING FIRST CREATED THE TABLE THE PROGRAM WILL FAIL. USE A LOGICAL ORDER WHEN RUNNING THE PROGRAM OR YOU WILL HAVE TO STOP AND RESTART THE DEBUGGING PROCESS IN VISUAL STUDIO

The code for our first pass (too tightly coupled)

```
1   Public Class frmDBExample
2       Dim db As DB
3
4       Private Sub btnFromCode_Click(ByVal sender As System.Object, ByVal
5   e As System.EventArgs) Handles btnConnFromCode.Click
6           db = New DB()
7           db.MyDataSource = "IBM-26AA6302EA1\SQLEXPRESS"
8           db.MyDatabase = "cosc444"
9           db.ConnectToSql()
10      End Sub
11
12      Private Sub btnDisConnFromCode_Click(ByVal sender As System.Object,
13  ByVal e As System.EventArgs) Handles btnDisConnFromCode.Click
14          db.DisconnectFromSql()
15      End Sub
16
17      Private Sub btnCreateTable_Click(ByVal sender As System.Object,
18  ByVal e As System.EventArgs) Handles btnCreateTable.Click
19          db.CreateSampleTable()
20      End Sub
21
22      Private Sub btnInsertRow_Click(ByVal sender As System.Object, ByVal
23  e As System.EventArgs) Handles btnInsertRow.Click
24          db.InsertRow()
25      End Sub
26
27      Private Sub btnSelectAll_Click(ByVal sender As System.Object, ByVal
28  e As System.EventArgs) Handles btnSelectAll.Click
29          txtSelectResult.Text &= db.SelectAll()
30      End Sub
31
32      Private Sub btnDeleteTable_Click(ByVal sender As System.Object,
33  ByVal e As System.EventArgs) Handles btnDeleteTable.Click
34          db.DeleteSampleTable()
35      End Sub
36  End Class
```

```
37  Public Class DB
38      Public SSconn As New SqlClient.SqlConnection
39      Private ds As String
40      Private db As String
41
42      Public Sub New()
43          frmDBExample.lblFromCode.Visible = False
44          frmDBExample.lblDisFromCode.Visible = False
45          frmDBExample.lblSuccessfulCreate.Visible = False
46          frmDBExample.lblSuccessfulDelete.Visible = False
47          frmDBExample.lblSuccessfulInsert.Visible = False
48      End Sub
49
50      Public Property MyDataSource() As String
51          Get
52              Return ds
53          End Get
54          Set(ByVal value As String)
55              ds = value
56          End Set
57      End Property
58
59      Public Property MyDatabase() As String
60          Get
61              Return db
62          End Get
63          Set(ByVal value As String)
64              db = value
65          End Set
66      End Property
67
68      ' Visual Basic
69      Public Sub ConnectToSql()
70          frmDBExample.lblFromCode.Visible = False
71          frmDBExample.lblDisFromCode.Visible = False
72          ' TODO: Modify the connection string and include any
73          ' additional required properties for your database.
74      SSconn.ConnectionString &= _
75      "integrated security=SSPI;data source=" & Me.MyDataSource & ";" & _
76      "persist security info=False;database=" & Me.MyDatabase
77          ' original string
78          ' "integrated security=SSPI;data source=SQL Server Name;" & _
79          ' "persist security info=False;initial catalog=northwind"
80          ' whatis SSPI
81          ' Security Support Provider Interface (SSPI). SSPI allows
82          ' an application to use various security models available
83          ' on a computer or network without changing the interface
84          ' to the security system
85          Try
86              SSconn.Open()
87          Catch ex As Exception
88              frmDBExample.lblFromCode.Text = "Failed Connection"
89              frmDBExample.lblFromCode.Visible = True
90              MessageBox.Show("Failed to connect to data source")
91          Finally
92              frmDBExample.lblFromCode.Visible = True
93          End Try
```

```
94      End Sub
95
96      Public Sub DisconnectFromSql()
97          frmDBExample.lblFromCode.Visible = False
98          frmDBExample.lblDisFromCode.Visible = True
99          frmDBExample.lblSuccessfulCreate.Visible = False
100         frmDBExample.lblSuccessfulDelete.Visible = False
101         frmDBExample.lblSuccessfulInsert.Visible = False
102         frmDBExample.lblSuccessfulSelect.Visible = False
103         SSconn.Close()
104     End Sub
105
106     Public Sub CreateSampleTable()
107         ' example CREATE TABLE Person (LastName varchar,FirstName
108             ' varchar,Address varchar,Age int)
109         Dim catCMD As SqlClient.SqlCommand = New
110         SqlClient.SqlCommand("CREATE TABLE " & _
111         frmDBExample.txtTable.Text & " (Text1 varchar(32),Text2
112             varchar(32),myint int,mydecimal decimal(10,2))", SSconn)
113         catCMD.ExecuteNonQuery()
114         frmDBExample.lblSuccessfulCreate.Visible = True
115     End Sub
116
117     Public Sub InsertRow()
118     ' example INSERT INTO Persons VALUES('Hussein', 'Saddam', 'White
119        ' House')
120         Dim catCMD As SqlClient.SqlCommand = New
121             SqlClient.SqlCommand("INSERT INTO " & _
122             frmDBExample.txtTable.Text & " VALUES('" & _
123             frmDBExample.txtVarChar1.Text & "','" & _
124             frmDBExample.txtVarChar2.Text & "','" & _
125             frmDBExample.txtInt.Text & "','" & _
126             frmDBExample.txtDecimal.Text & "')", SSconn)
127         catCMD.ExecuteNonQuery()
128         frmDBExample.lblSuccessfulInsert.Visible = True
129     End Sub
130
131     Public Function SelectAll() As String
132         ' example SELECT * FROM Persons
133         Dim catCMD As SqlClient.SqlCommand = New
134           SqlClient.SqlCommand("SELECT * FROM " &
135              frmDBExample.txtTable.Text, SSconn)
136         Dim reader As SqlClient.SqlDataReader = catCMD.ExecuteReader()
137         Try
138             While reader.Read()
139                 'Console.WriteLine("{0}", reader(0))
140                 frmDBExample.lblSuccessfulSelect.Visible = True
141                 Return reader(0)
142             End While
143         Catch ex As Exception
144             Return ex.Message
145             frmDBExample.lblSuccessfulSelect.Visible = False
146         Finally
147             reader.Close()
148         End Try
149         Return "no data"
150     End Function
```

```
151
152         Public Sub DeleteSampleTable()
153             ' DROP TABLE Person
154             Dim catCMD As SqlClient.SqlCommand = New
155               SqlClient.SqlCommand("DROP TABLE " & _
156                 frmDBExample.txtTable.Text, SSconn)
157             catCMD.ExecuteNonQuery()
158             frmDBExample.lblSuccessfulDelete.Visible = True
159             frmDBExample.lblFromCode.Visible = False
160             frmDBExample.lblDisFromCode.Visible = False
161             frmDBExample.lblSuccessfulCreate.Visible = False
162             frmDBExample.lblSuccessfulInsert.Visible = False
163             frmDBExample.lblSuccessfulSelect.Visible = False
164         End Sub
165         '    Dim SQLQuery As String = "SELECT * FROM " &
166         '           frmDBExample.txtTable.Text
167         '    Dim GetData As New OleDb.OleDbCommand(SQLQuery, SSconn)
168         '    Dim data_reader As OleDb.OleDbDataReader
169         '        data_reader = FindCustomer.ExecuteReader()
170         '        If data_reader.HasRows Then
171         '            While data_reader.Read
172         '                If Not TypeOf data_reader("Text1") Is DBNull Then
173         '                txtSelectResults.Text = CStr(data_reader("Text1"))
174         '                End If
175         '            End While
176         '        Else
177         '            : MsgBox("no Match found")
178         '        End If
179         '        data_reader.Close()
180     End Class
```

Refactoring the current program to make a two tier architecture

If we separate the view from the model, we will be able to use the DB class anywhere. It will have no direct references or knowledge of what is in the frmDBExample class.

Summary of our first pass:

The frmDBExample class

At this point the class does not do all that much
- It sets properties in DB class for server name and database name
- It calls methods in the DB class that do all of the work for creating, inserting, connecting, changing the view display, etc..

The DB class

DB does all the work at this point
Nothing is passed as a parameter from the frmDBExample class
The display in the frmDBExample is altered from here

The goal

The DB class is the model

Should not care what class has instantiated it.
Nothing in the DB class will directly reference anything in any user written class.

The frmDBExample class is the view

Has control over changing the display in the view
Has control over what is done with any data, the db class simply return what is in the reader and the frmDBExample decides what to do with it

The Row class is a container to represent data

In the original attempt our select statement did not display all of the data all that well. One way of manipulating data programmatically is to make a wrapper or a data structure for this data. For this example we have made a class called Row to represent the current data. If we wanted to we could make an array of Row objects to represent an entire result set.

The new DB class

```
Public class DB
    Public SSconn As New SqlClient.SqlConnection
    Private ds As String
    Private db As String

    Public Property MyDataSource() As String
        Get
            Return ds
        End Get
        Set(ByVal value As String)
            ds = value
        End Set
    End Property

    Public Property MyDatabase() As String
        Get
            Return db
        End Get
        Set(ByVal value As String)
            db = value
        End Set
    End Property

    ' Visual Basic
    Public Sub ConnectToSql(ByVal ds As String, ByVal db As String)
        SSconn.ConnectionString &= _
         "integrated security=SSPI;data source=" & ds & ";" & _
         "persist security info=False;database=" & db
        ' original string
        ' "integrated security=SSPI;data source=SQL Server Name;" & _
        ' "persist security info=False;initial catalog=northwind"
        ' whatis SSPI
        ' Security Support Provider Interface (SSPI). SSPI allows
        ' an application to use various security models available
        ' on a computer or network without changing the interface
        ' to the security system
        Try
            SSconn.Open()
            ' Insert code to process data.
        Catch ex As Exception
            frmDBExample.lblFromCode.Text = "Failed Connection"
            frmDBExample.lblFromCode.Visible = True
            MessageBox.Show("Failed to connect to data source")
        End Try
    End Sub

    Public Sub DisconnectFromSql()
        SSconn.Close()
    End Sub

    Public Sub CreateSampleTable(ByVal tableName As String)
        ' example CREATE TABLE Person (LastName varchar,FirstName
varchar,Address varchar,Age int)
```

```
        Dim catCMD As SqlClient.SqlCommand = New
SqlClient.SqlCommand("CREATE TABLE " & _
            tableName & " (Text1 varchar(32),Text2 varchar(32),myint
int,mydecimal decimal(10,2))", SSconn)
        catCMD.ExecuteNonQuery()
    End Sub

    Public Sub InsertRow(ByRef row As Row)
        ' example INSERT INTO Persons VALUES('Hussein', 'Saddam',
'White House')
        Dim catCMD As SqlClient.SqlCommand = New
SqlClient.SqlCommand("INSERT INTO " & _
            row.Table1 & " VALUES('" & _
            row.Text1 & "','" & _
            row.Text2 & "','" & _
            row.Int1 & "','" & _
            row.Decimal1 & "')", SSconn)
        catCMD.ExecuteNonQuery()
    End Sub

    Public Function SelectAll(ByVal t As String) As
SqlClient.SqlDataReader
        Dim i As Integer = 0
        ' example SELECT * FROM Persons
        Dim catCMD As SqlClient.SqlCommand = New
SqlClient.SqlCommand("SELECT * FROM " & t, SSconn)
        Dim reader As SqlClient.SqlDataReader = catCMD.ExecuteReader()
        reader.Read()
        Return reader
    End Function

    Public Sub DeleteSampleTable(ByVal t As String)
        ' DROP TABLE Person
        Dim catCMD As SqlClient.SqlCommand = New
SqlClient.SqlCommand("DROP TABLE " & _
            t, SSconn)
        catCMD.ExecuteNonQuery()
    End Sub
End Class
```

The New frmDBExample Class

```
Public Class frmDBExample
    Dim db As DB
    Dim sample As Row

    Public Sub New()
        InitializeComponent()
        lblFromCode.Visible = False
        lblDisFromCode.Visible = False
        lblSuccessfulCreate.Visible = False
        lblSuccessfulDelete.Visible = False
        lblSuccessfulInsert.Visible = False
    End Sub
    Private Sub btnFromCode_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnConnFromCode.Click
```

```
        db = New DB()
        db.MyDataSource = "IBM-26AA6302EA1\SQLEXPRESS"
        db.MyDatabase = "cosc444"
        db.ConnectToSql(db.MyDataSource, db.MyDatabase)
        lblFromCode.Visible = True
        lblSuccessfulInsert.Visible = False
    End Sub

    Private Sub btnDisConnFromCode_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnDisConnFromCode.Click
        lblFromCode.Visible = False
        lblDisFromCode.Visible = True
        lblSuccessfulCreate.Visible = False
        lblSuccessfulDelete.Visible = False
        lblSuccessfulInsert.Visible = False
        lblSuccessfulSelect.Visible = False
        db.DisconnectFromSql()
    End Sub

    Private Sub btnCreateTable_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnCreateTable.Click
        db.CreateSampleTable(txtTable.Text)
        lblSuccessfulCreate.Visible = True
        lblSuccessfulInsert.Visible = False
        lblSuccessfulDelete.Visible = False
    End Sub

    Private Sub btnInsertRow_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnInsertRow.Click
        lblSuccessfulInsert.Visible = False
        sample = New Row()
        sample.Table1 = txtTable.Text
        sample.Text1 = txtVarChar1.Text
        sample.Text2 = txtVarChar2.Text
        sample.Int1 = txtInt.Text
        sample.Decimal1 = txtDecimal.Text
        db.InsertRow(sample)
        lblSuccessfulInsert.Visible = True
    End Sub

    Private Sub btnSelectAll_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnSelectAll.Click
        Dim reader As SqlClient.SqlDataReader
        reader = db.SelectAll(txtTable.Text)
        Try
            If reader.HasRows Then
                sample.Text1 = reader.GetString(0)
                sample.Text2 = reader.GetString(1)
                sample.Int1 = reader.GetInt32(2)
                sample.Decimal1 = reader.GetDecimal(3)
            Else
                MessageBox.Show("no rows found")
            End If
        Catch ex As Exception
            MessageBox.Show( _
            "Exception looking for row" & vbCrLf & ex.Message & vbCrLf
& vbCrLf & ex.StackTrace, _
```

```
            ex.GetType().ToString())
        Finally
            reader.Close()
        End Try
        txtSelectResult.Text = sample.Text1 & " " & sample.Text2 & " "
& sample.Int1 & " " & sample.Decimal1
        lblSuccessfulInsert.Visible = False
        'lblSuccessfulSelect.Visible = False
    End Sub

    Private Sub btnDeleteTable_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnDeleteTable.Click
        db.DeleteSampleTable(txtTable.Text)
        lblSuccessfulDelete.Visible = True
        lblFromCode.Visible = False
        lblDisFromCode.Visible = False
        lblSuccessfulCreate.Visible = False
        lblSuccessfulInsert.Visible = False
        lblSuccessfulSelect.Visible = False
    End Sub
End Class
```

The new Row class

```
Public Class Row
    Private tablea As String
    Private texta As String
    Private textb As String
    Private inta As Integer
    Private decimala As Decimal

    Public Property Table1() As String
        Get
            Return tablea
        End Get
        Set(ByVal value As String)
            tablea = value
        End Set
    End Property
    Public Property Text1() As String
        Get
            Return texta
        End Get
        Set(ByVal value As String)
            texta = value
        End Set
    End Property
    Public Property Text2() As String
        Get
            Return textb
        End Get
        Set(ByVal value As String)
            textb = value
        End Set
    End Property
    Public Property Int1() As Integer
```

```
        Get
            Return inta
        End Get
        Set(ByVal value As Integer)
            inta = value
        End Set
    End Property
    Public Property Decimal1() As Decimal
        Get
            Return decimala
        End Get
        Set(ByVal value As Decimal)
            decimala = value
        End Set
    End Property
End Class
```

Conclusion

You can now use the DB class and the Row class outside of this project as a reusable, extensible, object oriented artifact. The view is specific to this application. The view did not change in our refactored program.