Connect to an Oracle Database from Visual Basic 6 (Part 2)

Preface

This is Part 2 in a 2 part series on using Visual Basic 6 to connect to an Oracle database. In Part 1, I showed you how to use an ADO Data Control to make the connection. In this article, I show you how to use ADO Objects to make the connection. Not only will the article be useful for the Oracle programmers in the audience, it will also show you how to use ADO objects in your program.

ADO Objects

What's an Object? Hopefully you've read by well-received book, Learn to Program with Visual Basic 6 Objects. As much as I would love to spend a lot of time discussing object variables in detail, I'm going to presume that you know what they are and how, in a nutshell, they refer to an object instantiated from a class or template.

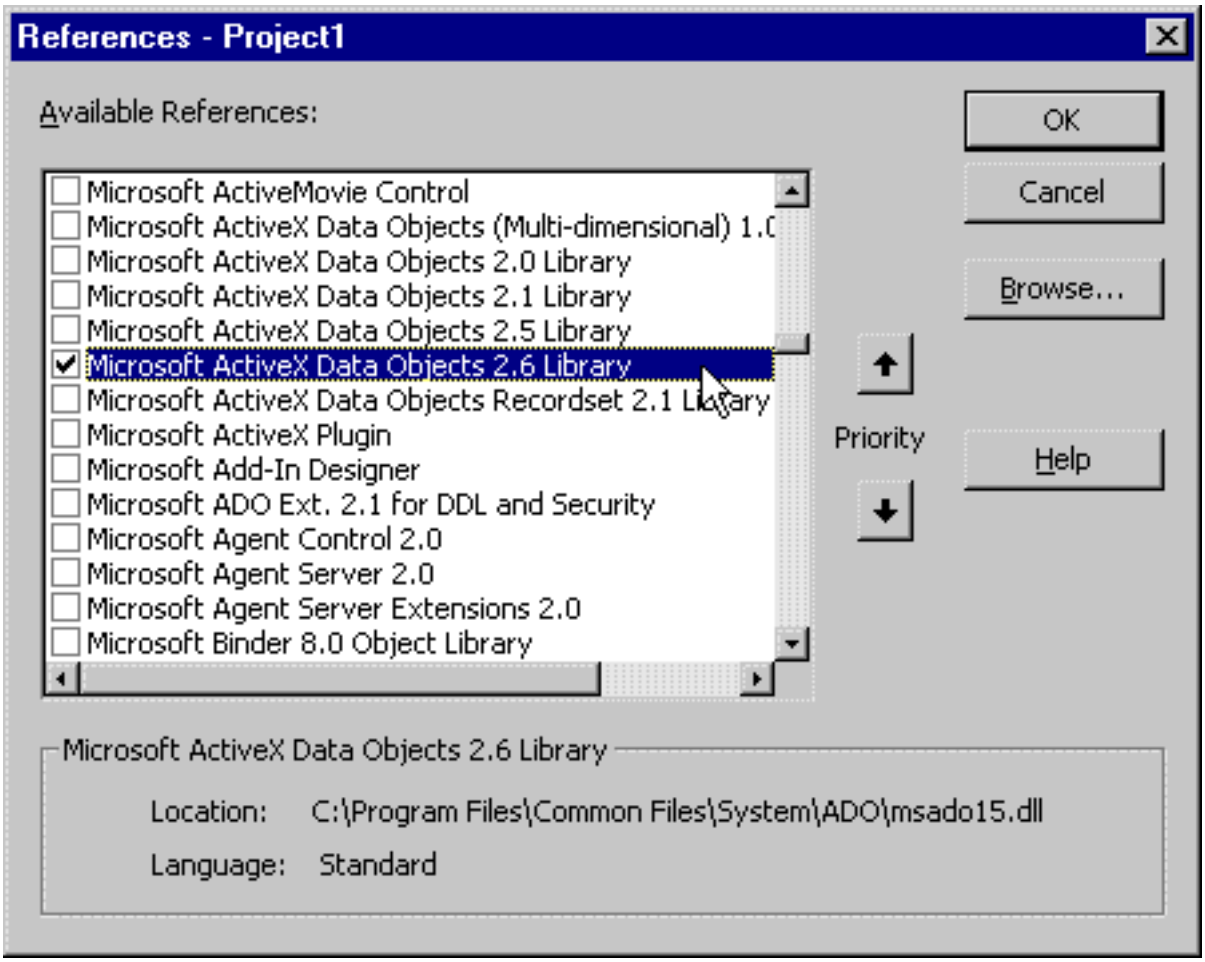
Having said that, I will tell you that the ADO Object model can be confusing, particularly because there are frequently many ways to achieve the same functionality.

In this article, we will create an ADO Connection Object, then use an ADO Command Object to return the results of a query to an ADO Recordset object. Finally, as an added bonus, I'll show you something that your average programmer doesn't know can be done--we'll set the Recordsource of an ADO DataGrid to the recordset we produce in code. In other words, without the ADO Data Control, we'll populate a DataGrid.

Similar to what we did when we added the ADO Data Control to our Visual Basic Toolbox, in order to use ADO Objects we must first set a reference to the ADO Object Library. The version of the ADO Object Library installed on your PC will vary depending upon a number of factors-the PC I'm using as I write this article contains ADO Version 2.x.

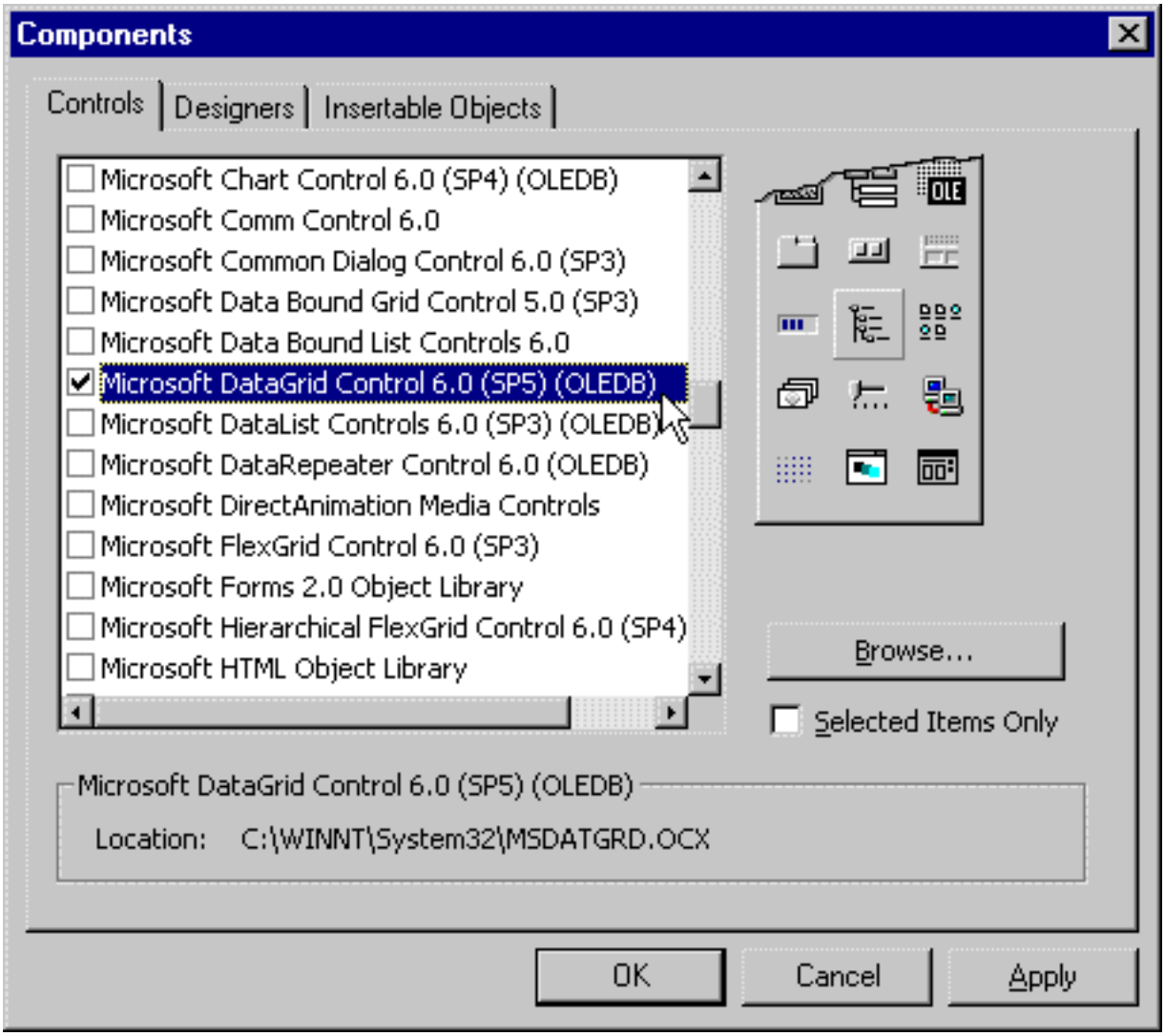
To add a reference, select **Project-References** from the Visual Basic Menu Bar, then select Microsoft ActiveX Data Objects 2.6 Library. If the version number doesn't match yours exactly, that's fine---just be sure to select the ActiveX Data Objects Library, not the ActiveX Data Objects Recordset Library that follows it in the list. Be sure that you select Project-References, NOT Project-Components. Object libraries have no visible interface, and so they won't appear in the

Components section.

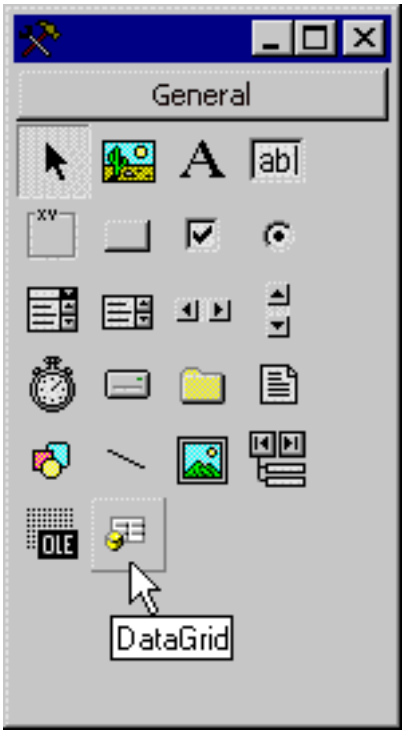


When you click on the OK button, nothing obvious happens--but you now have access to ADO Objects in your program.

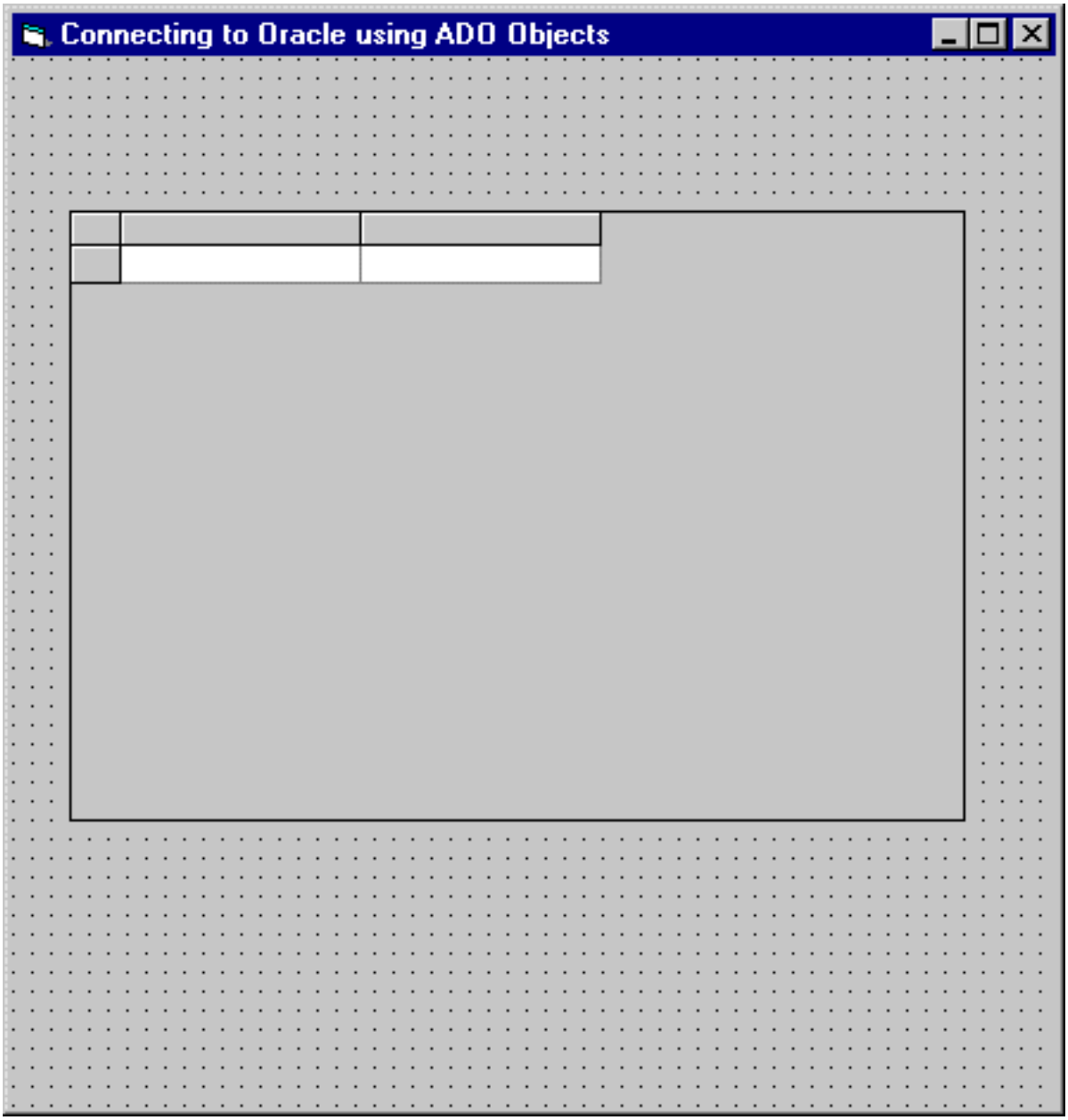
Before we get too far, let's start the process of adding the ADO DataGrid Control to our form. In order to do that, we need to select **Project-Components** (yes, that's right, Components this time) from the Visual Basic Menu Bar, and select Microsoft DataGrid Control 6.0.



When you click on the OK button, the DataGrid Control (the ADO version) is added to the Visual Basic Toolbox.



With the DataGrid control in your Toolbox, now it's time to add it to your form...



Working with ADO Objects to achieve your Oracle Connection

The Connection Property

It was at this point in Part 1 of this article that we adjusted properties of the ADO Data Control to achieve a Connection to an Oracle Database, and to build a Recordset which was then used to populate the ADO DataGrid. Here, we'll use an ADO Connection Object and a Recordset Object to achieve the same functionality. For demonstration purposes, let's place this code in the Load Event Procedure of the Form...

```
Dim oconn As New ADODB.Connection
```

```
Dim rs As New ADODB.Recordset
Dim strSQL As String

strSQL = "SELECT * FROM EMPLOYEES"

Set oconn = New ADODB.Connection
oconn.Open "Provider=msdaora;Data Source=John.world;User Id=jsmiley;
Password=neveruseyourdogsnameasyourpassword;"

rs.CursorType = adOpenStatic
rs.CursorLocation = adUseClient
rs.LockType = adLockOptimistic

rs.Open strSQL, oconn, , , adCmdText

Set DataGrid1.DataSource = rs
```

What's going on?

Let me explain. These first two lines of code declare 2 object variables. The first is an ADO Connection Object

```
Dim oconn As New ADODB.Connection
```

followed by an ADO Recordset Object

```
Dim rs As New ADODB.Recordset
```

Both of these declarations are necessary in order to build a Recordset--which is a virtual (in memory) representation of an actual Oracle table in our Database. The names for these two object variables are pretty standard, and you see them all the time in code and in books and articles. In fact, if you search for oconn or rs using a Google search, you'll gets lots of results for ADO.

The final variable declaration is a String variable that we will use to 'hold' the SQL statement used to build our recordset.

```
Dim strSQL As String
```

Once we've declared the strSQL variable, it's time to assign a SQL statement to it. You may remember this statement from Part 1 of this article--it's used to retrieve every record from the Employees table in my Oracle database...

```
strSQL = "SELECT * FROM EMPLOYEES"
```

Our next step is to open our ADO Connection. We do that by executing the

Open method of our Connection Object. The Open method is looking for four parameters: the Provider name, the Data Source (or HostName), the User ID and the Password of the database in question.

```
oconn.Open "Provider=msdaora;Data Source=John.world;User Id=jsmiley;
Password=neveruseyourdogsnameasyourpassword;"
```

Do the details of the Open Method look familiar? They should. In Part 1 I told you that the Connection String that the ADO Data Control wizard built for us could be used to open a Connection in code. In fact, some programmers use the ADO Data Control wizard to build their Connection Strings for them, and then copy and paste them into the code window.

Before we build the Recordset object (not the same as the Connection object) we need to adjust three properties of the Recordset object, the CursorType, CursorLocation and LockType. If you want to know more about the details of the Recordset object, I highly recommend Wrox's ADO 2.6 Programmer's Guide.

```
rs.CursorType = adOpenStatic
rs.CursorLocation = adUseClient
rs.LockType = adLockOptimistic
```

Now it's time to open the Recordset. If you are confused about the difference between a Connection and a Recordset, think of the Connection as the Database window you see when you first start Access. The Database contains a window with a list of tables that you can then select. When you select one, a Data Window opens up with just the information from that table. This is the equivalent of the Recordset.

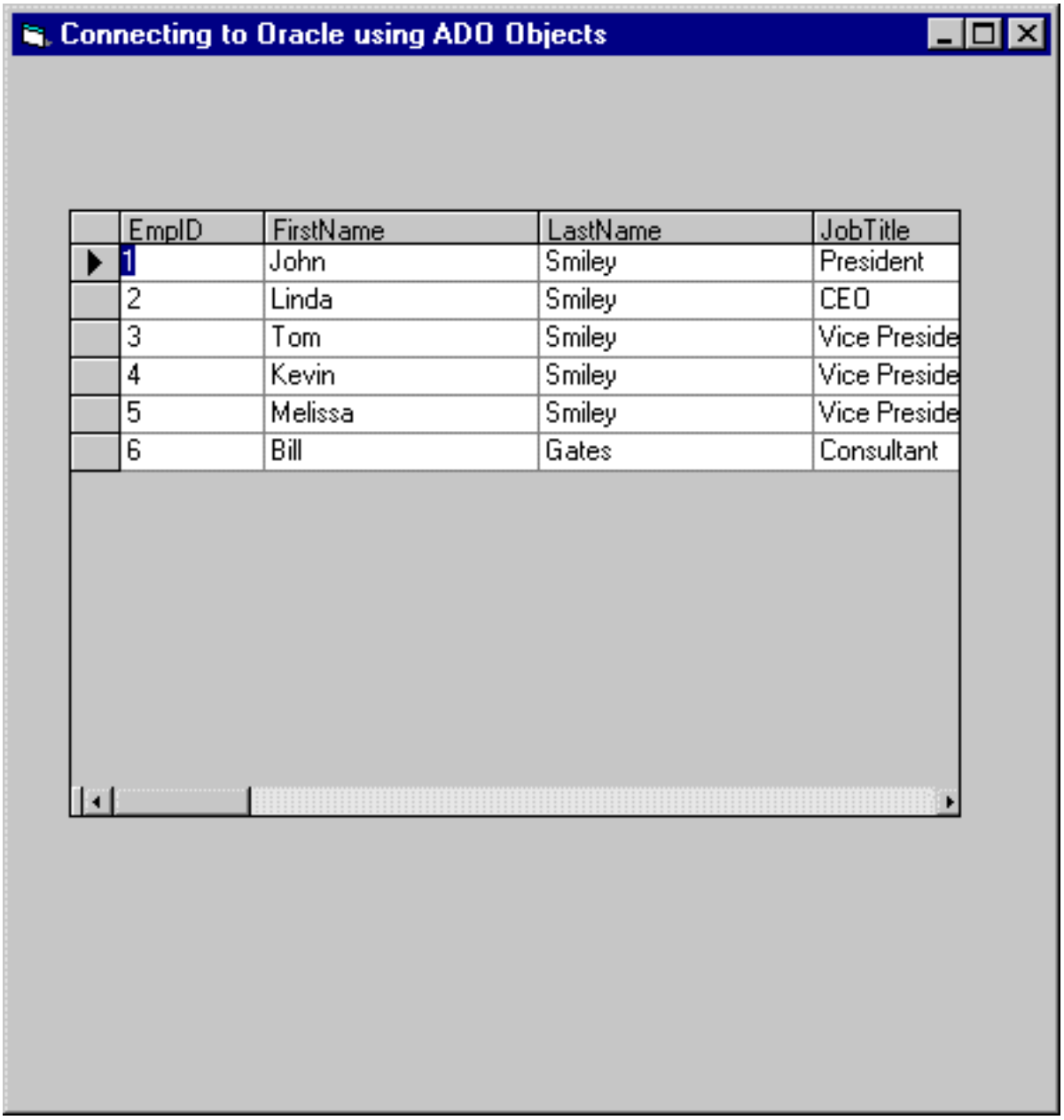
```
rs.Open strSQL, oconn, , , adCmdText
```

In theory, we now have a Recordset object built containing all of the data in the Employees table of my Oracle database. How do we see the Recordset?

We can use the Set statement to assign the Recordset object to the DataSource property of our DataGrid.

```
Set DataGrid1.DataSource = rs
```

If we now run the program, the code in the Load Event procedure of the form executes. A Connection object is created, initiating the Connection to my Oracle Database. Then a Recordset object is created, retrieving Employee records. Finally, the DataSource property of the DataGrid is set to point to the Recordset object. Here's our form after the code executes--as you can see, the DataGrid is now populated.



Summary

I hope you've enjoyed this article.