PREV
3. Testing XML

NEXT
ueri...

Establishing and closing a database connection

In this recipe, we will show how to establish a DB connection using VBScript code. We will build a simple custom class, `DB_Handler` that be instantiated using a global scope variable, `oDBHandler`. This globa object will serve as the basis for all our DB operations.

Getting ready

From the **File** menu, navigate to **New** | **Function Library...** or use t *Alt* + *Shift* + *N* shortcut. Name the new function library `DB_Func.vbs`.

How to do it...

The following code handles creating, opening, and closing a DB connection using `ADODB`:

```
Const C_ADODB_OBJ = "ADODB.Connection"
Dim oDBHandler

Function createDBHandler(p, ds, ic, uid, pwd)
    On error resume next
    Set oDBHandler=new DB_Handler

    call oDBHandler.Init(p, ds, ic, uid, pwd)

    createDBHandler=eval("err.number=0")
End Function

Class DB_Handler
    private m_oDBConnection
    Public Provider
    Public DataSource
    Public InitialCatalog
    Public Username
    Public Password

    Function Init()
        With me
            .Provider=p
            .DataSource=ds
            .InitialCatalog=ic
            .Username=uid
            .Password=pwd
        End With
    End Function

    Function openDBConnection()
        m_oDBConnection.open(createConnectionString())
    End function

    Function closeDBConnection()
        m_oDBConnection.close()
    End Function

    Function createConnectionString()
        createConnectionString = "Provider="&Provider& _
                                 ";Data Source="&DataSource& _
                                 ";Initial Catalog="&InitialCatalog& _
                                 ";uid="&myUserName&";pwd="&Password
    End Function

    Function createDBConnection()
        If not lcase(typename(m_oDBConnection)) = lcase(C_ADODB_OBJ) Then
            Set m_oDBConnection = CreateObject(C_ADODB_OBJ)
        End If
    End Function
    Function disposeDBConnection()
        closeDBConnection()
        Set m_oDBConnection = nothing
    End Function

    Private Sub Class_Initialize()
        createDBConnection()
    End Sub
    Private Sub Class_Terminate()
        disposeDBConnection()
    End Sub
```

```
End Class
```

Now, we can run `Action1` with the following lines of code:

```
dim p, ds, ic, uid, pwd

p="{SQL Server}"
ds="yourServername"
ic="yourDatabasename"
uid="yourUsername"
pwd="yourPassword"

if not createDBHandler(p, ds, ic, uid, pwd) then
    Reporter.ReportEvent micFail, "DB Init", "Failed to create a DB Handl
    exittest
end if

'Open the connection
oDBHandler.openDBConnection()

'Here goes code to query the DB (see recipe Using SQL queries programmati
'...
'...

'Close the connection
oDBHandler.closeDBConnection()

'Dispose the DB Handler
set oDBHandler=nothing
```

How it works...

We first declare the variables that we need to define our connection string. The `Provider` connection string tells `ADODB` the database type we intend to use (Access, SQL Server, Oracle, and so on). `DataSource` is the server or path on which our DB is located. `InitialCatalog` is the DB we wish to use upon connecting to the server. `Username` and `Password` are, of course, the credentials with which we log in to the DB.

We then call the `createDBHandler(p, ds, ic, uid, pwd)` method that serves as our constructor for the `DB_Handler` class. We pass the arguments listed here, and the method takes care of creating a new instance and initializing it with our parameters by calling the class member function `init(p, ds, ic, uid, pwd)`. If an error occurs, then the method returns `false` (`eval="err.number=0"` would evaluate to `false`), and the code would exit the test after reporting the event to the test results.

With our object already initialized, we can proceed to call its `openDBConnection()` function, perform the SQL queries we need, and at the end take care of calling `closeDBConnection()`. At the end, we dispose of our `DBHandler` object.