www.mbfoster.com

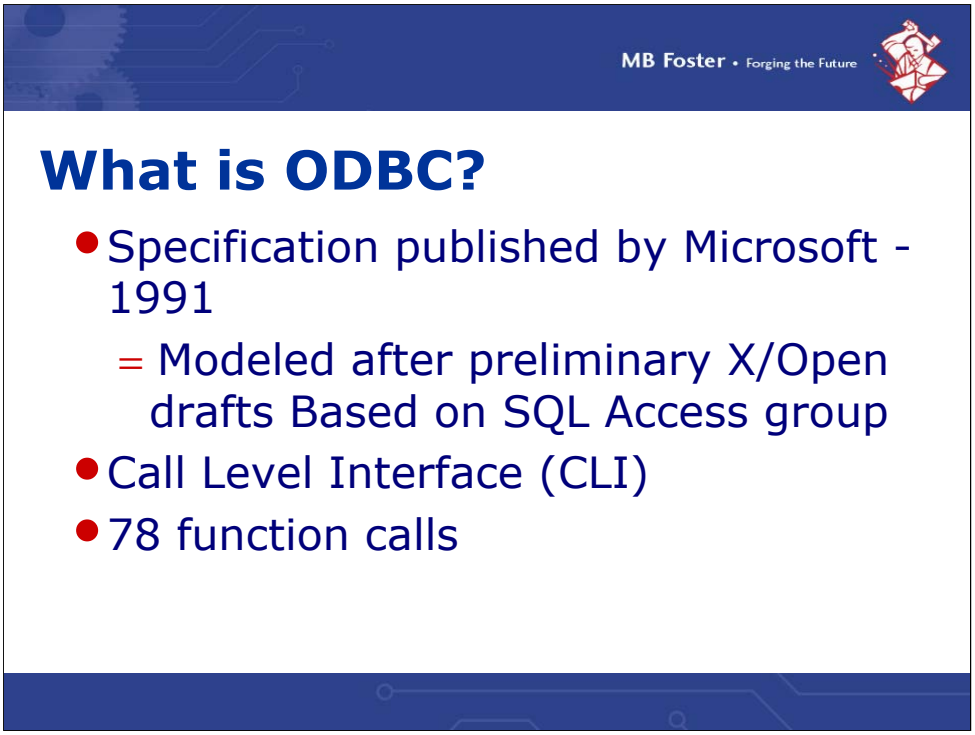
Solution Symposium
April 2002

VISUAL Basic with ODBC

Presented by John Middelveen
Technical Mgr. Core Product Development
MBFoster

Agenda

- Overview of ODBC
- Client Setup
- Managing/Tuning connections
- Troubleshooting
- ADO
- RDO
- Direct API programming



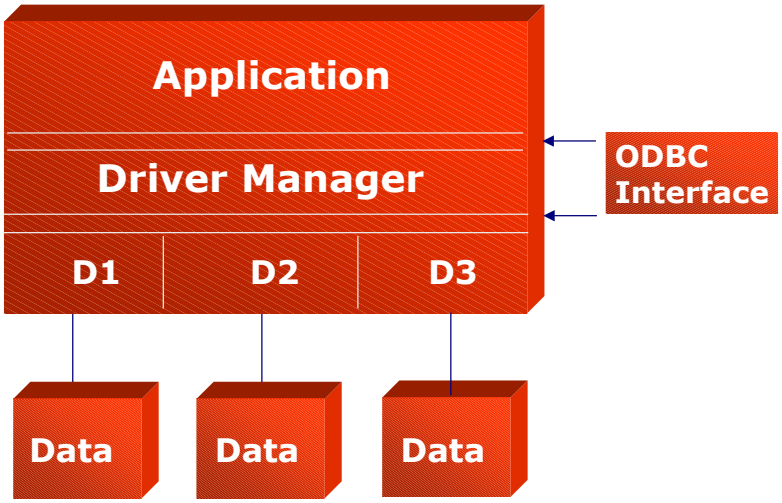
Open Database Connectivity (ODBC) provides a Universal Database Connectivity application programming interface (API) that enables applications to access data in a wide range of proprietary databases. Based on the X/Open SQL Access Group's Call Level Interface (CLI) specification, ODBC is an open, **vendor-neutral** way to uniformly access data stored in different formats and database engines.

ODBC is one of the the most widely used interfaces to relational data.

As the standard interface to relational data, your application can access a lot of data using ODBC. But ODBC does require that your data look like a relational database, so for non relational data such as documents or E-mais it's not the best way to expose data. It is difficult to write an ODBC driver to expose non relational data because you have to write a relational engine on top of the existing data structure.

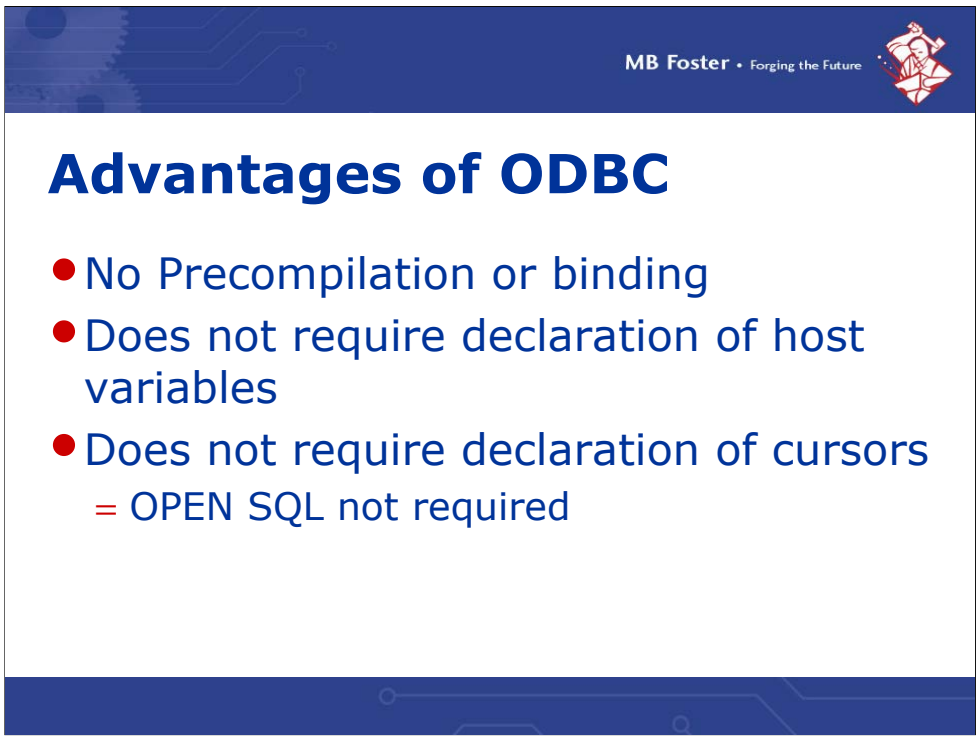
ODBC Interface

MB Foster • Forging the Future

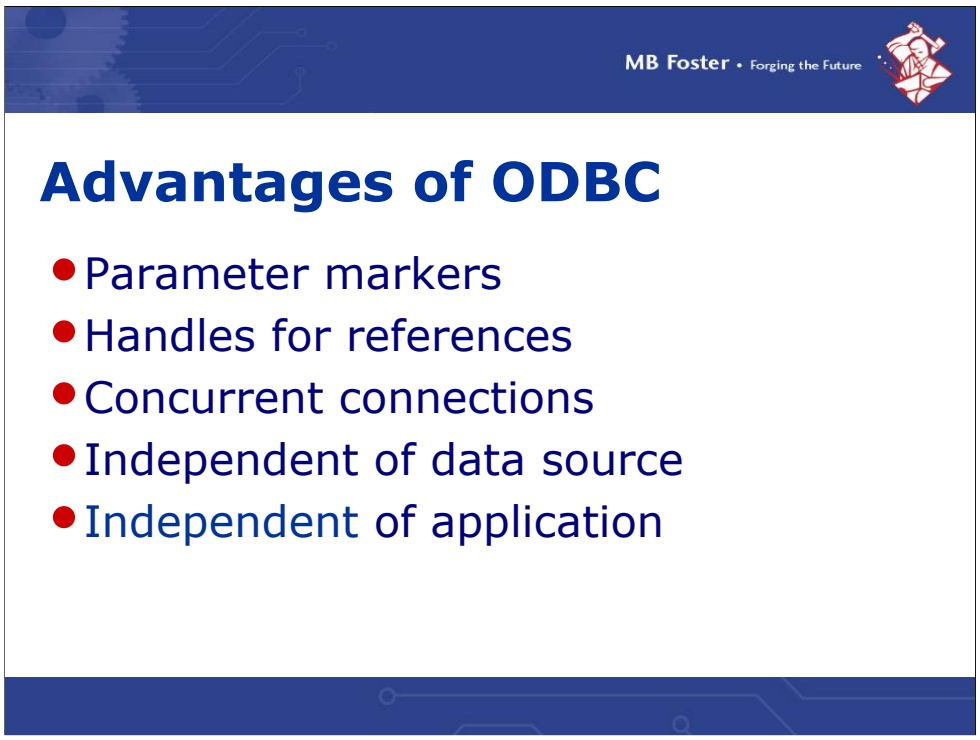


What ODBC is not

- Designed to use database capabilities, not supplement them
- Not a heterogeneous join engine
- Not a distributed transaction processor
- Can be built into tools



Host variables - no explicit linkage section required - can create problems if local variable



Parameter markers P1? P2?

Statement handles - abstract object - no need for product specific structures

Environment and connection handles - reference global variables and connection information

MB Foster • Forging the Future

Disadvantages of ODBC

- Extra layers
  - = Slower
  - = More places where things can go wrong
- Does not take advantage of all functions available in API

NEXT - ARCHITECTURE...

Functions of ODBC Drivers

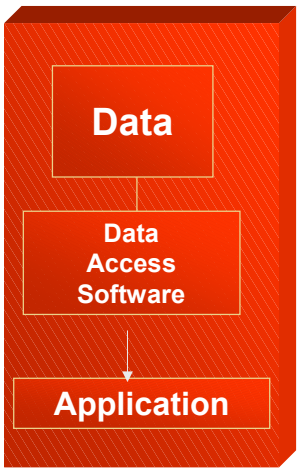
- Libraries
  - Connecting/Disconnecting
  - Error Checking
  - Initiating Transactions
  - Submitting SQL
  - Sending/Receiving Data
  - Mapping Errors

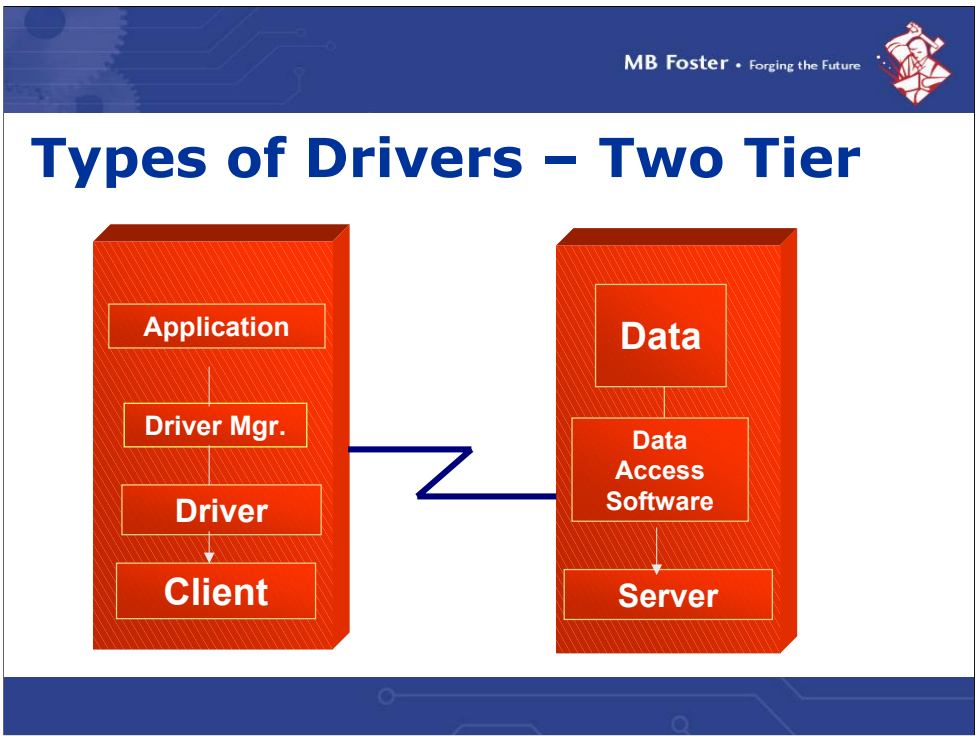
Driver Conformance Levels

- API
  - = Core
  - = Level 1
  - = Level 2
- SQL
  - = Minimum
  - = Core
  - = Extended
- Data
  - = Data Types

Types of Drivers

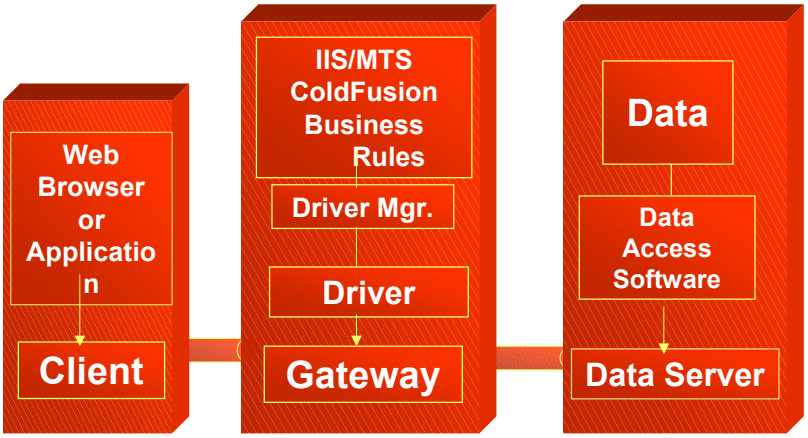
- Single Tier
  - = All on one platform
    - Application
    - Driver
    - Data





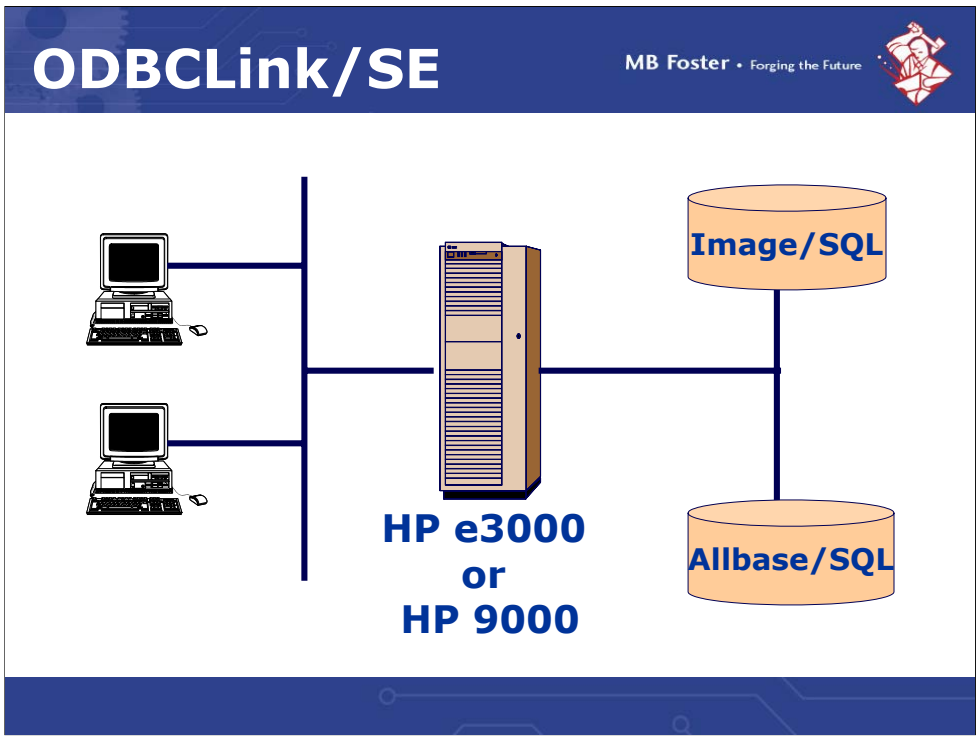
Types of Drivers – Two Tier

N-Tier Client Server



Components of ODBC
The Driver Manager

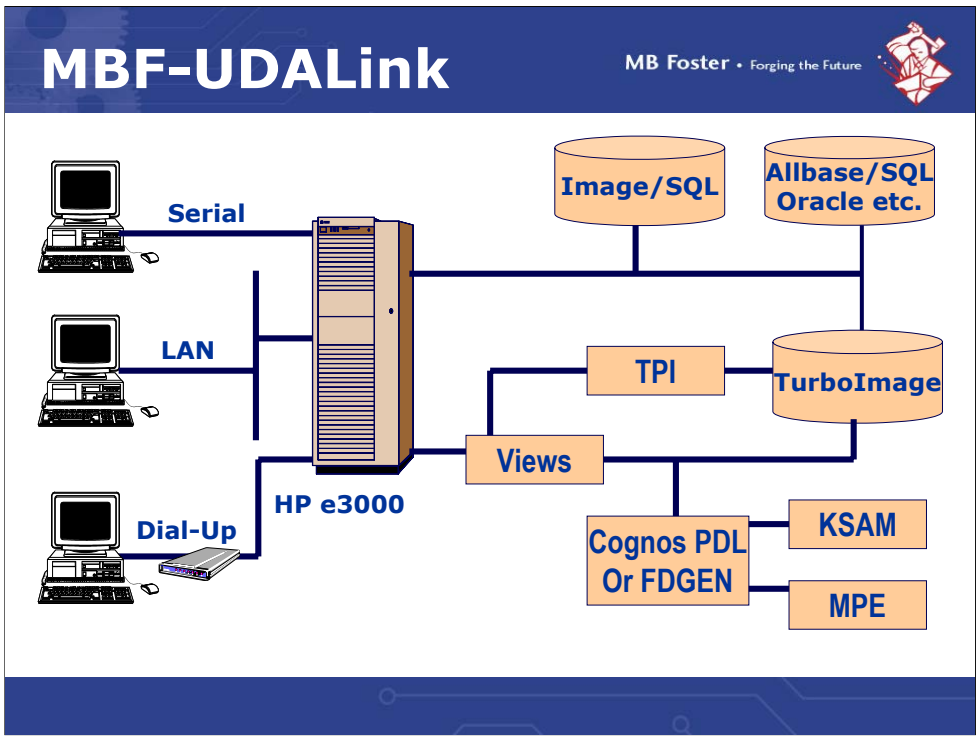
- Special Library
- Responsible for Routing calls
- Loads Drivers
- Examines calls
- Disconnects (or not)
- Some error checking



UDALink Vs ODBC/SE - differences

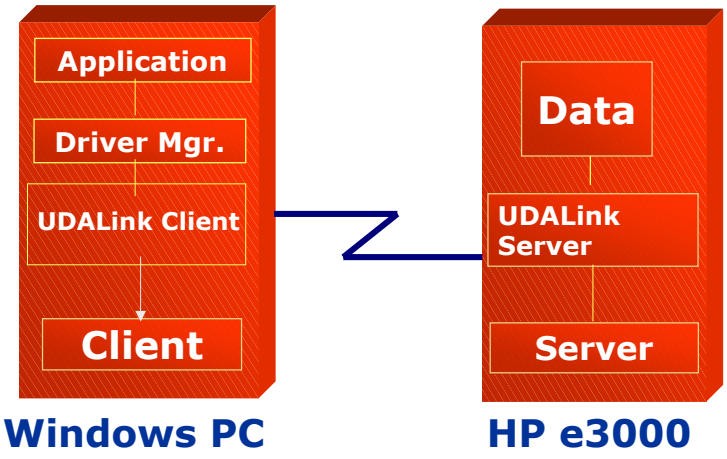
- ODBC/SE Cannot handle multiple environments at the same time and are limited to file type.

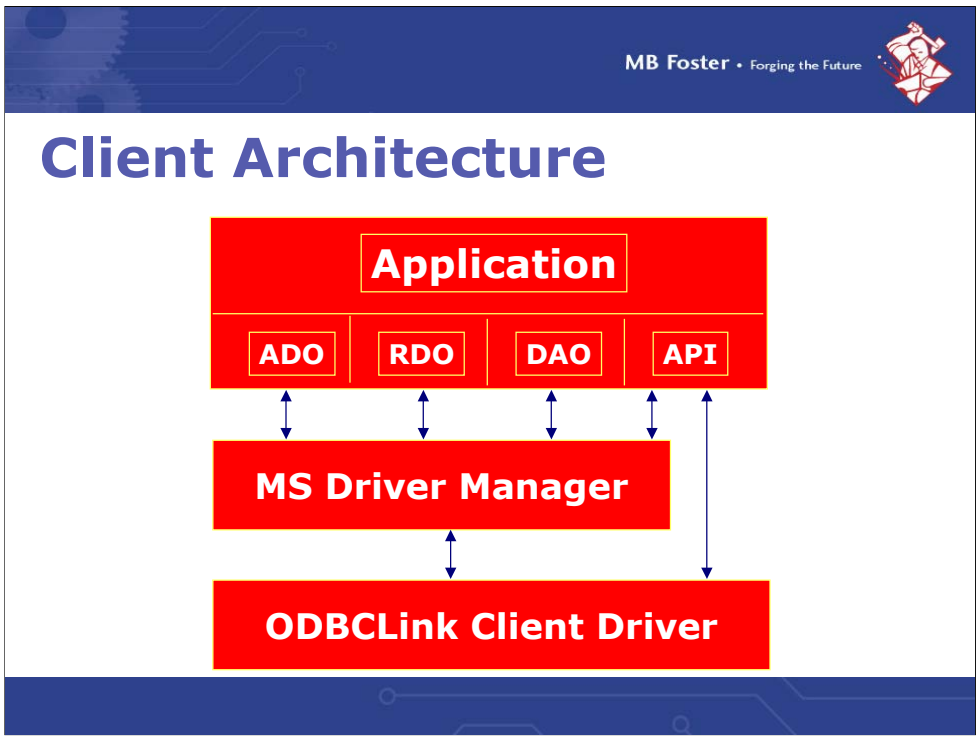
-



TPI must be turned on for support by UDALink.

MBF-UDALink Architecture

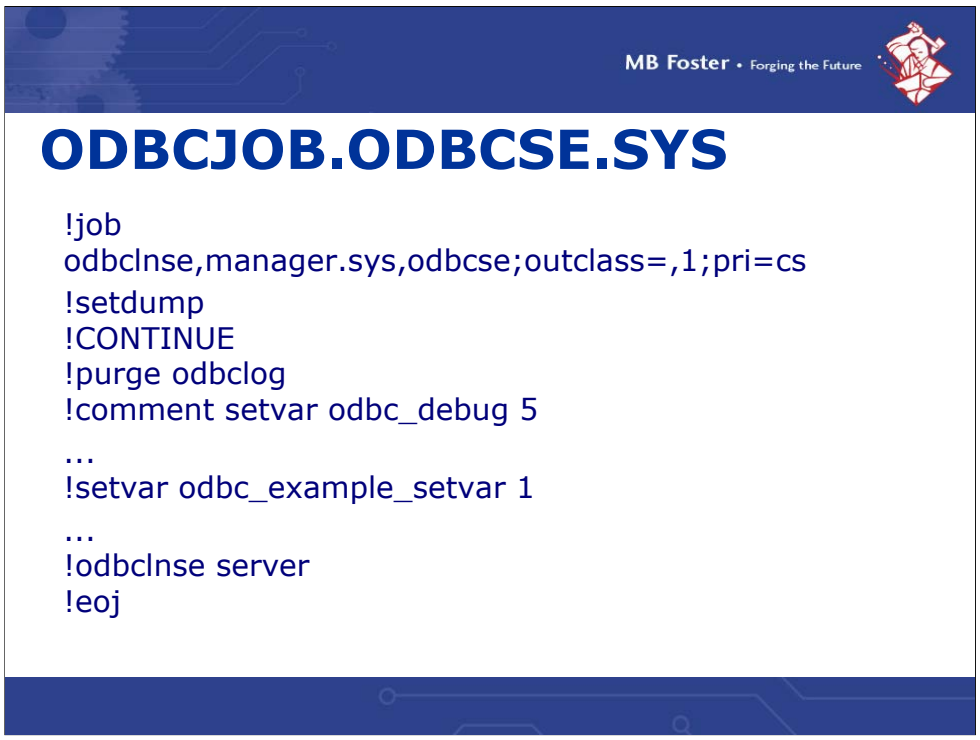




Can write API apps that links directly to the ODBC driver, bypassing MS Driver Mgr. Major increase in performance.

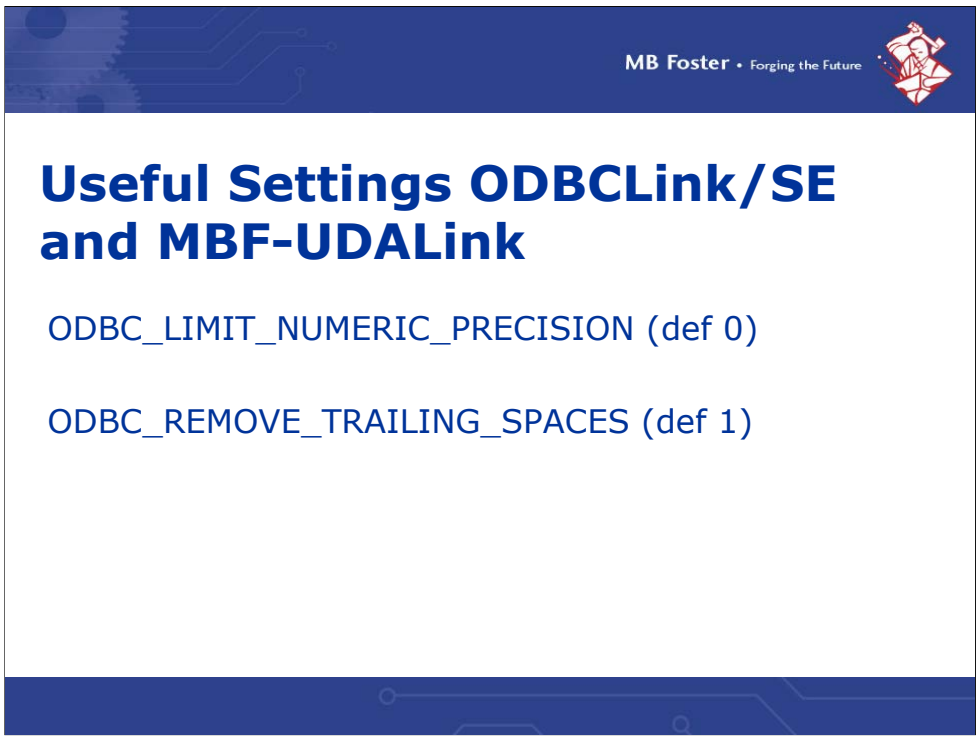
- Explain ADO, RDO, DAO, API????
-

Configuring the Listener



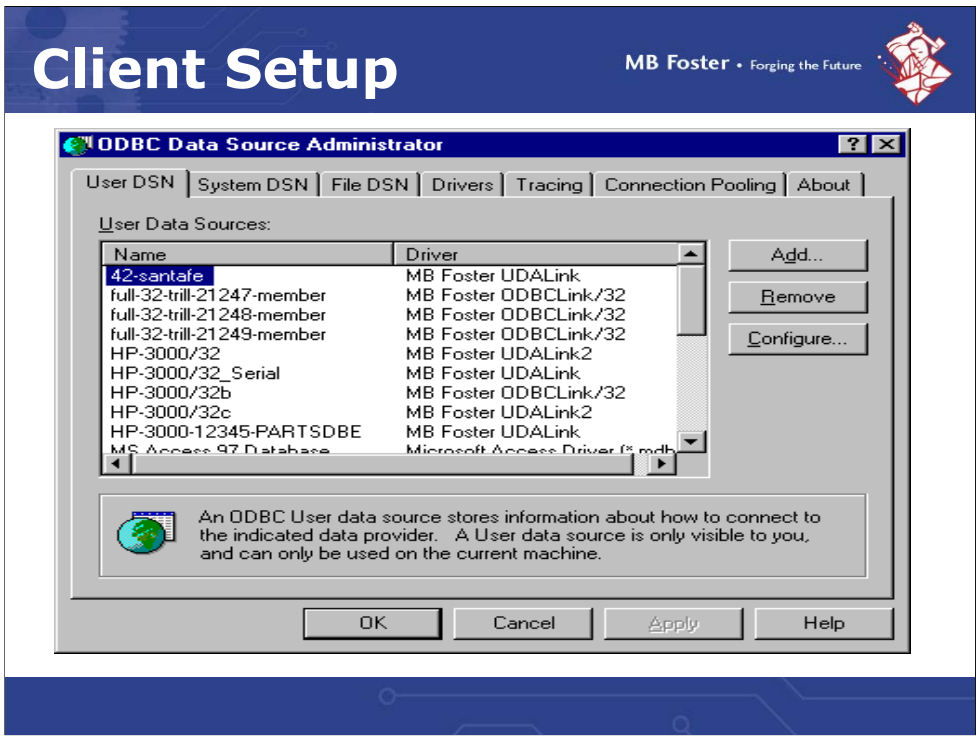
As you can see in the example putting a ‘comment’ in front of the setvar will comment the line out.

If you have a number of setvars (especially when using the full driver) it is useful to make a template with the more commonly used setvar in it, and just remove the comment as needed.



We mentioned ODBC_LIMIT_NUMERIC_PRECISION in the MS Access Section Previously.

The ODBC_REMOVE_TRAILING_SPACES is a param that was added for SQLServer Users. If set to 0 it will include all the trailing spaces in a fixed character field that the driver normally truncates before sending.



The first program that anyone dealing with any ODBC Driver must be familiar with is the ODBC Administrator. For Windows platforms, this is provided by Microsoft. Its main function is to allow users to configure data-sources for any ODBC driver that exists on the system.

Normally this program is invoked from the ODBC Administrator (ODBC Data Sources) icon in the Control Panel.

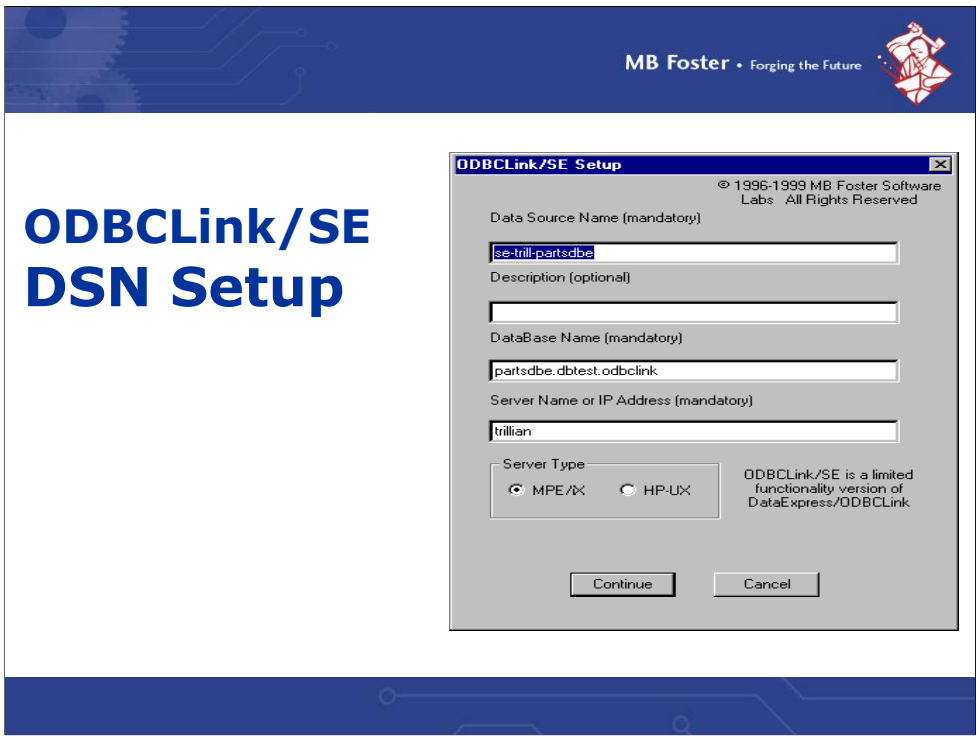
For Windows 2000 there is a subdirectory for all the Administrative Tools, and the ODBC Administrator is invoked by clicking on the “Data Sources(ODBC)” icon

The ODBC Administrator program allows you to configure data-sources, turn logging on and off; and for ODBC3.0 versions allows you to turn connection pooling on. You can also find out about what versions of ODBC drivers are installed on your PC.

To setup an ODBCLink/SE DSN you click on the Add button of the ODBC Administrator.

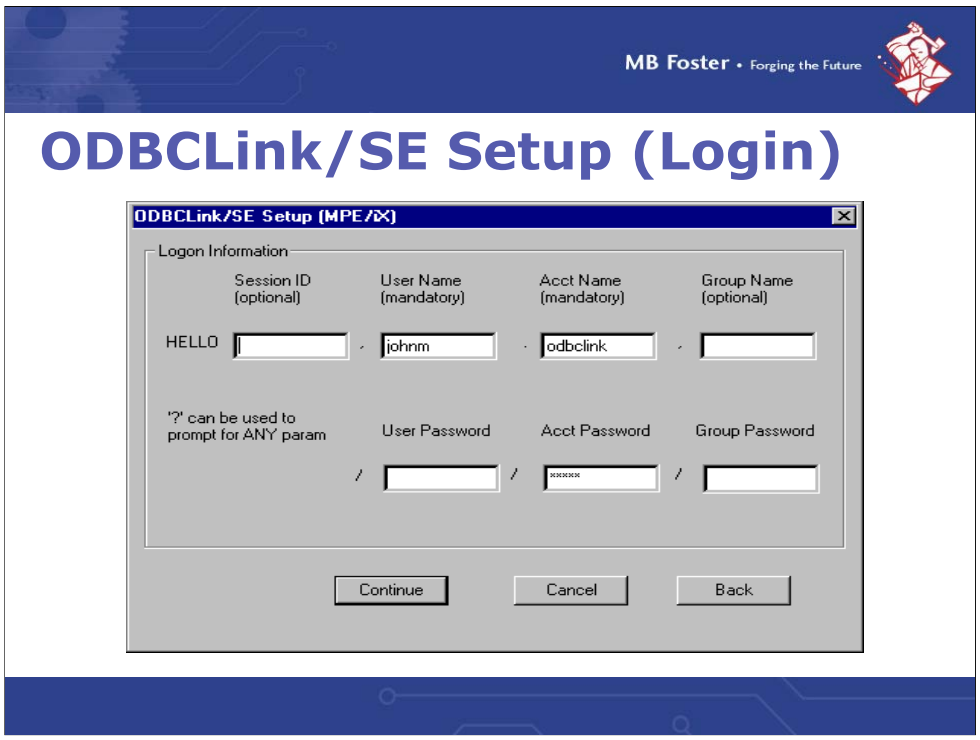
Then you select the driver that you wish to setup a DSN for; in this case the ODBCLink/SE-32 Driver.

Next click <Finish>.

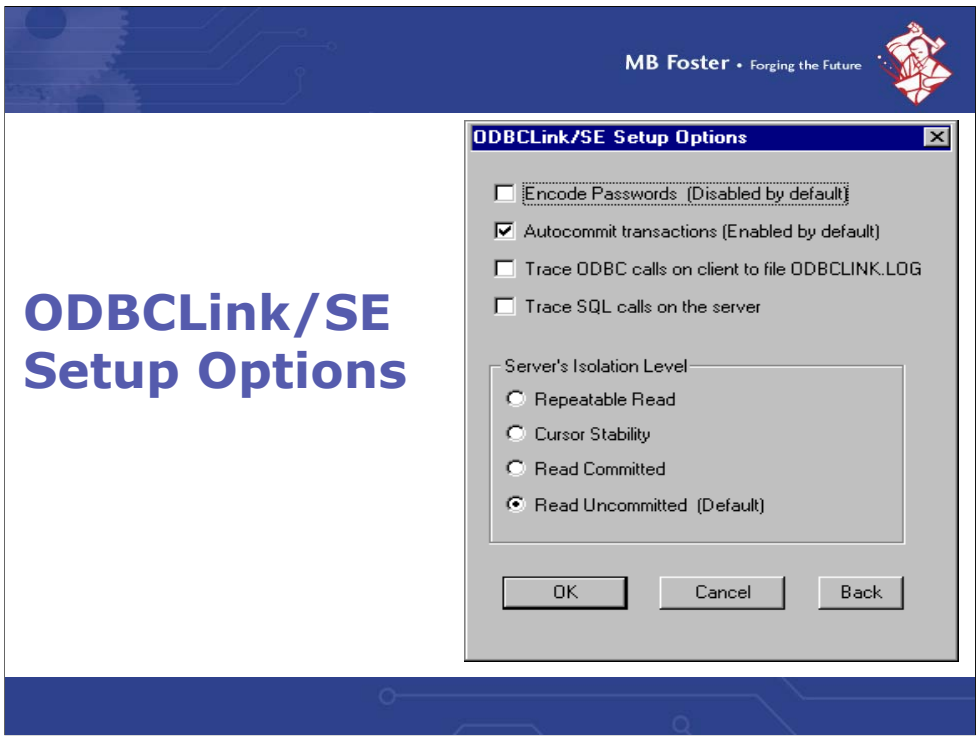


Enter the DSN Name, DBE name (including path or group/account), Server Name (or IP address) and select the proper Server Type;

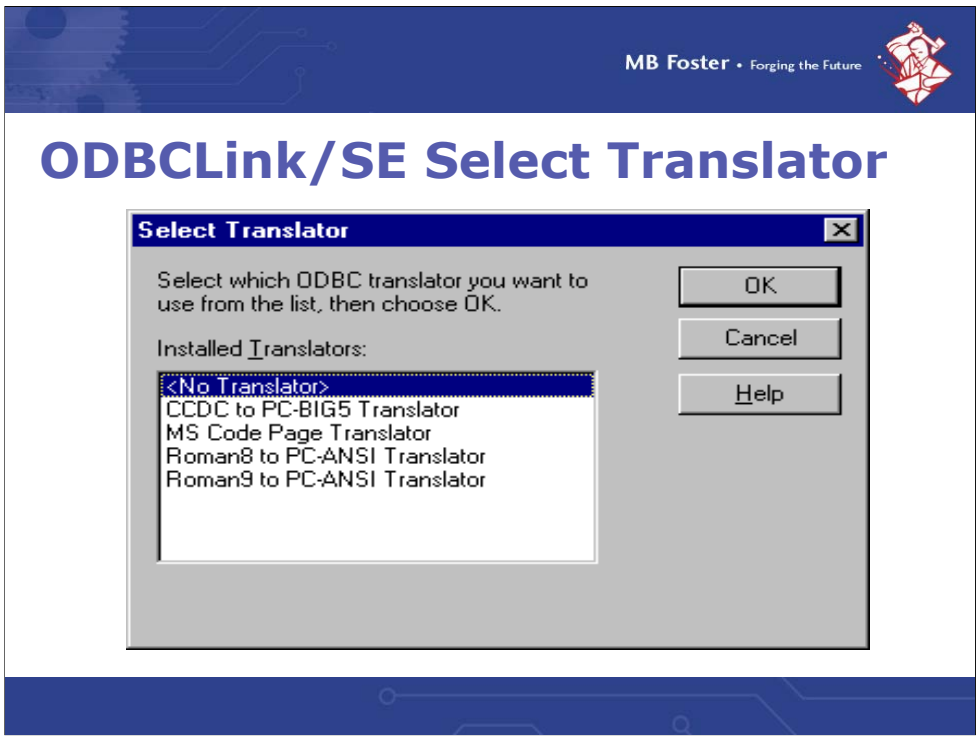
Then Click <Continue>



A MPE or UNIX login screen will appear. (depending on the OS type as setup in the previous screen)



After the Login Info a window will open allowing you to add Transaction Isolation Auto-Commit and Allow you to turn Debug ON or OFF



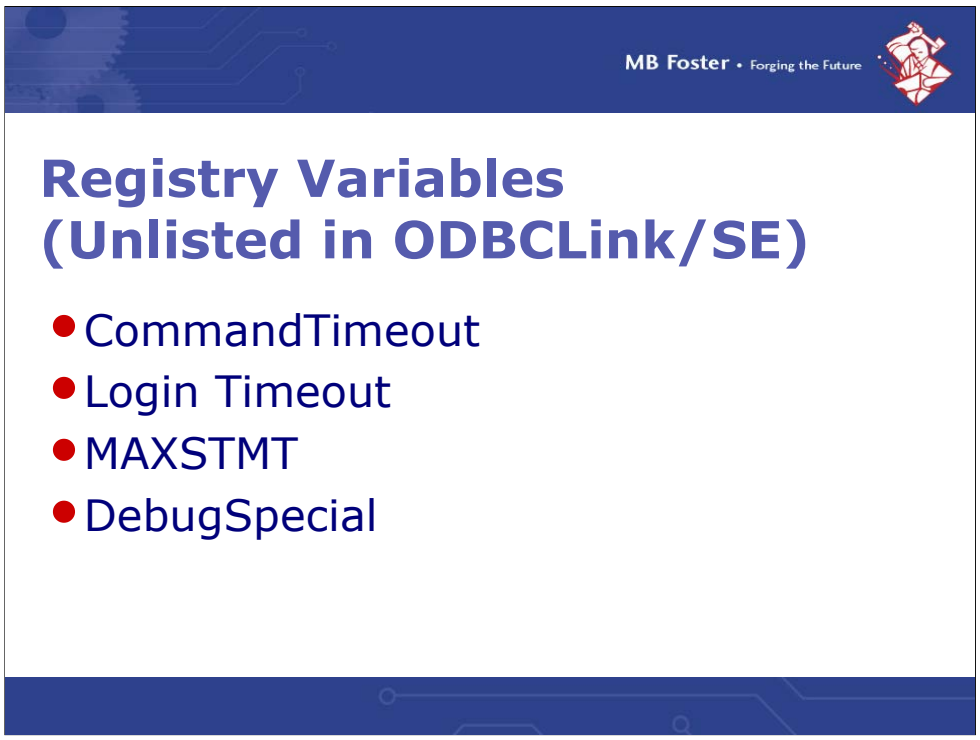
The final screen is the Translator setup screen which allows you to select any of the translators that have been setup on your PC.

The SE Driver comes with the BIG5 translator and the r8 to PC ANSI translator.

The Roman9 to PC-ANSI is identical except that it has the EURO symbol added in place of the sputnik. (As of 5.57.08)

Modify Registry Variables

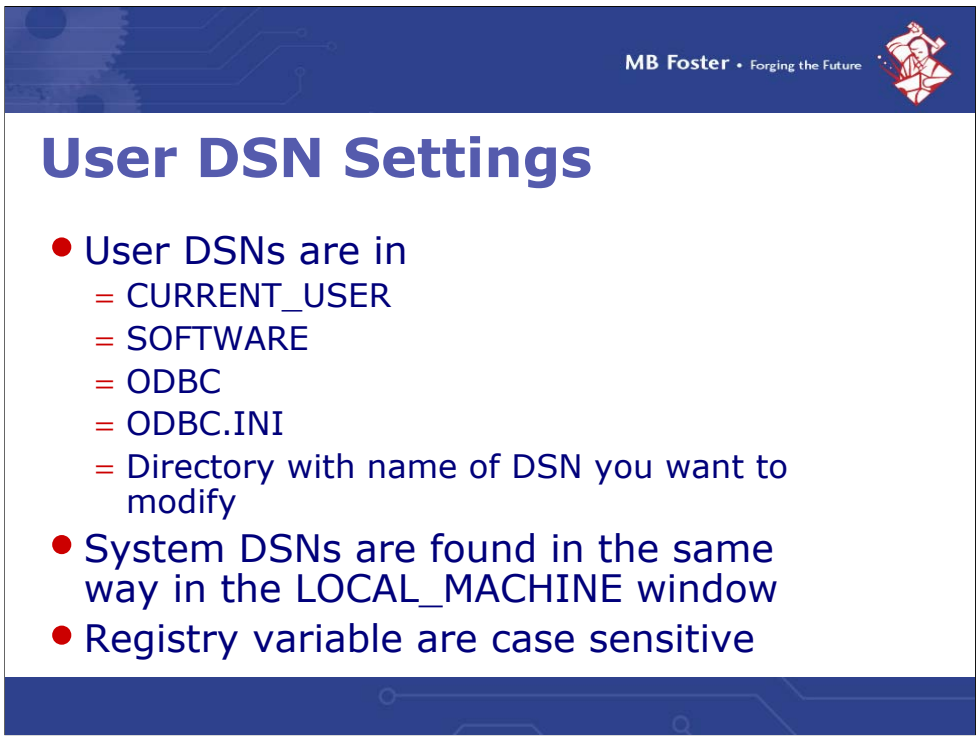
- DSN attributes are stored in the registry
- Normally changed from client setup window (via the ODBC Administrator odbcad32.exe)
- Use the registry editor program to edit settings – USE CAUTION



Usually you will not need to change any of these parameters.

These show up in Registry for the full version of the driver (as well as many other settings) but not the SE version

A few special parameter settings such as DebugSpecial must be added MANUALLY to the registry.



As you may know there are 3 types of DSNs

User, System and File DSN

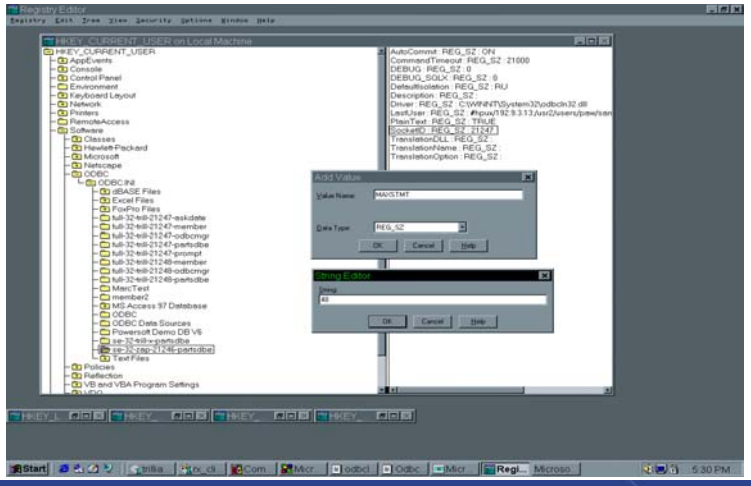
The User and System DSN Info is stored in the registry

File DSNs info is stored in small text files with '.dsn' extensions.

These could be anywhere (depending on how things are set up)

There are also 'DSNless' connections where all the connect information is entered in the Connection string

Editing the Registry Variables

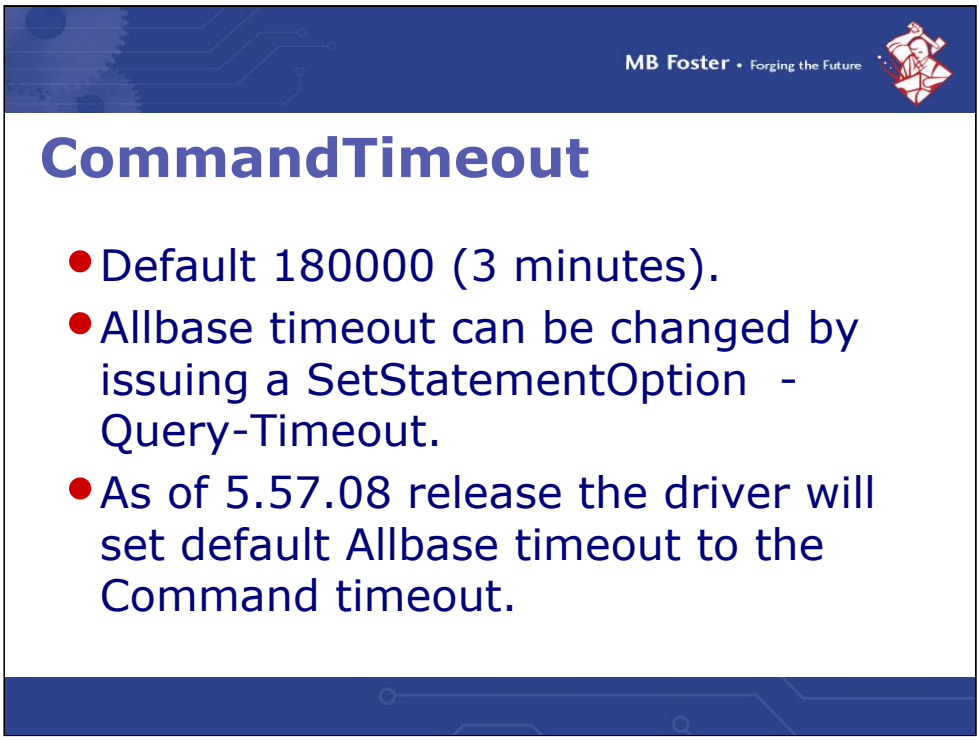


Here is a Shot of what it looks like using Regedit to Add a parameter to the Registry.

Depending of what version of the Administrator you are using, once you add the new variable it may prompt you for a value or you may have to double-click on the parameter to open an edit box.

REMEMBER THAT THESE PARAMETERS ARE CASE SENSITIVE!

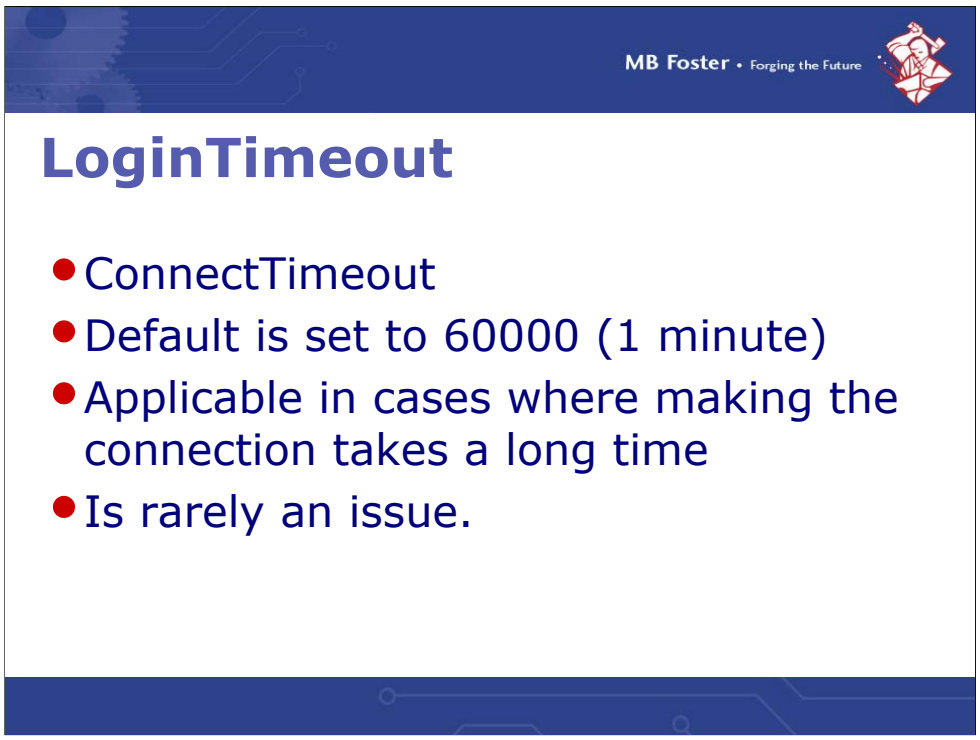
All Variables are string (REG_SZ)



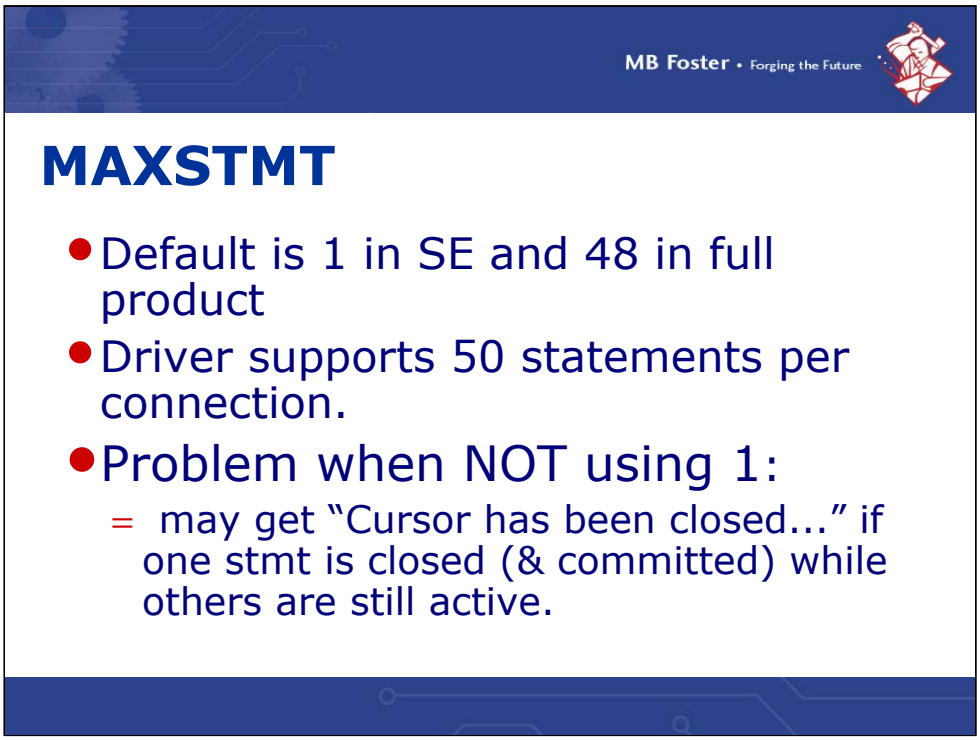
Notes:

In older versions (pre 5.57.08) this does NOT change ALLBASE timeout.

ALLBASE also has a MAXIMUM timeout setting (user settable). If this timeout is higher than the ALLBASE MAX then ALLBASE will just use the max (and you will see an error in the Server log if Logging is ON)



Sometimes this is useful if you have a LOT of Image/SQL tables attached to the DBE. In some Allbase versions these take a long time to get the info for and so the connection takes a long time.



Notes:

Although Setting the MAXSTMT to 1 cures problems associated with some Applications when accessing Allbase, there are some disadvantages to doing this.

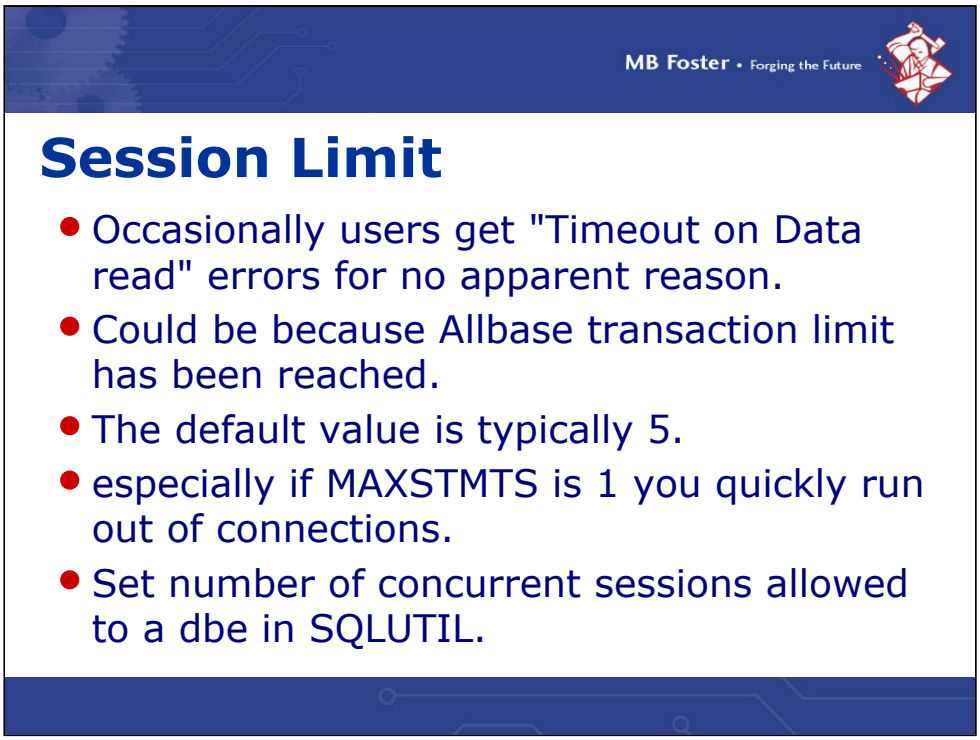
Disadvantages:

- There is more overhead required

- You will reach your max ALLBASE concurrent transaction limit sooner.

Other Tidbits

- Session Limit
- Queue Tuning
- Connecting to other DBEs
- File DSN/ DSNLess Connections



Notes:

We had a customer who ran into this problem once. Their test box was set to 50 users, and the production box was set to a 5 concurrent user limit.

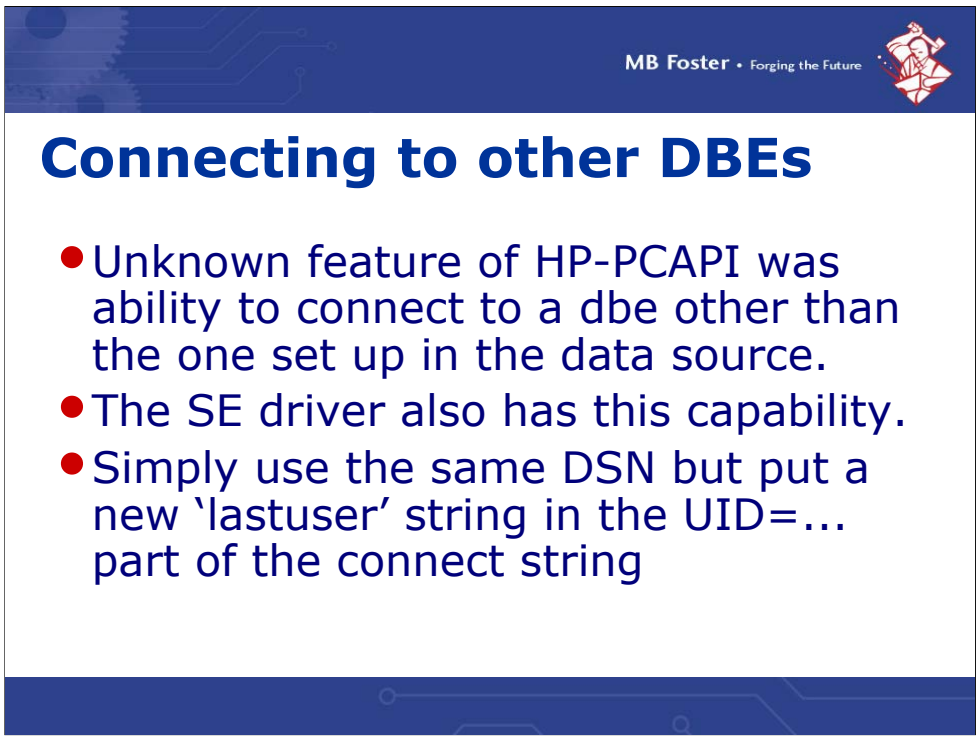
There was a fair bit of man hours spent trying to figure that one out, and it was a simple matter of having different configuration settings.

Queue Tuning

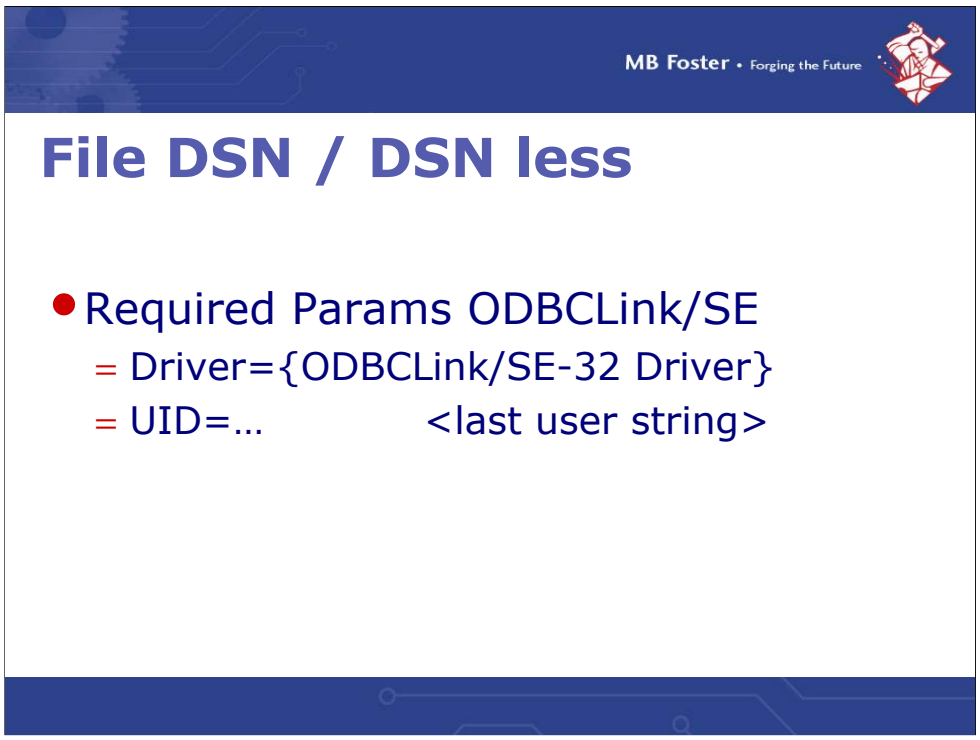
- On the first job statement of the listener job the command 'PRI=CS' is included.
- has no effect unless system manager issues the command 'JOBPRI CS'.
- System manager may elect to tune system queues so batch and online user queues overlap.
- Also possible to set up a queue (E)for the listener to allow special tuning between batch and online.

Notes:.

One way to tell if you are really in the Queue you expect is to look at the start of the spool file. You will see a `Priority = DS;` if you are in the D Queue instead of the C queue.



This ability was added in the early E.57.xx versions and so virtually everyone should have this ability now.



The Last User string is the one that you would have made from a registry DSN connecting to the same datasource. (I recommend making a Registry DSN and then simply Copy and Paste the last user string.into the DSN string)

You can make up your own User Specific LastUser String by prompting the user for login information, provided you put all the information in the proper format.

The reason we require that the LastUser String is placed in the DSN String is to make the SE Driver Compatible with applications that used this hidden feature in HP-PCAPI 16 bit ODBC Driver.

Optional Parameters:

- PWD HostConf PWD (not for SE)
- Debug Client Debug level (0-15)
- MAXSTMT 1 to 48
- Debug_SQLX Host Debug Level 0-5

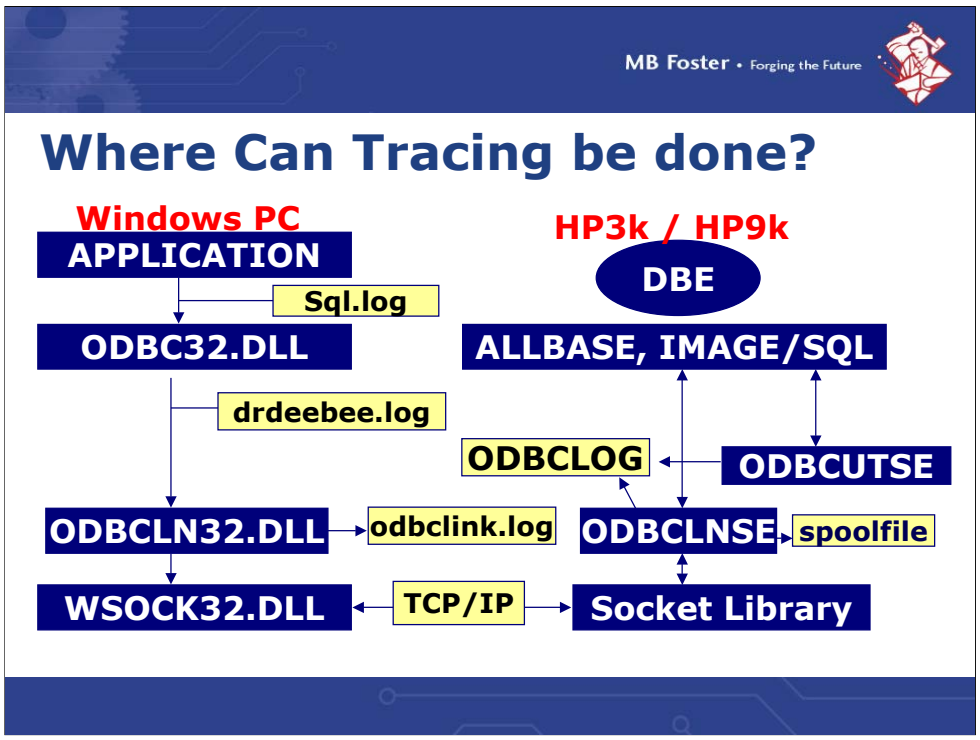
Optional Parameters

- LoginTimeout Connect timeout (msec)
- CommandTimeout Query timeout (msec)
- TranslationDLL Trans DLL location
- TranslationName Trans Name
- TranslationOption Trans option value

Troubleshooting

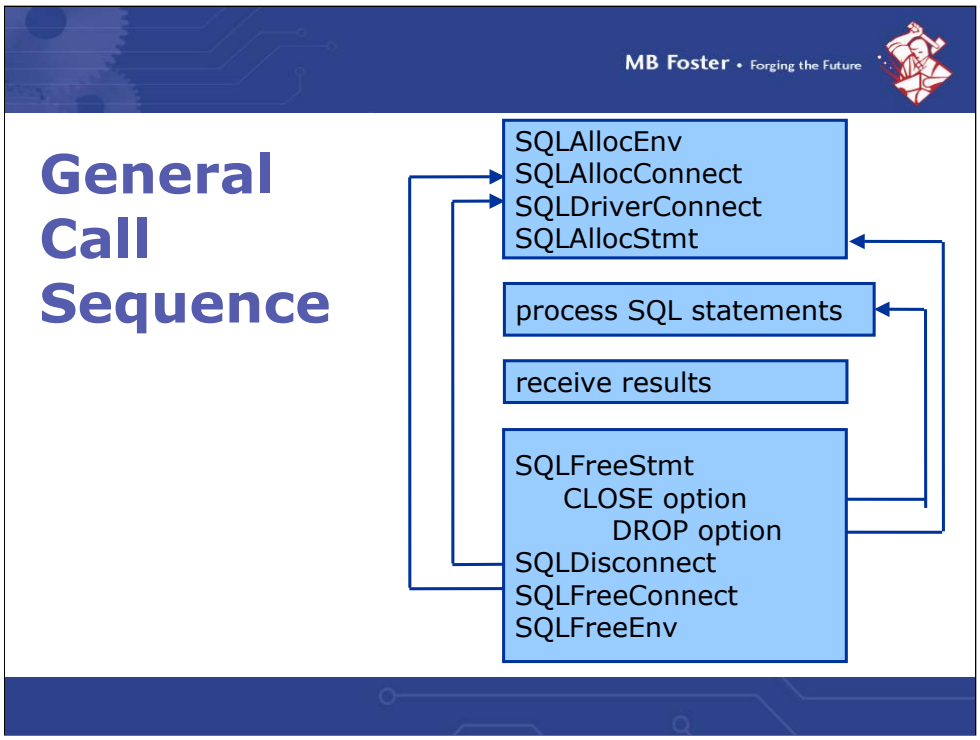
- Client Debugging
- Server Debugging
- Reading Logfile

Notes:



Notes:

Newer versions of the sql.log will show the call both into and out of the odbc32.dll.



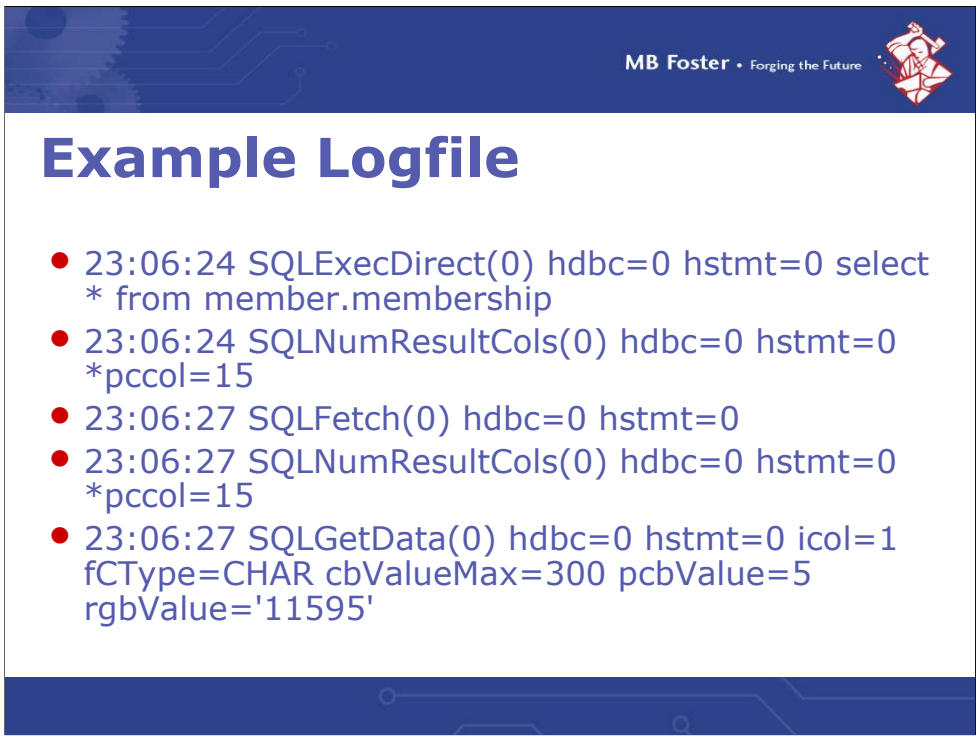
Notes:

This is extremely simplified.

If you look at a typical log of MSAccess or VB doing a single SQL Statement there will be perhaps even hundreds of ODBC calls. Most of them will be for setting and getting Connection or Statement Options.

ODBCLINK.LOG

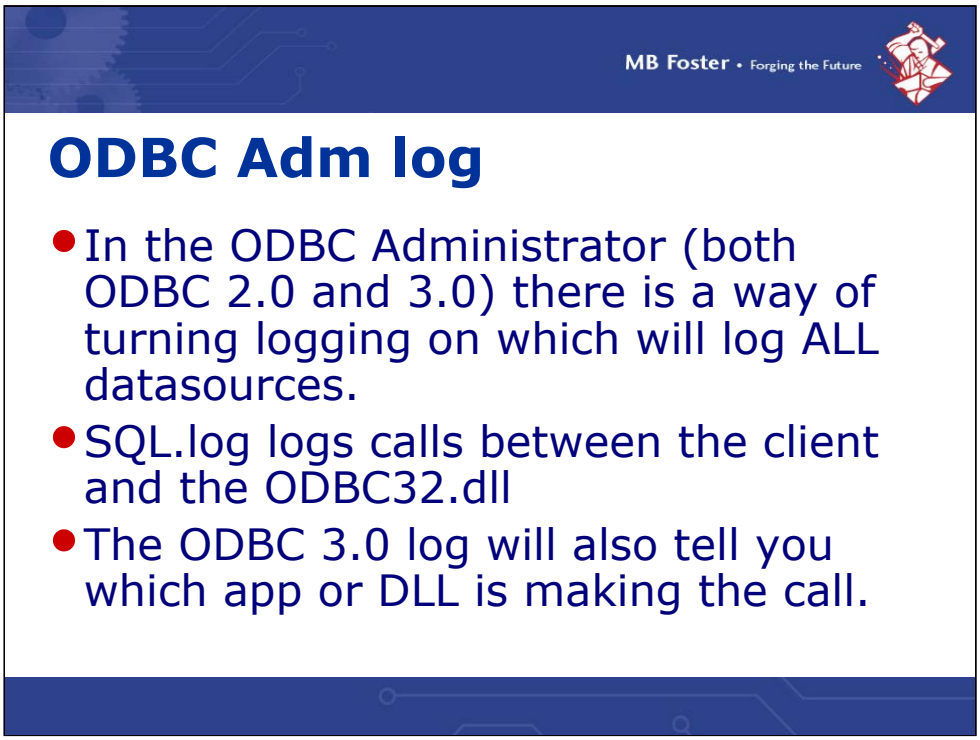
- Created when Debug Client box in the setup is checked.
- logfile 'odbclink.log' located in the app. working directory (eg : MSAccess will use the My Documents directory)
- The registry variable DEBUG for the particular data source you are using will be set to 3. Level 3 is adequate for most users needs, and gives all ODBC Calls.



Here is an example of an odbclink.log of a few ODBC Calls done to execute an SQL Statement and Fetch the first row of data.

What does it all mean?

- Generally¼a client lognote contains
  - = Time - Call (return code) connection_handle, stmt_handle, other call specific information
  - = 14:22:25 SQLColAttributes(0) hdbc=0 hstmt=0 fDescType=7
  - = The most common return codes are:
    - 0 Success
    - -1 Failure
    - 1 Success with Info
    - 100 End of Data

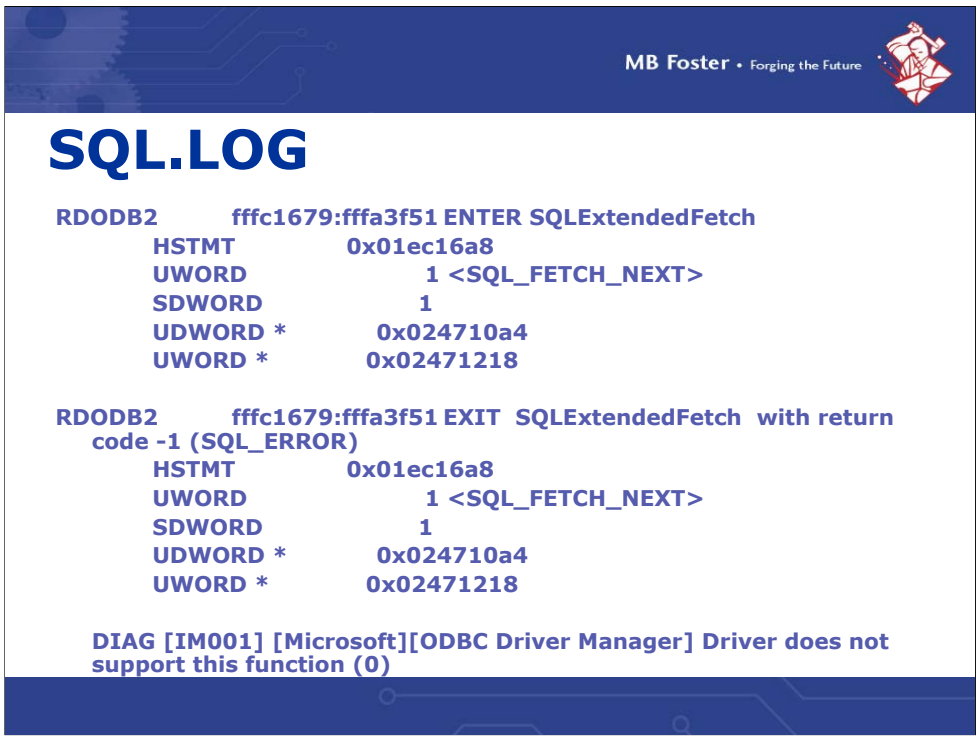


You use the "Tracing" Tab of the Administrator to turn on the sql.log.

The default name is SQL.LOG but you can change it (as well as where the log is written to).

One useful thing about the SQL.log is that you can reduce the output by not turning it on until just before you get to the problem you want to debug (provided you are able to control the app and that the bug happens shortly after some kind of user input)

The control button toggles between "Start Tracing Now" and "Stop Tracing Now"



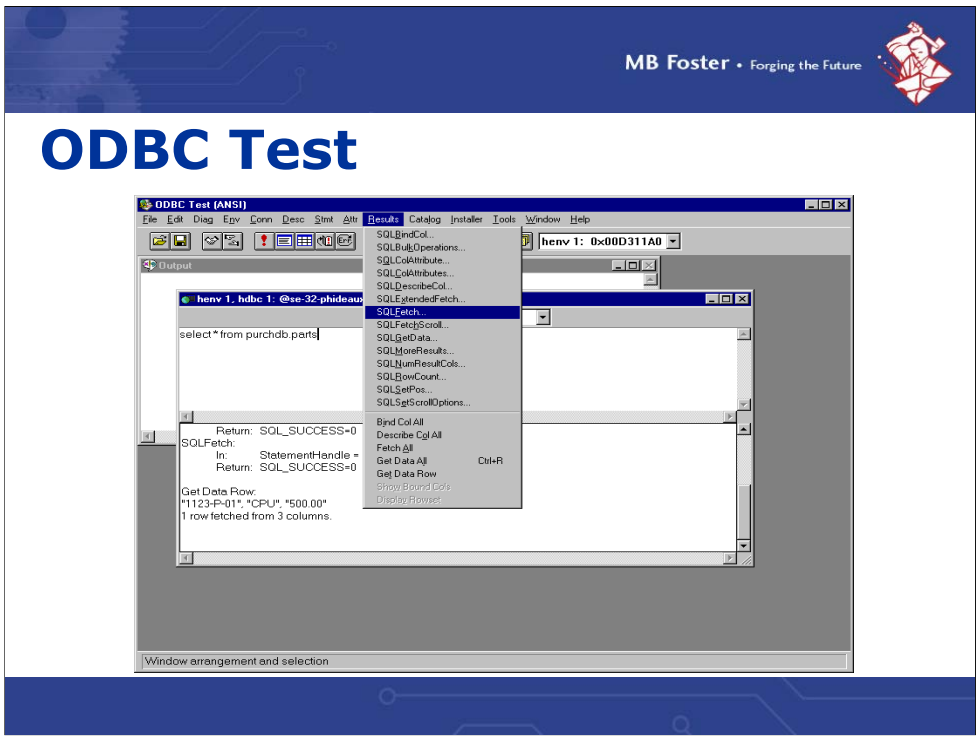
In this case we see a call to SQLExtended Fetch

THE ODBCLink Driver does not support this call.

Provided that the ODBC Cursor Library is installed the odbc32.dll will translate the Extended Fetch call into a series of SQLFetch Calls.

If the ODBC Cursor Library is NOT turned on then you will get an error as shown in this slide.

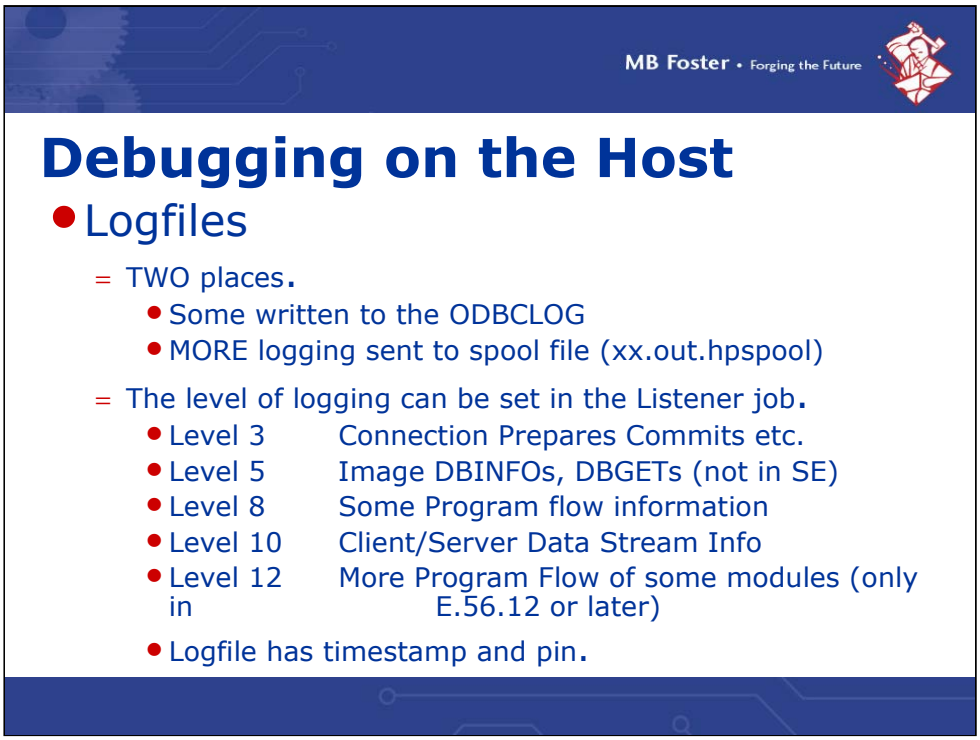
Note that this is one case where if you would have looked in the odbclink.log you would not have seen any indication of the error, This is because the error came from the odbc32.dll not the driver.



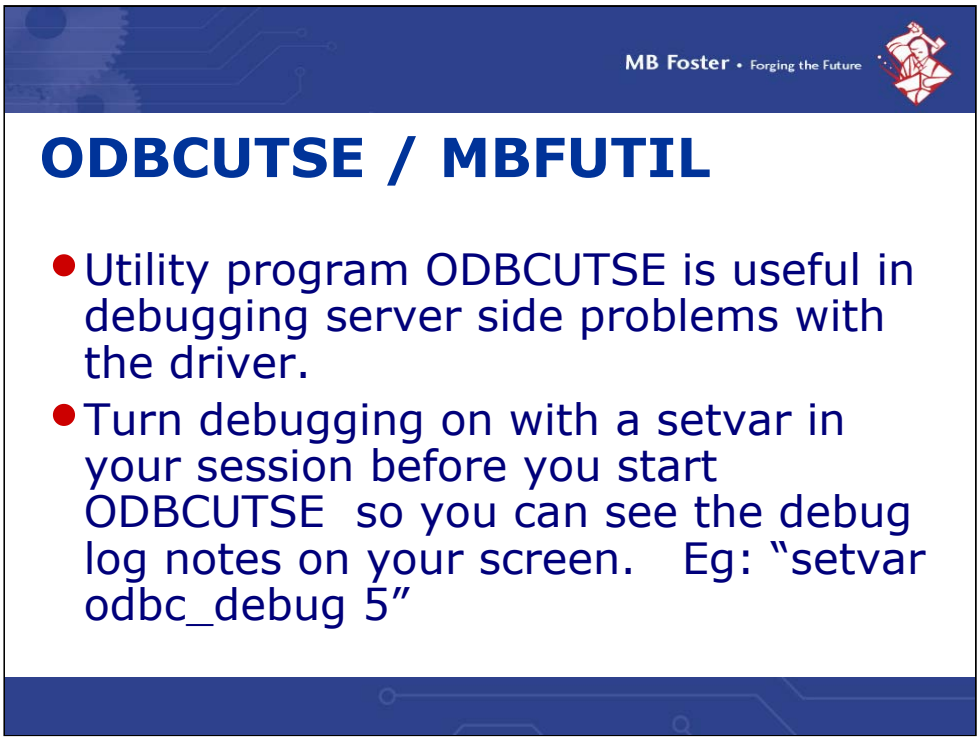
This is one of the Applications we use most in the lab to debug problems.

It will allow you to call any ODBC call.

You can connect as an ODBC2 Application or an ODBC3 Application



As mentioned before level 3 is the default(if you click debug_sqlx in the setup) There are a few debug levels even higher than 12 but these are rarely used.



There are only a few commands that' are likely to be of interest. We use this program often at the lab to debug server side problems with the x debugger, in which case we use the unoptimized compile.

To get a list of commands type help.

You only need to enter enough of a command to differentiate it from another command (usually 2 letters is enough)

eg sh for show (to show a list of tables) or SH <owner.table>

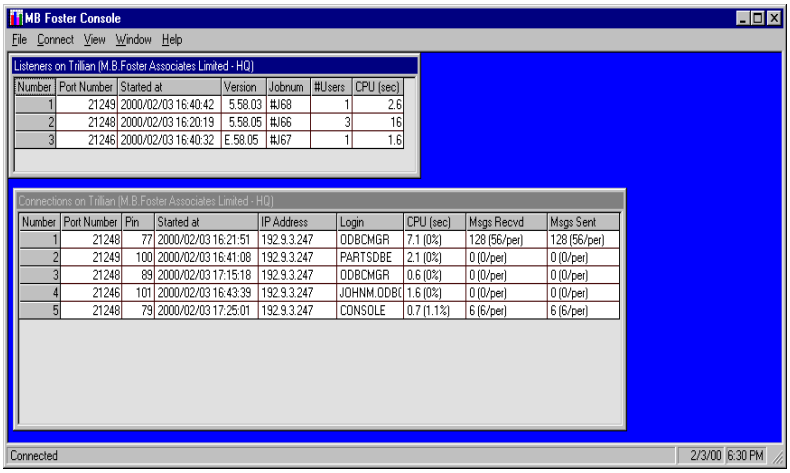
to show details of a specific table.

You can then enter select statements and then a FE to fetch a row

or FE ALL to fetch all the rows

MB Foster • Forging the Future

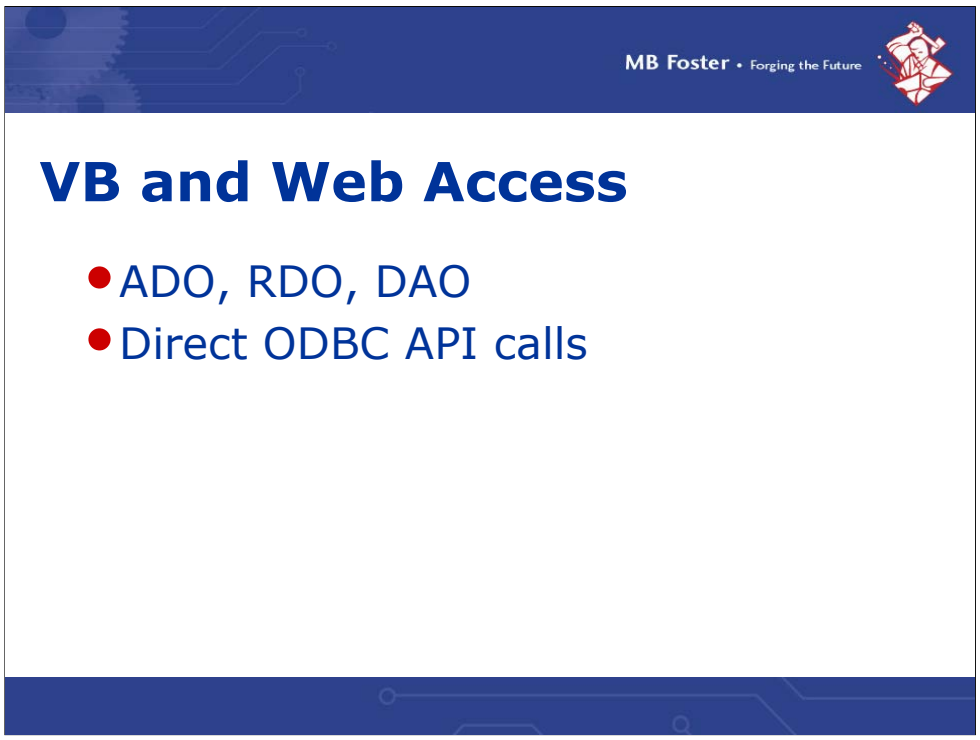
Managing the Connections



MB Foster has created a Management Console for the ODBCLink/SE and UDALink Drivers that will allow you to View all listeners and all connections for each listener.

In the example above the Console shows that there are 3 Listeners running, two UDALink Listeners and one ODBCLInk/SE Listener.

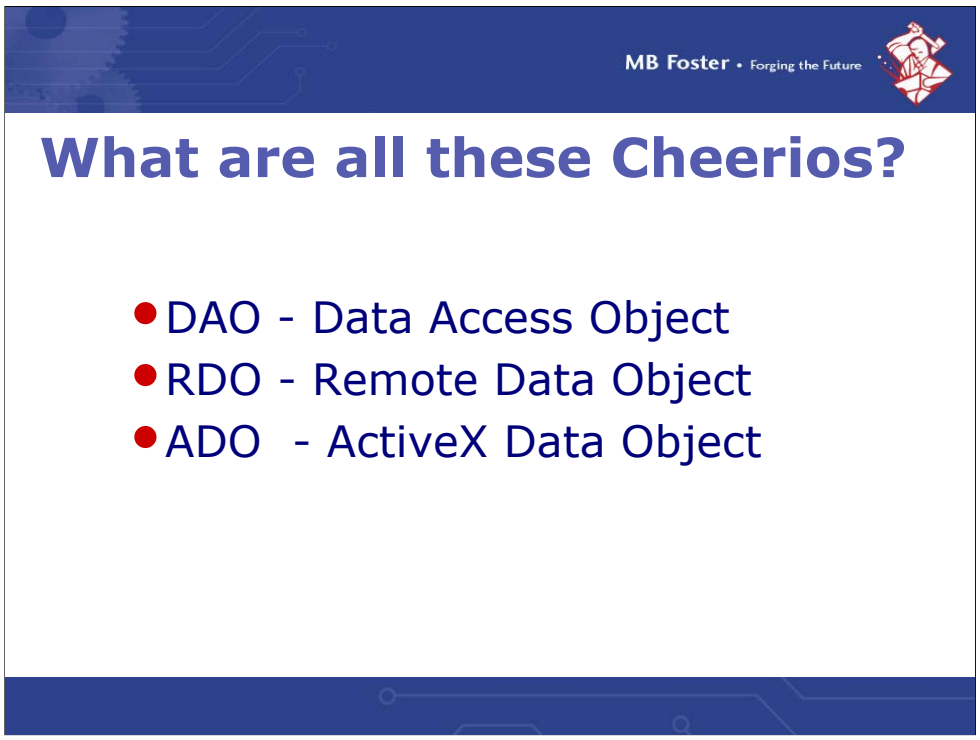
It also provides some statistics about each connection and allows the manager to turn debugging on for a particular connection, and review the server logging for a particular connection, or even to kill a connection.



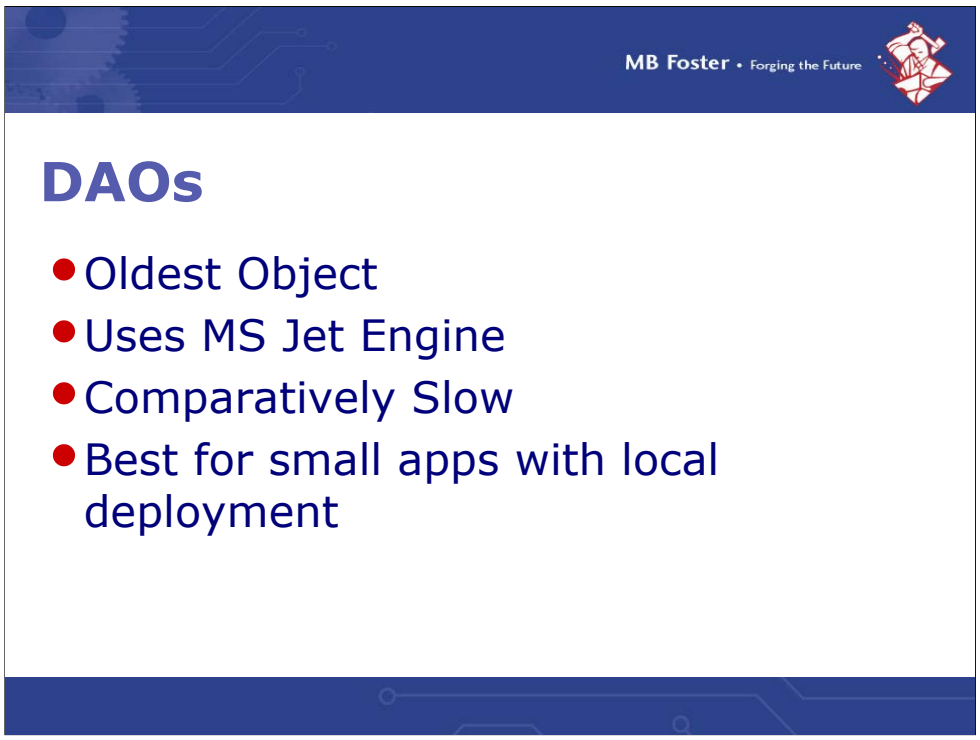
There are several types of connections that can be made using ODBC with Visual Basic. These include ADOs, RDOs, and DAOs. You can also explicitly code your own ODBC Calls if desired.

Direct API

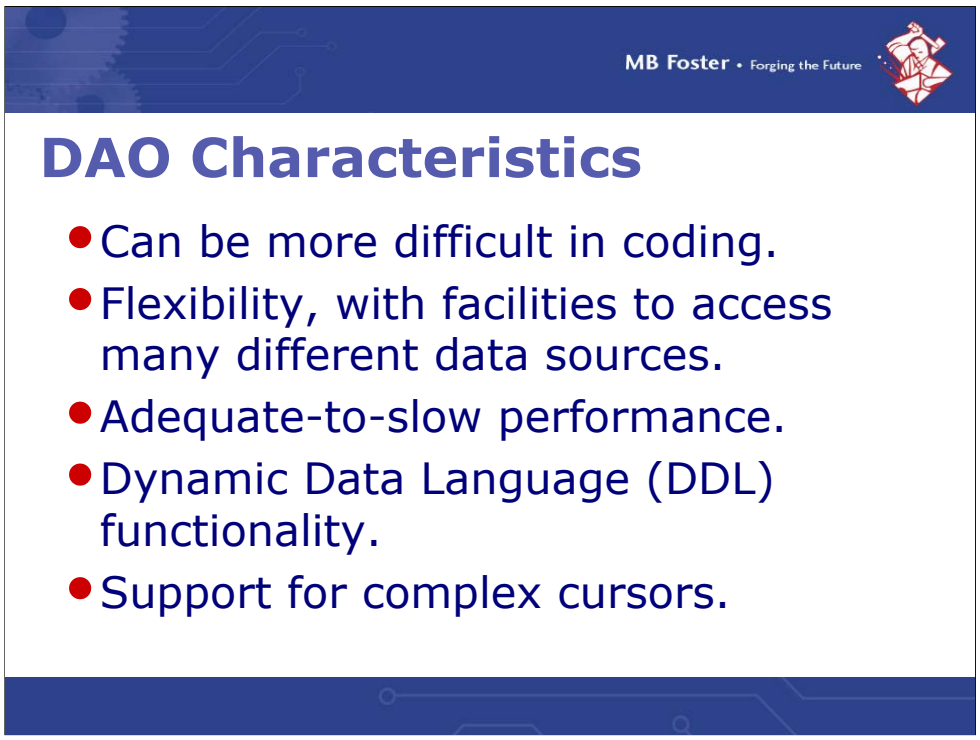
- Fastest access
- Include only call you want/need
- Tighter control
- Can be more complex



(Data Access Objects) was the first object-oriented interface that exposed the Microsoft Jet database engine (used by Microsoft Access) and allowed Visual Basic developers to directly connect to Access tables - as well as other databases - through ODBC. DAO is suited best for either single-system applications or for small, local deployments.



(Data Access Objects) was the first object-oriented interface that exposed the Microsoft Jet database engine (used by Microsoft Access) and allowed Visual Basic developers to directly connect to Access tables - as well as other databases - through ODBC. DAO is suited best for either single-system applications or for small, local deployments.



Data Access Objects (DAO) provides data access to native Microsoft Jet engine databases (.mdb files), selected ISAM databases, and any ODBC data source.

Historically, DAO is a popular solution when it comes to using Microsoft® Access (.mdb) and ISAM data sources such as Btrieve, FoxPro, Paradox, and dBase.

The general characteristics of DAO are:

· More Difficulty in coding.

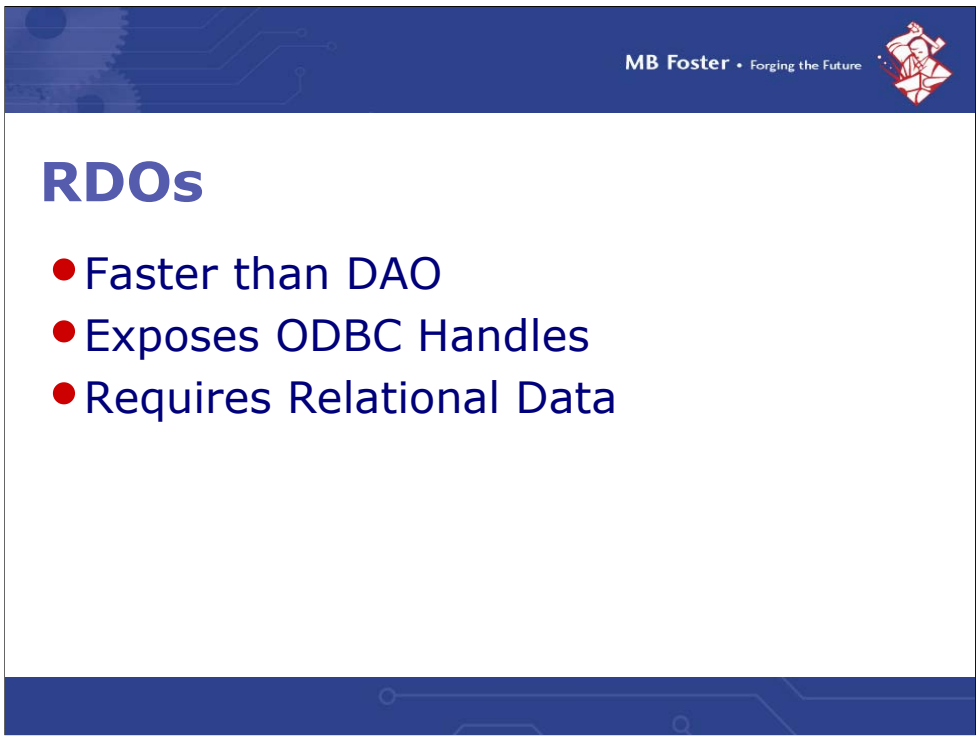
· Flexibility, with facilities to access many different data sources.

· Adequate-to-slow performance.

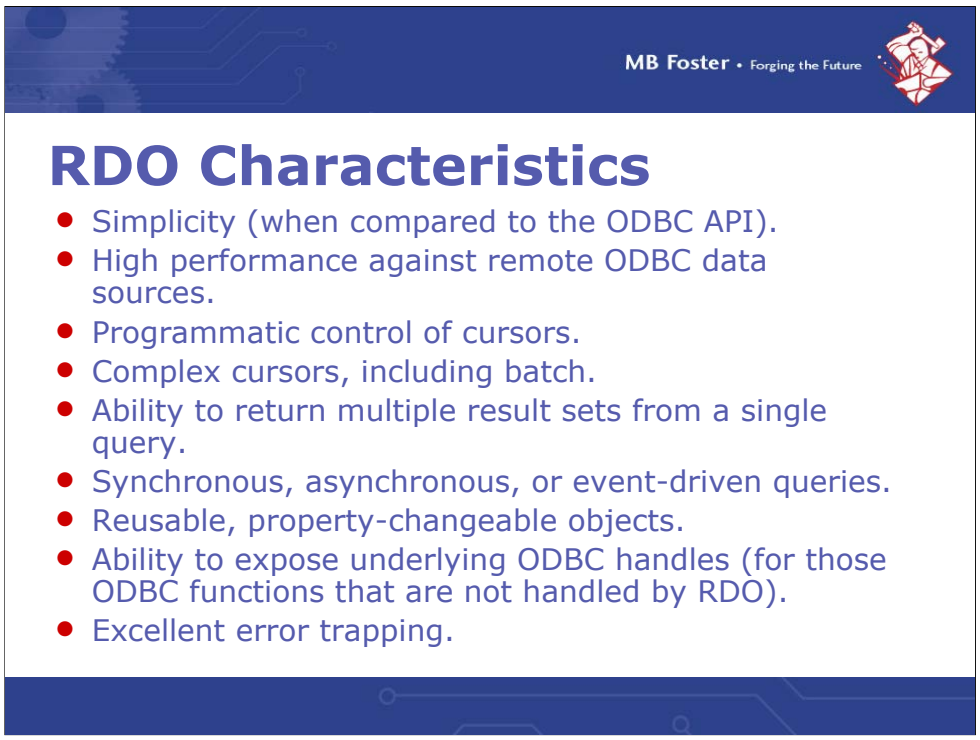
· Dynamic Data Language (DDL) functionality.

· Support for complex cursors.

Compared to the newer ActiveX Data Objects (ADO) or Remote Data Objects (RDO) technologies, Data Access Objects (DAO) is a slower, less capable data access alternative. DAO (and its companion, the Microsoft Jet database engine) was originally designed to handle remote ISAM data access. DAO is tied to the Microsoft Jet engine because it uses the Microsoft Jet engine query and result set processors.



(Remote Data Objects) is an object-oriented data access interface to ODBC combined with the easy-to-use style of DAO, providing an interface that exposes virtually all of ODBC's low-level power and flexibility. RDO is limited, though, in that it doesn't access Jet or ISAM databases very well, and that it can access relational databases only through existing ODBC drivers. However, RDO has proven to be the interface of choice for a large number of SQL Server, Oracle, and other large relational database developers. RDO provides the objects, properties, and methods needed to access the more complex aspects of stored procedures and complex result sets.



Remote Data Objects (RDO) is designed to access remote ODBC relational data sources, and makes it easier to use ODBC without complex application code. RDO is a means of accessing SQL Server, Oracle, or any relational database that is exposed with an ODBC driver.

The general characteristics of RDO are:

· Simplicity (when compared to the ODBC API).

· High performance against remote ODBC data sources.

· Programmatic control of cursors.

· Complex cursors, including batch.

· Ability to return multiple result sets from a single query.

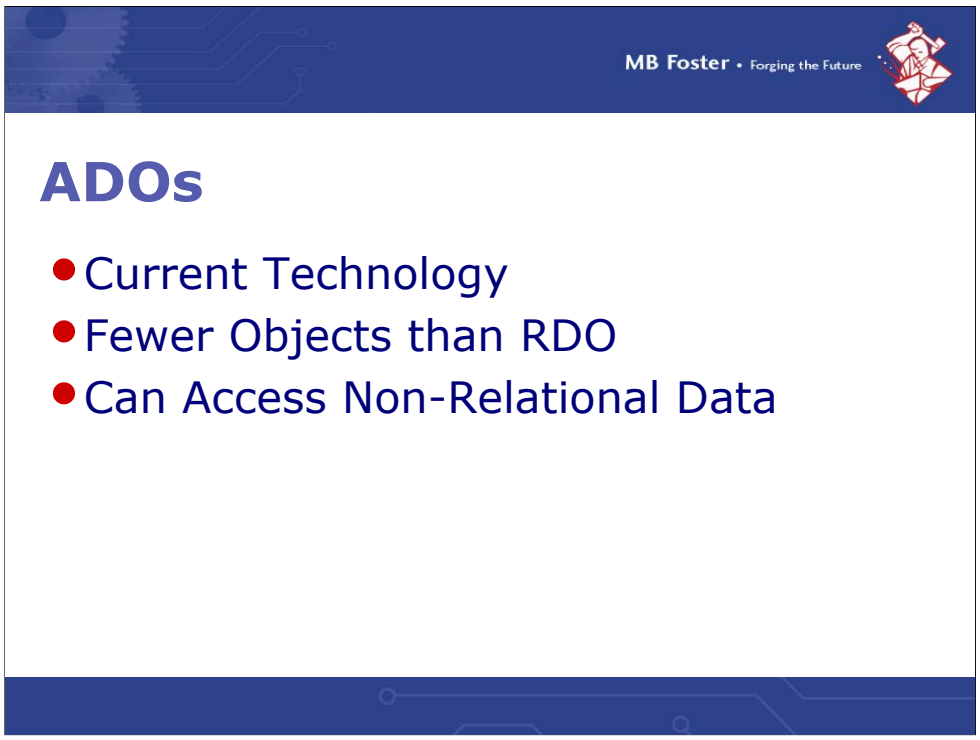
· Synchronous, asynchronous, or event-driven query execution.

· Reusable, property-changeable objects.

· Ability to expose underlying ODBC handles (for those ODBC functions that are not handled by RDO).

· Excellent error trapping.

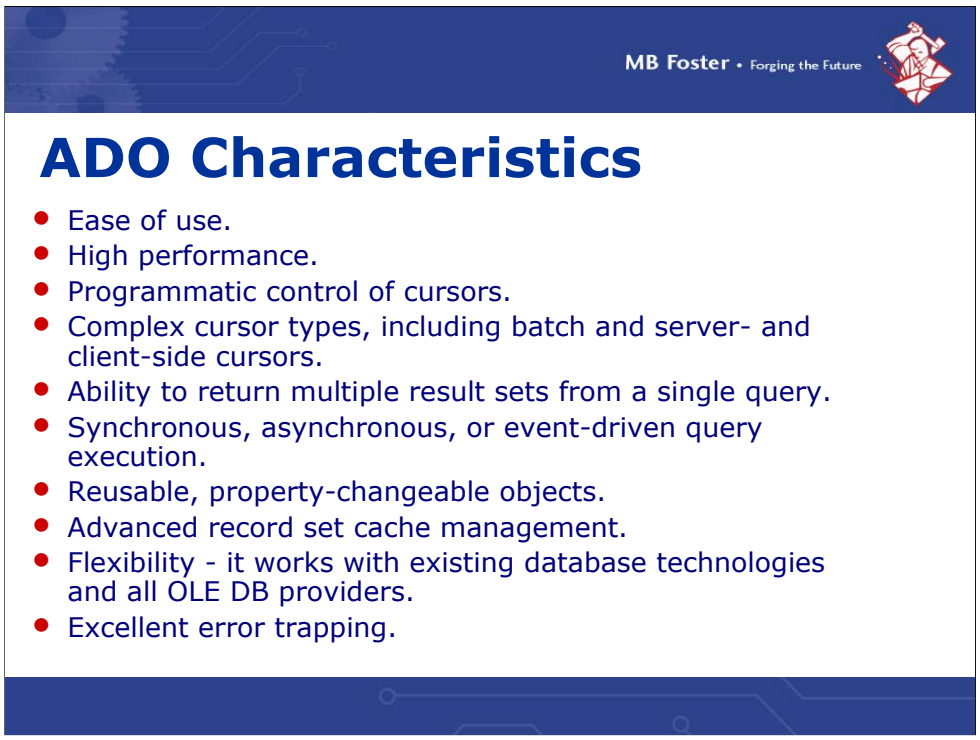
Compared to the older Data Access Objects (DAO) technology, RDO is a smaller, faster, more sophisticated alternative. RDO is especially capable of building and executing queries against stored procedures and handling all types of result sets, including those generated by multiple result set procedures, those returning output arguments and return status, and those requiring complex input parameters.



(Active-X Data Object) is the successor to DAO/RDO. Functionally ADO 2.0 is most similar to RDO, and there's generally a similar mapping between the two models. ADO "flattens" the object model, containing fewer objects and more properties, methods (and arguments), and events. For example, ADO has no equivalents to the rdoEngine and rdoEnvironment objects, which exposed the ODBC driver manager and hEnv interfaces. Nor can you currently create ODBC data sources from ADO, despite the fact that your interface might be through the ODBC OLE DB service provider.

Much of the functionality contained in the DAO and RDO models was consolidated into single objects, making for a much simpler object model. Because of this, however, you might initially find it difficult to find the appropriate ADO object, collection, property, method, or event. Unlike DAO and RDO, although ADO objects are hierarchical, they can also be created outside the scope of the hierarchy.

It should be noted that ADO currently doesn't support all of DAO's functionality. ADO mostly includes RDO-style functionality to interact with OLE DB data sources, plus remoting and DHTML technology. ADO doesn't currently support data definition (DDL), users, groups, and so forth.



ActiveX Data Objects (ADO) is designed to be an easy-to-use application-level interface to any OLE DB data provider, including relational and non-relational databases, e-mail and file systems, text and graphics, and custom business objects, as well as existing ODBC data sources.

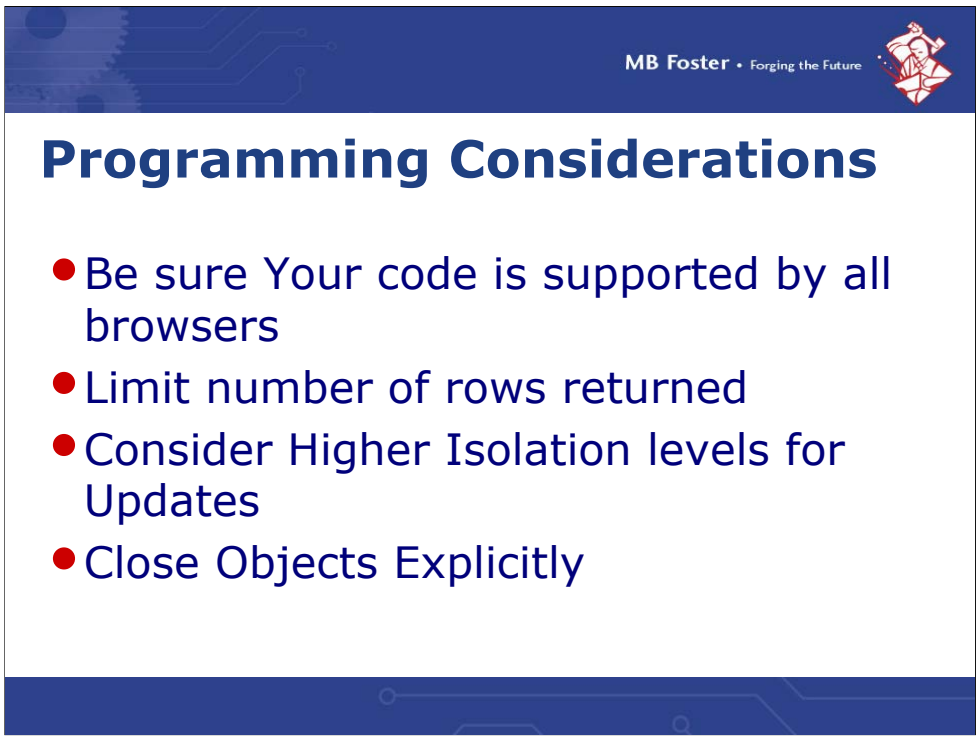
ADO is implemented for minimal network traffic in key Internet scenarios, and a minimal number of layers between the front-end and data source all to provide a small, high-performance interface. ADO is called the OLE Automation interface. And ADO uses conventions and features similar to DAO and RDO, simplified that make it easy to learn.

ADO isn't automatically code compatible with your existing data access applications. While ADO encapsulates the functionality of DAO and RDO, you must convert many of the language elements over to ADO syntax. In some cases, this will mean only a simple conversion of some functions of your existing code. In other cases, it might be best to rewrite the application using ADO's new features.

ADO is easy to use, language-independent, implemented with a small footprint, uses minimal network traffic, and has few layers between the client application and the data source - all to provide lightweight, high-performance data access.

The general characteristics of ADO are as indicated above

The simple semantics of ADO and universal application mean minimal developer training, rapid application development, and inexpensive maintenance.



Programming Considerations

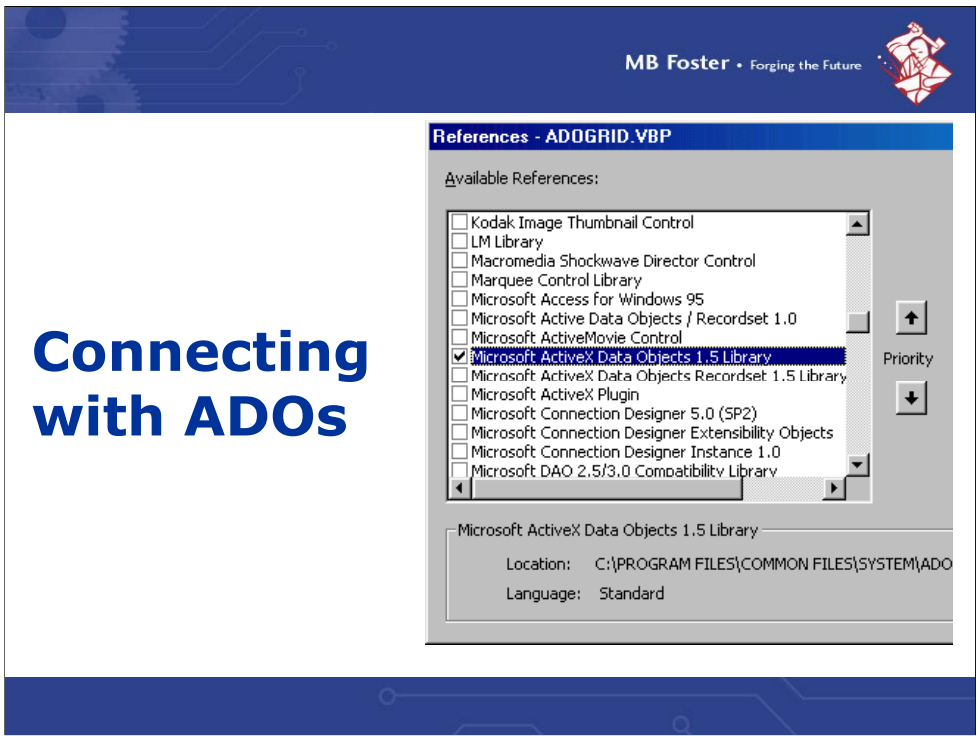
When designing an application in which uses ODBC to gain access to data on the server there are a number of things to be considered BEFORE you start coding your application.

- You can use the MSFlexGrid in an ASP but not all browsers support it.

- If you are connecting to a DB that has thousands of rows you should code your select statements or set the statement options to limit the amount of the rows that will come back. If you can reduce the number of rows returned your application will run faster.

-If you are creating an application that could have many users updating data you will want a higher isolation level than if you are simply reading data.

Explicitly close all your objects (with the .Close Method), rather than assuming they will be closed when the form or Application Terminates. This will reduce the chance of memory leaks



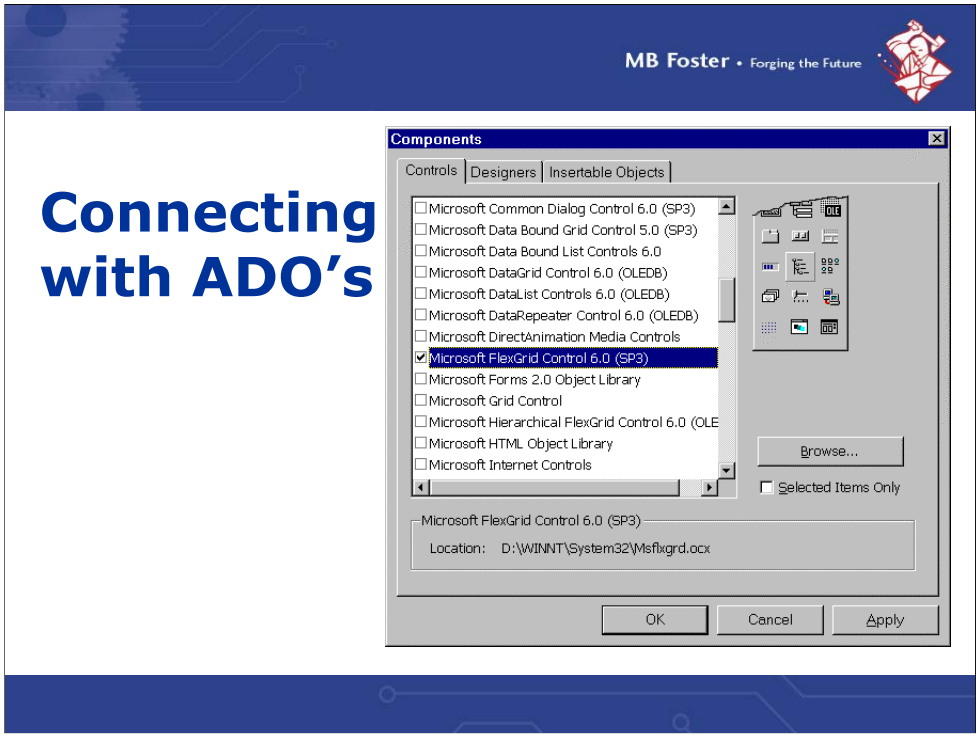
VBA/VB5/VB6:

ADO objects can be created with the DIM statement and/or the CreateObject(ProgID) function. ProgID names the desired class.

Make sure you have installed Microsoft ADO objects on your system. If you have done so then Microsoft ActiveX Data Objects ver X.X Library can be selected in the references dialog of the tools menu, as shown above.

To add the object reference for ADO X.X …(X.X is version number):

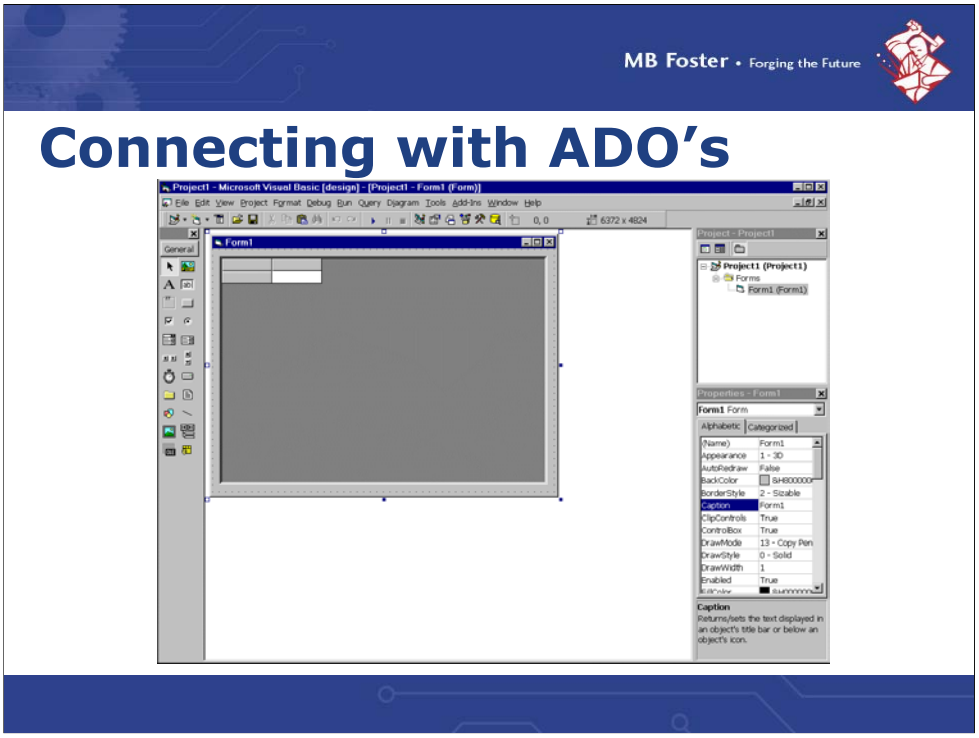
1. On the menu bar go to ‘Project’
2. Select ‘References’.
3. Find and select "Microsoft ActiveX Data Objects X.X".



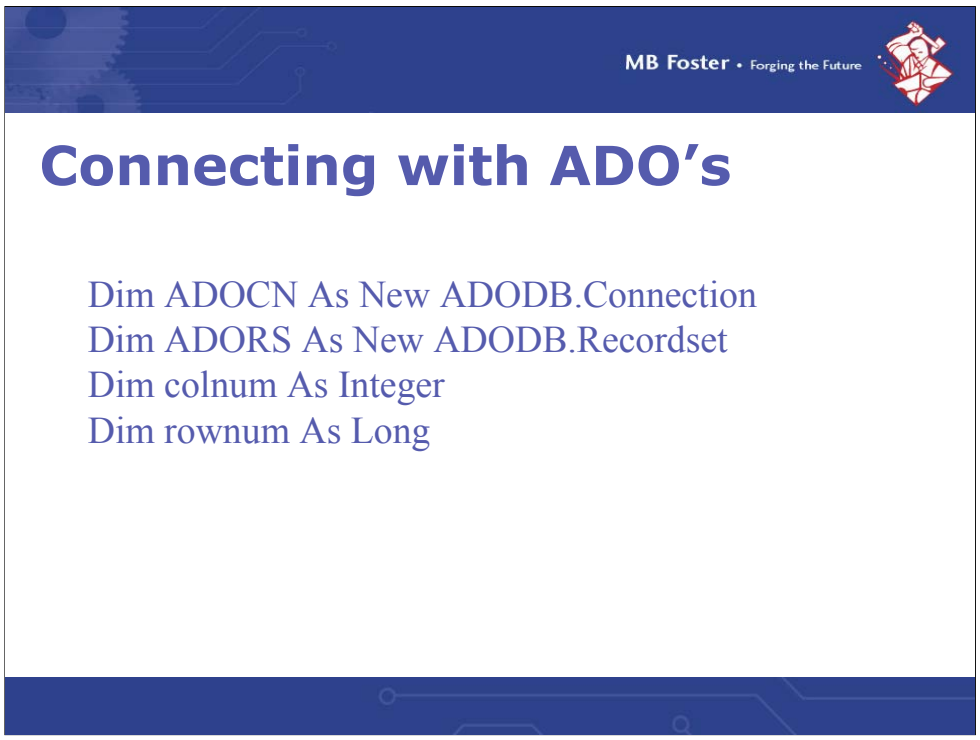
For our example we will use the MSFLEXGRID.

Add the component MSFLEXGRID to the tool bar.

1. On the menu bar go to 'Project'
2. Select 'Components'.
3. Find and select the Microsoft Flex Grid Control.

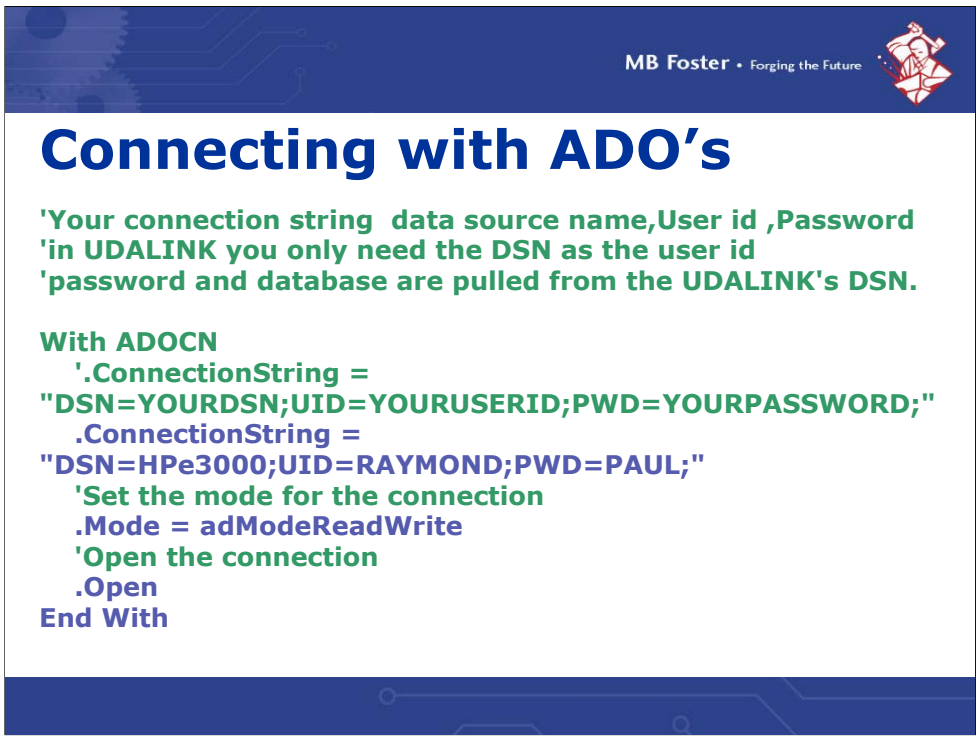


Add a MSFlexGrid (the yellow grid icon you just added to the tool bar) to the form. You should then resize it to fill most of your form.



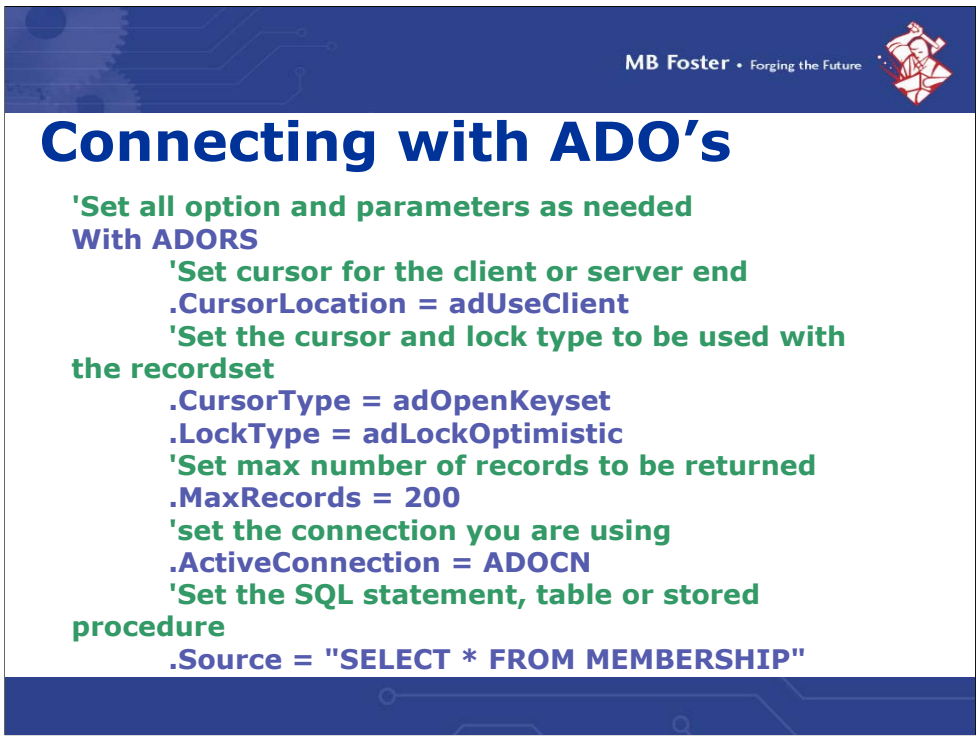
DECLARE YOUR VARIABLES FOR THE CONNECTION

This can be done in the 'Form Load' or the 'General Declarations" section.

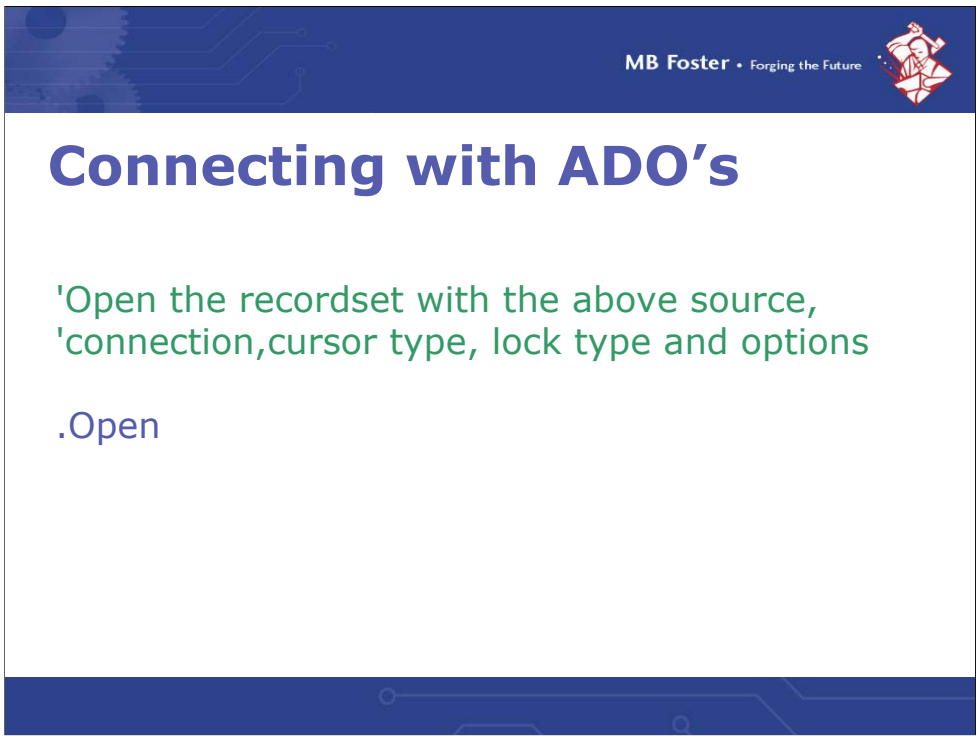


OPEN THE CONNECTION TO THE DATABASE

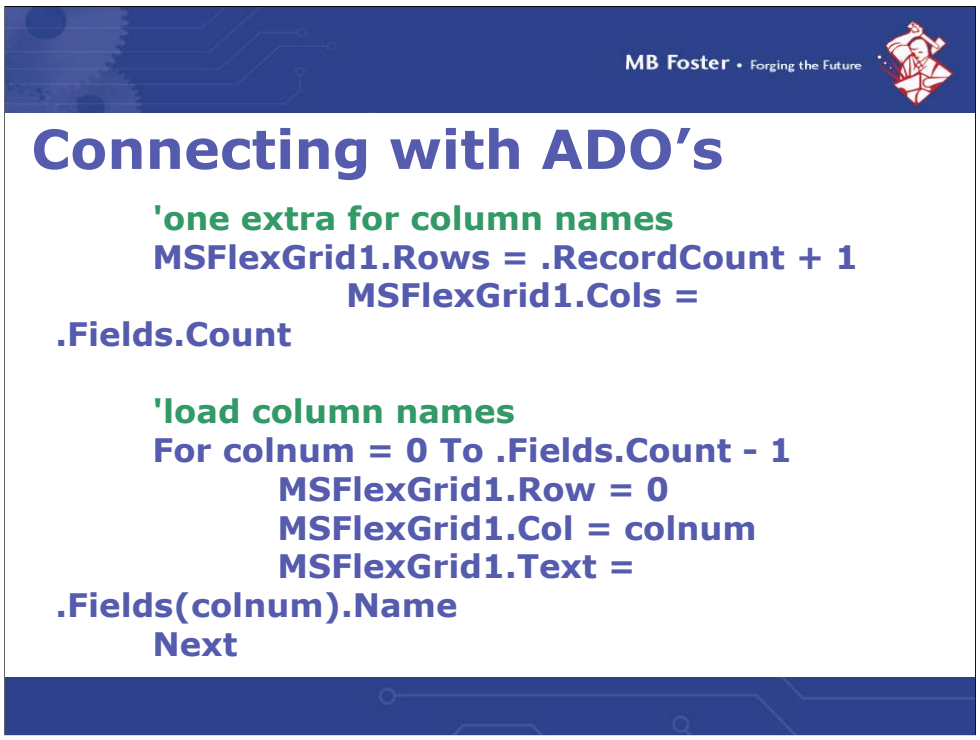
Do this in the 'Form Load' section.



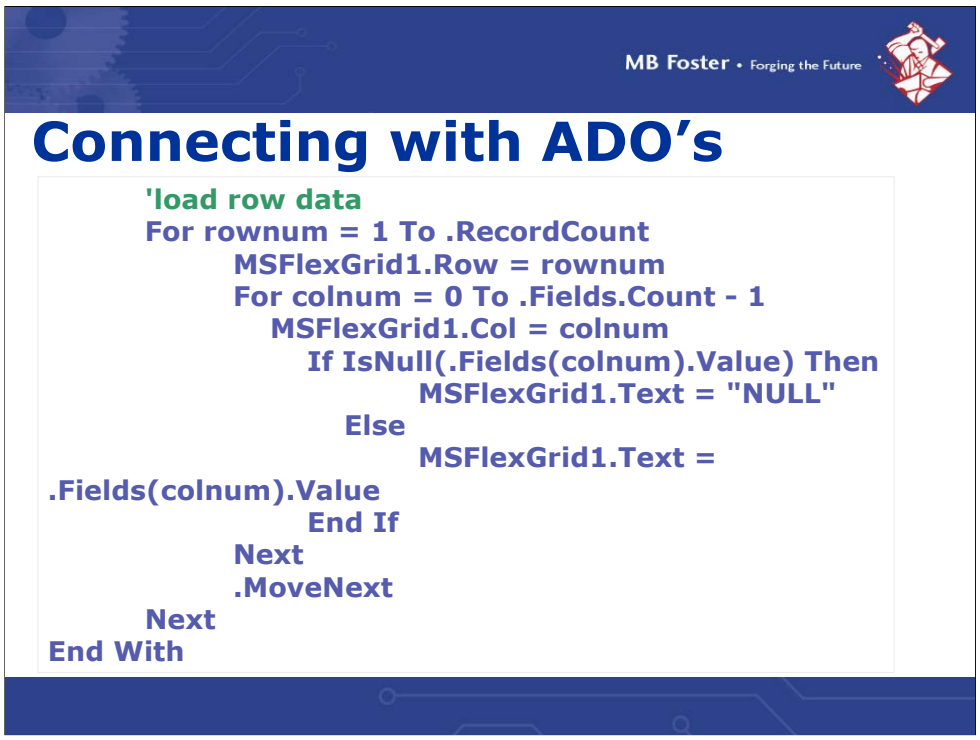
SET THE ADO RECORDSET OPTIONS



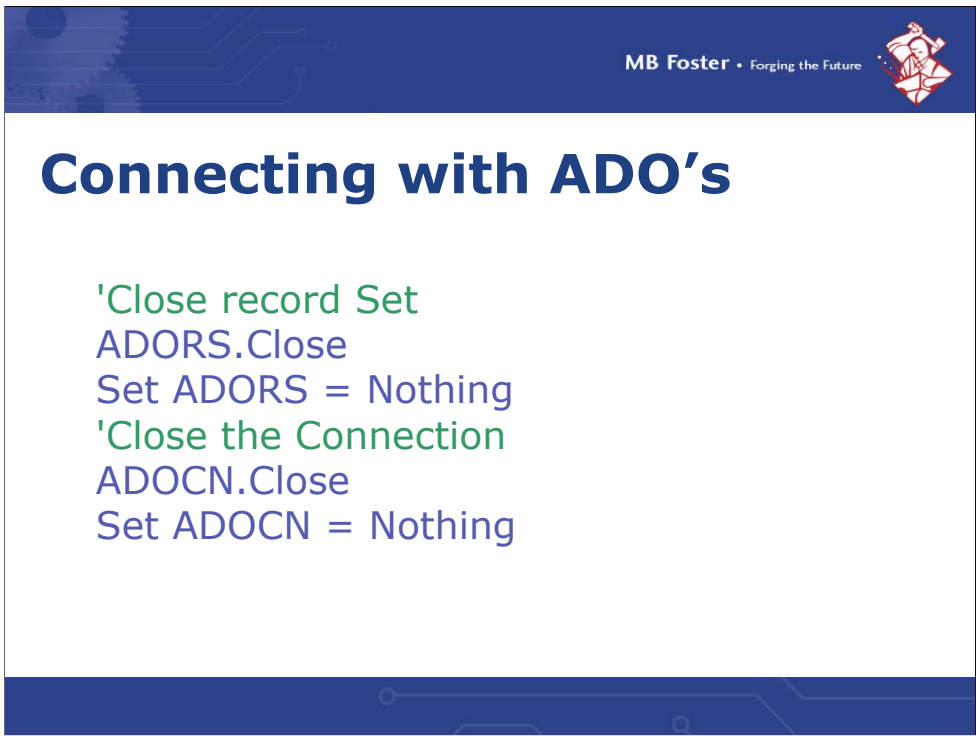
OPEN THE ADO RECORDSET



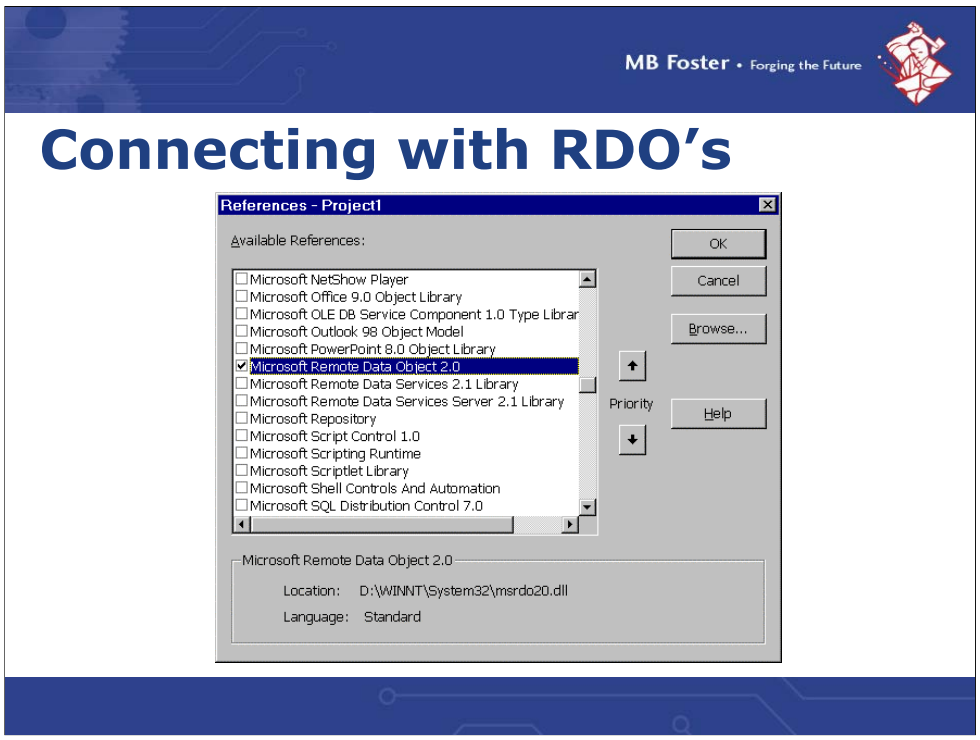
USE THE RECORDSET INFORMATION YOU NEED AND PROCESS IT ACCORDINGLY



USE THE RECORDSET INFORMATION YOU NEED AND PROCESS IT ACCORDINGLY



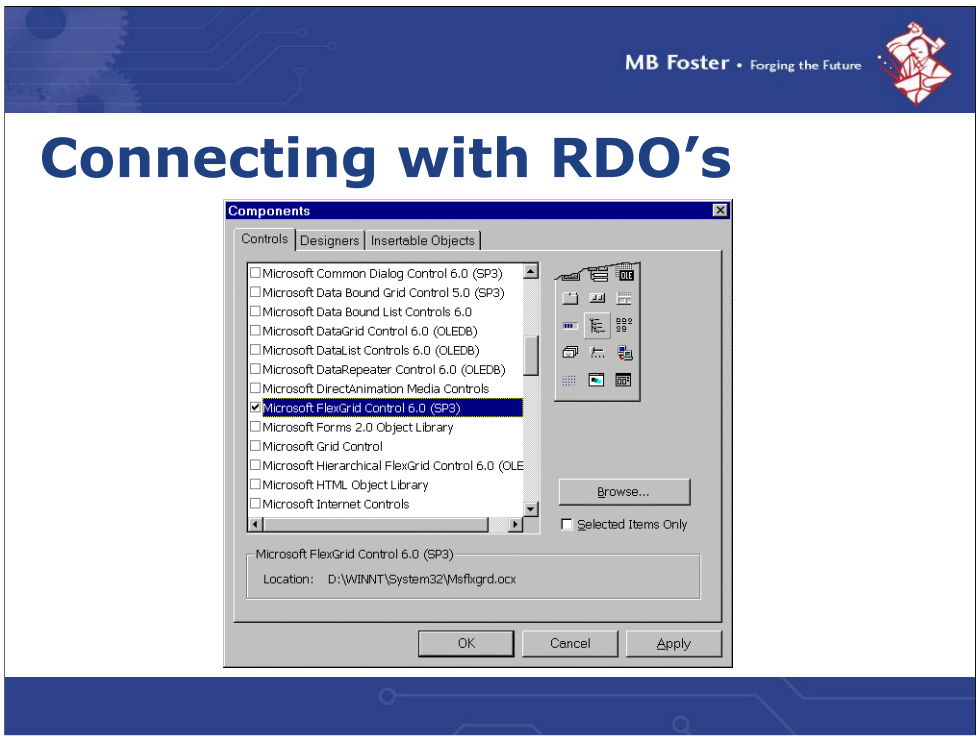
MAKE SURE YOU CLOSE ANY ODJECTS THAT YOU HAVE CREATED WHEN ERRORS OCCUR, OR WHEN EXITING



You must ADD the object reference for RDO 2.0.

To add the object reference for RDO 2.0

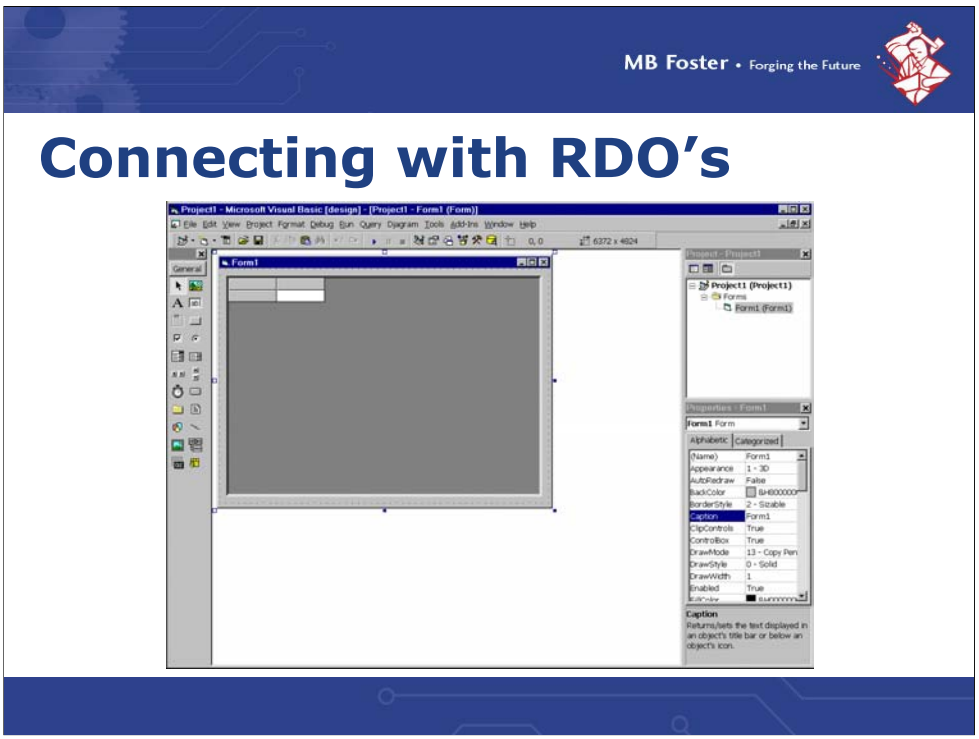
1. On the menu bar go to 'Project'
2. Select 'References'.
3. Find and select "Microsoft Remote Data Objects 2.0".



For our example we will use the MSFLEXGRID.

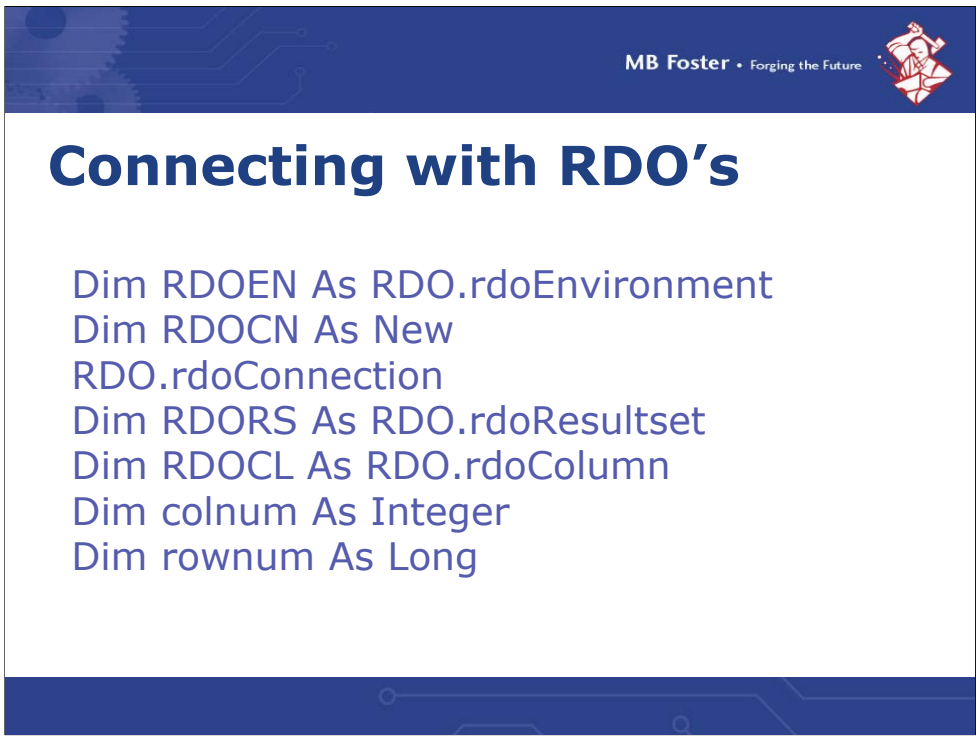
Add the component MSFLEXGRID to the tool bar.

1. On the menu bar go to 'Project'
2. Select 'Components'.
3. Find and select the Microsoft Flex Grid Control.



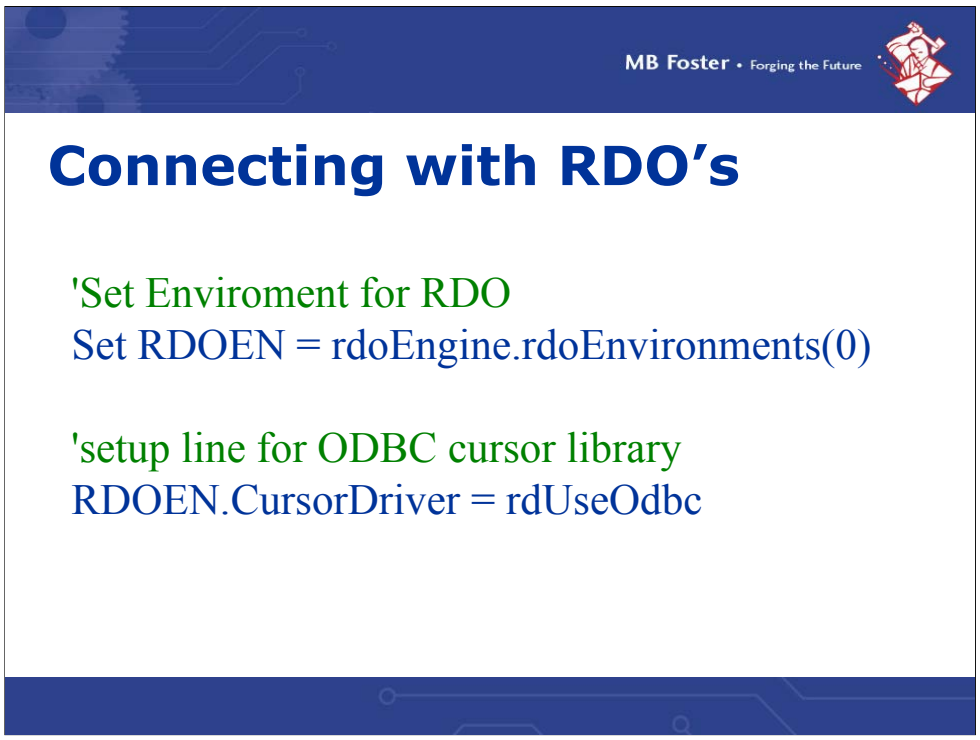
Connecting with RDO's

Add a MSFlexGrid (the yellow grid icon you just added to the tool bar) to the form. You should then resize it to fill most of your form.



DECLARE YOUR VARIABLES FOR THE CONNECTION

This can be done in the 'Form Load' or the 'General Declarations" section.



SET THE RDO ENIVROMENT

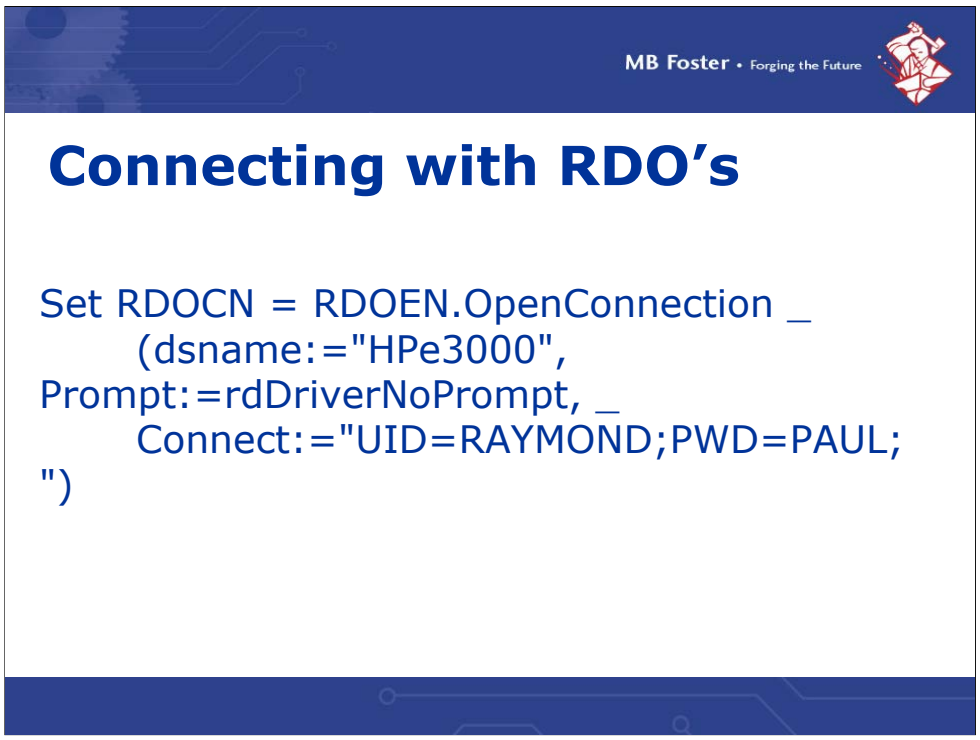
Do this in the 'Form Load' section.

RDOs generate calls to SQLExtendedFetch.

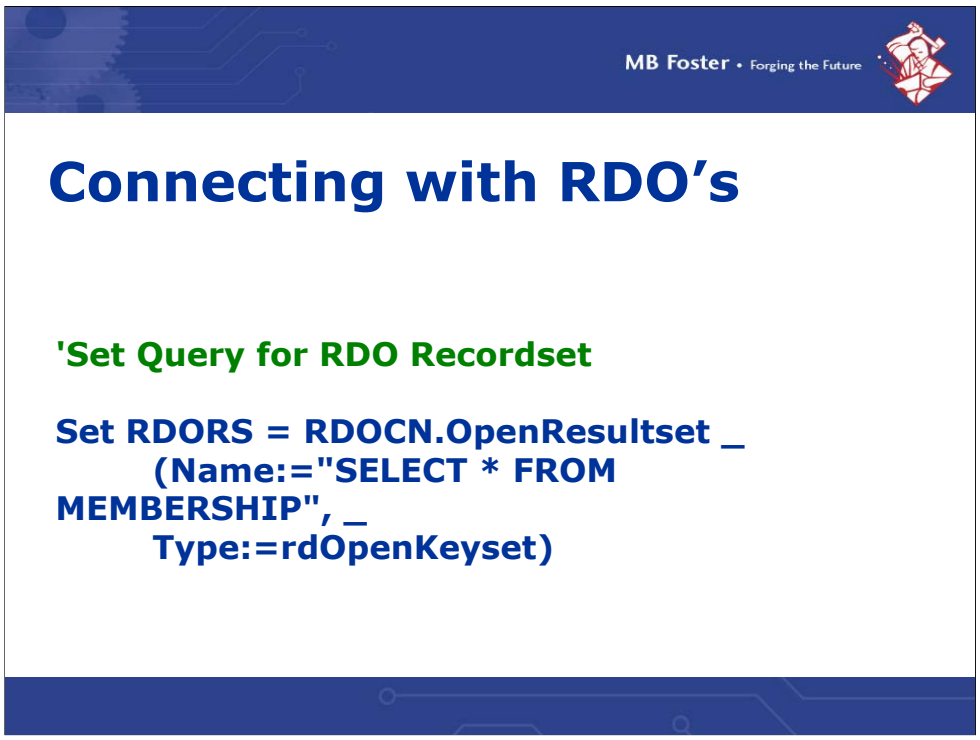
If it is not implemented in the driver, you can use the Microsoft ODBC Cursor Library to handle the SQLExtendedFetch call. It will translate the extended fetch into multiple fetches.

To do this you must set the object properties prior to connecting using the following line:

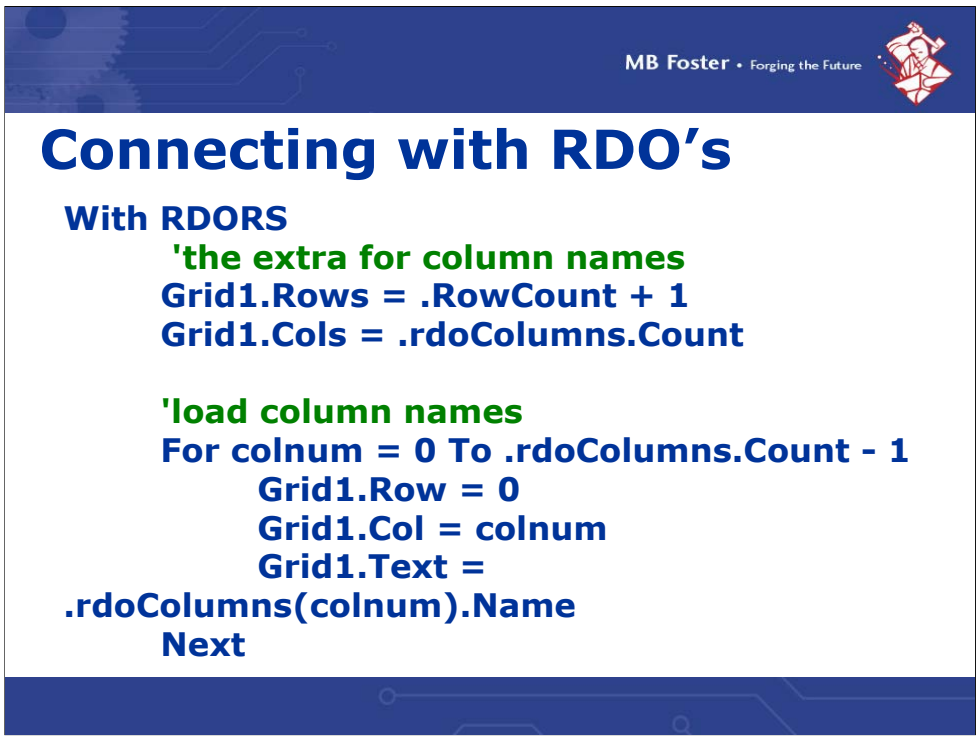
rdoEngine.rdoEnvironments(0).CursorDriver = rdUseOdbc



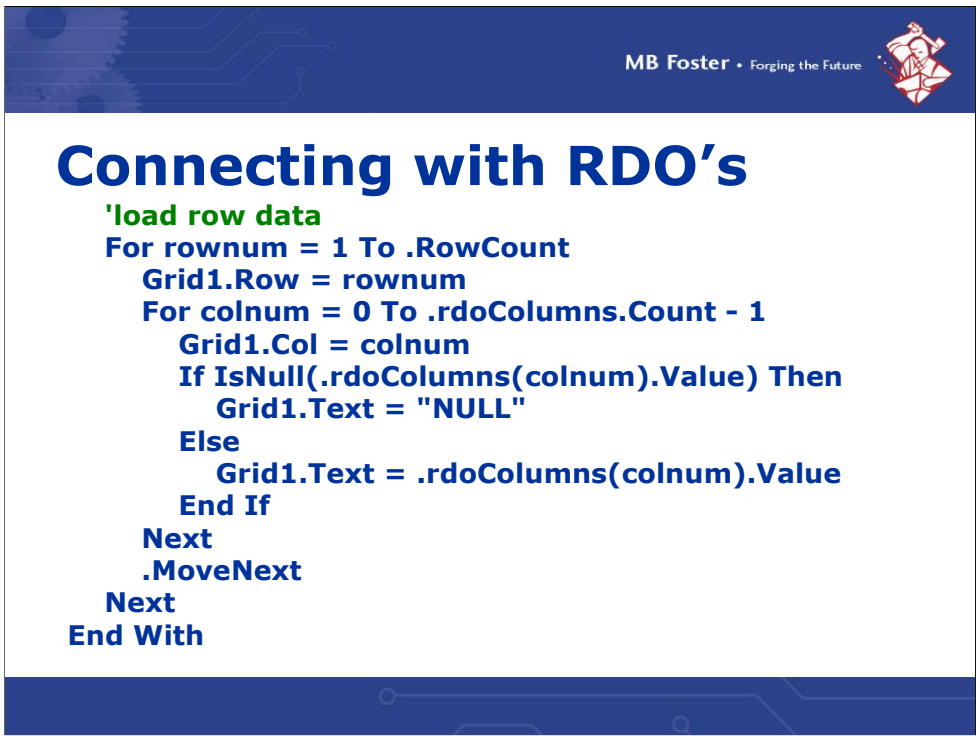
SET AND OPEN THE CONNECTION TO THE DATABASE



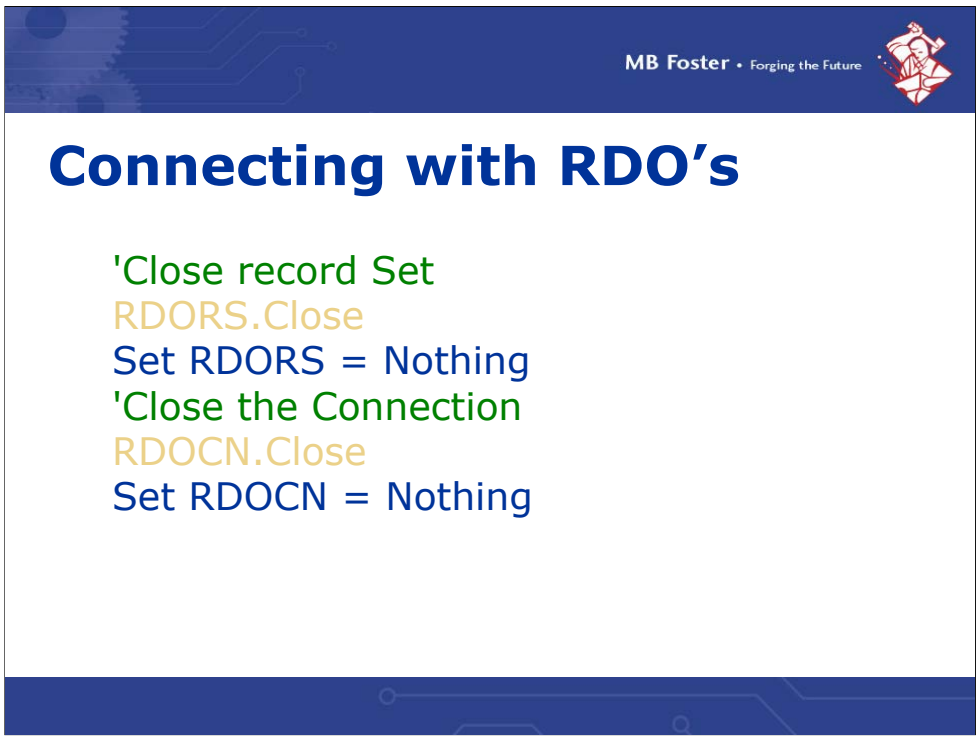
SET AND OPEN THE RECORDSET FOR THE TABLE



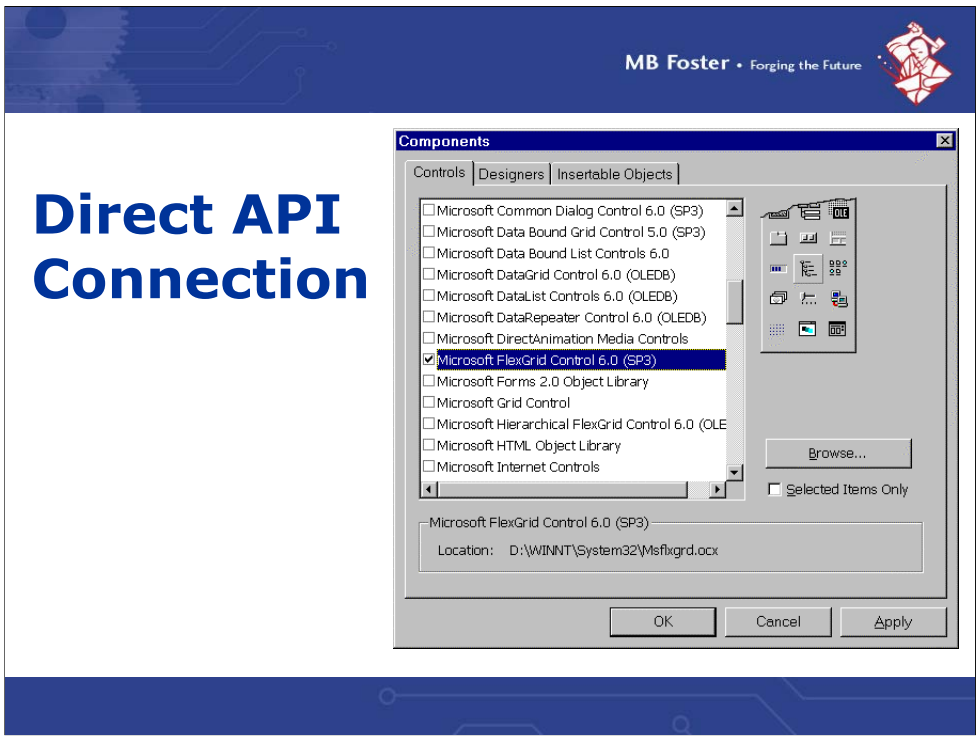
USE THE RECORDSET INFORMATION YOU NEED AND PROCESS IT ACCORDINGLY



USE THE RECORDSET INFORMATION YOU NEED AND PROCESS IT ACCORDINGLY



MAKE SURE YOU CLOSE ANY ODJECTS THAT YOU HAVE CREATED WHEN ERRORS OCCUR, OR WHEN EXITING



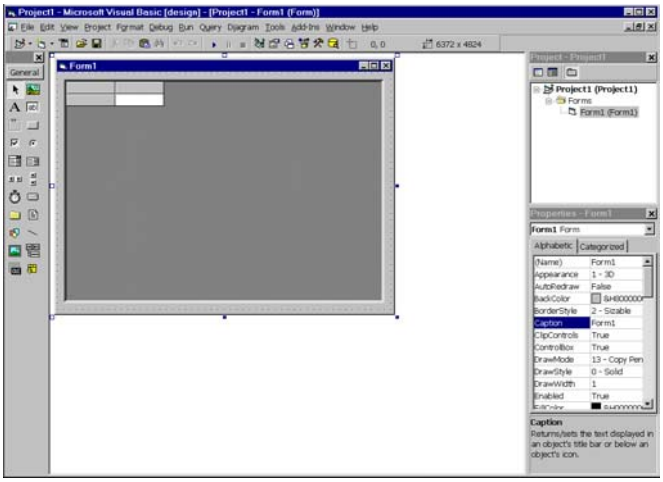
For our example we will use the MSFLEXGRID.

Add the component MSFLEXGRID to the tool bar.

1. On the menu bar go to 'Project'
2. Select 'Components'.
3. Find and select the Microsoft Flex Grid Control.

MB Foster • Forging the Future

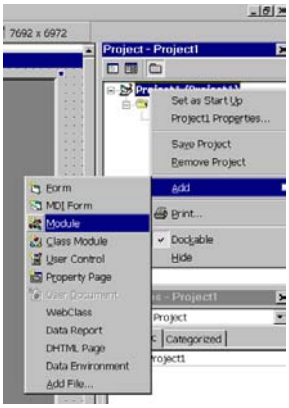
Direct API Connections

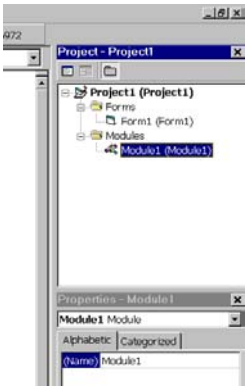


Add a MSFlexGrid (the yellow grid icon you just added to the tool bar) to the form. You should then resize it to fill most of your form.

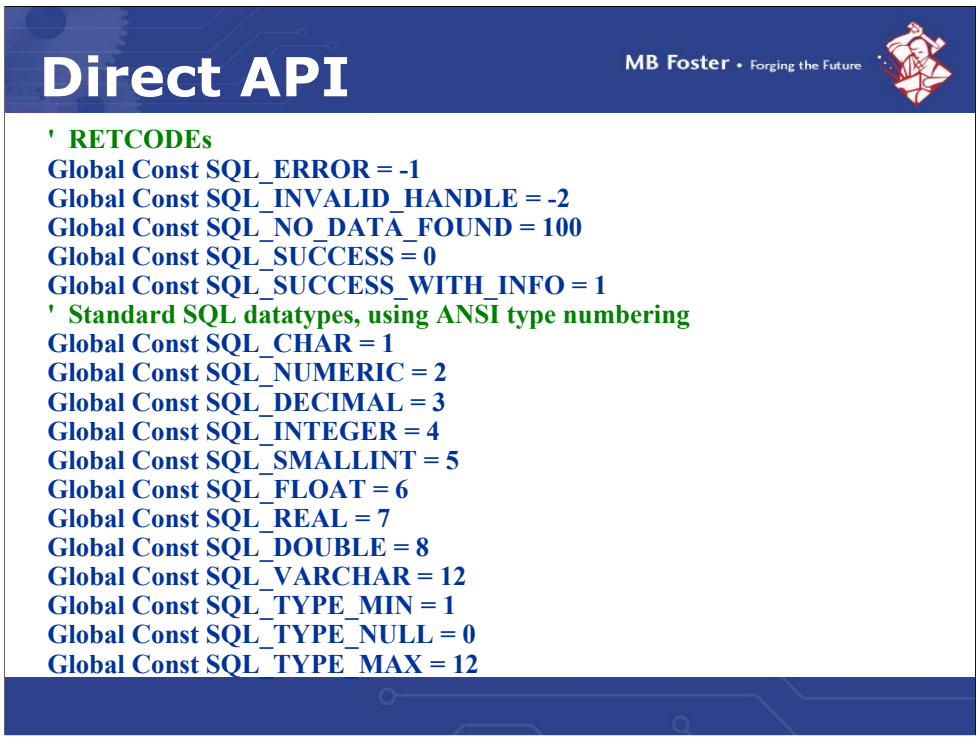
MB Foster • Forging the Future

Direct API Connections





Add a new module for Global Constants



GLOBALS

Do this in the ‘General Declarations” section of the module.

Direct API

MB Foster • Forging the Future

```
' C datatype to SQL datatype mapping    SQL types
Global Const SQL_C_CHAR = SQL_CHAR       ' CHAR, VARCHAR,
    DECIMAL, NUMERIC
Global Const SQL_C_LONG = SQL_INTEGER    ' INTEGER
Global Const SQL_C_SHORT = SQL_SMALLINT  ' SMALLINT
Global Const SQL_C_FLOAT = SQL_REAL      ' REAL
Global Const SQL_C_DOUBLE = SQL_DOUBLE   ' FLOAT, DOUBLE
' NULL status constants.  These are used in SQLColumns,
    SQLColAttributes,
' SQLDescribeCol, and SQLSpecialColumns to describe the nullablity of a
' column in a table.
Global Const SQL_NO_NULLS = 0
Global Const SQL_NULLABLE = 1
Global Const SQL_NULLABLE_UNKNOWN = 2

' Special length values
Global Const SQL_NULL_DATA = -1
Global Const SQL_DATA_AT_EXEC = -2

' Miscellaneous Constants
Global Const SBufferLen = 256          ' Default String Buffer Length
Global Const MAX_WIDTH = 255
```

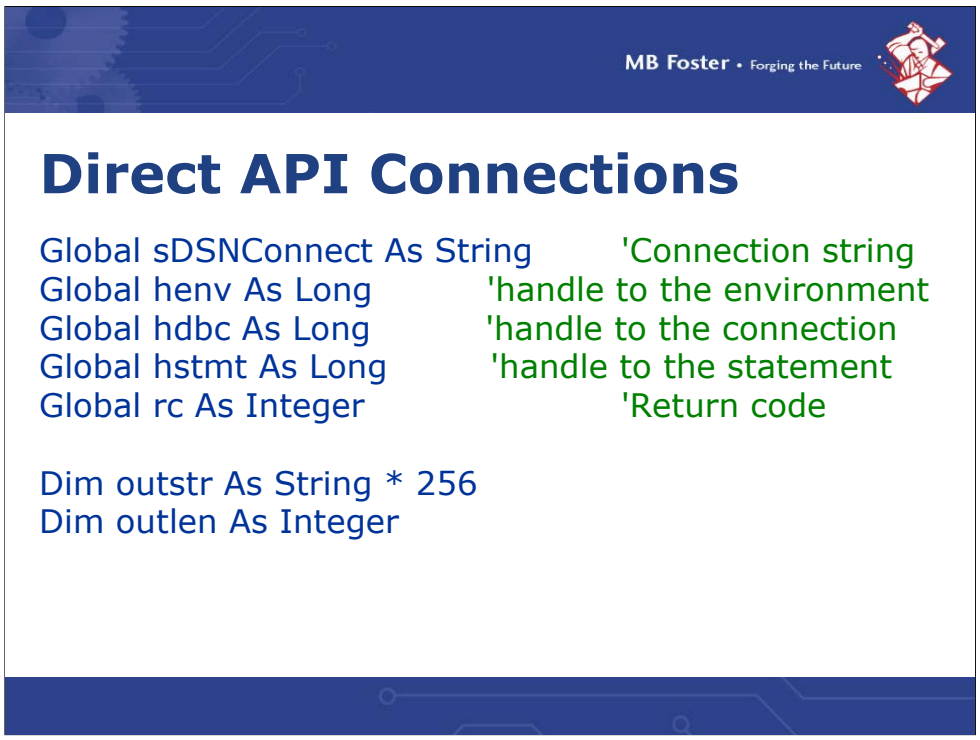
GLOBALS

Do this in the ‘General Declarations” section of the module.

Direct API

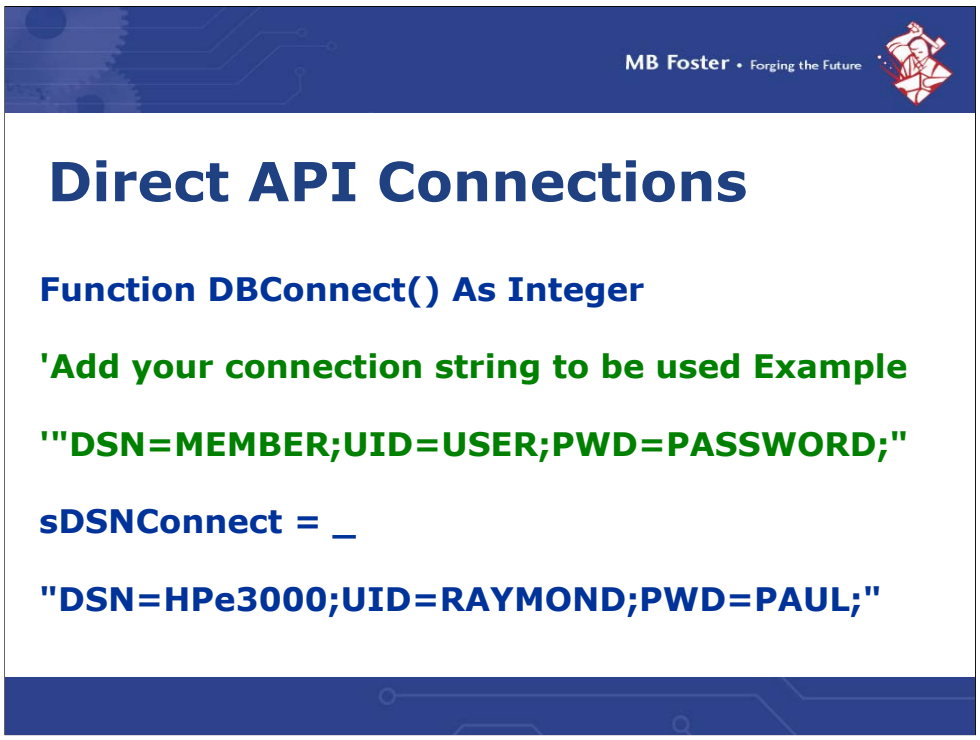
MB Foster • Forging the Future

```
Declare Function SQLAllocEnv Lib "odbc32.dll" (env As Long) As Integer
Declare Function SQLFreeEnv Lib "odbc32.dll" (ByVal env As Long) As Integer
Declare Function SQLAllocConnect Lib "odbc32.dll" (ByVal env As Long, hdbc As Long) As Integer
Declare Function SQLDriverConnect Lib "odbc32.dll" (ByVal hdbc As Long, ByVal hWnd As Long, _
        ByVal szCSIn As String, ByVal cbCSIn As Integer, ByVal szCSOut As String, _
        ByVal cbCSMax As Integer, cbCSOut As Integer, ByVal f As Integer) As Integer
Declare Function SQLFreeConnect Lib "odbc32.dll" (ByVal hdbc As Long) As Integer
Declare Function SQLDisconnect Lib "odbc32.dll" (ByVal hdbc As Long) As Integer
Declare Function SQLAllocStmt Lib "odbc32.dll" (ByVal hdbc As Long, hstmt As Long) As Integer
Declare Function SQLFreeStmt Lib "odbc32.dll" (ByVal hstmt As Long, ByVal EndOption As Integer)
    _
    As Integer
Declare Function SQLExecDirect Lib "odbc32.dll" (ByVal hstmt As Long, ByVal sqlString As String, _
        ByVal sqlstrlen As Long) As Integer
Declare Function SQLNumResultCols Lib "odbc32.dll" (ByVal hstmt As Long, NumCols As Integer) _
        As Integer
Declare Function SQLDescribeCol Lib "odbc32.dll" (ByVal hstmt As Long, ByVal colnum As Integer, _
        ByVal colname As String, ByVal buflen As Integer, colnamelen As Integer, dtype As Integer, _
        dl As Long, ds As Integer, n As Integer) As Integer
Declare Function SQLFetch Lib "odbc32.dll" (ByVal hstmt As Long) As Integer
Declare Function SQLGetData Lib "odbc32.dll" (ByVal hstmt As Long, ByVal col As Integer, _
        ByVal wConvType As Integer, ByVal lpbBuf As String, ByVal dwbuflen As Long, _
        lpcbout As Long) As Integer
Declare Function SQLError Lib "odbc32.dll" (ByVal env As Long, ByVal hdbc As Long, _
        ByVal hstmt As Long, ByVal SQLState As String, NativeError As Long, ByVal Buffer As String, _
        ByVal buflen As Integer, outlen As Integer) As Integer
Declare Function SQLRowCount Lib "odbc32.dll" (ByVal hstmt As Long, rows As Long) As Integer
```



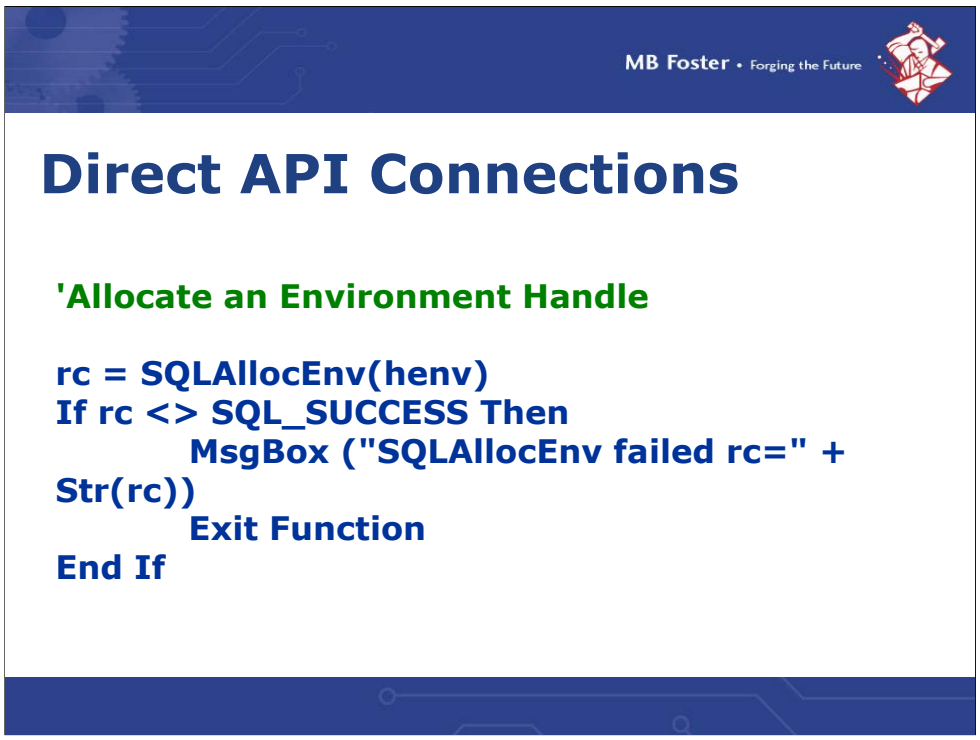
DECLARE YOUR VARIABLES FOR THE CONNECTION

Do this in the ‘General Declarations” section of the module.

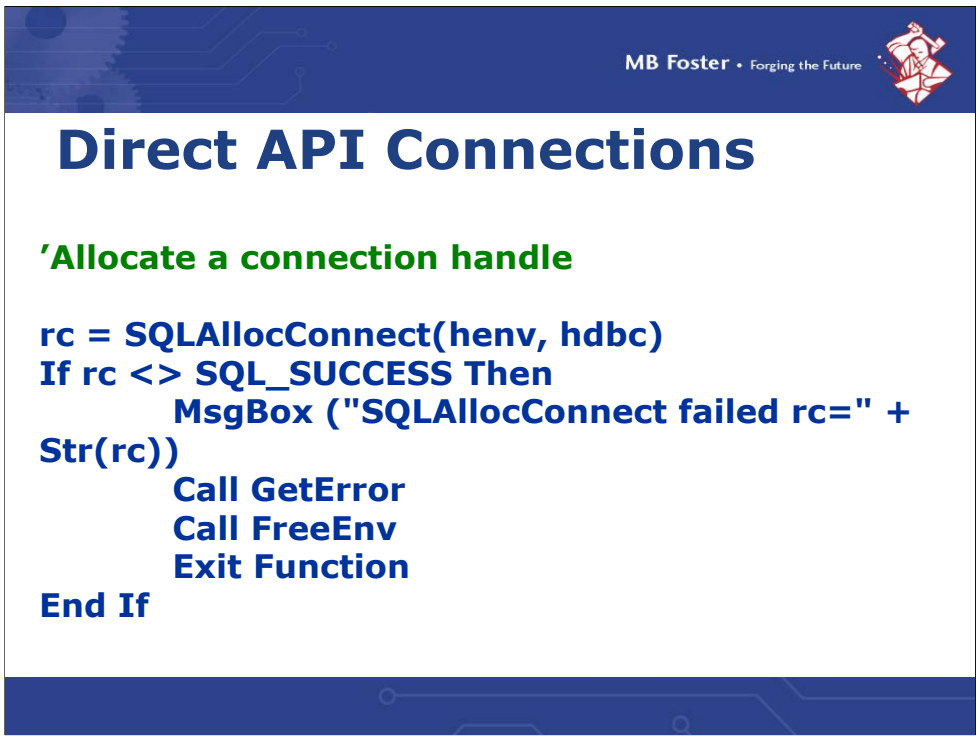


CREATE YOUR CONNECTION STRING

Do this in the ‘General Declarations” section of a module.



ALLOCATE AN ENVIROMENT HANDLE



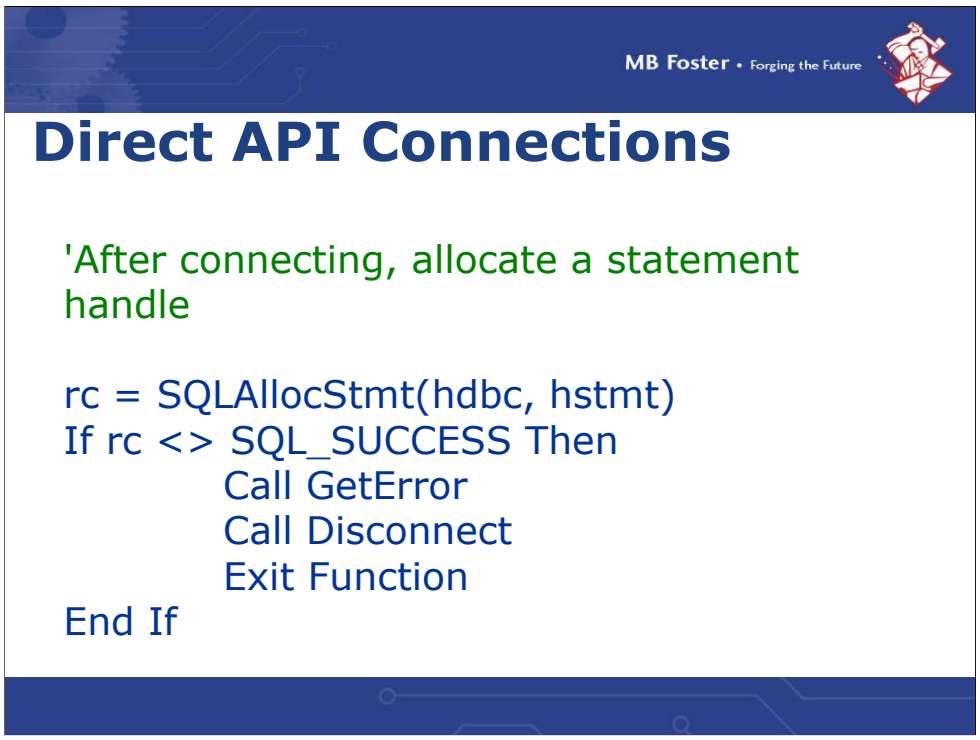
ALLOCATE A CONNECTION HANDLE

Direct API

MB Foster • Forging the Future

```
'Allocate the connection and pass in the
   Connection string

rc = SQLDriverConnect(hdbc, Form1.hWnd,
   sDSNConnect, _
        Len(sDSNConnect), outstr, 256, outlen, 3)
If rc <> SQL_SUCCESS Then
    If rc = SQL_NO_DATA_FOUND Then
       Exit Function 'User cancelled dialogue
    End If
    Call GetError
    Call Freeconnect
    Call FreeEnv
    Exit Function
End If
```

CONNECT TO YOUR DSN WITH SQLDriverConnect



ALLOCATE A STATEMENT HANDLE

Direct API

MB Foster • Forging the Future

```
'Now call an SQL query to select your data.

rc = SQLExecDirect(hstmt, "Select * from
   membership", _
      Len("Select * from member.membership"))
If rc <> SQL_SUCCESS Then
      Call GetError
      MsgBox _"Unable to connect to the
   Database Environment!", _
                                 vbCritical
         Call Disconnect
         Exit Function
End If

End Function
```

QUERY THE DATASOURCE USING SQLExecDirect

Direct API

MB Foster • Forging the Future

```
Sub GetError()

Dim error_str As String * 256
Dim SQLState As String * 20
Dim outlen As Integer
Dim NativeError As Long
Dim msg As String

rc = SQLError(henv, hdbc, SQL_NULL_HSTMT, SQLState,
   NativeError, _
      error_str, 256, outlen)
sDSNConnect = Space(256)
If rc <> SQL_NO_DATA_FOUND Then
         msg = Left(error_str, outlen)
      If gl_SQLStatement <> "" Then
            msg = msg + " (" + gl_SQLStatement + ")"
      End If
         MsgBox msg
End If

End Sub
```

CALL AN ERROR ROUTINE IF AN ERROR OCCURS

Direct API

MB Foster • Forging the Future

```
Sub Freeconnect()
rc = SQLFreeConnect(hdbc)
If rc <> SQL_SUCCESS Then
   Call GetError
End If
End Sub
```

```
Sub FreeEnv()
rc = SQLFreeEnv(henv)
If rc <> SQL_SUCCESS Then
   Call GetError
End If
End Sub
```

```
Sub Freestmt()
rc = SQLFreeStmt(hstmt, SQL_DROP)
If rc <> SQL_SUCCESS Then
   Call GetError
End If
End Sub
```

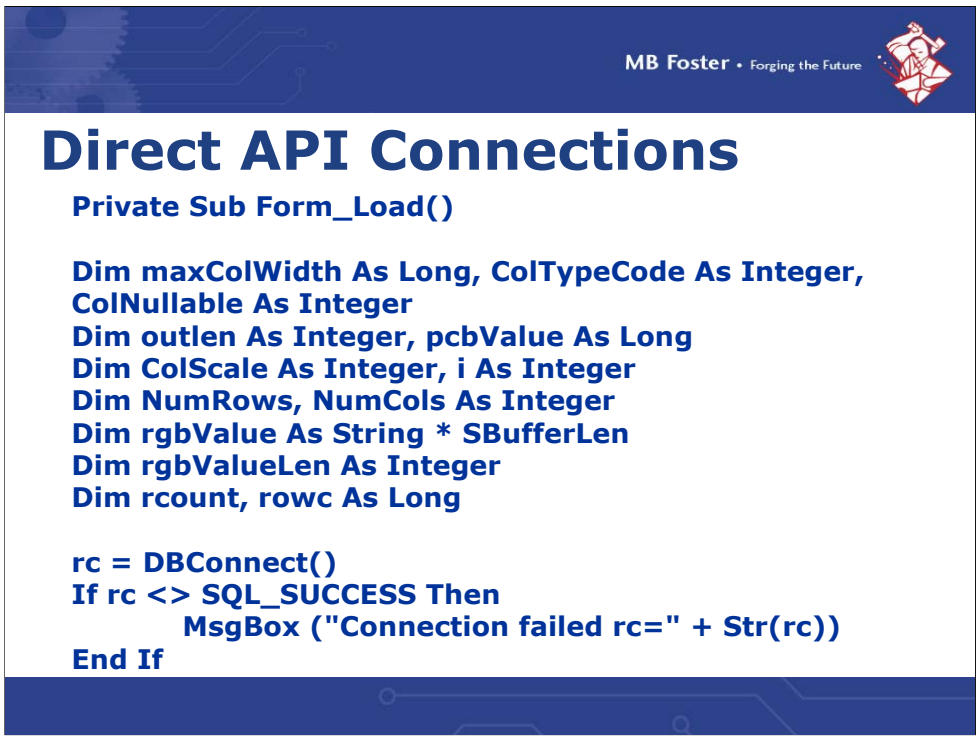
MAKE SURE YOU RELEASE ANY ENVIROMENT THAT YOU HAVE CREATED WHEN ERRORS OCCUR(or on exiting)

Direct API

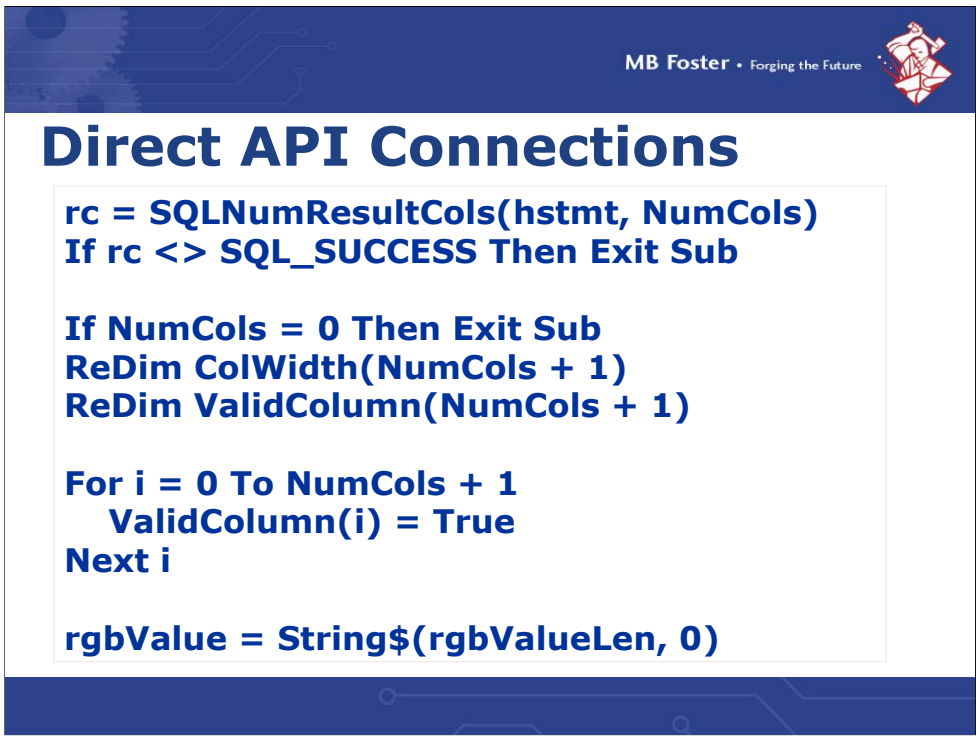
MB Foster • Forging the Future

```
Sub Disconnect()

rc = SQLDisconnect(hdbc)
If rc <> SQL_SUCCESS Then
   Call GetError
End If

rc = SQLFreeConnect(hdbc)
If rc <> SQL_SUCCESS Then
      MsgBox ("SQLFreeConnect failed rc=" + Str(rc))
      Exit Sub
End If

rc = SQLFreeEnv(henv)
If rc <> SQL_SUCCESS Then
      MsgBox ("SQLFreeEnv failed rc=" + Str(rc))
      Exit Sub
End If

End Sub
```

MAKE SURE YOU RELEASE ANY ENVIROMENT THAT YOU HAVE CREATED WHEN ERRORS OCCUR(or on exiting)


MB Foster • Forging the Future
Direct API Connections
Private Sub Form_Load()
Dim maxColWidth As Long, ColTypeCode As Integer,
ColNullable As Integer
Dim outlen As Integer, pcbValue As Long
Dim ColScale As Integer, i As Integer
Dim NumRows, NumCols As Integer
Dim rgbValue As String * SBufferLen
Dim rgbValueLen As Integer
Dim rcount, rowc As Long
rc = DBConnect()
If rc <> SQL_SUCCESS Then
MsgBox ("Connection failed rc=" + Str(rc))
End If

FORM LOAD


MB Foster • Forging the Future
Direct API Connections
rc = SQLNumResultCols(hstmt, NumCols)
If rc <> SQL_SUCCESS Then Exit Sub
If NumCols = 0 Then Exit Sub
ReDim ColWidth(NumCols + 1)
ReDim ValidColumn(NumCols + 1)
For i = 0 To NumCols + 1
ValidColumn(i) = True
Next i
rgbValue = String$(rgbValueLen, 0)

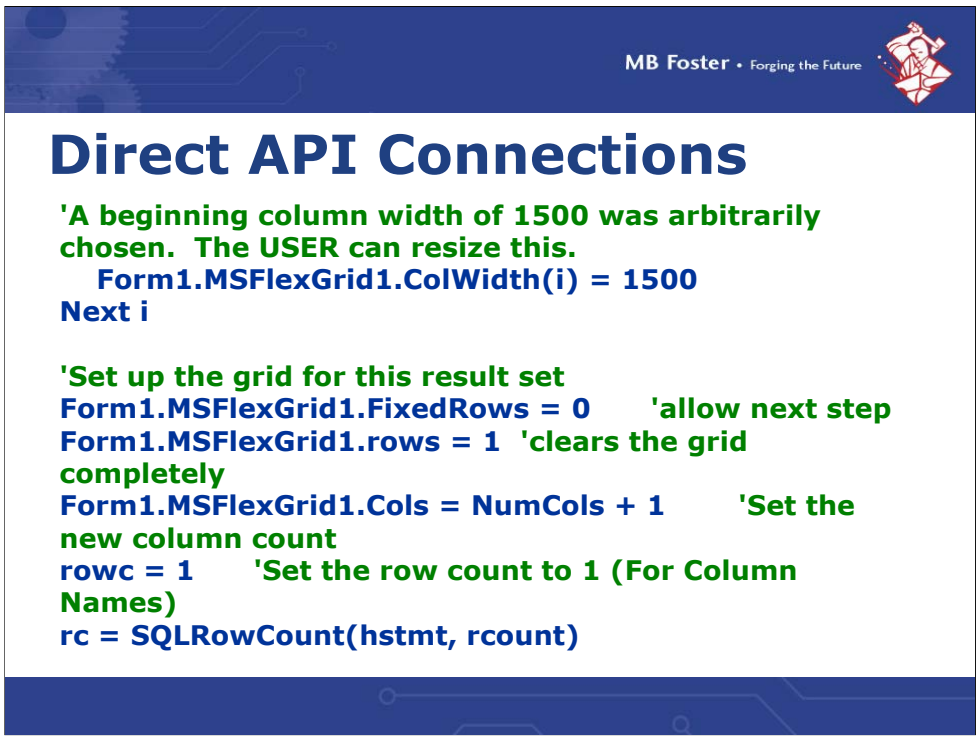
FORM LOAD

Direct API

MB Foster • Forging the Future

```
' Retrieve the Value Data for each column in the result set
' and initialize the column widths of the grid.
Form1.MSFlexGrid1.rows = 1
Form1.MSFlexGrid1.Cols = NumCols + 1
'Column zero is the row number column.  It is 8
characters wide.
ColWidth(0) = 8
For i = 1 To NumCols
   Form1.MSFlexGrid1.col = i  'Move to current column
   rc = SQLDescribeCol(hstmt, i, rgbValue, SBufferLen,
outlen, _
         ColTypeCode, maxColWidth, ColScale, ColNullable)
   Form1.MSFlexGrid1.Text = Left$(rgbValue, outlen)
   If maxColWidth > MAX_WIDTH Then _
      maxColWidth = MAX_WIDTH
   ColWidth(i) = maxColWidth  'Record the maximum text length
for the column
```

FORM LOAD

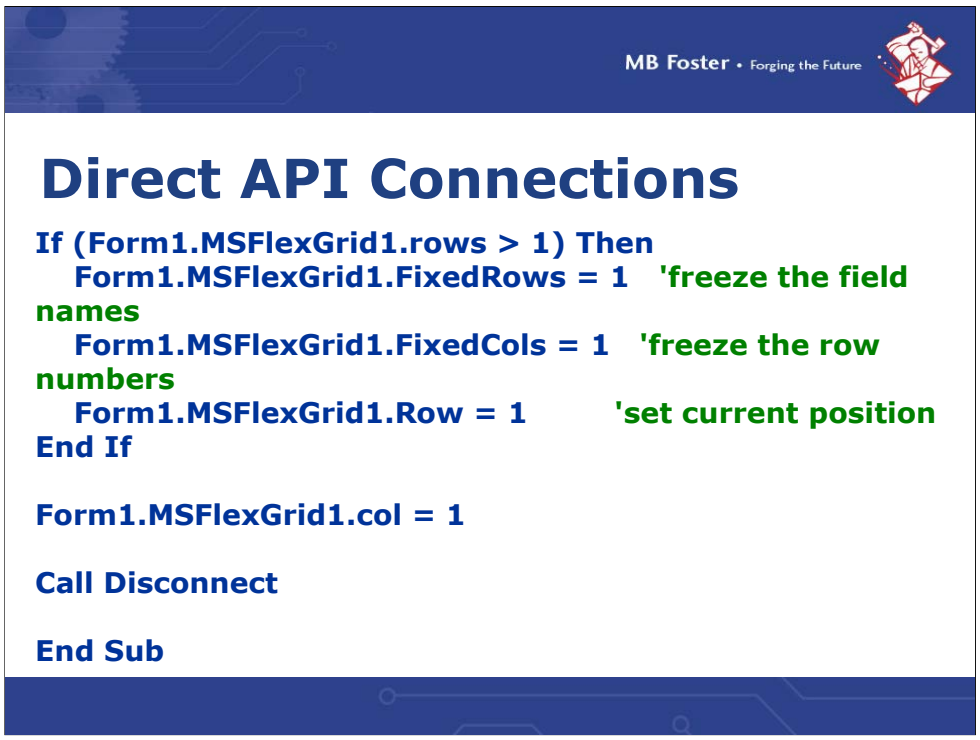

MB Foster • Forging the Future
Direct API Connections
'A beginning column width of 1500 was arbitrarily chosen. The USER can resize this.
Form1.MSFlexGrid1.ColWidth(i) = 1500
Next i
'Set up the grid for this result set
Form1.MSFlexGrid1.FixedRows = 0 'allow next step
Form1.MSFlexGrid1.rows = 1 'clears the grid completely
Form1.MSFlexGrid1.Cols = NumCols + 1 'Set the new column count
rowc = 1 'Set the row count to 1 (For Column Names)
rc = SQLRowCount(hstmt, rcount)

FORM LOAD

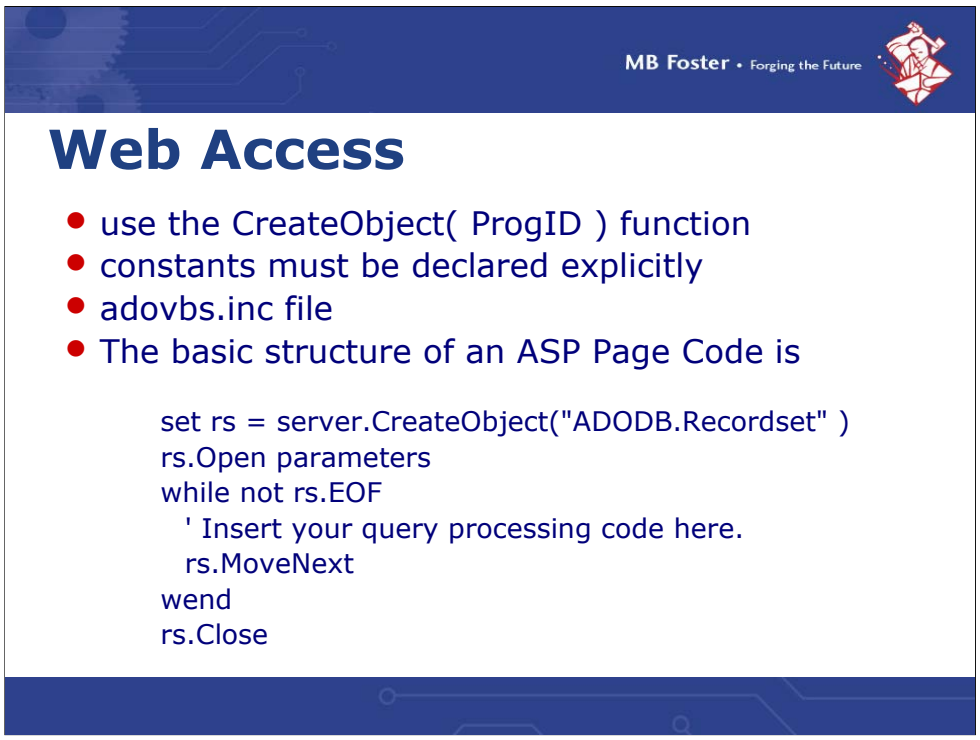
Direct API

MB Foster • Forging the Future

```
Do   'Repeat this loop until all result rows are fetched
   rc = SQLFetch(hstmt)
   If rc <> SQL_SUCCESS Then Exit Do

   Form1.MSFlexGrid1.rows = rowc + 1  'Increment the row counter
   Form1.MSFlexGrid1.Row = rowc       'and move to the next row
   Form1.MSFlexGrid1.col = 0
   Form1.MSFlexGrid1.Text = CStr(rowc)
   For i = 1 To NumCols
    Form1.MSFlexGrid1.col = i
    rc = SQLGetData(hstmt, i, SQL_C_CHAR, ByVal rgbValue, SBufferLen,
    pcbValue)
    If rc <> SQL_SUCCESS Then
       If rc = SQL_ERROR Then
         MsgBox "Result Column " + Str$(i) + " will be ignored"
       End If
    End If
    If rc <> SQL_ERROR Then
     If pcbValue = SQL_NULL_DATA Then
       Form1.MSFlexGrid1.Text = "NULL
     Else
       Form1.MSFlexGrid1.Text = Trim$(rgbValue)
     End If
    End If
   Next i
   rowc = rowc + 1
Loop
```


MB Foster • Forging the Future
Direct API Connections
If (Form1.MSFlexGrid1.rows > 1) Then
Form1.MSFlexGrid1.FixedRows = 1 'freeze the field names
Form1.MSFlexGrid1.FixedCols = 1 'freeze the row numbers
Form1.MSFlexGrid1.Row = 1 'set current position
End If
Form1.MSFlexGrid1.col = 1
Call Disconnect
End Sub

FORM LOAD



There are numerous Applications that allow you to use an ASP (Active Server Page) to post a web page on your or another PC.

In an .asp file, use the CreateObject(ProgID) function on the server object to create new ADO objects.

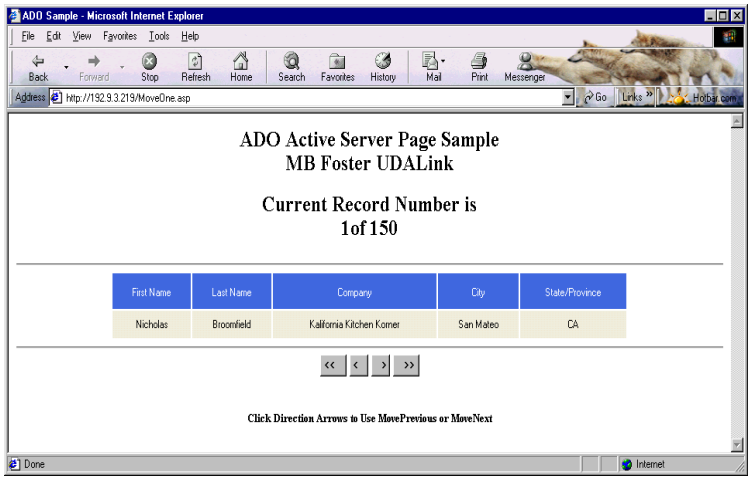
ADO constants must be declared explicitly because they are not loaded automatically from the typelib. The ADO constants are declared in the adovbs.inc file and can be manually inserted into your script. Not including the adovbs.inc file will give you inconsistent problems. Alternatively you can simply declare the constants you need within the page which might lower the amount of resources that you ASP page consumes, and thus increase its performance.

The basic structure of an ASP Page Code is

```
set rs = server.CreateObject("ADODB.Recordset" )
rs.Open parameters
while not rs.EOF
        ' Insert your query processing code here.
        rs.MoveNext
wend
rs.Close
```

MB Foster • Forging the Future

ADO in ASP Sample



The example above illustrates an ADO in an Active Server Page accessed with Microsoft Internet Explorer. The example connects to an MPE server and can scroll through any one of the records in an Image Table

This ASP Page demonstrates how to use ADO 2.1 with the MB Foster Driver. Below you will see the creation of the Connection and Recordset Objects. The MEMBERSHIP table is used for this sample. It demonstrates how to move through and display one row with the MoveFirst, MoveLast, Move Previous, and MoveNext Methods in ADO.

ADO in ASP

MB Foster • Forging the Future

```
<HTML>
<HEAD>
<META name=VI60_defaultClientScript content=VBScript>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<TITLE>ADO Sample</TITLE>
</HEAD>
<BODY><FONT face="MS SANS SERIF" size=2>
<CENTER>
<H3><FONT face="Times New Roman" size=5>ADO Active Server Page
    Sample<BR>MB Foster UDALink</FONT>  </H3>
<!-- Create Connection and Recordset Objects on Server -->

<%@ Language=VBScript %>
<%

'declare the constants explicitly or include the adovbs.inc include file
Const adUseClient = 3
Const adLockOptimistic = 3
Const adOpenKeyset = 1
```

ADO in ASP

MB Foster • Forging the Future

```
'Create and Open Connection Object
Set ADOCN = Server.CreateObject("ADODB.Connection")
ADOCN.Open "DSN=HPe3000;"

'Create and Open Recordset Object
Set ADORS = Server.CreateObject("ADODB.Recordset")
With ADORS
   .ActiveConnection = ADOCN
   .CursorLocation = adUseClient
   .CursorType = adOpenKeyset
   .LockType = adLockOptimistic
   .Source = "SELECT * FROM MEMBER.MEMBERSHIP;"
End With

ADORS.Open
```

ADO in ASP

MB Foster • Forging the Future

```
'Check Request.Form collection to see if any moves are recorded
If Not IsEmpty(Request.Form("MoveAmount")) Then
    'Keep track of the number and direction of moves this session
    Session("Moves") = Session("Moves") +
    Request.Form("MoveAmount")
    Clicks = Session("Moves")
    'Move to last known position
    ADORS.Move CInt(Clicks)
    'Check if move is + or - and do error checking
    If CInt(Request.Form("MoveAmount")) = 1 Then
          If ADORS.EOF Then
                Session("Moves") = ADORS.RecordCount
         ADORS.MoveLast
      End If
          ADORS.MoveNext
   End If
   If Request.Form("MoveAmount") < 1 Then
          ADORS.MovePrevious
   End If
```

ADO in ASP

MB Foster • Forging the Future

```
'Check if First Record or Last Record Command Buttons Clicked
   If Request.Form("MoveLast") = 3 Then
         ADORS.MoveLast
         Session("Moves") = ADORS.RecordCount
   End If
   If Request.Form("MoveFirst") = 2 Then
         ADORS.MoveFirst
         Session("Moves") = 1
   End If
End If

' Do Error checking for combination of Move Button clicks
If ADORS.EOF Then
    Session("Moves") = ADORS.RecordCount
    ADORS.MoveLast
    Response.Write "This is the Last Record"
End If
```

ADO in ASP

MB Foster • Forging the Future

```
If ADORS.BOF Then
    Session("Moves") = 1
    ADORS.MoveFirst
    Response.Write "This is the First Record"
End If

%>

<H3><FONT face="Times New Roman" size=5>Current Record Number
    is <BR>
<!-- Display Current Record Number and Recordset Size -->
<%
If IsEmpty(Session("Moves"))  Then
    Session("Moves") = 1
End If
%>

<%Response.Write(Session("Moves") )%>of
    <%=ADORS.RecordCount%></FONT></H3>
```

ADO in ASP

MB Foster • Forging the Future

```
<HR>
<CENTER>
<TABLE border=0 cellPadding=5 COLSPAN="8" bgColor=#ffffff
   borderColor=#ffffff style="BACKGROUND-COLOR: white;
   HEIGHT: 16px; WIDTH: 673px">
<!-- BEGIN column header row for MEMBERSHIP Table-->
 <TR>
  <TD align=middle bgColor=#4169e1><FONT color=#ffffff
  size=1 style="ARIAL: ">First Name</FONT> </TD>
  <TD align=middle bgColor=#4169e1><FONT color=#ffffff
  size=1 style="ARIAL: ">Last Name</FONT> </TD>
  <TD align=middle bgColor=#4169e1><FONT color=#ffffff
  size=1 style="ARIAL: ">Company</FONT> </TD>
  <TD align=middle bgColor=#4169e1><FONT color=#ffffff
  size=1 style="ARIAL: ">City</FONT> </TD>
  <TD align=middle bgColor=#4169e1><FONT color=#ffffff
  size=1 style="ARIAL: ">State/Province</FONT>
  </TD></TR><!--Display ADO Data from MEMBERSHIP Table-
  ->
```

ADO in ASP

MB Foster • Forging the Future

```
<TR>
   <TD align=middle bgColor=#f7efde><FONT size=1 style="ARIAL:
    "><%= ADORS("First_Name")%></FONT></TD>
   <TD align=middle bgColor=#f7efde><FONT size=1 style="ARIAL:
    "><%= ADORS("Last_Name")%></FONT></TD>
   <TD align=middle bgColor=#f7efde><FONT size=1 style="ARIAL:
    "><%= ADORS("Company")%></FONT></TD>
   <TD align=middle bgColor=#f7efde><FONT size=1 style="ARIAL:
    "><%= ADORS("City")%></FONT></TD>
   <TD align=middle bgColor=#f7efde><FONT size=1 style="ARIAL:
    "><%= ADORS("State")%></FONT></TD></TR>
   </TABLE>
   </FONT>
<HR>
<INPUT name=cmdFirst type=button value="<<   ">
<INPUT name=cmdDown type=button value="<   ">
<INPUT name=cmdUp type=button value="   >">
<INPUT name=cmdLast type=button value="   >>">

<BR> <BR>
```

ADO in ASP

```
<!-- Use Hidden Form Fields to send values to Server -->

<FORM Method = post Action="MoveOne.asp" Name=Form>
<INPUT Type="hidden" Size="4" Name="MoveAmount" Value = 0>
<INPUT Type="hidden" Size="4" Name="MoveLast" Value = 0>
<INPUT Type="hidden" Size="4" Name="MoveFirst" Value = 0>
</FORM></CENTER></CENTER>
</BODY>
<Script Language = "VBScript">
'Set Values in Form Input Boxes and Submit Form
Sub cmdDown_OnClick
   Document.Form.MoveAmount.Value = -1
   Document.Form.Submit
End Sub

Sub cmdUp_OnClick
   Document.Form.MoveAmount.Value = 1
   Document.Form.Submit
End Sub
```

ADO in ASP Sample

```
Sub cmdFirst_OnClick
    Document.Form.MoveFirst.Value = 2
    Document.Form.Submit
End Sub

Sub cmdLast_OnClick
   Document.Form.MoveLast.Value = 3
   Document.Form.Submit
End Sub
</Script></HTML>
```

Contact Us

By Phone: *1-800-ANSWERS (267-9377)*
613-448-2333
By Fax: *613-448-2588*
By E-Mail: *Support@mbfoster.com*
On the Web: *www.mbfoster.com*

Suggested Reading

- Microsoft ODBC 2.0/3.0 - Programmers Reference and SDK Guide

  Publisher: Microsoft Press

- ODBC 3.5 Developer's Guide

  By: Roger E. Sanders

  Publisher: McGraw-Hill

Suggested Reading

- Inside ODBC
  By: Kyle Geiger
  Publisher: Microsoft Press

- SQL Instant Reference
  By: Martin Gruber
  Publisher: Sybex

Suggested Reading

- Client/Server Databases - Enterprise Computing

  By: James Martin / Joe Leben

  Publisher: Prentice-Hall Inc.,

- Client Server - A Managers Guide

  By: Laurence Shafe

  Publisher: Addison-Wesley