Class Builder

Author: **Andrew McKay**
Date: **May 27th 2002**

Member of
The Institute of
Analysts and Programmers
http://www.iap.org.uk

Table Of Contents

Table Of Figures

Table Of Tables

1 Introduction

1.1 Overview

The Class Builder Project (herein referred to as CBP) is a web application which is designed to produce VBScript source code for performing basic database table operations. These operations are INSERT/UPDATE, LOAD, and DELETE.

CBP can operate as supplied with either a MySql database or Microsoft SQL Server. However this isn't a limitation of CBP and if you have other types of database available you could easily modify the project to encapsulate the functionality required for just about any database type or vendor.

The source code produced by CBP is implemented as a standard VBScript class which can be included in any other Active Server Page (ASP). A typical approach used by web developers is to add all of the necessary code for performing database manipulation on each and every web page where the database is to be interrogated or updated – a highly wasteful approach from a design and implementation perspective, as well as leading to overwhelming complexity for support purposes.

To give some idea of how CBP can reduce coding effort and improve web developer productivity consider the two examples given below. Both perform an identical operation on a database table – to insert a new row into a table.

1.2 Limitations

CBP is going to produce a very standard VBScript "template" which will perform some basic operations on a single database table. These operations are limited to the following:

- Save()
- Load()
- Delete()

Other methods can easily be added to the script class by the developer to perform additional functionality.

Additionally the script class will expose each of the fields of the database table as a read-only property.

Some care may be required in naming the columns of the table to which the class refers. For example it may be perfectly okay for a database table to have a column called "Delete", but as the class generated by CBP will have a method called "Delete" there will be a conflict. The developer will need to bear this in mind when they design their database.

Finally, in order for CBP to work successfully with a database table it is a mandatory requirement that the leftmost (first) column of the table is an identity column so that each record in the table can be uniquely identified. If

the leftmost column of the table is not an identity column CBP will not work, and any script code produced will not be viable.

1.3 Example Database Schema

For the purposes of the example detailed in this document a database table named tblTelDir (a simple telephone directory) is defined in the database having the schema shown below.

Table 1: Example Table Schema

Column Name	Column Type	Width	Description
ID	Int	4	Identity column
Surname	Varchar	20	Surname
Forename	Varchar	20	Forename
Telephone	Varchar	20	Telephone number

1.4 Typical ASP Script

A typical fragment for inserting a new record into our tblTelDir table would look similar to the following:

```
<%
    sSqlStr = "INSERT INTO tblTelDir (Surname, Forename, Telephone) VALUES (" & _
          "'" & sSurname & "', '" & sForename & "', '" & sTelephone & "')"

    Set oCmd = CreateObject("ADODB.Command")
    oCmd.ActiveConnection = mConnStr
    oCmd.CommandType = adCmdText
    oCmd.CommandText = sSqlStr
    oCmd.CommandText = "SELECT @@IDENTITY"
    Set oRst = oCmd.Execute

    NewId = oRst.Fields(0)

    Set oRst = Nothing
    Set oCmd = Nothing
%>
```

1.5 Using Class Builder

Performing exactly the same operation using CBP the above source code can be replaced thus:

```
<%
    Set oTelDir = New Class_tblTelDir
    NewId = oTelDir.Save(0, sSurname, sForename, sTelephone)
    Set oTelDir = Nothing
%>
```

1.6 Summary

The above examples relate to a very simple table having only four fields. Many database tables will have rather more fields, and the user will have to construct the necessary SQL commands and queries to match each tables schema.

This work isn't required using CBP, and by navigating just three simple web pages as described in this document the mechanical aspects of using database tables can be reduced to very simple operations.

2 Deploying Class Builder

2.1 Overview

CBP is supplied as a complete web project which can be deployed upon Microsoft IIS4/5 web servers. The user can add this project to a sub-folder of their own web site if required, or create a standalone web site for CBP if that is more appropriate.

2.2 Files Supplied

A zip file containing all of the required script and other files is supplied. Use WinZip (http://www.winzip.com) or another utility to unzip the files to a directory on your hard disk – making sure to preserve the folder structure – then add these files to your web project in exactly the same hierarchy as is supplied in the zip file.

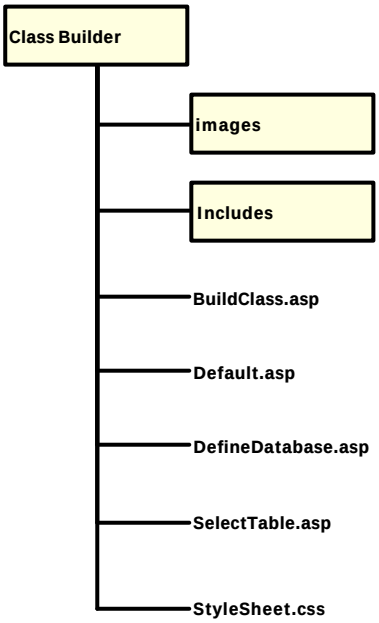

Figure 1: Class Builder Hierarchy

2.3 Author Acknowledgement

If CBP is used to help build your own web sites it would be greatly appreciated if suitable acknowledgement were provided to the author (see section 7 for further information). A link to the authors own web site at http://www.kazmax.co.uk would be greatly respected, and (subject to the target being of suitable quality) KazMax Ltd will be delighted to provide a reciprocal link.

3 Generating The Class Script

3.1 Overview

The process of creating a VBScript class for a table is very simple as shown below.

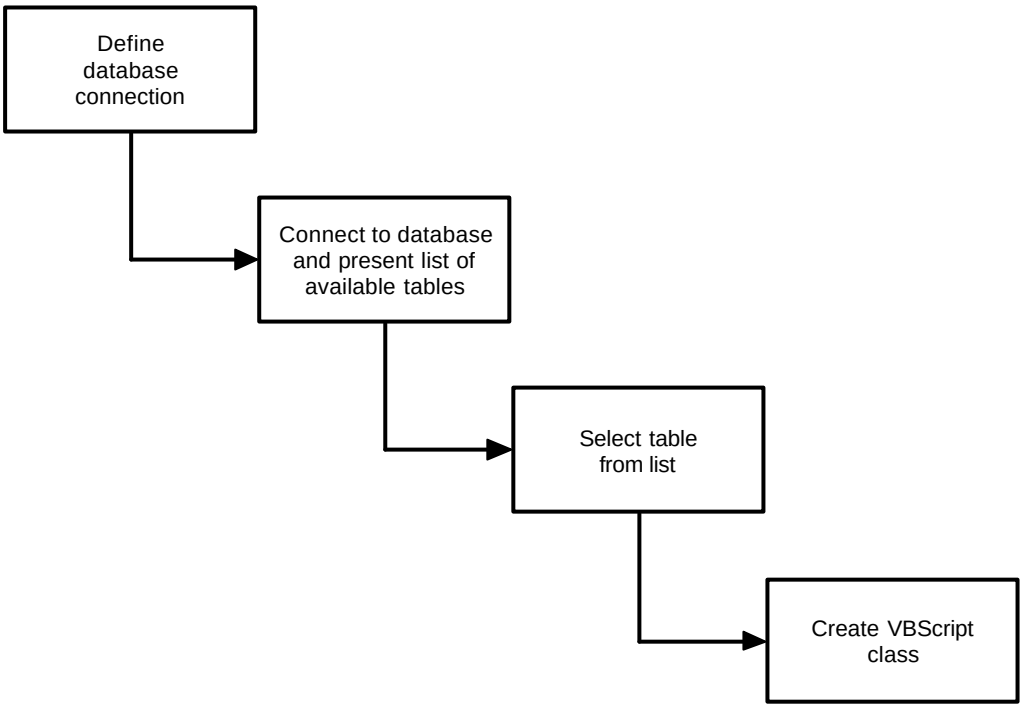

Figure 2: Scripting Process

Each of the steps shown in the figure above are explained below. The entire process starts from the Default.asp web page shown below.

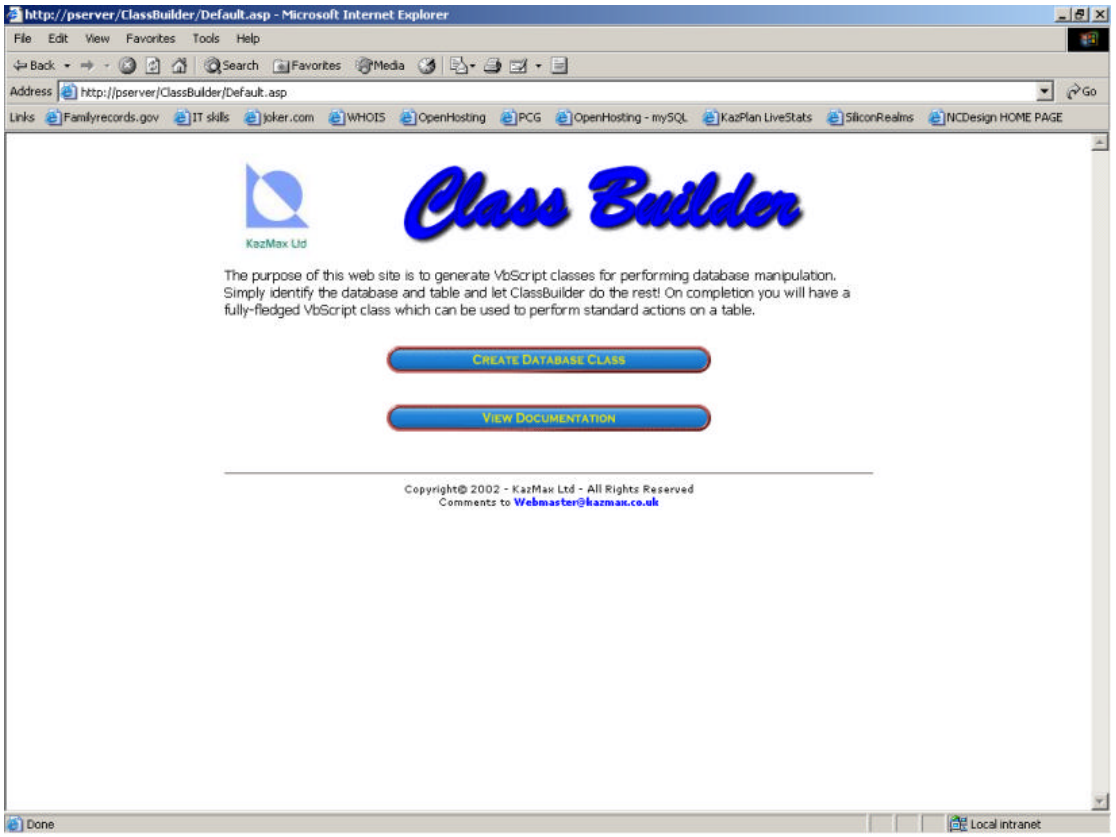

Figure 3: Default.asp

Click on the “Create Database Class” button to proceed to the DefineDatabase.asp web page where we collect the connection information for our database.

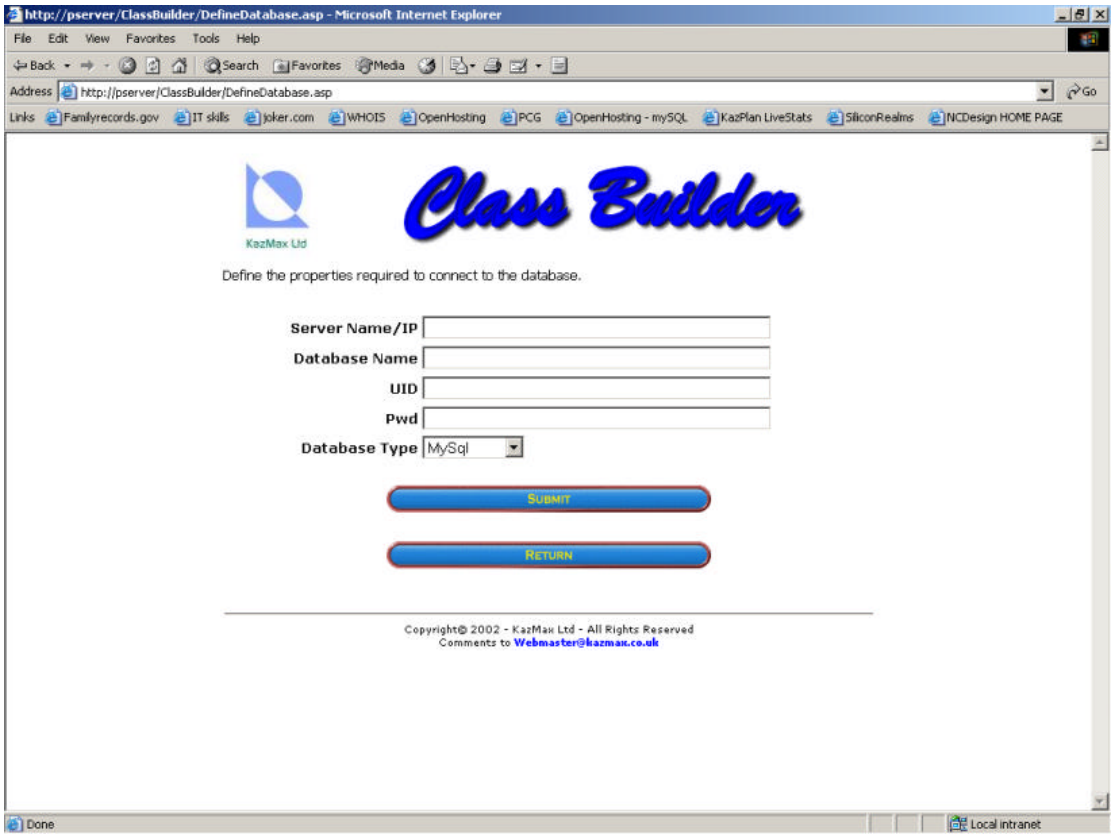

Figure 4: DefineDatabase.asp

DefineDatabase.asp collects the connection information for our database. For security reasons the screenshot shown above does not show the actual connection information, however you should know the information requested and thus be able to enter it as indicated.

Once the server, database name, UID, password and database type have been defined, click the “Submit” button to proceed to the SelectTable.asp web page.

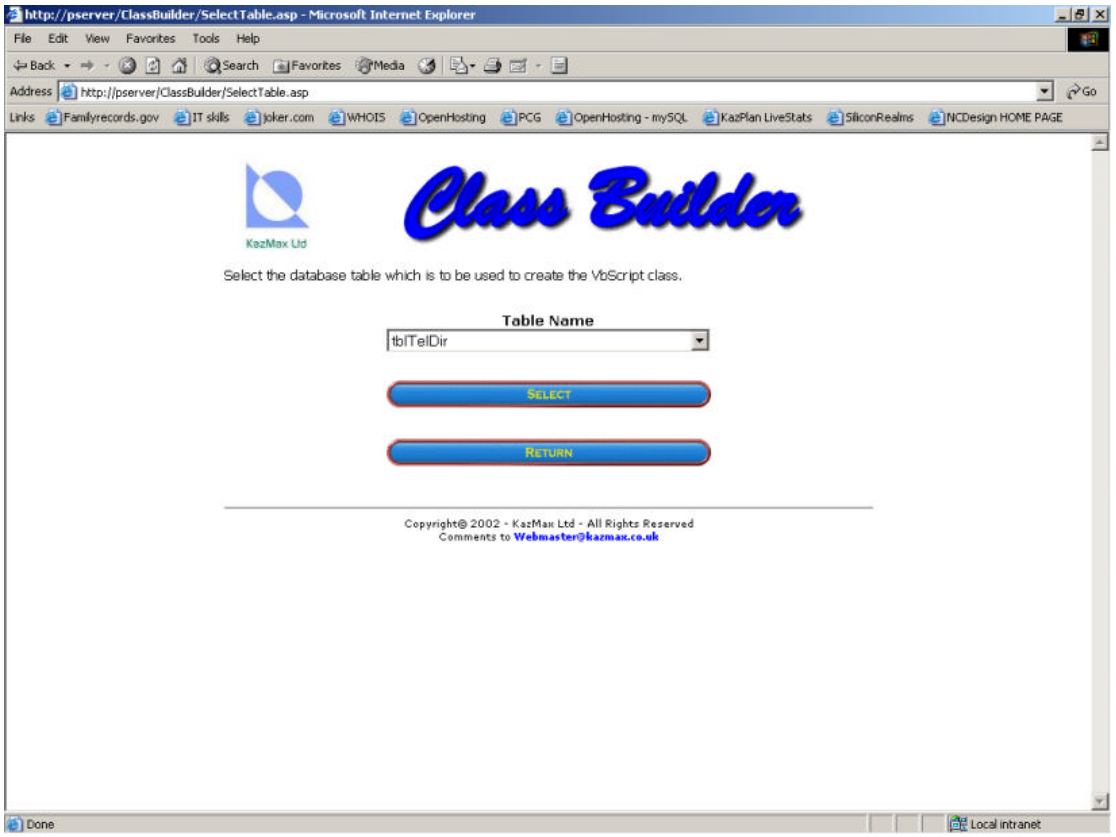

Figure 5: SelectTable.asp

Providing that the connection to the database is okay SelectTable.asp will provide a dropdown list of tables within the database. Select the table for which a class is to be generated to proceed to generate the script.

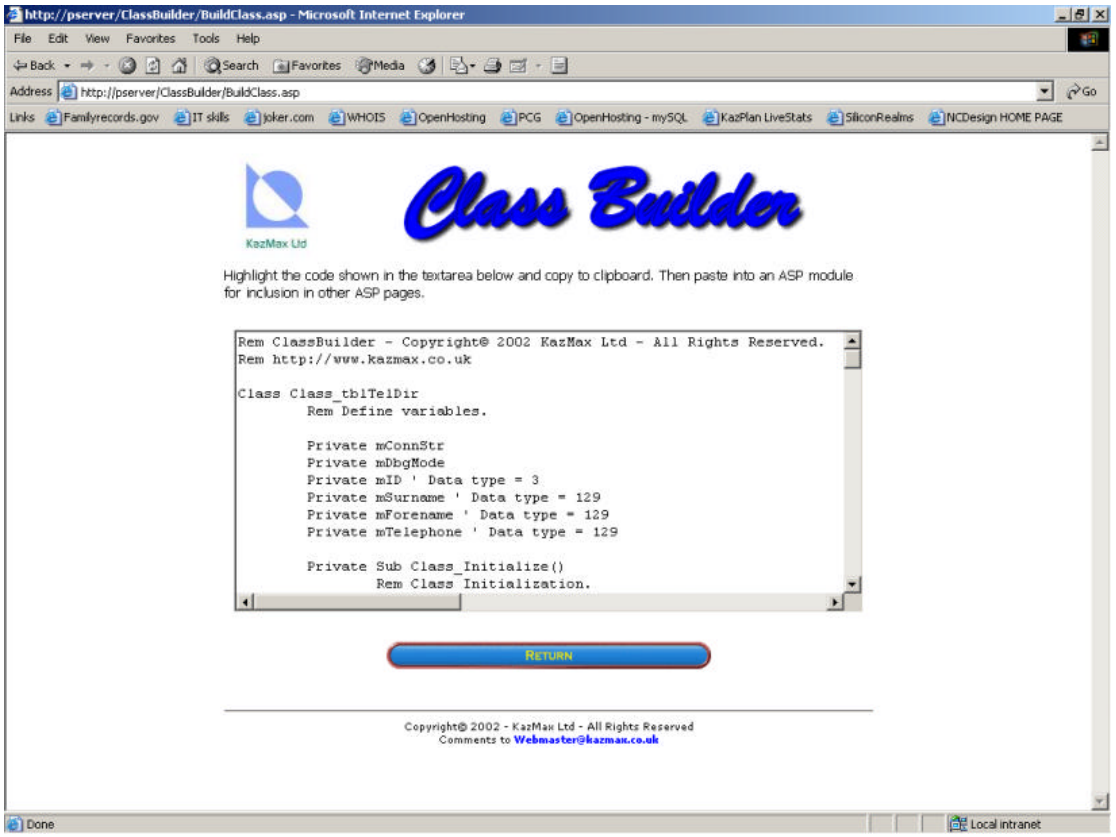

Figure 6: BuildClass.asp

BuildClass.asp produces the VBScript necessary to work with the database table. This should work fine "out of the box", though of course the user may wish to add some extra functionality.

To make use of this script highlight all of the text shown in the textarea on this page, copy it to the Windows clipboard, then paste it to a new ASP file within the web project (the script will need to be bounded by the "<%" and "%>" script delimiters).

3.2 Example Script

The listing below shows an example of the script which is produced by CBP.

```
Rem ClassBuilder - Copyright© 2002 KazMax Ltd - All Rights Reserved.
Rem http://www.kazmax.co.uk

Class Class_tblTelDir
          Rem Define variables.

          Private mConnStr
          Private mDbgMode
          Private mID ' Data type = 3
          Private mSurname ' Data type = 129
          Private mForename ' Data type = 129
          Private mTelephone ' Data type = 129

          Private Sub Class_Initialize()
                    Rem Class Initialization.

                    mDbgMode = False
                    mConnStr = "driver=SQL Server;server=ZZZ;uid=XXX;Pwd=XXX;database=XXX"
                    Call Init()
          End Sub

          Private Sub Class_Terminate()
                    Rem Class Termination.
```

```
End Sub

Private Sub Init()
          Rem Initialize variables.

          mID = 0
          mSurname = ""
          mForename = ""
          mTelephone = ""
End Sub

Public Function YyyyMmDd(xDate)
          Rem Returns date in form YYYY-MM-DD.

          YyyyMmDd = Right("0000" & Year(xDate), 4)
          YyyyMmDd = YyyyMmDd & Right("00" & Month(xDate), 2)
          YyyyMmDd = YyyyMmDd & Right("00" & Day(xDate), 2)
End Function

Public Function Quoted(xStr)
          Rem Translates single quotes to double-quotes.

          Quoted = "'" & Replace(xStr, "'", "''") & "'"
End Function

Public Function SqlQuery(xSql)
          Rem Performs SQL query on table. Returns recordset.

          Dim oCmd

          If mDbgMode Then
                    Response.Write xSql & "<br>"
          End If

          Set oCmd = CreateObject("ADODB.Command")
          oCmd.ActiveConnection = mConnStr
          oCmd.CommandType = &H0001
          oCmd.CommandText = xSql
          Set SqlQuery = oCmd.Execute
          oCmd.ActiveConnection = Nothing
          Set oCmd = Nothing
End Function

Public Function SqlAction(xSql)
          Rem Performs SQL action on table.

          Dim oCmd

          If mDbgMode Then
                    Response.Write xSql & "<br>"
          End If

          SqlAction = False
          Set oCmd = CreateObject("ADODB.Command")
          oCmd.ActiveConnection = mConnStr
          oCmd.CommandType = &H0001
          oCmd.CommandText = xSql
          oCmd.Execute
          oCmd.ActiveConnection = Nothing
          Set oCmd = Nothing
          SqlAction = True
End Function

Rem Define Properties.

Public Property Let DebugMode(xDbg)
          mDbgMode = xDbg
End Property

Public Property Get ID()
          ID = mID
End Property

Public Property Get Surname()
          Surname = mSurname
End Property

Public Property Get Forename()
          Forename = mForename
End Property

Public Property Get Telephone()
          Telephone = mTelephone
End Property

Public Function Save(xID, xSurname, xForename, xTelephone)
          Rem Saves record to table.

          Dim SqlStr
          Dim oRst

          Save = 0
          Call Init()

          If CLng(xID) = CLng(0) Then
                    SqlStr = "INSERT INTO tblTelDir(ID, Surname, Forename, Telephone) VALUES (" & _
                              xID & ", " & Quoted(xSurname) & ", " & Quoted(xForename) & ", " & _
                              Quoted(xTelephone) & ")"
                    Call SqlAction(SqlStr)

                    SqlStr = "SELECT MAX(ID) AS ID FROM tblTelDir ORDER BY ID DESC"
                    Set oRst = SqlQuery(SqlStr)
```

```
                        If Not oRst.EOF Then
                                Save = Load(oRst.Fields("ID"))
                        End If
                Else
                        SqlStr = "UPDATE tblTelDir SET Surname = " & Quoted(xSurname) & _
                                 ", Forename = " & Quoted(xForename) & _
                                 ", Telephone = " & Quoted(xTelephone) & " WHERE ID = " & xID
                        Call SqlAction(SqlStr)
                        Save = Load(xID)
                End If

                Set oRst = Nothing
        End Function

        Public Function Load(xID)
                Rem Loads record from table.

                Dim SqlStr
                Dim oRst

                Load = 0
                Call Init()

                SqlStr = "SELECT * FROM tblTelDir WHERE ID = " & xID
                Set oRst = SqlQuery(SqlStr)

                If Not oRst.EOF Then
                        mID = oRst.Fields("ID")
                        mSurname = oRst.Fields("Surname")
                        mForename = oRst.Fields("Forename")
                        mTelephone = oRst.Fields("Telephone")
                        Load = mID
                End If
        End Function

        Public Sub Delete(xID)
                Rem Deletes record from table.

                Dim SqlStr

                Call Load(xID)
                SqlStr = "DELETE FROM tblTelDir WHERE ID = " & xID
                Call SqlAction(SqlStr)
        End Sub
End Class
```

4 Using The Script Class

4.1 Overview

Having produced the script for table manipulation it is time to put it to work. For our example project we shall name the script file as Class_tblTelDir.asp and place this in a subfolder on our web site called “Includes”.

Using the script on an ASP web page requires the following steps to be performed:

- Include the script file containing our table class.
- Create an instance of the script.
- Call the methods and properties of the class.
- Terminate the instance of the script class.

Each of these steps is detailed below.

4.2 Include Script File

To include the script file in our ASP web page we insert the following line of code at the top of the web page, at line 2 of the ASP web page:

```
<!-- #include file=Includes/Class_tblTelDir.asp ✍
```

4.3 Create An Instance Of Script Class

Creating an instance of the script class involves adding the following line of code to our ASP web page:

```
<%
        Set oTbl = New Class_tblTelDir
%>
```

4.4 Call Methods/Properties Of Script Class

Having instantiated a copy of the script class for managing our database table we proceed to call the methods and/or properties of the class. An example of this would be similar to the following:

```
<%
        TelId = 0
        TelId = oTbl.Save(TelId, “Smith”, “John”, “0123-445566”)
%>
```

In this example we are saving a new telephone directory entry because we are calling the Save() method with a value of zero for the ID. This would cause the script class to create and execute a SQL INSERT statement.

If we had set a value for the ID as follows then the script class would create and execute an UPDATE statement instead.

```
<%
        TelId = 123
        TelId = oTbl.Save(TelId, “Smith”, “John”, “0123-445566”)
%>
```

The effect of this would be to replace an existing record having an ID of 123, rather than create a whole new record.

5 Class Methods

5.1 Overview

The script class generated by CBP will have the following standard methods defined.

- Quoted()
- SqlAction()
- SqlQuery()
- Save()
- Load()
- Delete()

5.2 Quoted()

A standard function which replaces a single quote in a string with a double quote. This is necessary for constructing SQL commands which are issued to the database.

This function takes a string as a parameter, and returns the same string with the single quotes replaced by double quotes.

5.2.1 Example

```
SqlStr = oTbl.Quoted("Drink O'Murphy's Guinness")
```

5.3 SqlAction()

A standard function enabling the host web page to issue a SQL action (for example a DELETE, INSERT or UPDATE on the database. This action does not have to be limited to the single table that the class defines – it could be database wide.

This function takes a string defining the SQL action which is to be performed. There is no return value from this function.

5.3.1 Example

```
Call oTbl.SqlAction("DELETE FROM tblXyz WHERE ID = 123")
```

5.4 SqlQuery()

A standard function enabling the host web page to request a recordset from the database via a SELECT statement. This request does not have to be limited to the single table that the class defines – it could be database wide.

This function takes a string defining the query which is to be performed (e.g. "SELECT * FROM tblXyz WHERE ID = 123"). A disconnected ADO recordset is returned to the caller.

5.4.1 Example

```
Set oRst = oTbl.SqlQuery("SELECT * FROM tblXyz WHERE ID = 123")
```

5.5 Save()

This function will save a record to the table. This acts upon the single table to which the class refers, and all column values must be supplied in the exact order of the columns defined in the table.

Save() figures out whether to use an INSERT or an UPDATE according to the first parameter which (as advised in section 1.2) must be an identity column. If this value is zero a new record is added to the table via an INSERT statement, otherwise an UPDATE statement is issued to the database.

On completion of the Save() operation the class will automatically perform a Load() of the record, thus making the class properties valid.

Save() will return the identity value for the record just saved, so if this were a new record (involving an INSERT statement) the calling process will be informed of that new records identity without further work being required.

5.5.1 Example

```
TelId = 0
TelId = oTbl.Save(TelId, “Smith”, “John”, “0123-445566”)
```

5.6 Load()

This function loads the contents of one record into the properties of the class. It requires a single parameter to be passed, this being the value of the identity column to be matched (see section 1.2). If the supplied identity value is not matched in the database (e.g. it can legally be zero) then the class properties are initialised to default values.

Load() returns the identity value for the record just loaded. This value should always be the value of the identity passed as a parameter, however in the event that the identity is not found in the database table then zero will be returned (this could be used to confirm that the requested record actually exists in the database table).

5.6.1 Example

```
TelId = 123
TelId = oTbl.Load(TelId)
```

5.7 Delete()

This subroutine performs a delete of the requested record from the database table. Prior to actually deleting the record the records contents will be automatically loaded via a call to the Load() function. Therefore the calling process could potentially make use of the deleted records contents after the record has been deleted from the database.

5.7.1 Example

```
TelId = 123
Call oTbl.Delete(TelId)
```

6 Class Properties

6.1 Overview

CBP will automatically generate read-only properties which are the names of the columns in the database table. This means that the contents of any column of a record in the table can be used by issuing a call to the Load() method, after which the properties will be valid.

One other “special” property is a write-only property, this being DebugMode as explained below.

6.2 DebugMode

During development it isn’t unusual for SQL scripting errors to cause problems, and these can be tricky to track down.

The database class exposes a write-only property called DebugMode. Setting this to True will cause any SQL issued by the class to be written verbatim to the host web page, immediately before that SQL is actually executed. The results of this will not be pretty, but it can be a lifesaver on a web development project.

6.2.1 Example

```
oTbl.DebugMode = True
```

7 About The Author

Andrew McKay is based in the United Kingdom and has over 25 years direct experience within the IT industry in various roles. He is a full member of the Institute of Analysts and Programmers (http://www.iap.org.uk).

After leaving school Andrew went on to college in Southampton, England, to gain his OND in Engineering followed by an HNC in Electronics. After four years working for Mullards Semiconductors in Southampton as a hardware engineer Andrew joined Hewlett Packard as a field-based customer engineer. In those days of the late 1970's and early 1980's HP had a desktop computer division which was eventually swallowed up into the computer division which continues to exist today. Andrew was to become one of the primary UK technical support personnel for the now obsolete HP1000 range of systems, many people will still remember with affection the HP1000 M/E/F series.

In 1986 Andrew changed from a hardware focussed career to an applications support role, looking after HP's Factory Automation products. This role involved pre-sales and support, designing and delivering customer training courses, and occasional attendance and stand duty at trade exhibitions at Earls Court, Olympia and the NEC.

In 1990 HP took a corporate decision to withdraw from offering software solutions to focussing their activities as a hardware vendor. After a short secondment to the business administration team where Andrew revolutionised reporting of consultancy revenue streams throughout HP Europe he moved on to join the Integrated Systems Division in Birmingham, UK, where he worked as part of a small project team continuing the development of the Rover TestBook product. This was a ruggedised PC with a touchscreen interface which can today be found in all Rover and Land Rover dealerships worldwide.

Andrew started his own consultancy company, KazMax Ltd, in 1997 since when he has focussed on delivering web based solutions, with particular emphasis on middle-tier component (MTS and COM+) and back end database design and development. KazMax Ltd offer web design and hosting for other small businesses and maintain web sites on behalf of other clients.

As a life member of the Association of Shareware Professionals Andrew continues to offer try-before-you-buy software that he has written to the world community. His KazPlan product (http://www.kazplan.com) is a resource planning and conference scheduling solution used all around the world. Available in two compatible versions for single user and corporate implementation, KazPlan utilises a Microsoft SQL Server database to host the functionality of supporting up to an unlimited number of users.

Andrew is available to undertake bespoke assignments for other organisations can be contacted via his company web site which can be found on the Internet at http://www.kazmax.co.uk.