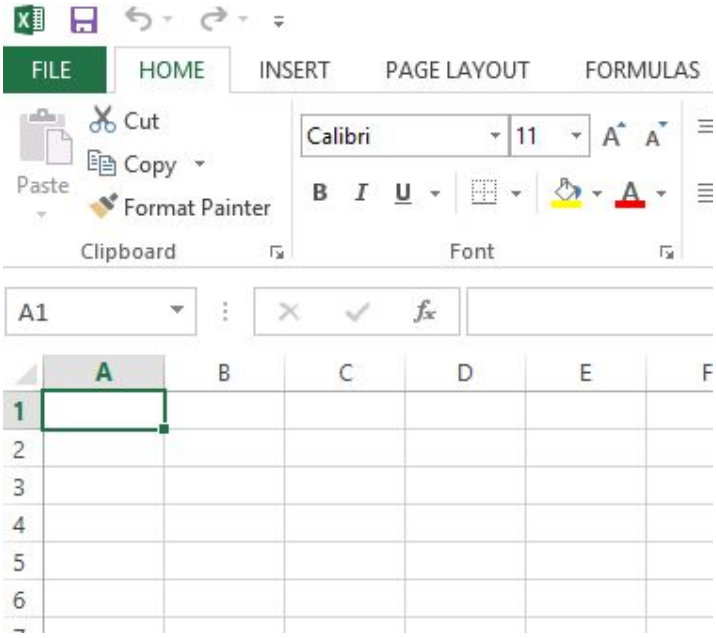
Chapter -12

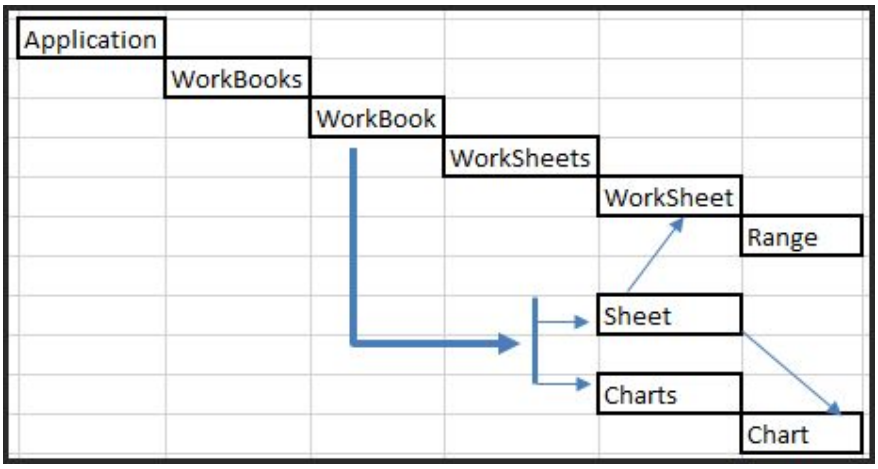
UFT – EXCEL



“Take the first step in faith. You don't have to see the whole staircase, just take the first step.” - Martin Luther King, Jr. quotes

1.0 In this chapter we will discuss the UFT and Excel sheet

UFT can work well with Microsoft excel. Microsoft has developed the object model [REF] for various application including Excel application. We can access and modify excel by VBScript using excel object model. We can access Microsoft Office Excel with the objects provided by the Excel object model.



The Application object represents the entire excel application, and each Workbook object contains a collection of Worksheet objects. Worksheet is the main object that contains cells and Range. There are many objects [REF] and features supported by MS Excel object model

UFT 11.52 support .xls and .xlsx excel format.UFT used to support only .xls prior to this version.

Exercise- Understanding Excel API

STEP1: Create excel object

Open Notepad | write following code | Save file as "ExcelProgramming.vbs"

```
Dim objExcel
Set objExcel = Createobject("Excel.Application")
objExcel.Application.Visible = true
objExcel.Workbooks.Add
wait 1
objExcel.ActiveWorkbook.SaveAs  "C:\Test\KrishanExcel.xls"
objExcel.Application.Quit
Set objExcel=nothing
```

STEP2: Open script "ExcelProgramming" | Copy the code | navigate to UFT | File | New | Test | Paste the code | Run

Concept of Exercise- Understanding Excel API

In STEP1, we are using excel api. API is application programming Interface. In nutshell, it provides the method and object that can be used to access the features of the application. Various application including Databases, hardware provide sophisticated API. Most of the application also provide GUI to

intract with application. Broadly API is a library which comprises of classes and support many methods and object to manage, control and manipulate the application.

Line-1: We have declare the variable objExcel. It will set in line-2. Objects always use “Set” for initialization. So the following line will return the object and can assign to the variable at left by using “Set” keyword.

```
Set objMyBrowser = Broswer(“TechnoPreacher”)
Set objMyPage = Broswer(“TechnoPreacher”).Page(“QuickTestPro”)
Set objMyLink = Broswer(“TechnoPreacher”).Page(“QuickTestPro”).Link(“Home”)
```

Line-2: Here we are creating the excel application object. It is a parent of all the object to access subsequent objects. We can create more than one object so it is not like that if we created one excel. Application object then that will only point to access other excel objects. VBScript uses CreateObject to create object of COM API (Component Object Model) or OLE automation. Basically CreateObject is a starting point to interact with any com application. Every COM application object model is different from other and we should find the relevant information in developer notes of the application or directly from the product providers company.

Line-3: Excel.Application.Visible = True or False will set the visible property accordingly. If we choose false then it will not mean that excel object stop working, It will just work in silent mode. Yes, It is little unusual to think that everything is happening in excel without seeing it. But you can go in task manager and find the EXCEL.exe process there.

Line-4: Until line-3, we were using the root excel application object. Here we are associating a workbooks with application. Please revisit object model diagram once again. Workbooks is a collection object and can contains many Workbook. Object Workbooks support various method including Add, Open, Close etc. Here “Add” method creates a new workbook and the new workbook becomes the active workbook. Workbooks can contains many workbook in excel.

Line-5: This is hardcoded waiting time.

Line6: ActiveWorkbook method returns the active WorkBook. We are calling “Save as” method to save the .xls file.

Line7: The Quit method quits Microsoft Excel. If unsaved workbooks are open, Excel will display a alert by asking whether we want to save the changes. We can prevent this by saving all workbooks before using the **Quit** method or by setting the DisplayAlerts property to **False**.

Line8: Line 7 and Line 8 looks trivial but they are very important. This line release the memory. VBScript takes care memory management for ordinary datatype but these external applications need explicit de-allocation of the memory. Setting any object to “Nothing” will deallocate memory and can be claim by VBScript garbage collector for other objects.

Try your Self: You can try to run Line-1 to Line-6 in for-loop with 1 million iteration and verify the system crash due to memory mismanagement.

In STEP2, we are running same code from UFT. It says that programming of external application worked well in UFT or .vbs file as it is pure VBScript code. This code is independent to any application or dependency.

Exercise- Working with Excel and UFT

STEP1: Write into already exist excel file

Open UFT | File | New | Test | "WriteInExcel" | write following code (make sure previous section created KrishanExcel.xls already) | Run

```
1   Set objExcel = createobject("Excel.Application")
2   objExcel.Workbooks.Open "C:\Test\KrishanExcel.xls"
3   objExcel.Application.Visible = true
4   set objSheet1 = objExcel.ActiveWorkbook.Worksheets("Sheet1")
5   objSheet1.cells(1,1).value ="Krish"
6   objSheet1.cells(1,2).value ="Aric"
7   objSheet1.cells(1,3).value ="Aadhya"
8   For i = 2 To 10
9       objSheet1.cells(i,1).value ="Krish" & i
10      objSheet1.cells(i,2).value ="Aric" & i
11      objSheet1.cells(i,3).value ="Aadhya" & i
12  Next
13  objExcel.ActiveWorkbook.Save
14  objExcel.Application.Quit
15  Set objExcel=nothing
```

STEP2: Run the test

Navigate to "WriteInExcel" test in UFT | Run

Concept of Exercise- Working with Excel and UFT

In STEP1, we are writing values in excel file. Line-1 to Line-3 discussed in previous section. Worksheets is the important area where user want to store data. Worksheet reside inside workbook. objExcel.ActiveWorkbook returns active WorkBook and workbook.WorkSheet(<Sheet Name>) returns the worksheet object.
Line5: Worksheet support cells method that enable us to set or get any value from specific excel cell.

Try your Self: You can try to manipulate other sheets and write logic to write data in them. You can also try to get data from cell in MsgBox.

Exercise- Working with UsedRange in Excel

STEP1: Write into already exist excel file

Open UFT | File | New | Test | "RangeInExcel" | write following code | Go to "C:\Test" | Verify KrishanExcel.xls (otherwise create the file and wite few rows in the excel) | Go to UFT | Run

```
1   Set objSheet1 = createobject("Excel.Application")
2   objSheet1.Workbooks.Open "C:\Test\KrishanExcel.xls"
3   objSheet1.Application.Visible = true
4   set mysheet = objSheet1.ActiveWorkbook.Worksheets("Sheet1")
5   Row=mysheet.UsedRange.Rows.Count
6   Col=mysheet.UsedRange.columns.count
7   For  i= 1 to Row
8       For j=1 to Col
9           Print  mysheet.cells(i,j).value
10      Next
11  Next
12  objSheet1.ActiveWorkbook.Close
13  objSheet1.Application.Quit
14  Set mysheet =nothing
15  Set objSheet1 = nothing
```

	A	B	C
1	Krish	Aric	Aadhya
2	Krish2	Aric2	Aadhya2
3	Krish3	Aric3	Aadhya3
4	Krish4	Aric4	Aadhya4
5	Krish5	Aric5	Aadhya5
6	Krish6	Aric6	Aadhya6

Concept of Exercise- Working with UsedRange in Excel
In STEP1, we are reading the cell values. Excel contains variable data and we may find it difficult to know the exact rows and column of the data in excel sheet. “UsedRange” takes care this problem and can return rows number and column numbers for the used cells. We can then iterate rows and column and imply our logic to find the appropriate information from cell.

Exercise- Working with Range in Excel

STEP1: Write into already exist excel file

Open UFT | File | New | Test | “RangeInExcel” | write following code | Run

```
Set objExcel = CreateObject("Excel.Application")
objExcel.Visible = True
objExcel.Workbooks.Add

objExcel.Cells(1, 1).Value = "Krish"
objExcel.Cells(1, 1).Font.Bold = TRUE
objExcel.Cells(1, 1).Interior.ColorIndex = 30
objExcel.Cells(2, 1).Value = "SecondRow_FirstCol"
objExcel.Cells(2, 2).Value = "SecondRow_SecondCol"

Set objRange = objExcel.Range("A1","B3")
objRange.Font.Size = 12
objRange.Font.Size = 25
```

Concept of Exercise- Working with Range in Excel
In STEP1, we explored one of the most important aspect of excel object model, Range and cell modification. Range is Row X Column so A1 to B3 means first 3 rows with two columns (A and B). Cells support various formatting properties e.g Interior.ColorIndex and Font etc.

Benefit of Excel Programming by VBScript

Traditionally tester used excel sheet to write test cases and user inputs. Excel is easy to demonstrate tabular data and conceptual representation of specific step. The concept of data table in UFT is borrowed from the same reason. Datatable is versatile but we need to open UFT test and manipulate data inside UFT. Working with Excel resolve this issue and different resource can create test data in excel sheet while the automater can just connect excel sheets and run the test suite. So,effectively , business analyst or subject matter analyst can create test with data in excel and automater can run the tests based on that information. Excel is not the only data source. We can design our data source with text files, Database, dictionary, Word, datatable, xml files and many other ways.

You can find more sample code here UFT installation folder>\CodeSamplesPlus\UsingExcel.vbs usually UFT installation folder can be found here C:\Program Files (x86)\HP\Unified Functional Testing

<Revise figure of weight lifting>

Let's do coding of common Excel programs

Exercise-FSO Check if Excel file exist otherwise create new excel

```
fileCheckorCreate = "C:\Test\KrishShukla.xlsx"
Set fso = CreateObject("Scripting.FileSystemObject")
Set objExcel = CreateObject("Excel.Application")

isFileExist = fso.FileExists(fileCheckorCreate)

   If isFileExist Then
      Set objWorkBook = objExcel.Workbooks.Open(fileCheckorCreate)
      ' User defined logic
      objWorkBook.Save
   Else
      Set objWorkBook = objExcel.Workbooks.Add
      'User defined logic
      objWorkBook.SaveAs fileCheckorCreate
   End If

objWorkBook.Close
objExcel.Quit
Set objExcel = Nothing
```

Exercise- Search text in Excel

```
fileCheckorCreate = "C:\Test\KrishShukla.xlsx"
Set fso = CreateObject("Scripting.FileSystemObject")
Set objExcel = CreateObject("Excel.Application")
searchString = InputBox("Please provide searc criteria")
isFileExist = fso.FileExists(fileCheckorCreate)
If isFileExist Then
objExcel.Visible = True
Set objWorkBook = objExcel.Workbooks.Open(fileCheckorCreate)
Set objWorkSheet = objExcel.ActiveWorkbook.Worksheets("Sheet1")
Set isTextExist =  objWorkSheet.UsedRange.Find (searchString)
If isTextExist is nothing Then
    MsgBox "String Not Found"
Else
    MsgBox "String Found in Row Number: " & isTextExist.Row
End If
objWorkBook.Save
Else
Set objWorkBook = objExcel.Workbooks.Add
objWorkBook.SaveAs fileCheckorCreate
End If
objWorkBook.Close
objExcel.Quit
Set objExcel = Nothing
```

	A	B	C
1	Country	City	Name
2	USA	bentonville	Richa
3	USA	Bella Vista	Aric
4	UK	Watford	Ruby
5	India	PrathviPur	**Ashish**
6	India	Jhansi	Meena
7	India	Jhansi	Mahavir
8	India	Gurha	O MAA

<Revise: Delete this code , Image is fine as above>

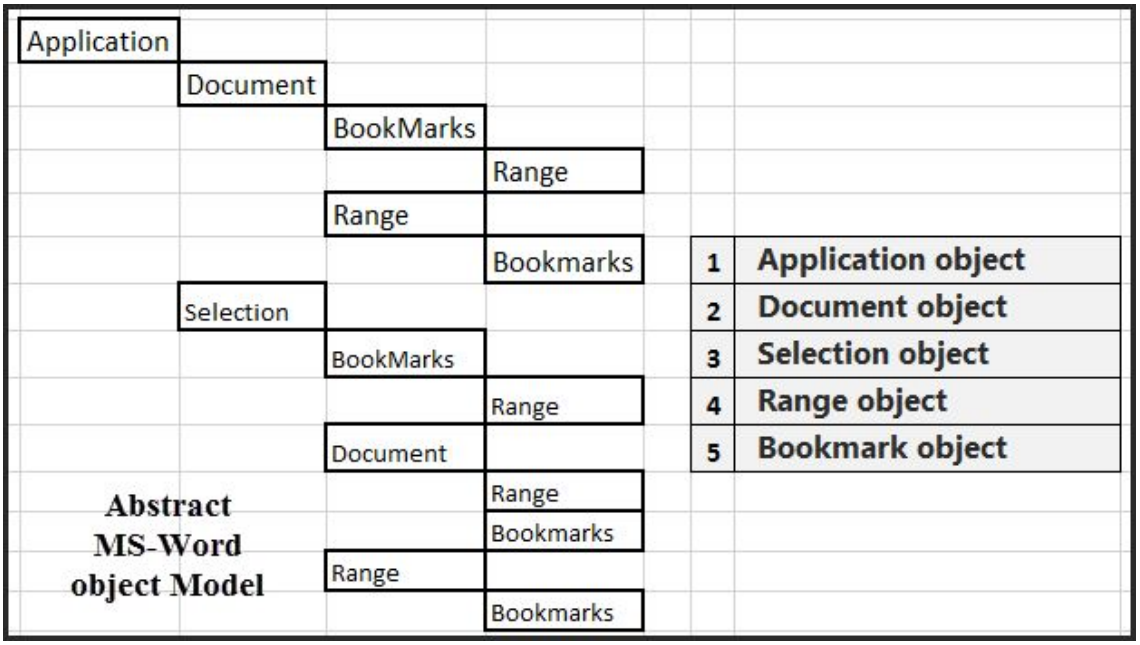
```
fileCheckorCreate = "C:\Test\KrishShukla.xlsx"
Set fso = CreateObject("Scripting.FileSystemObject")
Set objExcel = CreateObject("Excel.Application")
searchString = InputBox("Please provide searc criteria")
isFileExist = fso.FileExists(fileCheckorCreate)
  If isFileExist Then
    objExcel.Visible = True
    Set objWorkBook = objExcel.Workbooks.Open(fileCheckorCreate)
    Set objWorkSheet = objExcel.ActiveWorkbook.Worksheets("Sheet1")
    Set isTextExist =  objWorkSheet.UsedRange.Find (searchString)


              If isTextExist is nothing Then
                MsgBox "String Not Found"
              Else
                MsgBox "String Found in Row Number: " & isTextExist.Row
              End If
            objWorkBook.Save
          Else
            Set objWorkBook = objExcel.Workbooks.Add
            objWorkBook.SaveAs fileCheckorCreate
          End If
        objWorkBook.Close
```

```
objExcel.Quit
Set objExcel = Nothing
```

REF: Excel Object Model Overview http://msdn.microsoft.com/en-us/library/wss56bz7(v=vs.90).aspx
REG: Excel Object Model objects http://msdn.microsoft.com/en-us/library/bb149081(v=office.12)

MS Word and UFT

In this section we will explore the MS Word object model programming [REF] and the usage in UFT. Application object is the top most "Parent" object in word application. The supported method and properties of application object apply for whole word application. Document is a central point for word automation. Application use activedocument property of document object to enable manipulation of word document. Selection is the area of document that currently selected while range object represent continuous area of document and there may be multiple ranges in a selection.



Let's explore the word object model programming. You can run the whole program in UFT.

1. Create a Word Application object

   ```
   Set objWord = CreateObject("Word.Application")
   ```

2. Use application object property visible

```
objWord.Visible = True
```

3. To create a new document by using Document object

```
Set objMSDoc = objWord.Documents.Add()
```

4. Write a line into word document

```
Set oWordSelection = objWord.Selection
oWordSelection.TypeText "Welcome to Automation"
```

To write a text, we need to make the selection by using Selection object. We are using TypeText method to write into word document. There are many methods and property supported by Selection object [REF]

5. Write a table in range

```
Set objRange = oWordSelection.Range
objMSDoc.Tables.Add objRange, 1, 3
Set objTable = objMSDoc.Tables(1)
```

We are creating range and table in that range.

6. Write cell data in table

```
objTable.Cell(1, 1).Range.Text = "TechnoPreacher"
objTable.Cell(1, 2).Range.Text = "QuickTestPro."
```

In this step, we are inserting text into table.

7. Save the document

```
oMSDoc.saveAs "C:\Test\Krishan.doc"
```

8. Close the application , Quit and release memory

```
objMSDoc.Close
objMSDoc.Quit
Set oWordSelection=Nothing
Set objMSDoc =Nothing
Set objMSWord=Nothing
```

In this step, we need to be careful in the order. We cannot release memory before closing it. Also we cannot release parent object before child object.

Exercise – Find the word in Word

STEP1: Find the content of the Word document

 *Go to "C:\Test" |Create Krish.doc file | Write the following text in it*

"Techopreacher advocate the technical testing in all kind of testing. It is very important to enable testing with tools and utilities e.g scripting, UFT, Loadrunner, SOAPUI, Jmeter and so on … There are many plug in and in-built utility to make testing effective."

STEP2: Write a code to get the content of the word document

Go to "UFT" |File | New | Test |"GetContent" | Write the following code | Run

```
Set oMSWord = CreateObject("Word.Application")
oMSWord.Visible = false
Set objKrishDoc = oMSWord.Documents.Open("C:\Test\Krishan.doc")
print objKrishDoc.Content
MsgBox objKrishDoc.Content
objKrishDoc.Close
Set oMSWord = Nothing
```

Concept of Exercise – Find the word in Word

In STEP1, We created a doc file. In STEP2, we opened the word document. Method "Content" returns the whole content from the document. Now we can write logic to find the specific value, count the word occurrence or any other logic as per testing need for example following code will count the word in the document.

```
strContent = objKrishDoc.Content
Set objRegExp = New RegExp
objRegExp.global = true
objRegExp.pattern = "\w"    ' Any word in the string
Set Matches = objRegExp.execute(strContent)
MsgBox Matches.count
```

REF: Word Object Model http://msdn.microsoft.com/en-us/library/kw65a0we(v=vs.90).aspx

REF: Selection Method http://msdn.microsoft.com/en-us/library/microsoft.office.interop.word.selection_methods(v=office.14).aspx

MS Outlook and UFT

We can manipulate all the data stored in the MS Outlook by using outlook object model [REF]. Outlook object model provides various object [REF] to control many features of MS Outlook user interface e.g. calendar, managing task and contacts etc. Story of Outlook automation is similar to other office application automation. We first need to find Manu (version of Adam) i.e. Create Outlook top most parent Object.

If we are not specifically testing the outlook itself then we may need minimal automation need of outlook. The most common task is to send email with attachment for example after finishing the "nightly-run" or encountering any exception. We should have outlook on the machine where code is running.

Exercise – Send Email by Outlook

STEP1: Create mail item in outlook object and send email

Go to "C:\Test" | Create a pdf "Krish.pdf" or copy any pdf and rename it | Go to UFT |File | New | Test | "SendMailByOutlook" |Write following code | Run

```
SendMail "contact@quicktestpro.co.uk","Hi","UFTSTAR,"C:\Test\Krish.pdf"
'Remember it is sub call so cannot use parenthesis

        Function SendMail(SendTo, Subject, Body, Attachment)
                Set ol=CreateObject("Outlook.Application")
                Set Mail=ol.CreateItem(0)
                Mail.to=SendTo
                Mail.Subject=Subject
                Mail.Body=Body
                If (Attachment <> "") Then
                        Mail.Attachments.Add(Attachment)
                End If
                Mail.Send
                ol.Quit
                Set Mail = Nothing
                Set ol = Nothing
        End Function
```

Concept of Exercise – Send Email by Outlook

In STEP 1, we have created Outlook object. Outlook object support CreateItem that creates and returns a new Microsoft Outlook item. The mail item can be supplied with receiver email id , subject, body and attachment.

Other way to send email

We can send email by other means as well. VBSript can automate COM component. Collaboration Data Objects for Windows NT Server (CDONTS) is a component in Windows (NT and 200x) based. CDONTS enables creating and sending e-mail messages from application (majorly web-application) if SMTP server installed locally. It is now deprecated in favour of Windows Server 2003 but you may still get lot of encounter of CDONTS code with QTP during your job.

```
SendMail "contact@quicktestpro.co.uk","Hi","UFTSTAR,"C:\Test\Krish.pdf"
```

```
Function SendMail(SendFrom, SendTo, Subject, Body)
  Set oMail=CreateObject("CDONTS.Newmail")
  oMail.From = SendFrom
  oMail.To = SendTo
  oMail.Subject = Subject
  oMail.Body = Body
  oMail.Send
  Set oMail = Nothing
End Function
```

Exercise – Access Outlook to find information from it.

STEP1: Open Outlook and find the reminder information

Go to UFT |File | New | Test | "AccessOutlook" |Write following code | Run

```
'Open Outlook if it is not already opened
Set objOutlookApplication = CreateObject("Outlook.Application")
Set objOutlookNameSpace = objOutlookApplication.GetNamespace("MAPI")
Msgbox objOutlookApplication.Version
Msgbox objOutlookApplication.DefaultProfileName

For folderIndex=3  to 6
'Get Default Outlook folders : Inbox, Sent, Trash etc
Set objFolder =  objOutlookNameSpace.GetDefaultFolder( folderIndex )
objFolder.Display
Next
Set objCurrentReminders = objOutlookApplication.Reminders
Msgbox  objCurrentReminders .Count
```

Concept of Exercise – Access Outlook to find information from it.

In STEP1, we are finding the information about and from outlook. Our code use Messaging API i.e. MAPI. It provides the messaging architecture for MS Outlook. Code or Applications can use MAPI to manipulate email data, to create email messages and the access folders data. We are displaying default folders of outlook.

REF: Outlook Object Model http://msdn.microsoft.com/en-us/library/office/aa221870(v=office.11).aspx

REF: Outlook Developer reference http://msdn.microsoft.com/en-us/library/office/ff866465.aspx

A Note on Microsoft Office Object Model

Aadhya: Aric, It seems we gone too far from our UFT world. Now I am feeling there are too many application and too much code from everywhere.
Aric: Microsoft create various applications and to publish object model to manage and manipulate automatically by object model. This concept known as OLE automation or just automation. So whenever you hear the word automation just don't restrict yourself with product automation. Automation can be used for task automation, process automation, and component automation and so on.
Aadhya: Ok. Where can I find the object model documentation?
Aric: The object models for the Office applications are documented in a Language Reference.
Aadhya: But I wish I could have some sample code
Aric: The Microsoft Knowledge Base is good place for finding automation code samples written in Visual Basic, Visual C++ and MFC.
Aadhya: But how do I know which classes, methods, and properties to use?
Aric: The help file is the best place to get the information about it. Application also provide Macro editor where we can record and replay application in VBA. That gives fair idea about the flow, object, methods and properties.

Aadhya: Do I need to know in depth about these application.
Aric: The more you know the better it is. It is essential to know the basic idea of OLE automation and then you can build your own logic by other references including help files and online research.
Aadhya: What is OLE
Aric: OLE automation allows programmer to expose feature of application in terms of “object”. So, these object can be manipulated by various languages including Visual Basic languages (VB, VBA, VBS etc.). OLE stands for Object Linking and Embedding. So it is like linking various application object and embedding it in code to manipulate application programmatically. The automation controller is the "client" and the application exporting the automation objects is known as "server". So do not confuse when you encounter servername in CreateObject specification.

CreateObject(<servername.typename>[,location])

Aadhya: Which micorosft office applications provide object model. I am aware with word, excel and outlook only.
Aric: Following is the list of applications. You may find more information in MSDN (Online help files) library

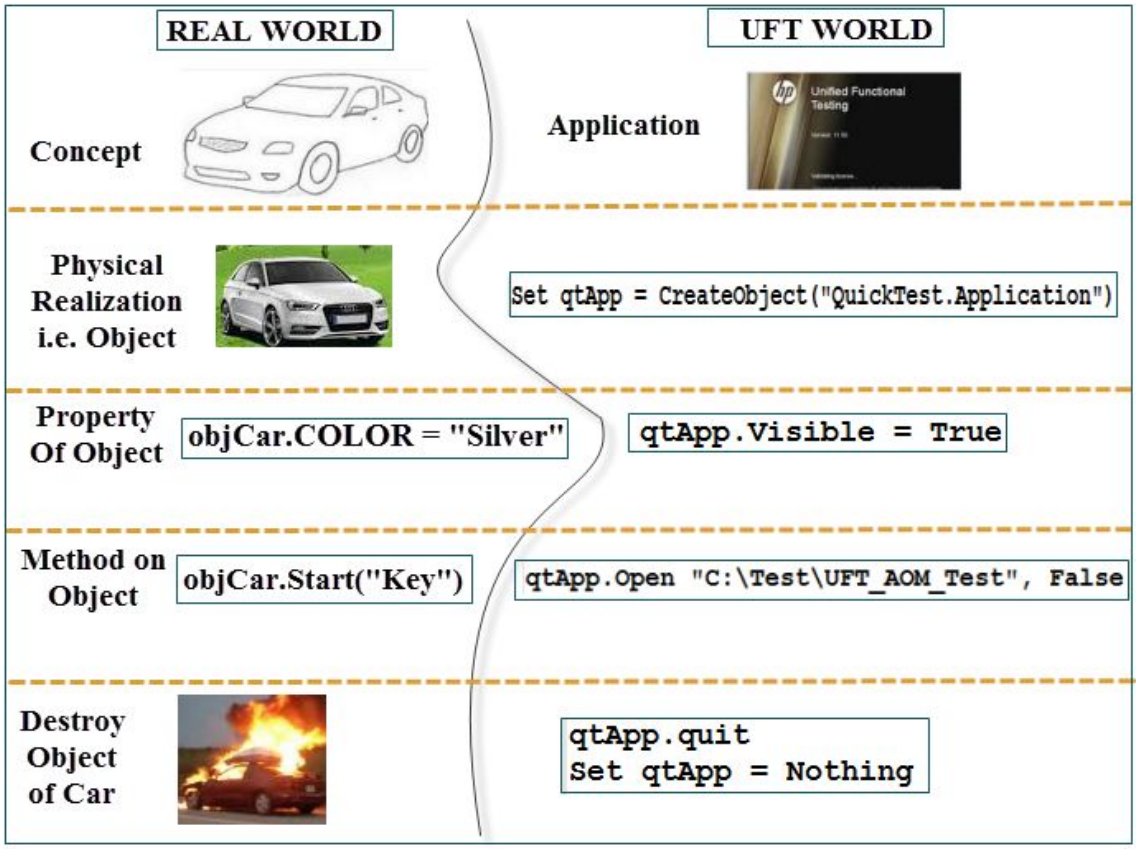
1	Microsoft Access Objects
2	Microsoft Excel Objects
3	Microsoft FrontPage Objects
	Create, deploy, modify, and manage Web sites using Microsoft® FrontPage®.
4	Microsoft Outlook Objects
	Create custom Microsoft® Outlook® objects and manipulate those objects from within Outlook or from another application using VBA code from within Outlook or another Microsoft® Office XP application by using Automation.
5	Microsoft PowerPoint Objects
6	Microsoft Project Objects
	Build powerful custom applications easily with the Microsoft® Project object model.
7	Microsoft Word Objects
8	Microsoft Visio Objects
	Design, model, and manage complex enterprise-level systems with the sophisticated tool set provided by Microsoft® Visio® products.


Go to UFT |File | New | Test | Create new GUITest as “UFT_AOM_Test” | Create | Record | Select “Record and run test on any browser” | OK | Press F5 key | Click on IE | “google.co.uk” in URL | Search with “UFT” | Enter | Click on Image Link | Close browser | Stop | Save | Modify Test by giving 5 second wait after Step entering “UFT” in script | File | Save Test As | Navigate to “C:\Test” | Save with “UFT_AOM_Test” name | Run

Create UFT Object and run the Test

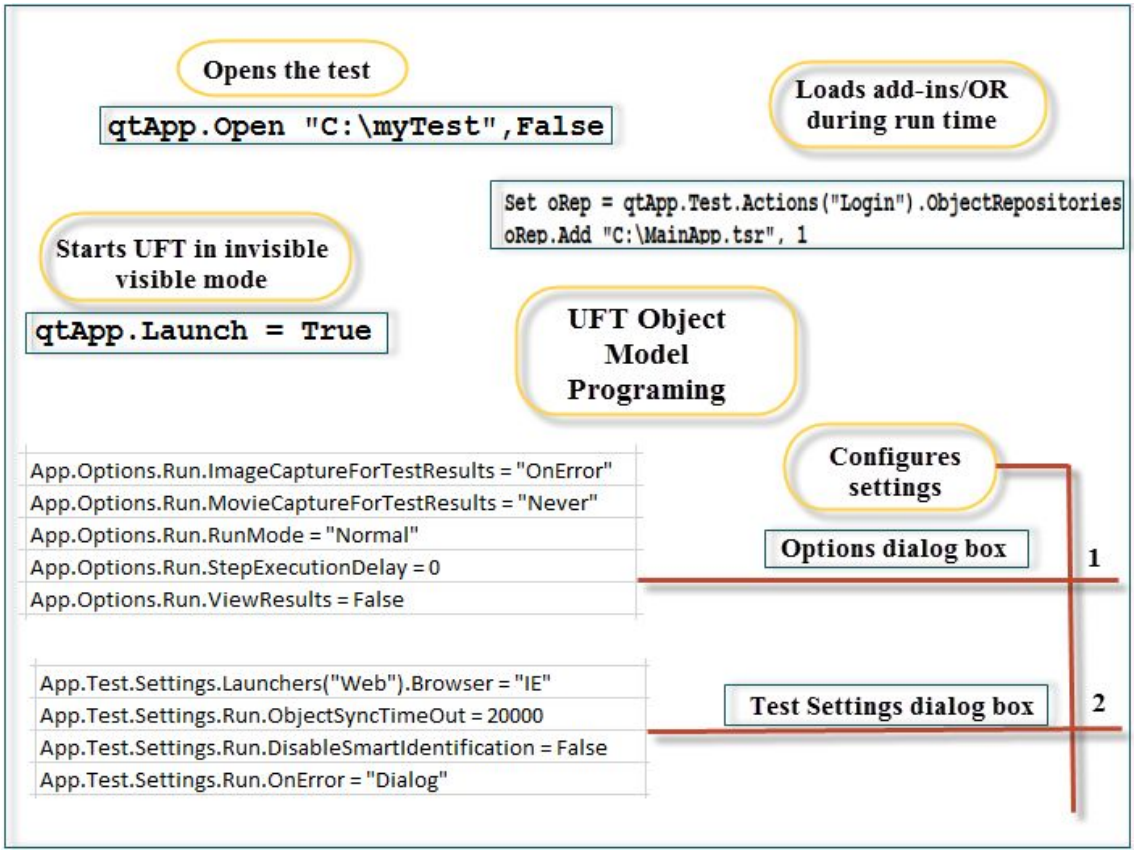
Go to "C:\Test" |Create text file | insert code as given below | Save as Run_UFT_BY_AOM.VBS | Close | Double click on "Run_UFT_BY_AOM.VBS"

```
Dim qtApp
Dim qtTest

'Create the QTP Application object
Set qtApp = CreateObject("QuickTest.Application")
qtApp.Launch
qtApp.Visible = True
qtApp.Open "C:\Test\UFT_AOM_Test", False

'set run settings for the test
Set qtTest = qtApp.Test
qtTest.Run
qtTest.Close
qtApp.quit

'Release Object
Set qtTest = Nothing
Set qtApp = Nothing
```



Few Example of what we can do by UFT Object Model

Example of What can be automate by Object Model



Exercise some example same as HP help file

Open Test and set the Setting properties

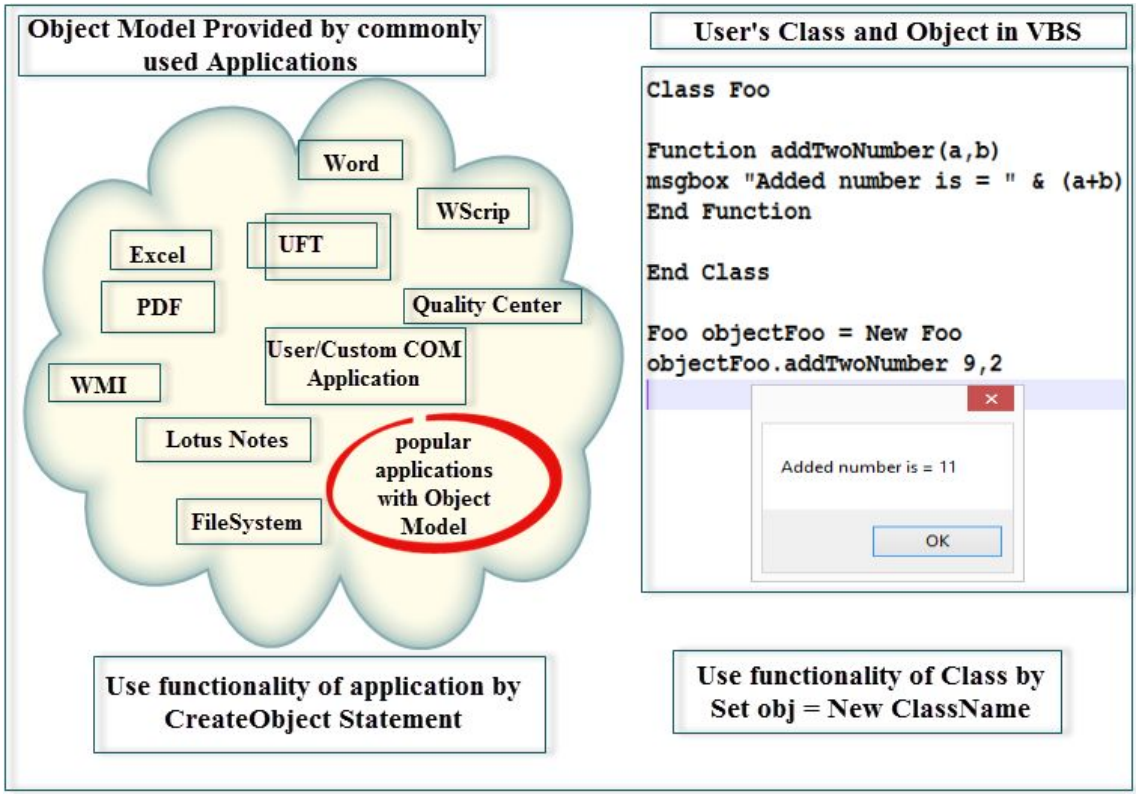
```
'*************************************************************************************
****************************************
'Description:
'
'This example opens a test, configures run options and settings,
'runs the test, and then checks the results of the test run.
'
'Assumptions:
'There is no unsaved test currently open in UFT.
'For more information, see the example for the Test.SaveAs method.
'When UFT opens, it loads the add-ins required for the test.
'For more information, see the example for the Test.GetAssociatedAddins method.
'*************************************************************************************
****************************************

Dim qtApp 'As QuickTest.Application ' Declare the Application object variable
Dim qtTest 'As QuickTest.Test ' Declare a Test object variable
```

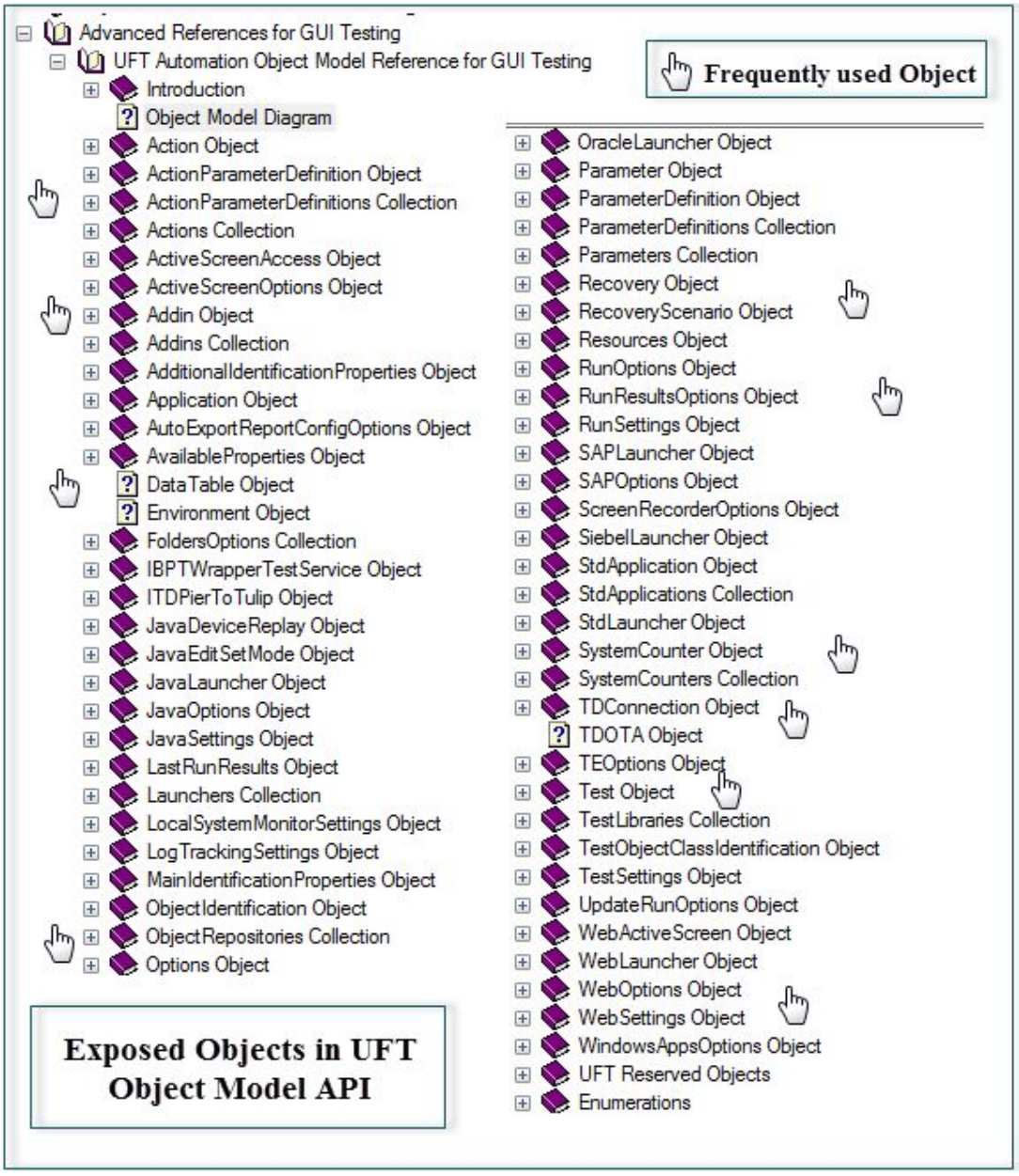
```
Dim qtResultsOpt 'As QuickTest.RunResultsOptions ' Declare a Run Results Options object variable
Dim qtAutoExportResultsOpts 'As QuickTest.AutoExportReportConfigOptions ' Declare the
Automatically Export Report Configuration Options object variable

Set qtApp = CreateObject("QuickTest.Application") ' Create the Application object
qtApp.Launch ' Start UFT
qtApp.Visible = True ' Make the UFT application visible

' Set UFT run options
qtApp.Options.Run.ImageCaptureForTestResults = "OnError"

qtApp.Options.Run.RunMode = "Fast"
qtApp.Options.Run.ViewResults = False

qtApp.Open "C:\Tests\Test1", True ' Open the test in read-only mode

' set run settings for the test
Set qtTest = qtApp.Test
qtTest.Settings.Run.IterationMode = "rngIterations" ' Run only iterations 2 to 4
qtTest.Settings.Run.StartIteration = 2
qtTest.Settings.Run.EndIteration = 4
qtTest.Settings.Run.OnError = "NextStep" ' Instruct UFT to perform next step when error occurs

Set qtResultsOpt = CreateObject("QuickTest.RunResultsOptions") ' Create the Run Results Options
object
qtResultsOpt.ResultsLocation = "C:\Tests\Test1\Res1" ' Set the results location

‘Set options for automatic export of run results at the end of every run session

Set qtAutoExportResultsOpts = qtApp.Options.Run.AutoExportReportConfig
qtAutoExportResultsOpts.AutoExportResults = True ' Instruct UFT to automatically export the run
results at the end of each run session
qtAutoExportResultsOpts.StepDetailsReport = True ' Instruct UFT to automatically export the step
details part of the run results at the end of each run session
qtAutoExportResultsOpts.DataTableReport = True ' Instruct UFT to automatically export the data
table part of the run results at the end of each run session
qtAutoExportResultsOpts.LogTrackingReport = True ' Instruct UFT to automatically export the log
tracking part of the run results at the end of each run session
qtAutoExportResultsOpts.ScreenRecorderReport = True ' Instruct UFT to automatically export the
screen recorder part of the run results at the end of each run session
qtAutoExportResultsOpts.SystemMonitorReport = False ' Instruct UFT not to automatically export
the system monitor part of the run results at the end of each run session
qtAutoExportResultsOpts.ExportLocation = "C:\Documents and Settings\All Users\Desktop" 'Instruct
UFT to automatically export the run results to the Desktop at the end of each run session
qtAutoExportResultsOpts.UserDefinedXSL = "C:\Documents and Settings\All
Users\Desktop\MyCustXSL.xsl" ' Specify the customized XSL file when exporting the run results data
qtAutoExportResultsOpts.StepDetailsReportFormat = "UserDefined" ' Instruct UFT to use a
customized XSL file when exporting the run results data
qtAutoExportResultsOpts.ExportForFailedRunsOnly = True ' Instruct UFT to automatically export run
results only for failed runs
```

```
qtTest.Run qtResultsOpt ' Run the test

MsgBox qtTest.LastRunResults.Status ' Check the results of the test run
qtTest.Close ' Close the test

Set qtResultsOpt = Nothing ' Release the Run Results Options object
Set qtTest = Nothing ' Release the Test object
Set qtApp = Nothing ' Release the Application object
Set qtAutoExportSettings = Nothing ' Release the Automatically Export Report Configuration Options
object
```

What is Object Model?

Automation is a Microsoft technology that makes it possible to access software objects inside one application from other applications. These objects can be created and manipulated using a scripting or programming language such as VBScript or VC++. Automation enables you to control the functionality of an application programmatically.


Object Model Provided by commonly used Applications
Word
WScrip
Excel
UFT
PDF
Quality Center
User/Custom COM Application
WMI
Lotus Notes
popular applications with Object Model
FileSystem
Use functionality of application by CreateObject Statement
User's Class and Object in VBS
Class Foo
Function addTwoNumber(a,b)
msgbox "Added number is = " & (a+b)
End Function
End Class
Foo objectFoo = New Foo
objectFoo.addTwoNumber 9,2
Added number is = 11
OK
Use functionality of Class by Set obj = New ClassName


Advanced References for GUI Testing
UFT Automation Object Model Reference for GUI Testing
Frequently used Object
Introduction
Object Model Diagram
Action Object
ActionParameterDefinition Object
ActionParameterDefinitions Collection
Actions Collection
ActiveScreenAccess Object
ActiveScreenOptions Object
Addin Object
Addins Collection
AdditionalIdentificationProperties Object
Application Object
AutoExportReportConfigOptions Object
AvailableProperties Object
DataTable Object
Environment Object
FoldersOptions Collection
IBPTWrapperTestService Object
ITDPierToTulip Object
JavaDeviceReplay Object
JavaEditSetMode Object
JavaLauncher Object
JavaOptions Object
JavaSettings Object
LastRunResults Object
Launchers Collection
LocalSystemMonitorSettings Object
LogTrackingSettings Object
MainIdentificationProperties Object
ObjectIdentification Object
ObjectRepositories Collection
Options Object
OracleLauncher Object
Parameter Object
ParameterDefinition Object
ParameterDefinitions Collection
Parameters Collection
Recovery Object
RecoveryScenario Object
Resources Object
RunOptions Object
RunResultsOptions Object
RunSettings Object
SAPLauncher Object
SAPOptions Object
ScreenRecorderOptions Object
SiebelLauncher Object
StdApplication Object
StdApplications Collection
StdLauncher Object
SystemCounter Object
SystemCounters Collection
TDConnection Object
TDOTA Object
TEOptions Object
Test Object
TestLibraries Collection
TestObjectClassIdentification Object
TestSettings Object
UpdateRunOptions Object
WebActiveScreen Object
WebLauncher Object
WebOptions Object
WebSettings Object
WindowsAppsOptions Object
UFT Reserved Objects
Enumerations
Exposed Objects in UFT Object Model API