**ADVANCE QTP TIPS**

**Mercuryquicktestprofessionalblogspot.com**

**Sub Procedures**

A sub procedure is a series of VBScript statements, enclosed by Sub and End Sub statements which perform actions but do not return a value. A sub procedure can take arguments. If a sub procedure doesn’t receive any arguments, its Sub statement must include an empty parenthesis().

The following Sub procedure uses two intrinsic, or built-in, VBScript functions, MsgBox and InputBox , to prompt a user for information. It then displays the results of a calculation based on that information. The calculation is performed in a Function procedure created using VBScript. The Function procedure is shown after the following discussion.  
  
Sub ConvertTemp()

temp = InputBox("Please enter the temperature in degrees F.", 1)

MsgBox "The temperature is " & Celsius(temp) & " degrees C."

End Sub

**Function Procedures**

A function procedure is a series of VBScript statements enclosed by the Function and End Function statements. **A function procedure is similar to a sub procedure but it can return value to the calling function.** A function procedure can take arguments (constants, variables or expressions that are passed to it by a calling procedure). If a function procedure has no arguments, it Function statement must include an empty set of parenthesis. A function returns a value by assigning a value to its name in one or more statements of the procedure. Since VBScript has only one base data type, a function always returns a variant.

In the following example, the Celsius function calculates degrees Celsius from degrees Fahrenheit. When the function is called from the ConvertTemp **Sub** procedure, a variable containing the argument value is passed to the function. The result of the calculation is returned to the calling procedure and displayed in a message box.

Sub ConvertTemp()

temp = InputBox("Please enter the temperature in degrees F.", 1)

MsgBox "The temperature is " & Celsius(temp) & " degrees C."

End Sub

Function Celsius(fDegrees)

Celsius = (fDegrees - 32) \* 5 / 9

End Function

**Tips:**

1. To get data out of a procedure, you must use a Function. Remember, a Function procedure can return a value; a Sub procedure can't.
2. AFunction in your code must always be used on the right side of a variable assignment or in an expression.
3. To call a **Sub** procedure from another procedure, type the name of the procedure along with values for any required arguments, each separated by a comma. The Call statement is not required, but if you do use it, you must enclose any arguments in parentheses.
4. The following example shows two calls to the MyProc procedure. One uses the Call statement in the code; the other doesn't. Both do exactly the same thing.

Call MyProc(firstarg, secondarg)  
  
MyProc firstarg, secondarg  
  
Notice that the parentheses are omitted in the call when the **Call** statement isn't used.

**Chapter – 3**

**AUTO EXPAND VB SCRIPT SYNTAX**

Tool- View Option – Auto Expand VB Script Syntax

Please make it enable.

**AN OVERVIEW OF WITH STATEMENT**

Tool -> Options -> “Automatically generate the with statement after recording”

Note: Desable/Enable

OR

Edit -> Advance -> Apply with to script.

With Dialog("Login")

.Activate

.WinEdit("Agent Name:").Set "fffff"

.WinEdit("Password:").SetSecure "4f50628457dfdfb25f542f728f94bbe36232f45a"

.WinButton("OK").Click

End With

Window("Flight Reservation").WinMenu("Menu").Select "File;Exit"

SystemUtil.Run "C:\Program Files\Mercury Interactive\QuickTest Professional\samples\flight\app\flight4a.exe","","C:\Program Files\Mercury Interactive\QuickTest Professional\samples\flight\app\","open"

**FINDING THE SYNTAX ERROR**

Tools -> Check Syntax

**EXICUTING THE SCRIPT FROM A SPECIFIC STEP**

Automation -> Run from step (To start the execution)

Break Points -> To stop the execution.

**ADDING BREAK POINTS**1. Place the cursor on the step where you want to add the break points.

2. Debug -> add/remove break points

Note : QTP pause the run session before executing the step.(Used for script debugging)

**Run Time Object Property :**

QTP uses the test objects property and value to indentify the run time objects in AUT.

GetToProperty : Retrieves the property and value of run time objects

<object>.GetTOProperty(<property>)

GerROProperty:

<object>.GetROProperty(<property>)

**PROGRAMS**

**Environment Variable:**

path1=Environment.Value("Flight")

SystemUtil.Run path1

Note :

**File -> Setting -> Enviornment -> User Defined -> +**

**Name :**

**Value: “application path”**

**Function Library:**

**File->New->Function Library(.qfl)**

**File ->associate function library with current test**

**Or**

**Resources-> associate function library**

**Note : function can be defined in external notepad too , we should save it in .vbs**

**Reusable Action :**

**We can call the reusable action from a new script.**

1. **Write click on expert view pane.**
2. **Call To existing action.**
3. **Select the reusable action.**

RunAction "Action1 [test1]", oneIteration

‘Descriptive Program : (Property – Value Pair)

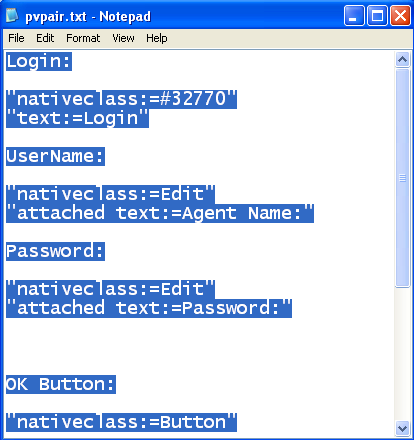
‘Descriptive program(Property-Value Pair)

‘Syntax : object("property:=value")

Dialog("text:=Login","nativeclass:=#32770").WinEdit("nativeclass:=Edit","attached text:=Agent Name:").Set"ppppp"

Dialog("text:=Login","nativeclass:=#32770").WinEdit("nativeclass:=Edit","attached text:=Password:").Set"mercury"

Dialog("text:=Login","nativeclass:=#32770").WinButton("nativeclass:=Button","text:=OK").Click



**Description – Object**

Dim l,u,p,b

Set l= Description.Create()

Set u=Description.Create()

Set p=Description.Create()

Set b=Description.Create()

l("nativeclass").Value="#32770"

l("text").Value="Login"

u("nativeclass").Value="Edit"

u("attached text").Value="Agent Name:"

p("nativeclass").Value="Edit"

p("attached text").Value="Password:"

b("nativeclass").Value="Button"

b("text").Value="OK"

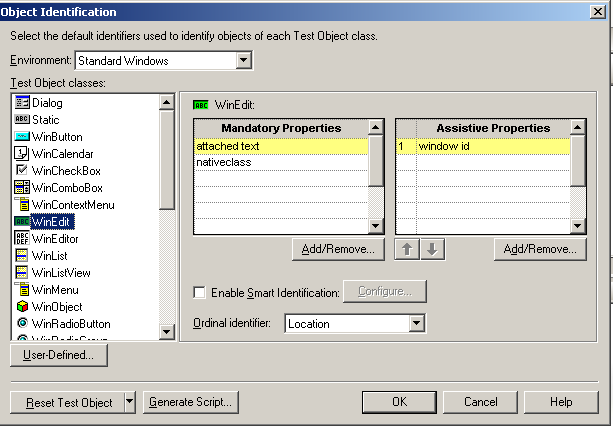
Dialog(l).WinEdit(u).Set "pppp"

Dialog(l).WinEdit(p).Set"mercury"

Dialog(l).WinButton(b).Click

Object Identification :

Tool -> Object Identification



Local Data Sheet:

File->Setting ->Run -> Run on all rows

Satish’s Program For DB CONNECT(OPTIONAL TO LEARN)

Test

DataTable.ImportSheet"D:\abc.xls",2,2 ' Iporting Data into Data Table from an Excel File

row = 1

varRowCount = DataTable.GetSheet(2).GetRowCount

For i = 1 to varRowCount

Window("Flight Reservation").Activate

Window("Flight Reservation").WinObject("Date of Flight:").Type "111112"

var\_CityFrom = DataTable.Value("CityFrom",2)

FlightFrom = get\_Flyfrom\_City(var\_CityFrom) ' call function to select From City

Window("Flight Reservation").WinComboBox("Fly From:").Select (FlightFrom)

var\_CityTo = DataTable.Value("CityTo",2)

FlightTo = get\_Flyto\_City(var\_CityTo) ' call function to select To City

Window("Flight Reservation").WinComboBox("Fly To:").Select (FlightTo)

Window("Flight Reservation").WinButton("FLIGHT").Click

var\_FlightNo = DataTable.Value("FlightNumber",2)

FlightNo = select\_Flight(var\_FlightNo) ' call function to select Flight Number

If FlightNo <> "" Then

Window("Flight Reservation").Dialog("Flights Table").WinList("From").Select(FlightNo)

Window("Flight Reservation").Dialog("Flights Table").WinButton("OK").Click

else

Reporter.ReportEvent micFail, "Flight Number", "Flight Number not Found"

DataTable.Value("Tester\_Comments",2) = "Flight Not Found"

DataTable.Value("Result",2) = "Fail"

Call Export\_Open() ' call function to Export the Results into an External Excel File(D:\results.xls) , Invalid Case

ExitTest

End If

Window("Flight Reservation").WinEdit("Name:").Set "sateesh"

Window("Flight Reservation").WinEdit("Tickets:").SetSelection 0,1

var\_Tickets = DataTable.Value("Tickets",2)

NumTickets = insert\_Ticket(var\_Tickets) ' call function to validate Tickets

Window("Flight Reservation").WinEdit("Tickets:").Set NumTickets

Window("Flight Reservation").WinButton("Insert Order").Click

Window("Flight Reservation").WinObject("Insert Done").WaitProperty "text", "Insert Done...", 10000

OrdNum = Window("Flight Reservation").WinEdit("Order No:").GetROProperty("text")

verifyOrderNo = verify\_InsertOrd(OrdNum) ' call function to check the Insert Order Number

If verifyOrderNo = True Then

DataTable.Value("Tester\_Comments",2) ="Order Number Found in the Database"

else

DataTable.Value("Tester\_Comments",2) ="Order Number NOT Found in the Database"

End If

' DataTable.Value("Tester\_Comments",2) = "Order Verified"

Window("Flight Reservation").Activate

Window("Flight Reservation").WinButton("Button\_5").Click

DataTable.Value("Result",2) = "Pass"

row = row+1

DataTable.SetCurrentRow(row)

Next

Call Export\_Open() ' call function to Export the Results into an External Excel File(D:\results.xls), Valid Case

Function Library:

Function get\_Flyfrom\_City(argFlyFrom)

Dim x, i, fromFound

x = Window("Flight Reservation").WinComboBox("Fly From:").GetItemsCount

i = 0

fromFound = False

Do Until fromFound Or (i>= x)

strItem1 = Window("Flight Reservation").WinComboBox("Fly From:").GetItem(i)

If strItem1 = argFlyFrom Then

fromFound = True

get\_Flyfrom\_City = strItem1

else

i = i+1

get\_Flyfrom\_City= Window("Flight Reservation").WinComboBox("Fly From:").GetItem(0)

End If

Loop

End Function

Function get\_Flyto\_City(argFlyTo)

Dim y, j, toFound

y = Window("Flight Reservation").WinComboBox("Fly To:").GetItemsCount

j = 0

toFound = False

Do Until toFound Or (j>= y)

strItem2 = Window("Flight Reservation").WinComboBox("Fly To:").GetItem(j)

If strItem2 = argFlyTo Then

toFound = True

get\_Flyto\_City = strItem2

else

j = j+1

get\_Flyto\_City = Window("Flight Reservation").WinComboBox("Fly To:").GetItem(0)

End If

Loop

End Function

Function select\_Flight(argFlightNo)

z = Window("Flight Reservation").Dialog("Flights Table").WinList("From").GetItemsCount

For k = 0 to z-1

strFlightRow = Window("Flight Reservation").Dialog("Flights Table").WinList("From").Getitem(k)

arrStr = split(strFlightRow, " ")

Flight\_No = arrStr(0)

If Flight\_No = argFlightNo Then

select\_Flight = k

exit for

else

select\_Flight = ""

End If

Next

End Function

Function insert\_Ticket(argTickets)

If 0>argTickets Or argTickets>10 Then

MsgBox "Only ten tickets may be ordered at one time"

argTickets = InputBox("Enter tickets")

Call insert\_Ticket(argTickets)

End If

insert\_Ticket = argTickets

End Function

Function verify\_InsertOrd(argVerifyOrder)

Set dbConnect = CreateObject("ADODB.Connection")

dbConnect.ConnectionString = "DSN=QT\_Flight32"

dbConnect.Open()

query = "select \* from Orders where Order\_Number ="&argVerifyOrder

Set res = dbConnect.execute(query)

Do until res.eof

strOrder = res.Fields("Order\_Number")

res.movenext

Loop

If strOrder = "" Then

verify\_InsertOrd = False

else

verify\_InsertOrd = True

End If

dbConnect.Close

Set dbConnect = nothing

End Function

Function Export\_Open()

DataTable.Export"E:\Results.xls"

SystemUtil.Run"E:\Results.xls"

End Function

Schedule :

1. Function Library chapter - 9(done)

2. Database Connectin Chapter - 8(Pending)

3. VB Script : ch 6,7(pending)

4. Working with Object property Ch-4(Pending)

5. Descriptive Programing - Ch-5(Done)

6. Step , Step into , step out , step over

7. Break points

8. Debuging the existing script

Use of GetROProperty :

Browser("Advantage Shopping").Page("Register Account").WebEdit("firstname").Set DataTable("A", dtGlobalSheet)

Browser("Advantage Shopping").Page("Register Account").WebEdit("lastname").Set DataTable("B", dtGlobalSheet)

Str1=Browser("Advantage Shopping").Page("Register Account").WebButton("Continue").GetROProperty("name")

msgbox Str1