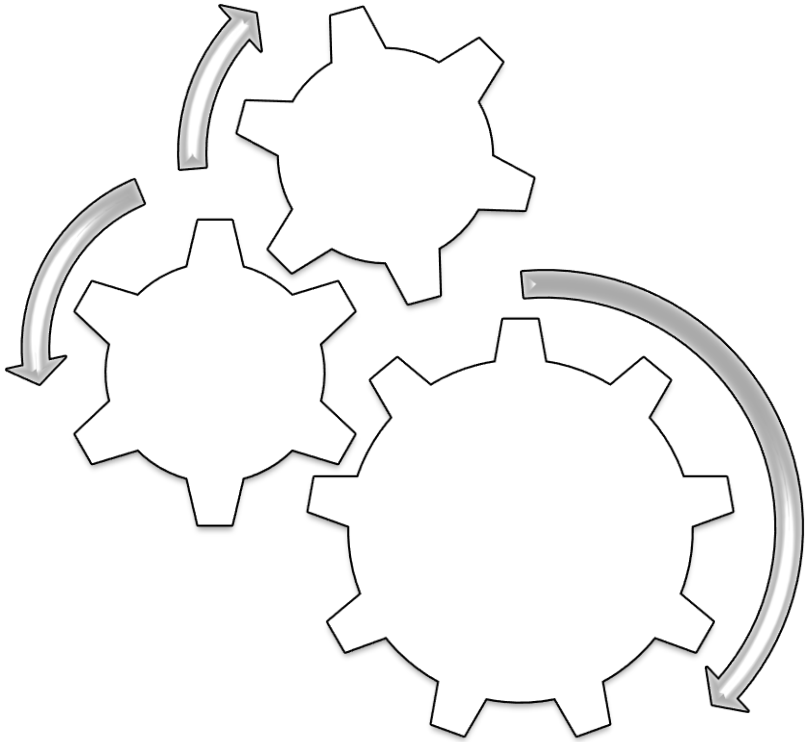


Dive into QTP

Weifeng Lu

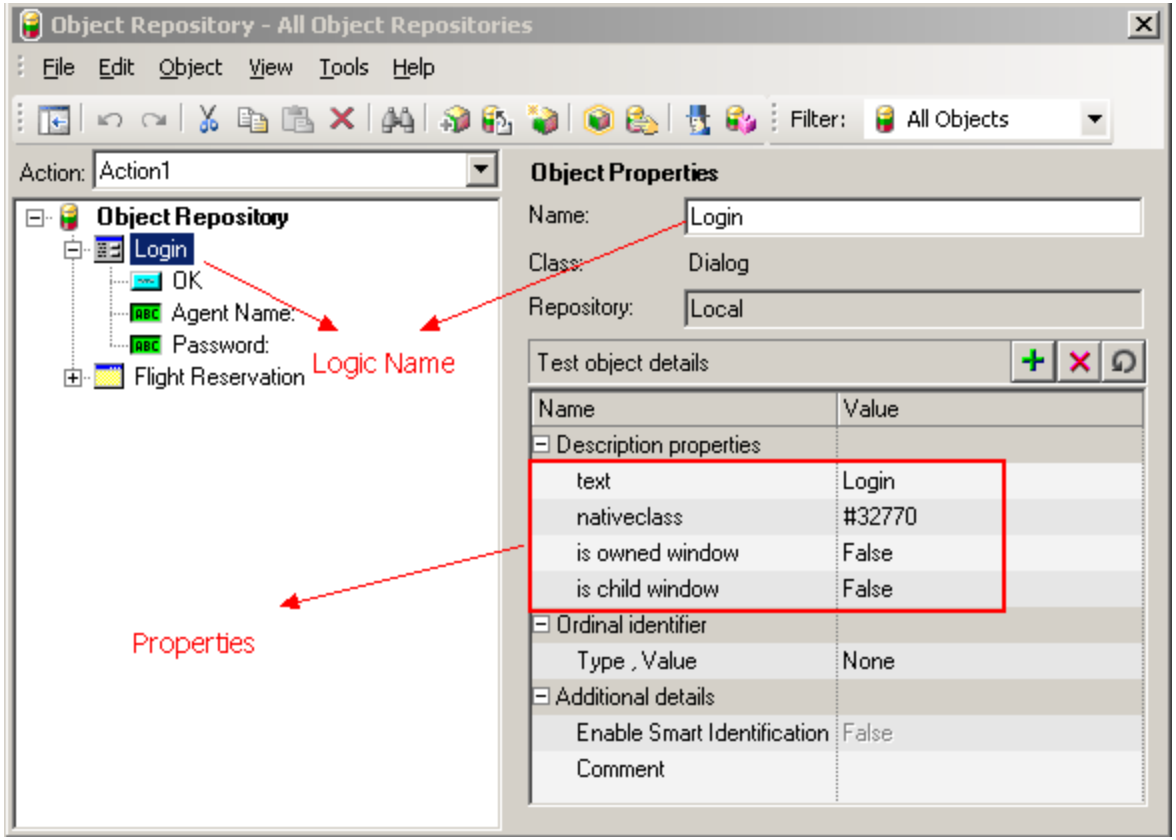
07/08/11

Agenda

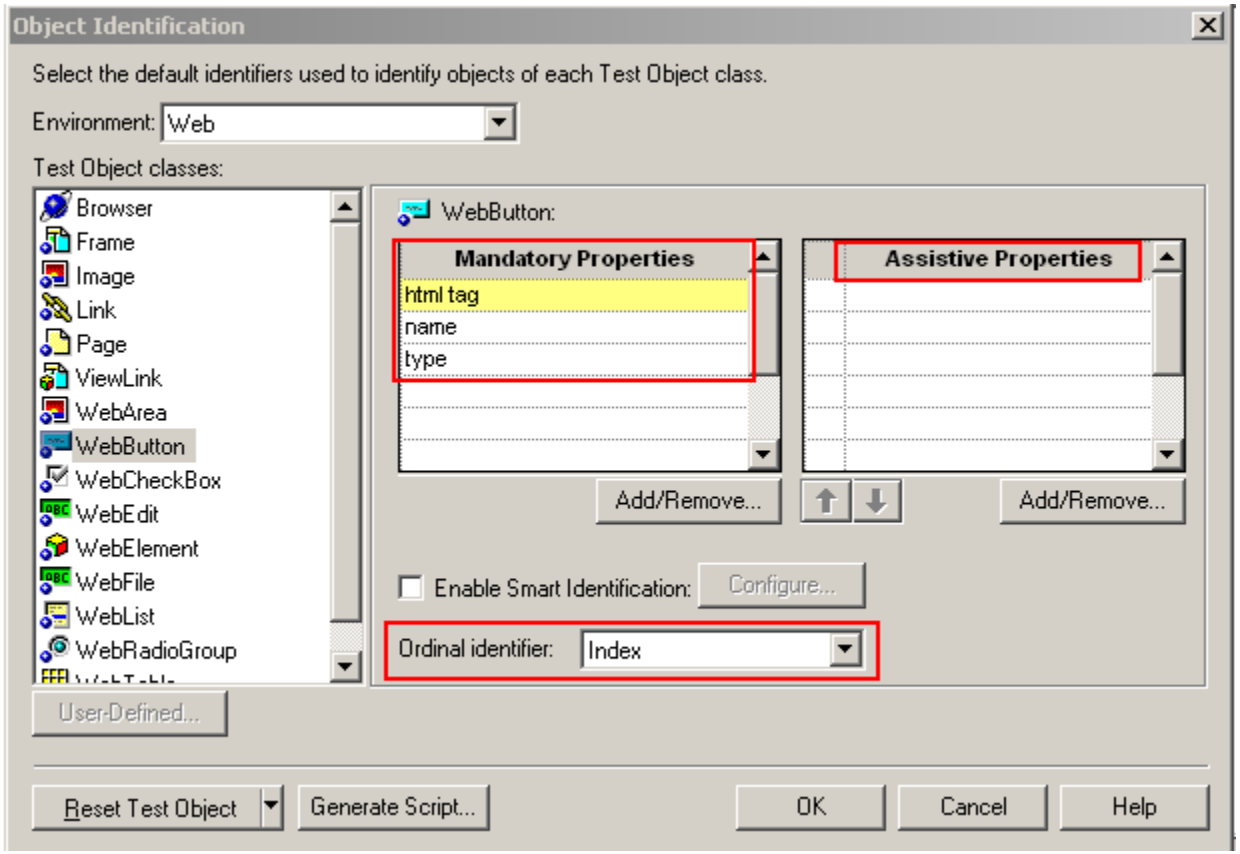
- Object Repository (OR)
- Descriptive Programming (DP)
- TO, RO and .object
- Text Recognition
- Reserved Objects
- Mouse & Keyboard
- Automation Object Model
- Error Handling & Exception Handling
- Design Pattern

Object Repository (OR)

- Object Repository is a place where QTP stores learned objects
- QTP uses default Object Identification properties: mandatory and assistive to learn objects into OR



Object Identification



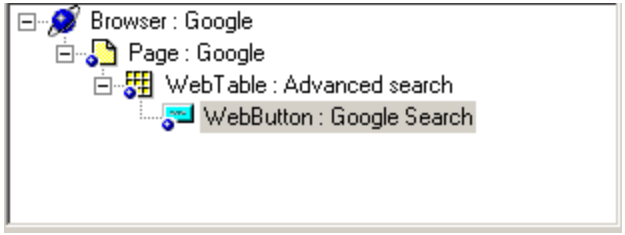
Script playback using OR

- QTP finds the Object in Object Repository using object Logical Name and Object Hierarchy
- QTP retrieves corresponding Object description properties from OR
- QTP searches actual Runtime Object with the same properties as the OR Object description properties and performs user action

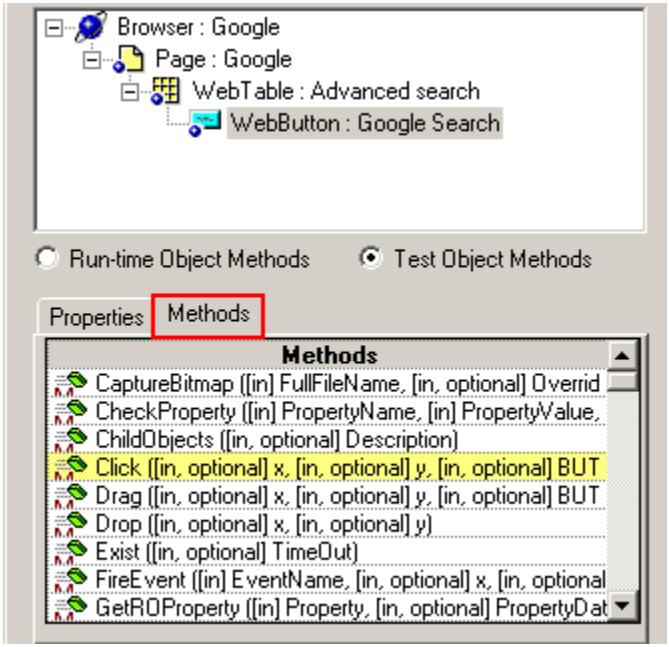
Descriptive Programming (DP)

1. Concepts

```
' Syntax
ObjectClassName("property1:=value1")
' Example
Browser("title:=Google")
' Of course you can use as many properties as you like:
ObjectClassName("property:=value", "property:=value", "property:=value")
' Example
WebEdit("name:=q", "html tag:=INPUT", "title:=demo")
```




```
' Connect all these descriptions and form a hierarchical tree to identify the WebButton
Browser("title:=Google").Page("title:=Google").WebTable("html tag:=TABLE").WebButton("html tag:=INPUT", "value:=Google Search")
' Skip the WebTable object to identify the WebButton
Browser("title:=Google").Page("title:=Google").WebButton("html tag:=INPUT", "value:=Google Search")
' Skip the Page and WebTable object to identify the WebButton
Browser("title:=Google").WebButton("html tag:=INPUT", "value:=Google Search")
```




```
' Add an event which can be described as action on target object
Browser("title:=Google").Page("title:=Google").WebButton("html tag:=INPUT", "value:=Google Search").click
```

Descriptive Programming (DP)

2. Regular Expressions

```
' Only using the first 2 words: getAll
' . Matches any single character except a newline character
' *  Matches the preceding character zero or more times. For example, "zo*" matches either "z" or "zoo"
Browser( "title:=MyTitle" ).Page( "title:=MyTitle" ).Image( "file name:=getAll.*" ).Click
' Using 1 word (Attributes) with the extension (JPG)
Browser( "title:=MyTitle" ).Page( "title:=MyTitle" ).Image( "file name:=.*Attributes.*JPG" ).Click
```

```
' Without using regular express
Browser("title:=Welcome: Mercury Tours").Page("title:=Welcome: Mercury Tours").Image("file name:=banner2.gif").Highlight

' Using regular express
' \w  Matches any word character including underscore and whitespace (spaces, tabs, and line breaks). Equivalent to "[A-Za-z0-9_]"
' +  Matches the preceding character one or more times. For example, "zo+" matches "zoo" but not "z"
' \D  Matches a non-digit character. Equivalent to [^0-9]
' \d  Matches a digit character. Equivalent to [0-9]
Browser("title:=Welc\w+\D+\w+").Page("title:=Welc\w+\D+\w+").Image("file name:=ban\w+\d+\.\w+").Highlight
```

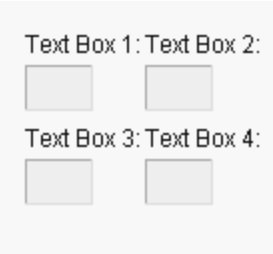
Descriptive Programming (DP)

3. Ordinal Identifiers

- An ordinal identifier assigns a numerical value to a test object that indicates its order or location relative to other objects with an otherwise identical description (objects that have the same values for all properties). This ordered value provides a backup mechanism that enables QuickTest to create a unique description to recognize an object when the defined properties are not sufficient to do so.
- The 3 types of ordinal identifiers

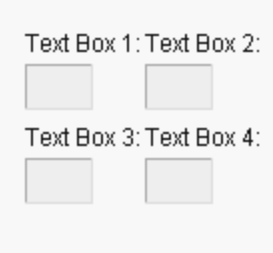
I. Location
II. Index
III. CreationTime

Location
from top to bottom, and left to right




```
'Text  Box  1
Browser("title:=.*Descriptive.*").Page("micclass:=Page").WebEdit("name:=dpTest","location:=0").Set  "1"

'Text  Box  3
Browser("title:=.*Descriptive.*").Page("micclass:=Page").WebEdit("name:=dpTest","location:=1").Set  "2"

'Text  Box  2
Browser("title:=.*Descriptive.*").Page("micclass:=Page").WebEdit("name:=dpTest","location:=2").Set  "3"

'Text  Box  4
Browser("title:=.*Descriptive.*").Page("micclass:=Page").WebEdit("name:=dpTest","location:=3").Set  "4"
```

Index
appearance of objects in the source code




```
'Text Box 1
Browser("title:=.*Descriptive.*").Page("micclass:=Page").WebEdit("name:=dpTest", "index:=0").Set "1"

'Text Box 2
Browser("title:=.*Descriptive.*").Page("micclass:=Page").WebEdit("name:=dpTest", "index:=1").Set "2"

'Text Box 3
Browser("title:=.*Descriptive.*").Page("micclass:=Page").WebEdit("name:=dpTest", "index:=2").Set "3"

'Text Box 4
Browser("title:=.*Descriptive.*").Page("micclass:=Page").WebEdit("name:=dpTest", "index:=3").Set "4"
```

Creation Time

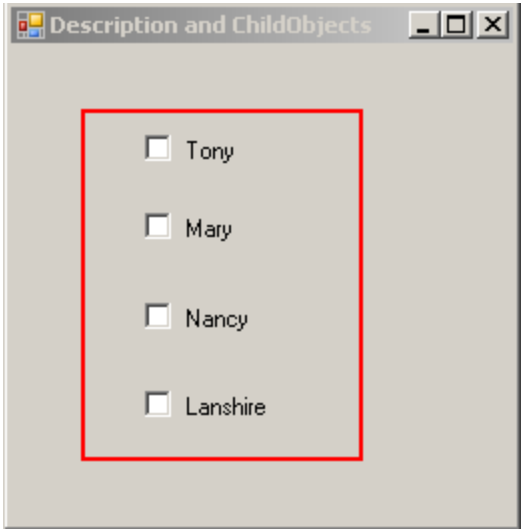
```
' CreationTime  0
SystemUtil.Run  "iexplore.exe",  "http://www.HP.com"
' CreationTime  1
SystemUtil.Run  "iexplore.exe",  "http://www.AdvancedQTP.com"
' CreationTime  2
SystemUtil.Run  "iexplore.exe",  "http://www.LinkedIn.com"
' Highlight  HP.com
Browser(  "creationtime:="  ).Highlight
' Highlight  AdvancedQTP.com
Browser(  "creationtime:=1"  ).Highlight
' Highlight  LinkedIn.com
Browser(  "creationtime:=2"  ).Highlight
```

Descriptive Programming (DP)

Description Object and ChildObjects method

```
' Description Object
Dim oDesc
' Object Collection
Dim colObject
' using the variable oDesc to create a description of something
Set oDesc = Description.Create
' Remember to always use 'micclass' and not 'class name'
oDesc( "micclass" ).value = "Link"
' Images
oDesc( "text" ).value = "I.*age.*"
' Retrive object collections which meet corresponding condition
Set colObject = Browser(
"title:=Google").Page("title:=Google").ChildObjects(oDesc)
' Retrieve # of the objects
Msgbox colObject.Count
```

Assume don't know how many are checkboxes (the checkboxes are created dynamically), but I want to mark all of them!




```
Set oDesc = Description.Create
oDesc("swfname").value = "CheckBox.*"
' Retrieve all CheckBox objects
Set oCheckBoxChildren =
SwfWindow("swfname:=form1").ChildObjects(oDesc)
' Iterator though oCheckBoxChildren objects and mark
them all
For i=0 To oCheckBoxChildren.count - 1
      oCheckBoxChildren(i).set "ON"
next
```

OR or Regular DP won't help - Don't know how to identify each checkbox

Descriptive Programming (DP)

Common Mistake - Using “Class Name” instead of “micclass”

```
' Below is the wrong way
Browser("Class Name:=Browser")

' Below is the right way
Browser("micclass:=Browser")

' Below is the wrong way
Set oDesc = Description.Create
oDesc("Class Name").Value = "Browser"
oDesc("title").Value = "My title"

' Below is the right way
Set oDesc = Description.Create
oDesc("micclass").Value = "Browser"
oDesc("title").Value = "My title"
```

Descriptive Programming (DP)

Common Mistake - Using strings with Pattern

```
' Let's assume we want to click a link "Logout (Demo)" on my web page. Two possible methods that can be used are
' Method 1
Browser("micclass:=Browser").Page("micclass:=Page").Link("text:=Logout (Demo)").Click
' Method 2
Set oDesc = Description.Create
oDesc("text").Value = "Logout (Demo)"
Browser("micclass:=Browser").Page("micclass:=Page").Link(oDesc).Click

' Method 1
Browser("micclass:=Browser").Page("micclass:=Page").Link("text:=Logout \(Tarun\)").Click
' Method 2
Set oDesc = Description.Create
oDesc("text").Value = "Logout \(Tarun\)"
Browser("micclass:=Browser").Page("micclass:=Page").Link(oDesc).Click
' Method 3
Set oDesc = Description.Create
oDesc("text").Value = "Logout (Tarun)"
'Do not treat the value as regular expression.
oDesc("text").RegularExpression = False
```

OR Pros and Cons

PROS:

- Easy to understand object hierarchies and maintain objects due to front end GUI
- No need to modify the script when object properties changes
- Can be created independently from scripts
- Support Auto-Complete feature
- Highlight in Application feature is great tool to walk the object tree

OR Pros and Cons

CONS:

- Unnecessary objects can be created
- Multiple users cannot concurrently save/write to the shared OR
- It won't eliminate the need for Descriptive Programming in most of cases

DP Pros and Cons

PROS:

- Code portability is high
- Easy to mass update(Copy/Paste)
- Compatible with different QTP versions
- Support Description Object and ChildObjects method

DP Pros and Cons

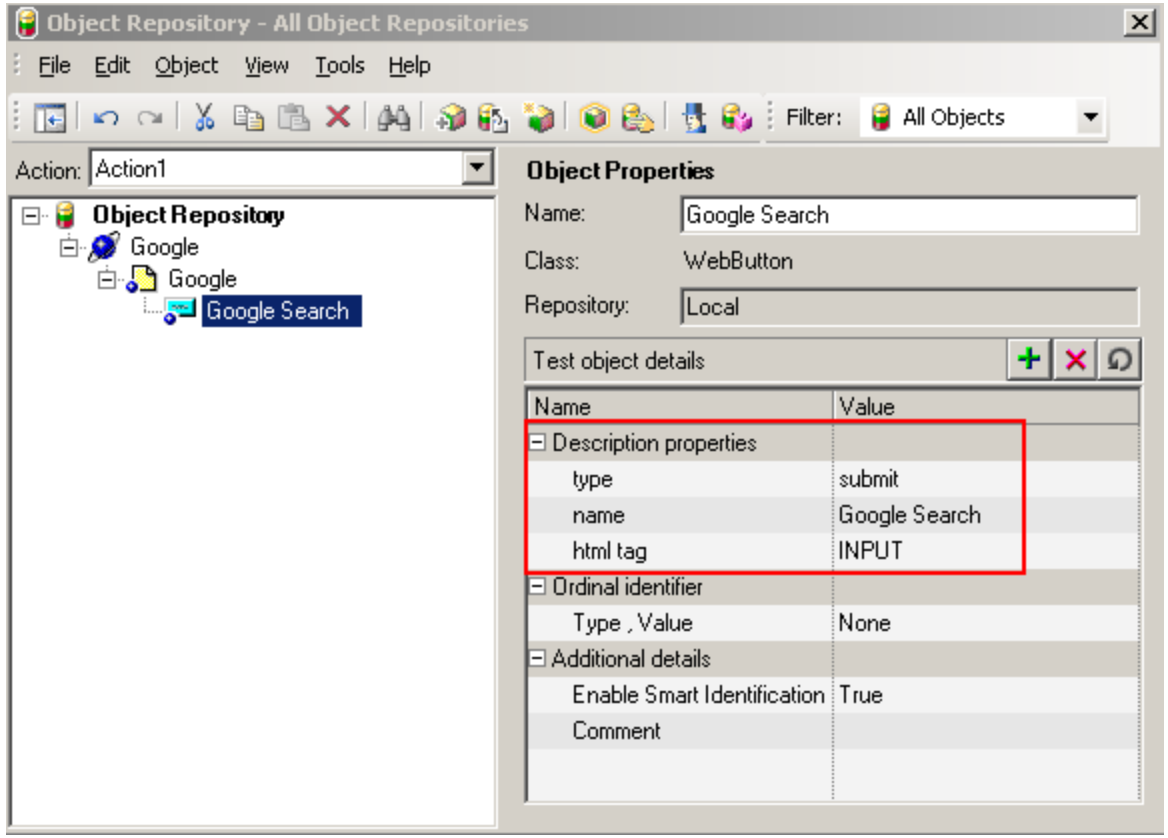
CONS:

- Lower Code Readability and requires more comments, like “what object is accessed”
- Potentially slower to create
- To highlight an object in the application requires utilizing the “Highlight” method

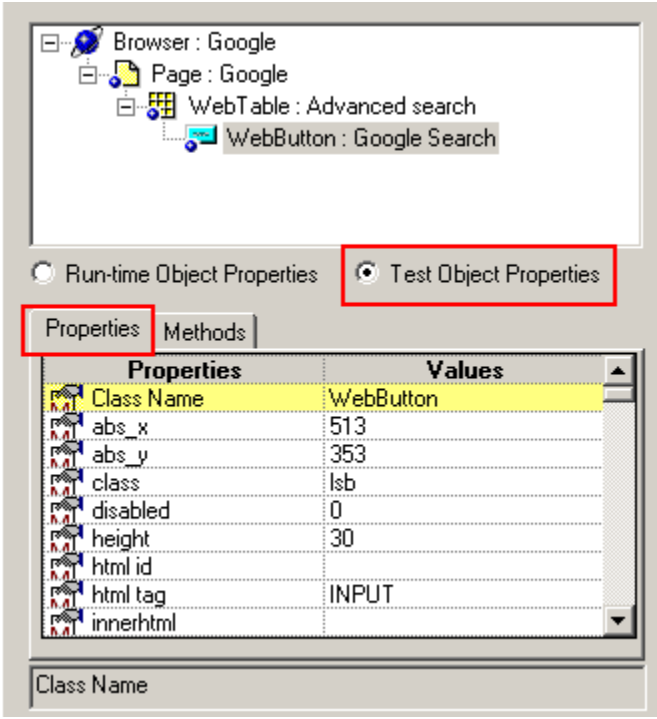
TO, RO and .object

- .GetTOproperty/SetToProperty refers to the **derived** property stored in OR
- .GetROProperty refers to the Runtime object **derived** property
- .method refers to the Runtime object **derived** method
- .object.<*property/method*> refers to the Runtime object **NATIVE** property/method

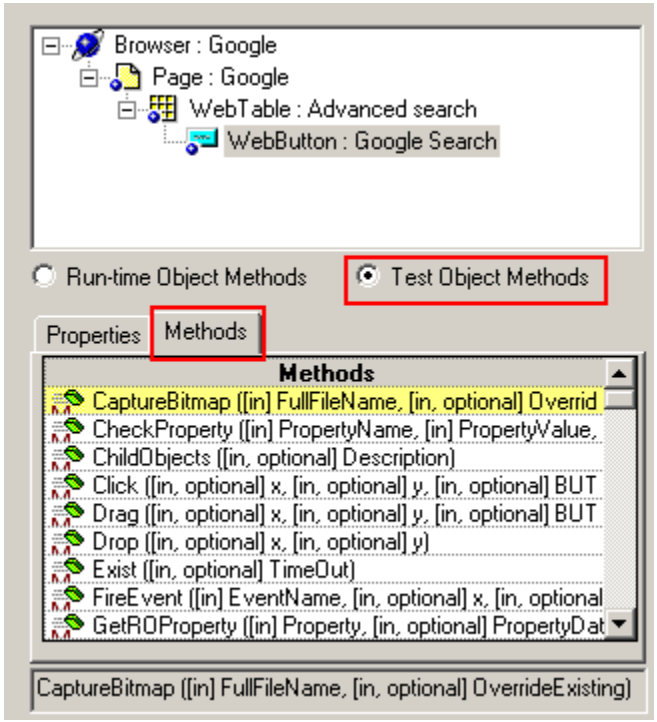
Derived property from OR (.GetTOproperty/SetToProperty)



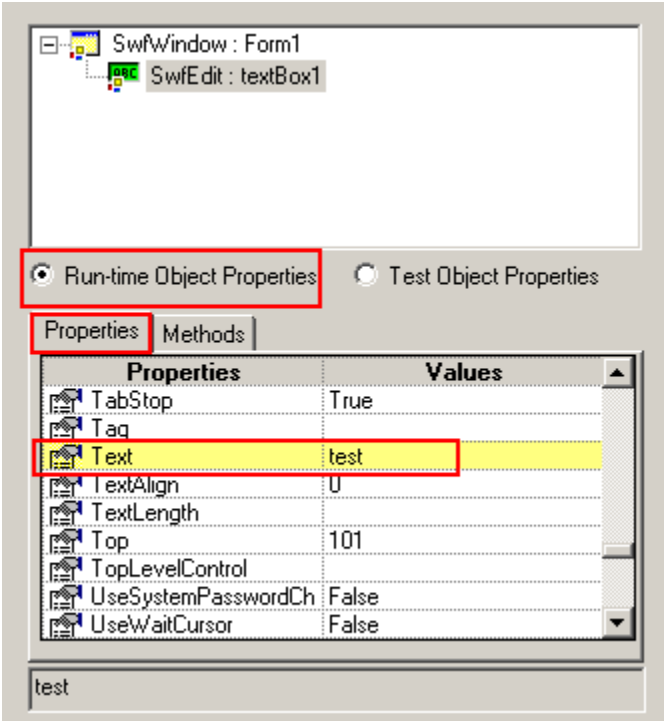
Runtime object derived property (.GetRoProperty)

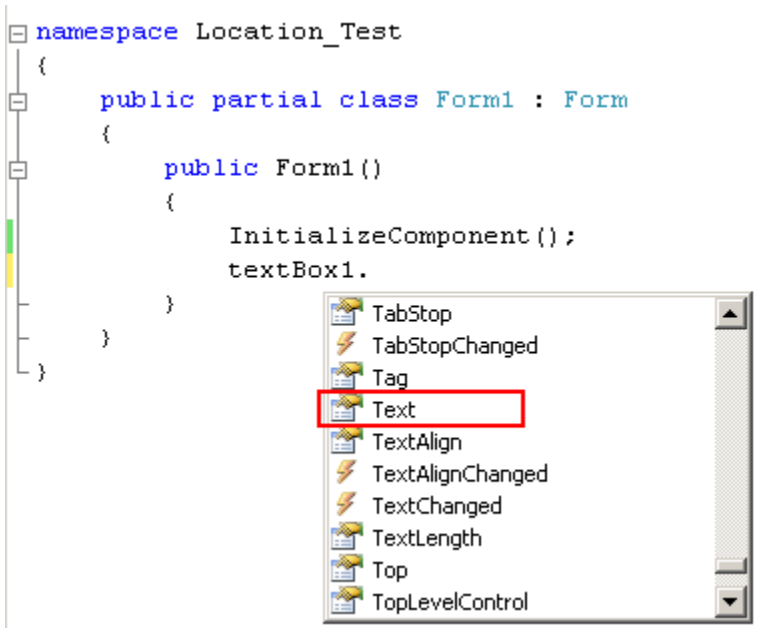


Runtime object derived method (.method)

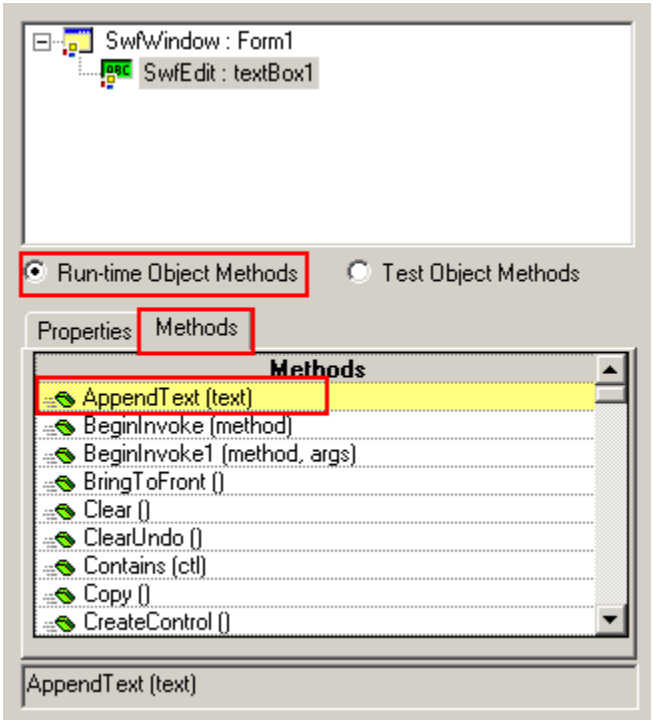


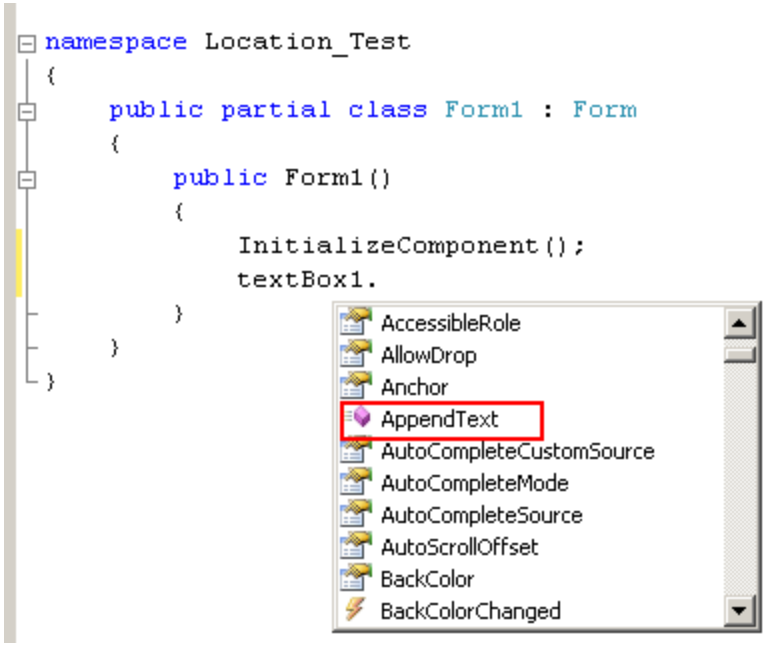
Runtime object native property (.object.property)





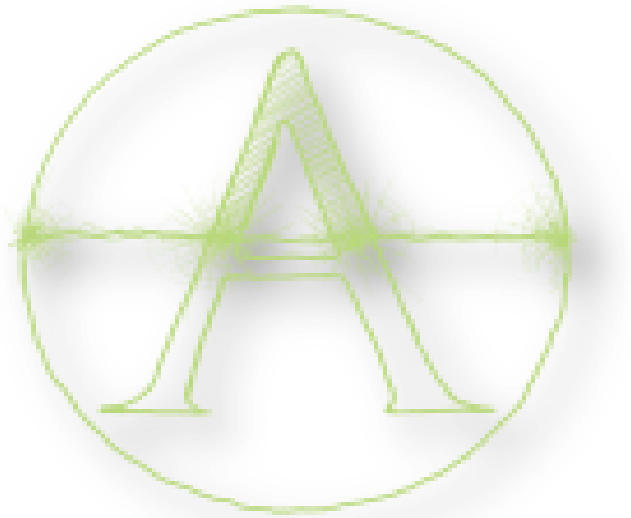
Runtime object native method (.object.method)

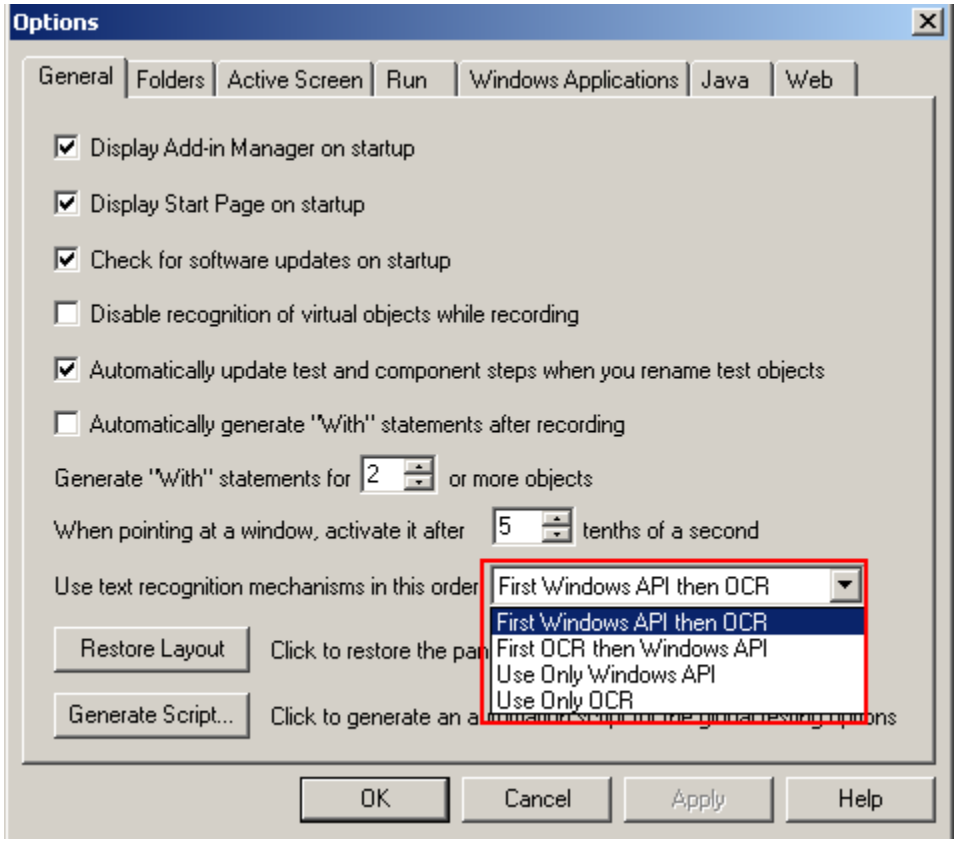




Text Recognition

- QuickTest tries to retrieve the text directly from the object using a Windows API-based mechanism (default mechanism). If QuickTest cannot capture the text this way (for example, because the text is part of a picture), it tries to capture the text using an OCR (optical character recognition) mechanism provided by ABBYY Software Ltd.
- Optical Character Recognition is the process of translating images of typewritten text into computer readable text. QTP can capture an image of an application screen and use OCR to “read” the text in it and convert it to usable ASCII or Unicode text. This text can be used to create solid, reliable tests.

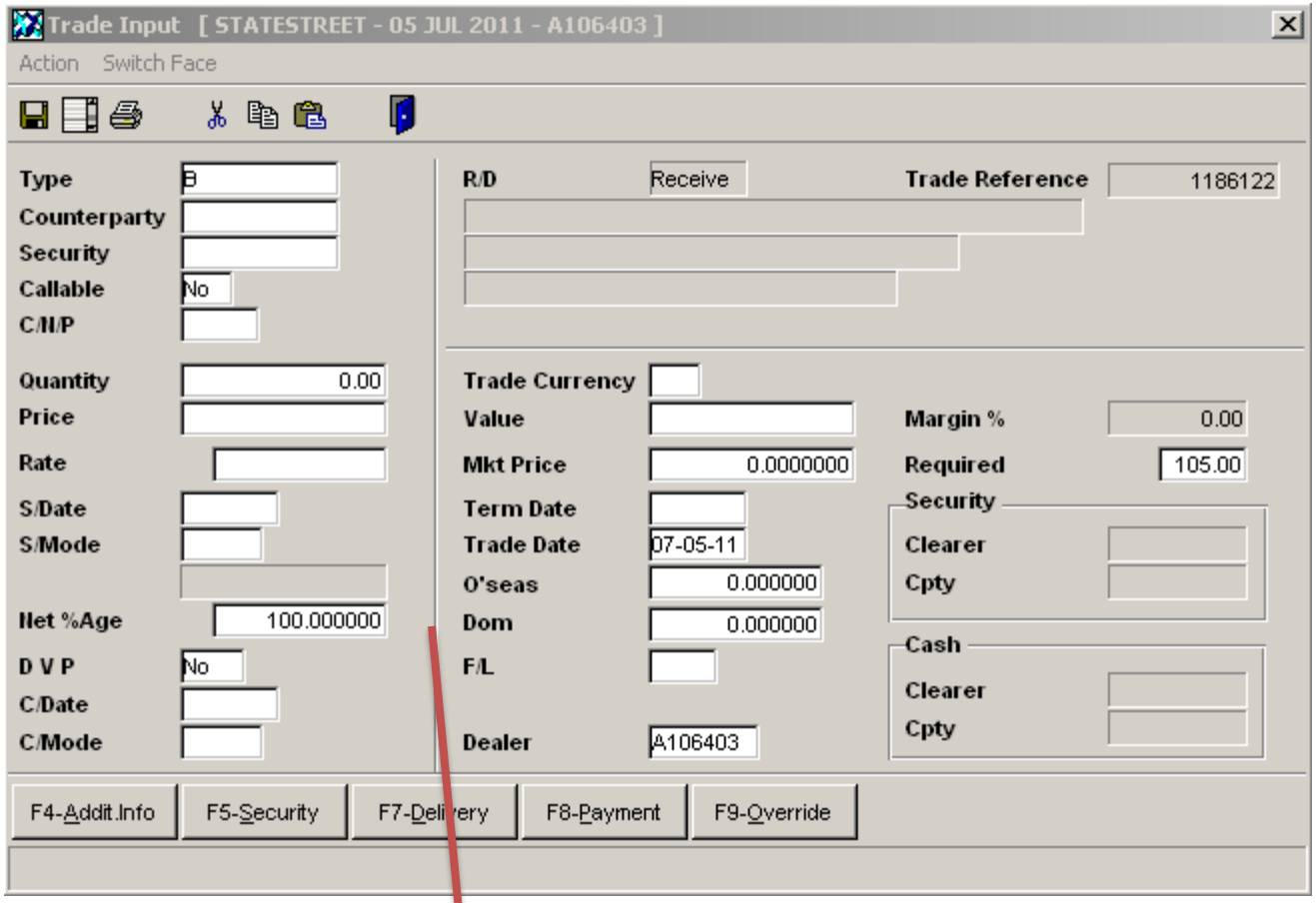



Options
General
Folders
Active Screen
Run
Windows Applications
Java
Web
Display Add-in Manager on startup
Display Start Page on startup
Check for software updates on startup
Disable recognition of virtual objects while recording
Automatically update test and component steps when you rename test objects
Automatically generate "With" statements after recording
Generate "With" statements for 2 or more objects
When pointing at a window, activate it after 5 tenths of a second
Use text recognition mechanisms in this order
First Windows API then OCR
First Windows API then OCR
First OCR then Windows API
Use Only Windows API
Use Only OCR
Restore Layout
Click to restore the pan
Generate Script...
Click to generate an a
OK
Cancel
Apply
Help

Text Method

- Object.GetVisibleText
- Object.GetTextLocation
- TextUtil.GetText
- TextUtil.GetTextLocation
- Object.ClickOnText
  - Extremely useful method! A merge between .GetTextLocation and .Click: It finds the specified text within the object, and clicks

```
''' <summary>
''' Click a control by its text
''' </summary>
''' <param name="Obj" type="Object">Parent Window control</param>
''' <param name="strText" type="String">Text of the control to be clicked</param>
''' <return></return>
''' <remarks></remarks>
Public Function ClickByText(ByVal Obj,ByVal strText)

    Dim hwnd, window_x, window_y, l, t, r, b x, y, Succeeded, dr
    l = -1
    t = -1
    r = -1
    b = -1
    hwnd = Obj.GetROProperty("HWND")
    window_x = Obj.GetROProperty("x")
    window_y = Obj.GetROProperty("Y")
    Succeeded = TextUtil.GetTextLocation( strText,hwnd,l,t,r,b)
    If Succeeded Then
        x = window_x +(l+r) / 2
        y = window_y +(t+b) / 2
        Set dr = CreateObject("Mercury.DeviceReplay")
        dr.MouseClick x,y,LEFT_MOUSE_BUTTON
    End If

End Function
```



One Custom dialog, All inner controls can’t be recognized

```
' Click "F8-Payment" button even though this button can't be recognized
Window("A106403 ]").ClickOnText("F8-Payment")
```

COM

- Component Object Model (COM) is a binary-interface standard for software component introduced by Microsoft in 1993. It is used to enable inter-process communication and dynamic object creation in a large range of programming languages.

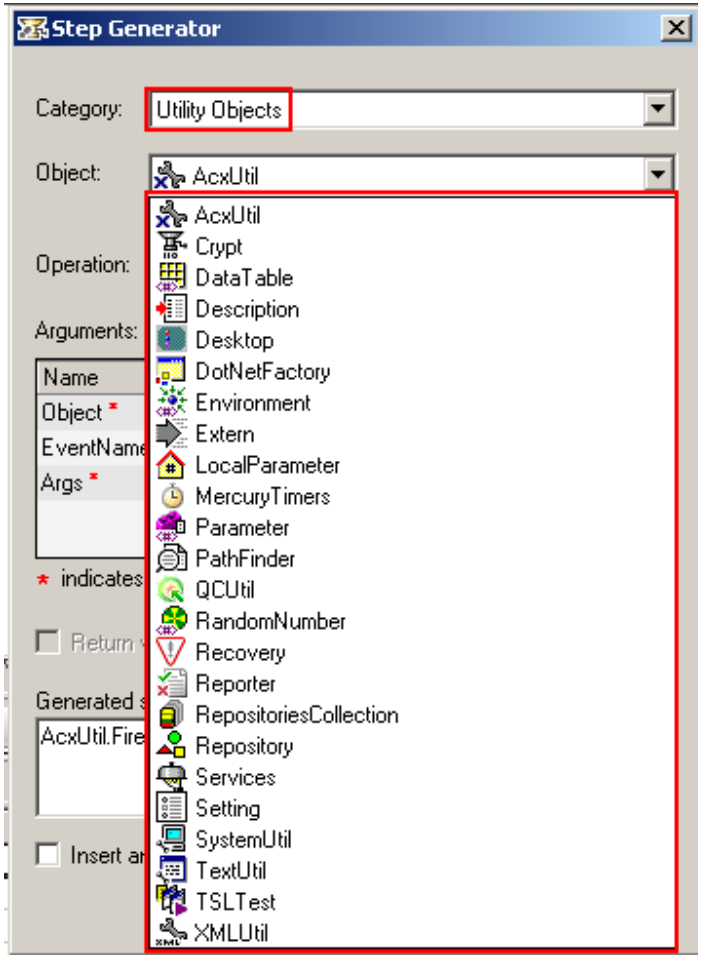
COM object

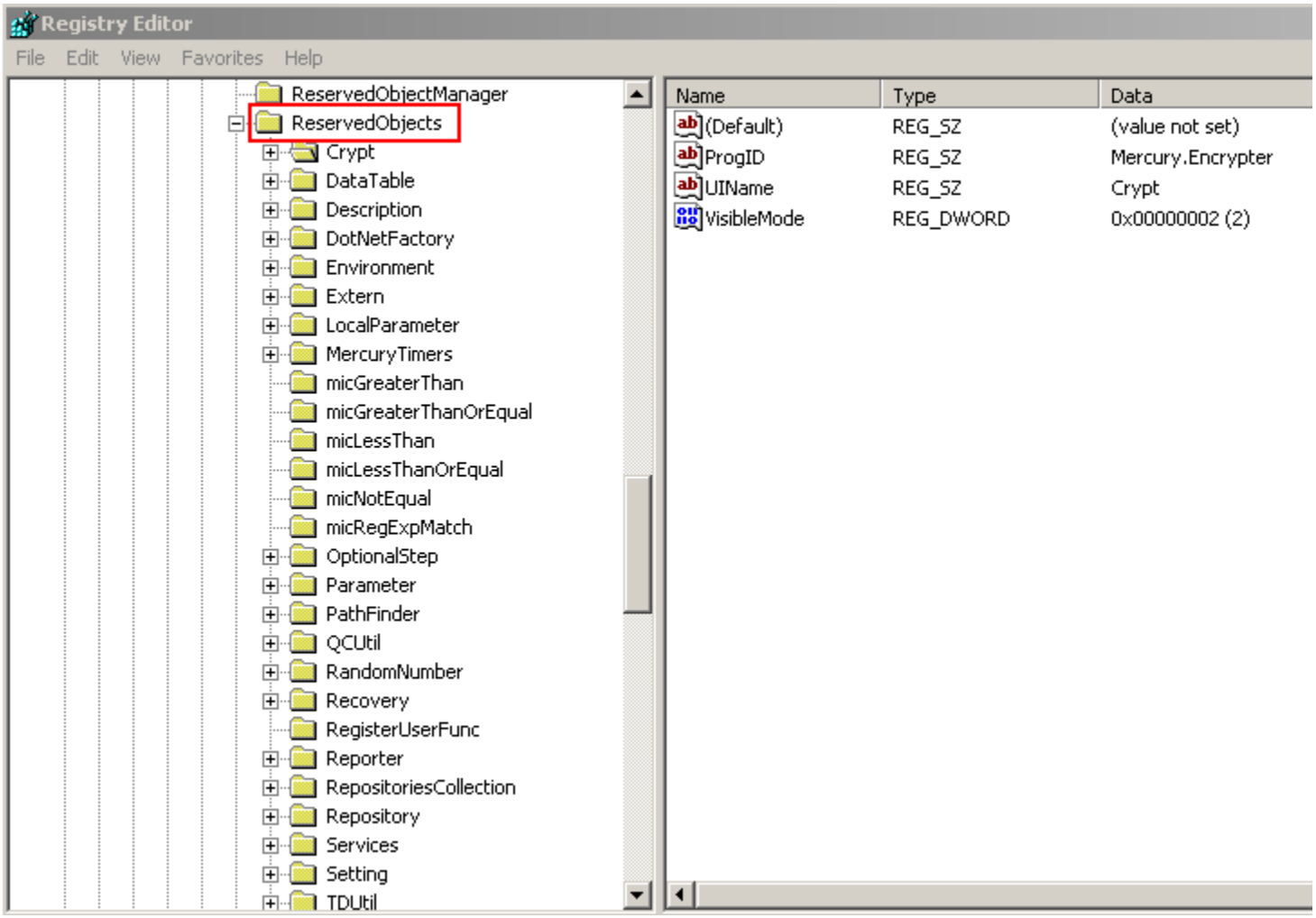
```
' Create a excel object
Set oExcel = CreateObject("Excel.Application")

' Create a word object
Set oWord = CreateObject("Word.Application")

' Create a FSO object
Set oFSO = CreateObject("Scripting.FileSystemObject")

' Create a dictionary object
Set oDict = CreateObject("Scripting.Dictionary")
```

Reserved Objects

Used to improved and simplify our scripts

- Singleton COM objects
- Created ONCE when a QTP script starts running
- Support QTP's auto-complete


Step Generator
Category:
Utility Objects
Object:
AcxUtil
Operation:
Arguments:
Name
Object *
EventName
Args *
* indicates
Return
Generated
AcxUtil.Fire
Insert ar
AcxUtil
Crypt
DataTable
Description
Desktop
DotNetFactory
Environment
Extern
LocalParameter
MercuryTimers
Parameter
PathFinder
QCUtil
RandomNumber
Recovery
Reporter
RepositoriesCollection
Repository
Services
Setting
SystemUtil
TextUtil
TSLTest
XMLUtil

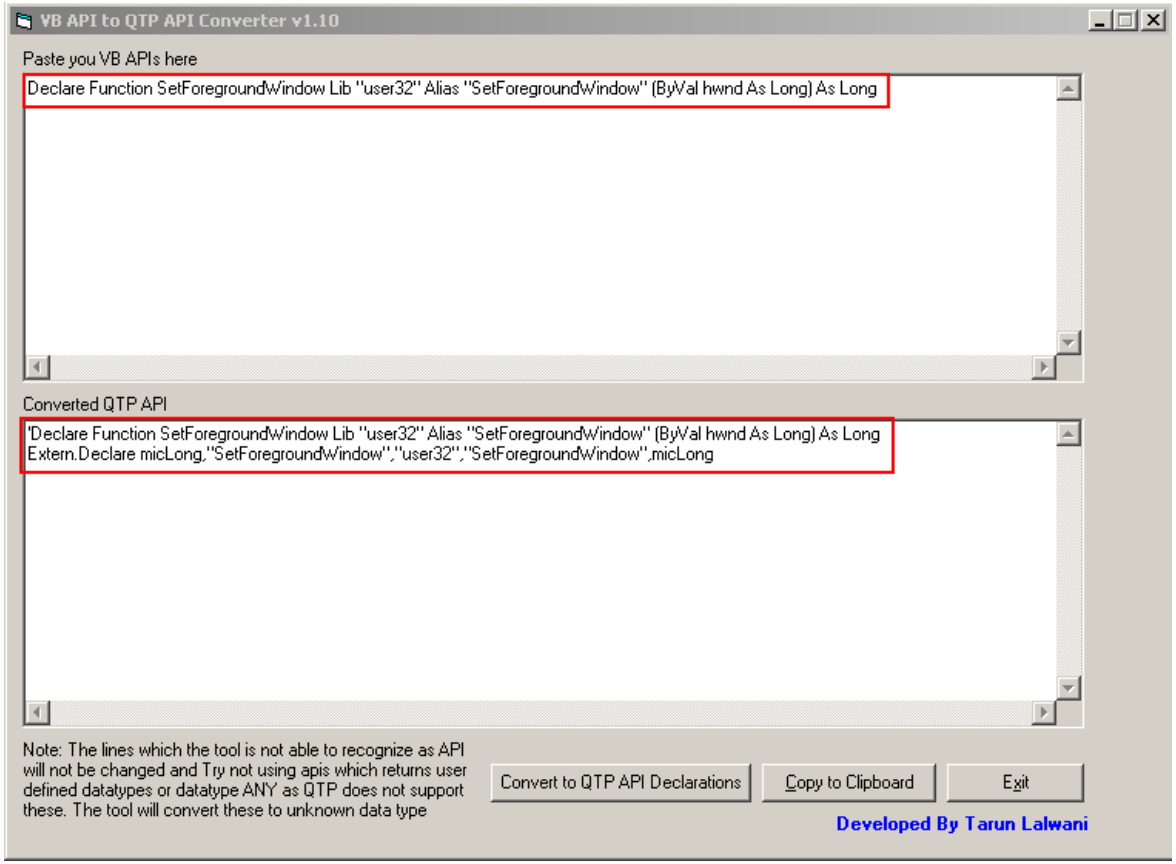

Registry Editor
File Edit View Favorites Help
ReservedObjectManager
ReservedObjects
Crypt
DataTable
Description
DotNetFactory
Environment
Extern
LocalParameter
MercuryTimers
micGreaterThan
micGreaterThanOrEqual
micLessThan
micLessThanOrEqual
micNotEqual
micRegExpMatch
OptionalStep
Parameter
PathFinder
QCUtil
RandomNumber
Recovery
RegisterUserFunc
Reporter
RepositoriesCollection
Repository
Services
Setting
TDUtil
Name
Type
Data
(Default)
REG_SZ
(value not set)
ProgID
REG_SZ
Mercury.Encrypter
UIName
REG_SZ
Crypt
VisibleMode
REG_DWORD
0x00000002 (2)

Extern

Extending the power of QTP by exposing all of the Win32 API

‘ Syntax

- Extern.Declare(RetType, MethodName, LibName, Alias [, ArgType(s)])



```
' Launch a notepad
systemutil.Run “Notepad.exe"

' Declare FindWindow method
Extern.Declare micHwnd, "FindWindow", "user32.dll", "FindWindowA", micString,
micString

' Declare SetWindowText method
Extern.Declare micLong, "SetWindowText", "user32.dll", "SetWindowTextA",
micHwnd, micString

' Get HWND of the Notepad window
hwnd = Extern.FindWindow("Notepad", vbNullString)
if hwnd = 0 then
     MsgBox "Notepad window not found"
end if

' Change the title of the notepad window
res = Extern.SetWindowText(hwnd, "kuku")
```

SystemUtil

An object used to control applications and processes during a run session.

‘ Runs a file or application

- Run

‘ Closes all processes opened by QuickTest

- CloseDescendentProcesses

‘ Closes a process according to its name

- CloseProcessByName

‘ Prevents keyboard and mouse input events

- BlockInput

‘ Re-enables keyboard and mouse input events

- UnblockInput

PathFinder

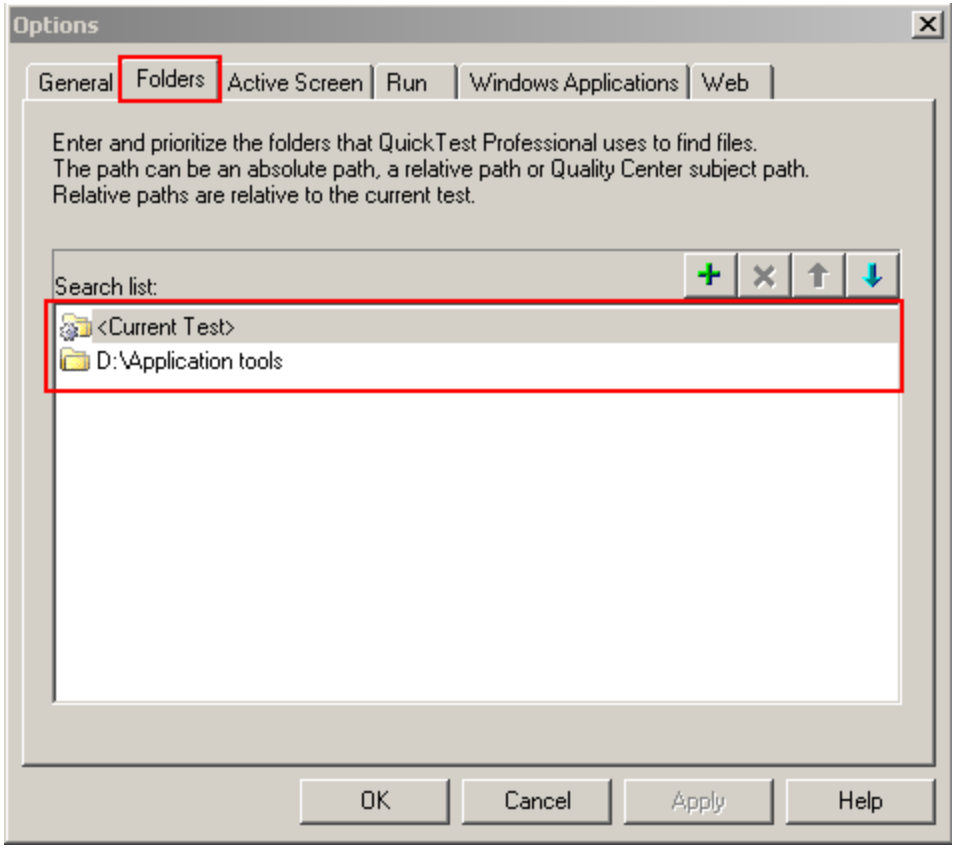
Enables you to find file paths

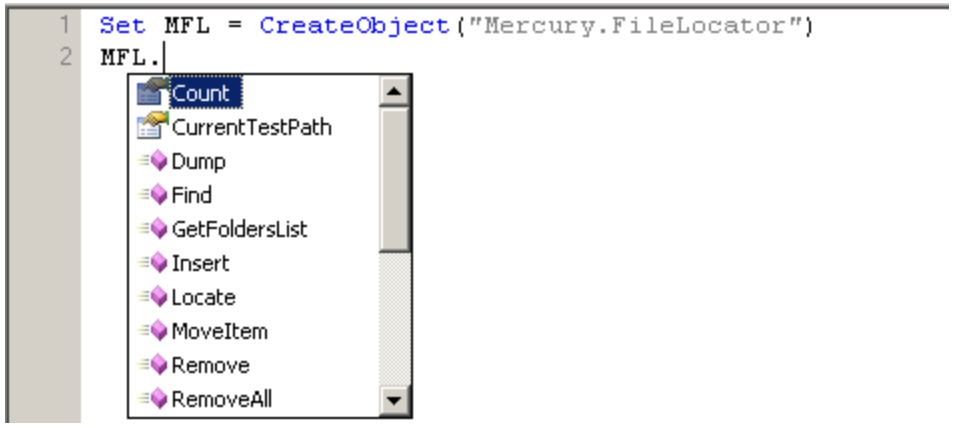
‘ Return current test path

- CurrentTestPath

‘ Returns the full file path

- Locate


Options
General
Folders
Active Screen
Run
Windows Applications
Web
Enter and prioritize the folders that QuickTest Professional uses to find files.
The path can be an absolute path, a relative path or Quality Center subject path.
Relative paths are relative to the current test.
Search list:
<Current Test>
D:\Application tools
OK
Cancel
Apply
Help


1 Set MFL = CreateObject("Mercury.FileLocator")
2 MFL.
Count
CurrentTestPath
Dump
Find
GetFoldersList
Insert
Locate
MoveItem
Remove
RemoveAll

XMLUtil

Used to access and return XML objects

‘ Creates and returns an object of type XMLData

- XMLUtil.CreateXML

‘ Creates and returns an object of type XMLData

- XMLUtil.CreateXMLFromFile(XMLFilePath)

```
XMLFileHeader = "<?xml version='1.0' encoding='UTF-8'?><?xml-stylesheet href='Report.xsl'
type='text/xsl'?><Report></Report>"
' Creates and returns an object of type XMLData
Set objXMLCustomReport = XMLUtil.CreateXML()
' Initializes an XML object using the specified XML string
objXMLCustomReport.Load XMLFileHeader
' Returns an XMLElement object, representing the block's root element.
Set objXMLroot = objXMLCustomReport.GetRootElement()
' Adds a new XMLAttribute item with the specified name and value to the element.
objXMLroot.AddAttribute "Header","Header"
' Adds a child element initialized with a tag and a value.
objXMLroot.AddChildElementByName "TestSuite", ""
' Return the first item of the root element
Set objXMLTestSuite = objXMLroot.ChildElements().Item(1)
' Adds a child element initialized with a tag and a value.
objXMLTestSuite.AddChildElementByName "TestCase", ""
' Saves the XMLData document to the specified file
objXMLCustomReport.SaveFile "c:\demo.xml"
```

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <?xml-stylesheet href='Report.xsl' type='text/xsl'?>
03 <Report Header="Header">
04   <TestSuite>
05     <TestCase>
06     </TestCase>
07   </TestSuite>
08 </Report>
```

MercuryTimer

Measures the passage of time in milliseconds

‘ Timer

- MercuryTimers(TimerName).Start
- MercuryTimers(TimerName).Stop
- MercuryTimers(TimerName).ElapsedTime

‘ VBScript Implement

- DateDiff("s",StartTime,EndTime)

DotNetFactory

Enables you to create an instance of a .NET object, and access its methods and properties

```
' Work with DateTime format
Set SystemDate = Dotnetfactory.CreateInstance("System.DateTime")
Set oDate = SystemDate.Parse("Fri, 9 Oct 2009")
FormattedDate = oDate.Day & "/" & oDate.Month & "/" & oDate.Year
' Output as "9/10/2009"
msgbox FormattedDate

' Work with File Dialog
Set fd = DotNetFactory.CreateInstance("System.Windows.Forms.OpenFileDialog",
"System.Windows.Forms")
fd.Filter="txt files (*.txt)|*.txt |All files (*.*) |*.*"
fd.ShowDialog()
msgbox fd.FileName
```

Create multi-threads using DotNetFactory

```
' This function will be invoked by a secondary thread in an asynchronous fashion
Public Function MultiThreadDemo(sParam1)
    MsgBox sParam1
End Function

Dim oFunctionRef
' Retrieve function referece
Set oFunctionRef = GetRef("MultiThreadDemo")

Dim oMultiThreadObject
' Create a instance of the Multi_Threads_Demo.MultiThread class writed by C#
' Which used to call function MultiThreadDemo though Reflection
Set oMultiThreadObject = DotNetFactory.CreateInstance("Multi_Threads_Demo.MultiThread",
"C:\Multi_Threads.dll")
oMultiThreadObject.Init(oFunctionRef)
oMultiThreadObject.Start()
wait 12
oMultiThreadObject.Stop()
wait 12
```

```
using System;

namespace Multi_Threads_Demo
{
   public class MultiThread
   {
      /// Our timer
      private System.Timers.Timer _Timer = new System.Timers.Timer(5000);

      /// The Funtion reference back to QTP
      private object _FuntionRef = null;

      public MultiThread()
      {
         //Basic configuration
         _Timer.AutoReset = true;
         _Timer.Elapsed += React_To_Tick;
      }

      /// Sends over the Funtion reference
      public void Init(object FuntionRef)
      {
         _FuntionRef = FuntionRef;
      }
```

```
        /// Fire QTP hook every time the timer ticks.
        public void React_To_Tick(object sender, System.Timers.ElapsedEventArgs e)
        {
            //Call the QTP hook

            try
            {
                //Send over the elapsed time since we've started listening
                _FuntionRef.GetType().InvokeMember("", System.Reflection.BindingFlags.InvokeMethod,
                            null, _FuntionRef, new object[] { e.SignalTime.ToShortTimeString() });
            }
            catch
            {
                //In case QTP has stopped listening to us
                _Timer.Stop();
            }
        }

        /// Start the timer and listining to its events.
        public void Start()
        {
            _Timer.Start();
        }

        /// <summary>
        /// Stop the timer
        /// </summary>
        public void Stop()
        {
            _Timer.Stop();
        }
    }
}
```

Mouse & Keyboard

- Sometimes we need to do specific action on the UI, for example a right click on an object. Also it can be useful to type symbols and letters from different languages, without installing special fonts or changing the keyboard layout, and this can be very useful for testing multi-language applications.
- For mouse operation the DragDrop method is very useful, to drag and drop items from one frame to another or between applications.
- In QTP, The deviceReplay object is used to simulate mouse clicks and movements and also keyboard input.

```
' Activate notepad and type a string.
Set deviceReplay = CreateObject("Mercury.DeviceReplay")
SystemUtil.Run "notepad.exe", "", "", "open"
Window("nativeclass:=Notepad", "index:=0").Activate micLeftBtn
deviceReplay.SendString( "DeviceReplay" )
Set deviceReplay = Nothing


' Activate the open menu of notepad using the hotkey and will close it
using Escape
Const VK_O = 24 : Const VK_F = 33
Const VK_CONTROL = 29 : Const VK_ESCAPE = 1 : Const VK_MENU = 56
Set deviceReplay = CreateObject( "Mercury.DeviceReplay" )
SystemUtil.Run "notepad.exe", "", "", "open"
Window("nativeclass:=Notepad", "index:=0").Activate micLeftBtn
Wait 1
' Opening the menu Alt + F + O
deviceReplay.PressKey VK_MENU
deviceReplay.PressKey VK_F
deviceReplay.PressKey VK_O
Wait 2
' Closing the menu
deviceReplay.PressKey VK_ESCAPE
deviceReplay.SendString "Open menu was closed."
Set deviceReplay = Nothing
```

```
''' <summary>
''' Clicks on the center of an Object using DeviceReplay
''' </summary>
''' <param name="Obj" type="Object">The object to be clicked on</param>
''' <return></return>
''' <remarks></remarks>
Function AsyncClick(ByVal Obj)

    'Check if the object exist or not
    If Not Obj.Exist Then
        'Reporter.ReportEvent micFail, "Object Does not Exist", "The object does not exist"
        Exit Function
    End If

    Dim x,y, DC

    'Get the position of the object on the screen
    x = obj.GetROProperty("abs_x")
    y = obj.GetROProperty("abs_y")

    If x < 0 or y < 0 or x = "" or y = "" Then
        'Reporter.ReportEvent micFail, "Object is not Visible", "The object is not visible "
        Exit Function
    End if

    width = obj.GetROProperty("width")
    height = obj.GetROProperty("height")

    x = x + width\2
    y = y + height\2

    Set DC = CreateObject("Mercury.DeviceReplay")

    'Click on the Middle of the button
    DC.MouseClick x,y, micLeftBtn

End Function
```

```
''' <summary>
''' Drag And Drop using DeviceReplay
''' </summary>
''' <param name="ObjFrom" type="Object">The object to be draged from</param>
''' <param name="ObjTo" type="Object">The object to be droped to</param>
''' <return></return>
''' <remarks></remarks>
Function DragAndDrop(ByVal ObjFrom, ByVal ObjTo)

    Dim DC
    Dim ObjFrom_abs_x, ObjFrom_abs_y, ObjFrom_center_x, ObjFrom_center_y
    Dim ObjFrom_width, ObjFrom_height, ObjTo_width, ObjTo_height
    Dim ObjTo_abs_x, ObjTo_abs_y, ObjTo_center_x, ObjTo_center_y
    'Get the position of the object on the screen
    ObjFrom_abs_x = ObjFrom.GetROProperty("abs_x")
    ObjFrom_abs_y = ObjFrom.GetROProperty("abs_y")
    ObjTo_abs_x = ObjTo.GetROProperty("abs_x")
    ObjTo_abs_y = ObjTo.GetROProperty("abs_y")

    ObjFrom_width = ObjFrom.GetROProperty("width")
    ObjFrom_height = ObjFrom.GetROProperty("height")
    ObjTo_width = ObjTo.GetROProperty("width")
    ObjTo_height = ObjTo.GetROProperty("height")

    ObjFrom_center_x = ObjFrom_abs_x + ObjFrom_width\2
    ObjFrom_center_y = ObjFrom_abs_y + ObjFrom_height\2
    ObjTo_center_x = ObjTo_abs_x + ObjTo_width\2
    ObjTo_center_y = ObjTo_abs_y + ObjTo_height\2

    Set DC = CreateObject("Mercury.DeviceReplay")

    'Drag from the center of the ObjFrom and drop to center of the ObjTo
    DC.DragAndDrop ObjFrom_center_x,ObjFrom_center_y,ObjTo_center_x,ObjTo_center_y,micLeftBtn

End Function
```

System.Windows.Forms.Control Class - retrieve the current mouse (cursor) position in the screen according to the limitation of the DeviceReplay object.

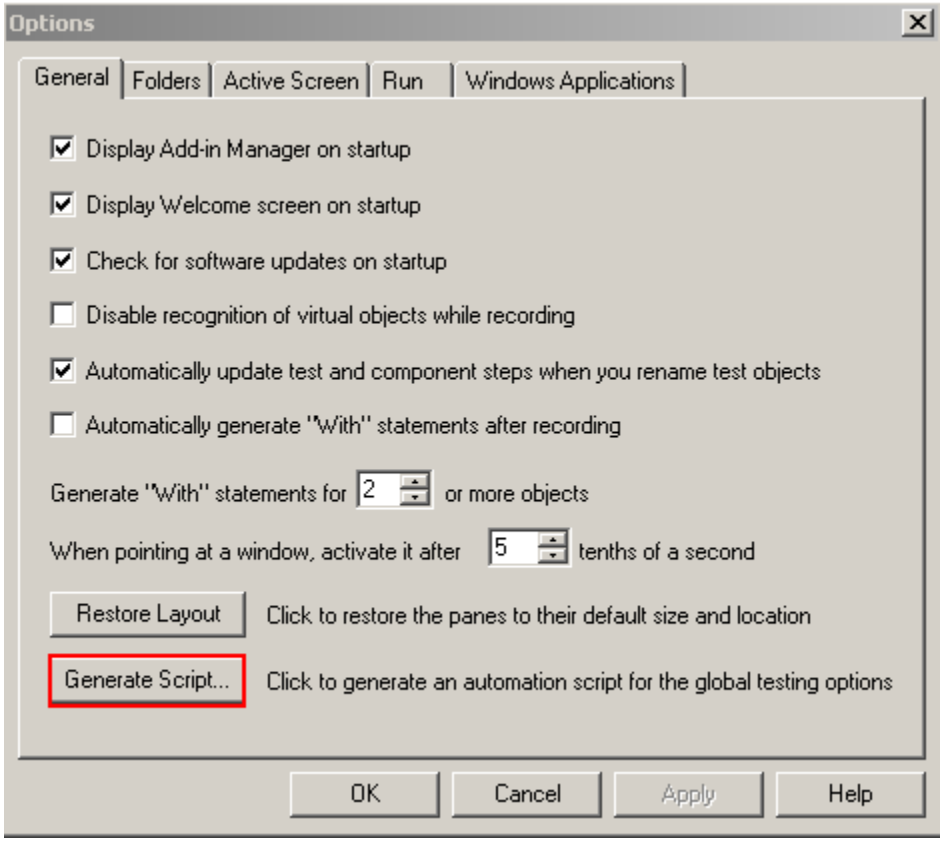
```
' where is my mouse
Set ctlr = DotNetFactory.CreateInstance("System.Windows.Forms.Control")
For i = 1 To 10
    Wait 1
    msgbox "1. X=" & ctlr.MousePosition.X & "; Y=" & ctlr.MousePosition.Y
Next
```

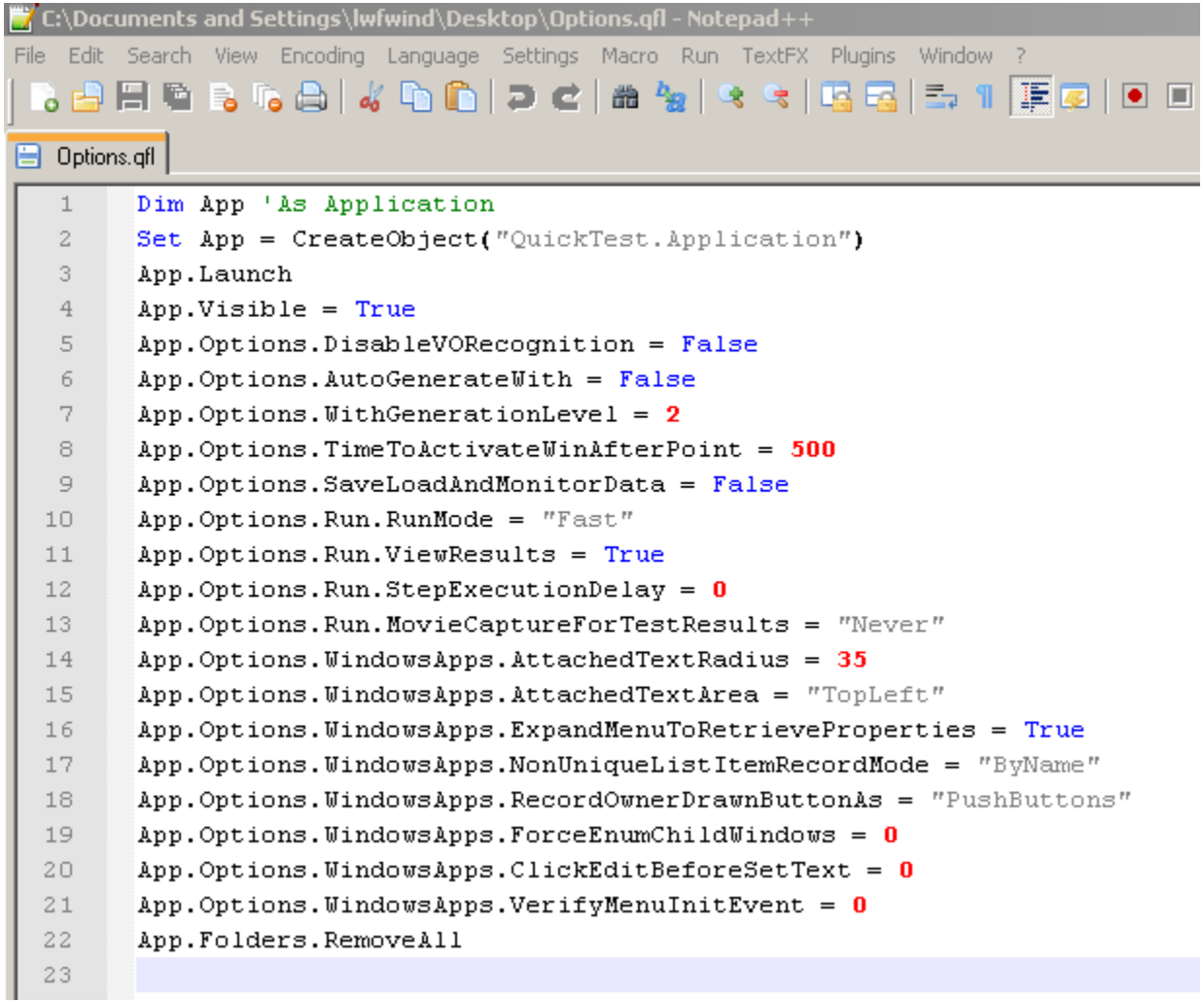
Microsoft.VisualBasic.Devices.Keyboard Class - to determine if a control key is already pressed according to the limitation of the DeviceReplay object.

```
' what is the state of the control key
 Set Keyboard = DotNetFactory.CreateInstance(
"Microsoft.VisualBasic.Devices.Keyboard", "Microsoft.VisualBasic" )
msgbox CBool(Keyboard.AltKeyDown)
msgbox CBool(Keyboard.CapsLock)
msgbox CBool(Keyboard.CtrlKeyDown)
msgbox CBool(Keyboard.NumLock)
msgbox CBool(Keyboard.ScrollLock)
msgbox CBool(Keyboard.ShiftKeyDown)
```

Automation Object Model

- Using the objects, methods, and properties exposed by the QuickTest automation object model, you can write scripts that configure QuickTest options and run tests or components instead of performing these operations manually using the QuickTest interface.
- Generating automation script
  I. Tools -> Options -> General Tab -> Generate Script
  II. File -> Settings -> General Tab -> Generate Script
  III. Tools -> Object Identification -> Generate Script


Options
General
Folders
Active Screen
Run
Windows Applications
Display Add-in Manager on startup
Display Welcome screen on startup
Check for software updates on startup
Disable recognition of virtual objects while recording
Automatically update test and component steps when you rename test objects
Automatically generate "With" statements after recording
Generate "With" statements for 2 or more objects
When pointing at a window, activate it after 5 tenths of a second
Restore Layout
Click to restore the panes to their default size and location
Generate Script...
Click to generate an automation script for the global testing options
OK
Cancel
Apply
Help


C:\Documents and Settings\lwfwind\Desktop\Options.qfl - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
Options.qfl
1 Dim App 'As Application
2 Set App = CreateObject("QuickTest.Application")
3 App.Launch
4 App.Visible = True
5 App.Options.DisableVORecognition = False
6 App.Options.AutoGenerateWith = False
7 App.Options.WithGenerationLevel = 2
8 App.Options.TimeToActivateWinAfterPoint = 500
9 App.Options.SaveLoadAndMonitorData = False
10 App.Options.Run.RunMode = "Fast"
11 App.Options.Run.ViewResults = True
12 App.Options.Run.StepExecutionDelay = 0
13 App.Options.Run.MovieCaptureForTestResults = "Never"
14 App.Options.WindowsApps.AttachedTextRadius = 35
15 App.Options.WindowsApps.AttachedTextArea = "TopLeft"
16 App.Options.WindowsApps.ExpandMenuToRetrieveProperties = True
17 App.Options.WindowsApps.NonUniqueListItemRecordMode = "ByName"
18 App.Options.WindowsApps.RecordOwnerDrawnButtonAs = "PushButtons"
19 App.Options.WindowsApps.ForceEnumChildWindows = 0
20 App.Options.WindowsApps.ClickEditBeforeSetText = 0
21 App.Options.WindowsApps.VerifyMenuInitEvent = 0
22 App.Folders.RemoveAll
23

```
''' <summary>
''' Auto Launch QTP, open an existing test and Run the Test
''' </summary>
''' <param name="sQTPProjetLocation" type="string">The location of the QTP Test</param>
''' <remarks></remarks>
Public Function AutoRunQTP(ByVal sQTPProjetLocation)

    'Create the QTP Application object
    Set qtApp = CreateObject("QuickTest.Application")
    'Make the QuickTest application visible
    qtApp.Visible = True
    'Set QuickTest run options
    'Instruct QuickTest to perform next step when error occurs
    qtApp.Options.Run.ImageCaptureForTestResults = "OnError"
    qtApp.Options.Run.RunMode = "Fast"
    qtApp.Options.Run.ViewResults = true
    'Open the test in read-only mode
    qtApp.Open sQTPProjetLocation, True
    'set run settings for the test
    Set qtTest = qtApp.Test
    'Instruct QuickTest to perform next step when error occurs
    qtTest.Settings.Run.OnError = "NextStep"
    'Run the test
    qtTest.Run
    'Close the test
    qtTest.Close
    'Close QTP
    qtApp.quit
    'Release Object
    Set qtTest = Nothing

End Function
```

Schedule AOM

```
    ''' <summary>
    ''' Add specific task using windows inner Scheduled Task
    ''' </summary>
    ''' <param name="sTaskName" type="string">Specifies a name for the task</param>
    ''' <param name="sStartTime" type="string">Specifies the time of day that the task starts in HH:MM:SS 24-hour format</param>
    ''' <param name="sSchedule" type="string">Specifies the schedule type. Valid values are MINUTE, HOURLY, DAILY, WEEKLY, MONTHLY, ONCE, ONSTART, ONLOGON, ONIDLE</param>
    Public Function AddTask(ByVal sTaskName, ByVal sStartTime, ByVal sSchedule)

        Dim WshShell, sScriptLocation, AddParemeters
        Set WshShell = CreateObject("WScript.Shell")

        sScriptLocation = "\" & Chr(34) & WshShell.CurrentDirectory & "\AutoRun.vbs" & "\" & Chr(34)
        AddParemeters = "/create /ru system /tn " & Chr(34) & sTaskName & Chr(34) & " /tr " & Chr(34) & sScriptLocation & Chr(34) &  " /st " & sStartTime & " /sc " & sSchedule
        WshShell.Run "schtasks.exe " & AddParemeters

    End Function
```

Error Handling

Error Handling, is when you know what errors can occur and you want to handle it

- Data Validation
  I. VarType - IsNumeric, IsDate, IsArray, IsObject, IsNull, IsEmpty
- Error Preventing
  I. Every If…Then..End If statement has the Else part, the same for Select Case. Use Case Else
- Synchronization
  I. Object.Exist
  II. Object.WaitProperty
  III. Custom timeout-exit mechanism

Data Validation and Error Preventing

```
‘ Only do the division if x or y are valid numbers
If IsNumeric(x) = true OR IsNumeric(y) = true then
    result = x/y
Else
    MsgBox "Error: Either x or y is not numeric"
End If
```

Synchronization

```
‘    Checks whether the object currently exists in the open application
If  Dialog("Login").Exist(10000)  Then
      Dialog("Login").WinEdit("Agent  Name:").Set  "dani"
      Dialog("Login").WinEdit("Password:").Set  "Mercury"
      Dialog("Login").WinButton("OK").Click
Else
      Reporter.ReportEvent  micFail,  "Sync  timeout",  "Dialog  'Login'  is  not
available."
End  If
With  Window("Flight  Reservation")
      .WinButton("Insert  Order").Click
      '  Synchronization  point  "Insert  Done…".
      .ActiveX("Threed  Panel  Control").WaitProperty  "text",  "Insert  Done...",  10000
      orderNum  =  .WinEdit("Order  No:").GetROProperty("text")
      MsgBox  orderNum,  vbInformation,  "Order  Number"
End  With
```

Custom timeout-exit mechanism

```
''' <summary>
''' Check that the current GUI context is loaded by specific time
''' </summary>
''' <param name="oDictGUIObjects" type="Dictionary">ChildObjects Dictionary</param>
''' <param name="Interval" type="int">Check once every specific Interval second</param>
''' <param name="TimeOut" type="int">Max timeout</param>
''' <return>True/False</return>
''' <remarks></remarks>
Public Function IsLoaded(ByVal oDictGUIObjects, ByVal Interval, ByVal TimeOut)

    Dim Starting, Ending, t
    Starting = Now
    Ending = DateAdd("s",TimeOut,Starting)
    IsLoaded = False
    Do
        t = DateDiff("s",Now,Ending)
        If IsContextLoaded(oDictGUIObjects) Then
            IsLoaded = true
            Exit Do
        End If
        wait Interval
    Loop Until t <= 0

End function
```

Exception Handling

Exception handling is for situations when there are errors that occur that you have not catered for

- On Error Resume Next
- Recovery Scenarios

On Error Resume Next

```
‘ A divide by 0 will be detected
If IsNumeric(x) = true OR IsNumeric(y) = true then
      On Error Resume Next
      result = x/y
      if Err.Number <> 0 then
            MsgBox "Exception: " & Err.Message
      End If
Else
      MsgBox "Error: Either x or y is not numeric"
End If
```

Design Pattern

Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem. Which make the derived code more parsimonious, scalable, and reusable, as well as more easily maintainable.

- Command Wrapper Pattern
- Singleton Pattern
- Factory Pattern

Command Wrapper Pattern

Implements a function as a class

- The code blocks embodied in class can be dynamically loaded, a feature that can be of great value in systems that are poor in resources. This is especially true when the to-be called function is rarely used. After creating the object, it is possible to execute its method according to need.

```
Public Function Sum(ByVal arrNumbers)
    Dim ix
    If (Not IsArray(arrNumbers)) Then 'Not an array, so nothing to do – exit function
        'Add your error handling code here
        Exit Function
    End If
    Sum = 0
    For ix = LBound(arrNumbers) To UBound(arrNumbers)
        If (IsNumeric(arrNumbers(ix))) Then
            Sum = Sum + arrNumbers(ix)
        Else
            'Add your error handling code here
        End If
    Next
End Function

'Test the function
MsgBox Sum(Array(23, 56, 78, 95, 114)), vbOKOnly, "Result" 'Display result returned by the Sum function
```

```
Class Sum

    Private m_arrVarNumbers
    Private m_varResult

    Public Property Get Numbers()
        Numbers = m_arrVarNumbers
    End Property
    Private Property Let Numbers(ByVal arrVarNumbers)
        m_arrVarNumbers = arrVarNumbers
    End Property
    Public Property Get Result()
        Result = m_varResult
    End Property
    Private Property Let Result(ByVal varResult)
        m_varResult = varResult
    End Property

    Private Sub Class_Initialize()
        'Initialize the Numbers member as an empty array
        ReDim m_arrVarNumbers(-1)
    End Sub

    Public Function Init(ByVal arrVarNumbers)
        Numbers = arrVarNumbers
    End Function
```

```
    Public Default Function Exec()
        Dim ix, arrNumbers
        If (Not IsArray(Numbers)) Then 'Not an array, so nothing to do – exit function
            'Add your error handling code here
            Exec = "Invalid data type was supplied to perform the operation."
            Exit Function
            If (UBound(arrNumbers) - LBound(arrNumbers) + 1 <= 1) Then
                'Array is empty or has single item - Add your error handling code here
                Exec = "Not enough data was supplied to perform the operation."
                Exit Function
            End If
        Else
            arrNumbers = Numbers
        End If
        Result = 0
        For ix = LBound(arrNumbers) To UBound(arrNumbers)
            If (IsNumeric(arrNumbers(ix))) Then
                Result = Result + arrNumbers(ix)
            Else
                'Add your error handling code here
            End If
        Next
        Exec = Result
    End Function

End Class
'This function behaves as a constructor and returns an initialized instance of the class
Public Function GetSum(ByVal arrNumbers)
    Set GetSum = New Sum
    GetSum.Init(arrNumbers)
End Function
```

Singleton Pattern

Ensure a class only has one instance, and provide a global point of access to it

- Just imagine the time wasted by creating an Excel COM object every time you report an event, or creating multiple database connection to the application's backbone. These objects might even lock the relevant resources, thereby failing the script in unpredictable ways.

```
' Define a global singleton excel instance
Public oExcel
Class Excel

    Private oExcelObject

    Private Sub Class_Initialize

        Dim bAlreadyInit
        bAlreadyInit = False
        If IsObject(oExcel) = True Then
            ' The object was once initialized
            If Not oExcel Is Nothing Then
                ' The object has not been destroyed
                bAlreadyInit = True
            End If
        End If

        ' Only create new object if needed
        If bAlreadyInit = False Then Set oExcel = CreateObject("Excel.Application")

        ' Set the local class excel reference to the global Singleton object
        Set oExcelObject = oExcel

    End Sub

    Private Sub Class_Terminate

        oExcelObject.Quit
        Set oExcelObject = Nothing

    End Sub
End Class
```

Factory Pattern

Define an interface for creating an object, but let the subclasses decide which class to instantiate. The Factory method lets a class defer instantiation to subclasses

- Easy to implement and maintain, and provide a central access point for multiple object creation

```
Function  ObjectFactory(sObjectName)
    Select  Case  sObjectName
        Case  "Save  Button"
            Set  ObjectFactory  =  VBWindow("vbname:=X").VBButton("text:=Save")
        Case  "Main  Window"
            Set  ObjectFactory  =  VBWindow("vbname:=X")
        Case  Else
            Set  ObjectFactory  =  Nothing
    End  Select
End  Function

ObjectFactory("Save  Button").Click
```

One major drawback – they don't remember the objects they've created. This is exactly the problem that the Singleton Pattern was set to solve, but it was built to manage the resources of only a single object.

```
'Provide a global access point
Public oOutputs
Set oOutputs = New clsOutputFactory
Class clsOutputFactory
    Public Channels 'Stores the output channel
    Private Sub Class_Initialize
        Set Me.Channels = CreateObject("Scripting.Dictionary")
    End Sub
    Private Sub Class_Terminate
        Set Me.Channels = Nothing
    End Sub
    Public Sub Construct(sChannelName, sChannelType)
        Dim oNewChannel
        If Me.Channels.Exist(sChannelName) Then Exit Sub
        Select Case sChannelType
            Case "Excel"
                'Create a new excel Singleton  class
                Set oNewChannel = New ExcelChannelClass
            Case "Text"
                'Text init code here
            Case "DB"
                'DB init code here
        End Select
        'Add the new channel to the storage
        Me.Channels.Add sChannelName, oNewChannel
    End Sub
End Class

oOutputs.Construct "Errors", "Excel"
oOutputs.Construct "Backup", "Text"
```

```
'Provide a global access point
Public oOutputs
Set oOutputs = New clsOutputFactory
Class clsOutputFactory
    Public Channels 'Stores the output channel
    Private Sub Class_Initialize
        Set Me.Channels = CreateObject("Scripting.Dictionary")
    End Sub
    Private Sub Class_Terminate
        Set Me.Channels = Nothing
    End Sub
    Public Sub Construct(sChannelName, sChannelType)
        Dim oNewChannel
        If Me.Channels.Exist(sChannelName) Then Exit Sub
        On Error Resume Next
        'Set oNewChannel = New Excel
        Execute "Set oNewChannel = New " & sChannelType
        If Err.Number <> 0 Then
            Reporter.ReportEvent micFail, " New Channel", "Failed to " &
                _ "create instance : " & sType
        Else
            'Add the new channel to the storage
            Me.Channels.Add sChannelName, oNewChannel
        End If
        On Error Goto 0
    End Sub
End Class

oOutputs.Construct "Errors", "Excel"
oOutputs.Construct "Backup", "Text"
```

Resource

' A Test Development Resource for HP QuickTest Professional
http://relevantcodes.com/

' Advanced QTP
http://www.advancedqtp.com/

' Linkedin for HP Mercury QTP
http://www.linkedin.com/groups/HP-Mercury-QTP-1697337?mostPopular=&gid=1697337

' SQA Forums for QTP
http://www.sqaforums.com/postlist.php?Cat=0&Board=UBB20&page=0

' Pragmatistic Testing
http://blog.csdn.net/Testing_is_believing

' iQuickTest Studio
http://www.iquicktest.com/

' software inquisition
http://www.softwareinquisition.com/

' Tarun Lalwani
http://knowledgeinbox.com/

' Work with QTP
http://rajivkumarnandvani.wordpress.com/