

UFT Features in Nutshell



“And over the course of the last six years, as I've directed more features and commercials, I've become better at articulating exactly how I want the audience to feel.”

- Jason Reitman

In this chapter we will discuss the basics of automation and UFT features to realize the automation. We will discuss UFT scripts, DataTable and Actions.

1.0 Journey from Manual Testing to Automate Testing

Automation can be considered as converting manual steps into programming scripts. Let's discuss a Google search test case with manual test steps.

ManualStep1: User opens Internet Explorer

ManualStep2: User navigates to Google.com

ManualStep3: User search "UFT Super star" and press enter to search on Google

ManualStep4: User Click Images Link and verify "Sign In" button.

ManualStep5: User Click on Search Link

ManualStep6: User repeat Step1 to Step5 with five different random 5 character search text in Step3.

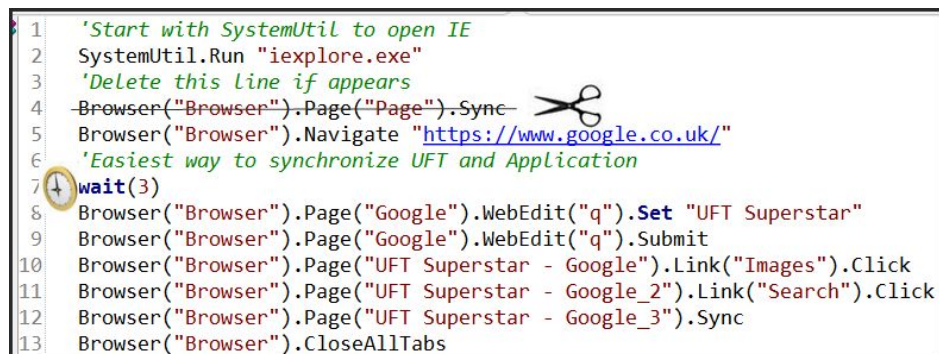
So, now to test this scenario, User will manually perform all these action five times. Now suppose, a robot help us with following thing automatically

1. Open browser and navigate user steps e.g. Step 1 and Step2
2. Generate data for various user input e.g. Step 3 and Step6
3. Verify the user's expected value on the application e.g. Step4
4. Bundle these steps to reuse these steps in any other test cases e.g. Step 1 to Step6 in One file.

UFT exactly is a helper robot for us. We will see how UFT provide all these facilities in the next sections.

2. UFT and Parameterization

Application requires data input from user during testing. Let's record the steps in UFT given in previous exercise and save the script with name UFTSuperStar:



```
1  'Start with SystemUtil to open IE
2  SystemUtil.Run "iexplore.exe"
3  'Delete this line if appears
4  Browser("Browser").Page("Page").Sync
5  Browser("Browser").Navigate "https://www.google.co.uk/"
6  'Easiest way to synchronize UFT and Application
7  ⌚ wait(3)
8  Browser("Browser").Page("Google").WebEdit("q").Set "UFT Superstar"
9  Browser("Browser").Page("Google").WebEdit("q").Submit
10 Browser("Browser").Page("UFT Superstar - Google").Link("Images").Click
11 Browser("Browser").Page("UFT Superstar - Google_2").Link("Search").Click
12 Browser("Browser").Page("UFT Superstar - Google_3").Sync
13 Browser("Browser").CloseAllTabs
```

In this example user provides "UFT SuperStar" text in WebEdit object. Suppose user wants to search with different text than he/she needs to amend the script with new text. We should remove the hard-coded user inputs from the script.

Let's think on the line below, if we replace hard-code text with variable than passing different value to this variable will not force script change. A Variable is used to store temporary information with pre-define datatype. UFT supports VBScripts. VBScript have only one datatype –variant. Variant can hold any type (string, integer, float etc.) of data.

```
8 Browser("Browser").Page("Google").WebEdit("q").Set "UFT-Superstar"  
9                                     Hard coded to Variable  
10 Browser("Google").Page("Google").WebEdit("q").Set myTextVariable
```

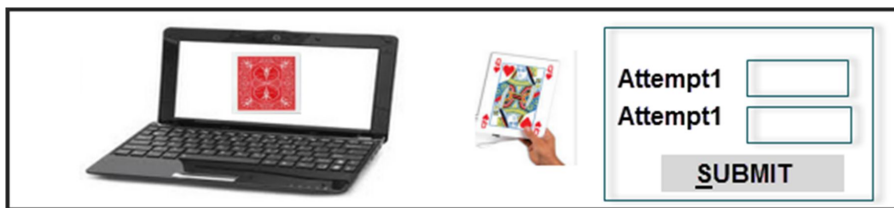
So, the process to change the hard-coded data value with parameter (or variable) known as parameterization. While we are discussing, there is anxious voice of Aadhya coming from the next door. What happened to her?



Aadhya: Hey Aric! Thank God you are here. My job is on stake, please help me

Aric: Hi, what happened Aadhya?

Aadhya: I have asked to test a Playing card Guessing-game program where Computer generate a hidden card number and I need to guess it with 2 attempts. I need to provide two numbers in to edit box and submit into system. I need to find the pattern to verify that how many times I am able to guess computer generated playing card number.



Aric: that is simple what is a problem?

Aadhya: Problem, I need to run the same program for 1 million times and generate the graph to depict the pattern. It will take my entire life if I will do it manually. Should I resign?

Aric: Hey! Hold on. Do you forgot your helper Mr. Robot, The UFT? UFT can insert data for you and run test for a million times.

Aadhya: but how UFT will take data? Will it take data from text file, Excel file, xml files, database or we need to generate random numbers in script.

Aric: Hey Aadhya! You will defeat computer for sure, your guesses are absolutely correct. They are all valid data sources. Apart from that UFT can parameterize data with UFT Data Table, Environment variables, Dictionary Objects, Test and Action parameters etc.

Aadhya: Wow! Great .So, my data can also drive the test. Based on the data value I can manipulate my script logic. Please tell me more about the most common way to parameterize data in UFT.

Aric: Exactly, That's why you can make Data Driven test case by employing parameterizations.

2.1 UFT and DataTable

UFT provides various utilities to accomplish various automation tasks. Data manipulation and storage is an integral part of testing. UFT provide various ways to interact with data. DataTable is one of the most easy and suitable way for test to interact with data. DataTable philosophy came from MS Excel. MS Excel provide a clear tabular way to represent data. DataTable also focus with same idea though it is not same as MS Excel and does not support all features of MS Excel. UFT support to import and export MS Excel (including .xlsx after UFT v11.52) and Datatable. Let's first do an exercise.

START - Exercise 5.1 (of 6 STEPS) – Take data from DataTable (Parameterization)

This exercise will create a Google search test and replace the search text in script with DataTable value. So user need not to go and change the script manually but script can interact with DataTable to modify any test input.

STEP1: Record a Test to search "UFT SuperStar" in Google



[Open UFT](#) | [File](#) | [New](#) | [Test](#) | [Provide "DataTableTest" in GUI Test](#) | [Create](#) | [Record](#) | [Navigate to google.co.uk](#) | [Search with "UFT SuperStar"](#) | [Enter](#) | [Click on Image link](#) | [Close IE](#) | [Stop](#) | [Save](#) | [Click Run*](#) (or F5 key)



***ConfuSense** – Also if your script run fast and your browser is slow (e.g. internet is slow) then you can insert wait(3) or wait(5) statement at that point to force UFT to wait for application to sync with browser)

STEP2: Explore Datatable



[Navigate to UFT](#) | [View](#) | [Data](#)

STEP3: Go to Global Sheet

 [Navigate to UFT | View | Data | Global](#)

STEP4: Write "Automation Star" in first cell


 [Navigate to UFT | View | Data | Global | write "Automation Star" in first cell.](#)

	A	B	C	D	E	F
1	Automation Star					
2						

STEP5: Replace the following line in Action script

```
8 ✖ Browser("Browser").Page("Google").WebEdit("q").Set "UFT Superstar"  
9 TO  
10 ✔ Browser("Browser").Page("Google").WebEdit("q").set DataTable.Value("A", dtGlobalSheet)
```

STEP6: Run the Test

 [Navigate to "DataTableTest" test in UFT | Run with F5 | View | Last Run Results | ALT + V + X | Close](#)

Concept of Exercise 5.1 – Take data from DataTable (Parameterization)

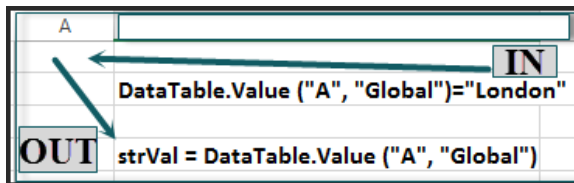
In STEP1, test created to search "UFT SuperStar" in Google. In STEP2, we focus on datatable. DataTable facilitates data for UFT test scripts. It is "similar" to MS Excel spreadsheets and can be used to run an action script multiple times. We will discuss actions later, but we can create many actions in one test. For example supposed to buy an online product we can create three actions - Login action, buy product action and Logout action.

Each test have one Global data sheet that is accessible to all actions inside the test and each action has its own private data table or known as local data table.

In STEP3, we focused on Global DataSheet. In STEP4, we insert "Automation Star" in first cell. This step will not make UFT script to know about "Automation Star". We need to modify script to take the data. In STEP5, We removed hard coded value in script and replace it with DataTable value.

Datatable.Value("A", Global)	
1	2
DataTable.Value(ParameterID, [SheetID])	
SheetID -Sheet name, index or dtLocalSheet, or dtGlobalSheet	

We can Get/Set the data from/in DataTable sheets during run time. Tester need to provide Action name if (s) he want to use action sheet instead of Global sheet.



Key Learning of Exercise: Datatable is a sibling of MS excel. But It is different from MS excel. It is core UFT feature though it can provide a way to import/export data from MS excel. It enables actions script to take data. There are Global and Actions datatables that thought as a way to modularize the global and per action data. Data in Global sheet provides global access of data for all actions while action sheet data modularize action specific data. DataTable supports various methods to devise the logic on it.

END of Exercise 5.1

Aadhya: I got it Aric! Local action sheet is for specific action and Global Sheet is for all action.

Aric: Aadhya, Actually It is misleading. It is possible to access any action's local data table from any other action.

Aadhya: oh! Then why Global and Local Sheets? It is confusing me to put data in many sheets.

Aric: Aadhya, UFT also advocate to keep things simple and self-explanatory. That is why you should use Global Sheet to keep global data used by every action e.g. URL, Constant values etc. that are reusable during whole test and keep the action specific data in Action sheets.

Aadhya: Can I just write one row in the exercise above?

Aric: Let's try to fill data in second cell as well and re-run the script

	A	B	C	D	E
1	Automation Star				
2	Test Automation				

Aadhya: Wow! My script ran two time with different data. So number of lines will equal to the number of script execution. My task of million run seems to be in a reach Aric.

Aric: Congrts Aadhya! Your test is derived by data now i.e. a Data-Driven test.

Types of DataTable - Run Time and Design Time DataTable

If you remember there was a concept of run-time object and design-time object. In Same manner, DataTable also distinguish itself in Run-time and Design-time datatable. During runtime datatable shows run-time value in datatable otherwise UFT shows design-time values in DataTable.

Exercise 5.2 Design Time DataTable

This exercise will explore design time DataTable.

STEP1: Provide multiple value in column or values in multiple columns



Navigate to "DataTableTest" test in UFT | Double Click on column "A" | Provide name "strGoogleSearch" | insert value as per figure | Save

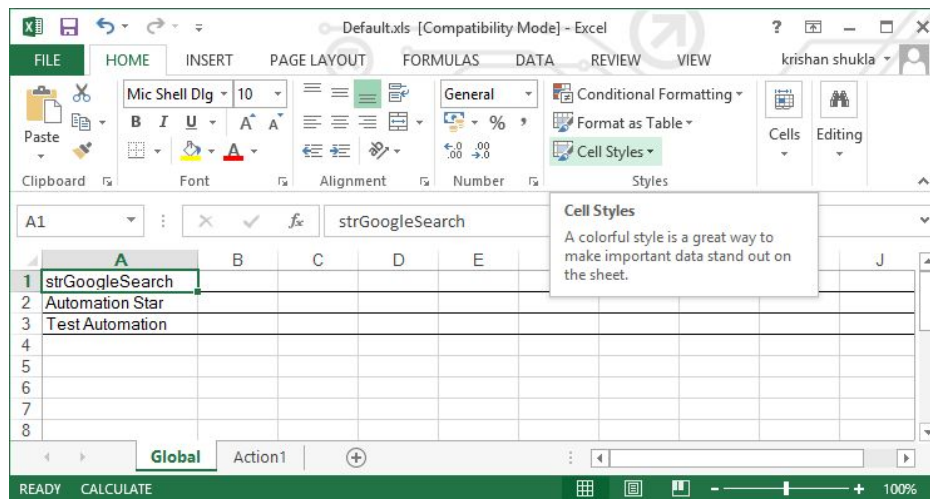
You can change the column name in DataTable e.g. column "A" can be change to any other name by double clicking on column (change to strGoogleSearch). This Column name is basically a parameter for script.

Data				
C2				
	strGoogleSearch	B	C	D
1	Automation Star			
2	Test Automation			

STEP2: Open Design-Time DataTable



Navigate to your Test Folder* | Open Default.xls




***ConfuseSense** – Default location C:\Users\<UserName>\Documents\Unified Functional Testing

Here point to notice that UFT first row of the sheet contains the data but in excel it shows column name in first row. So, when viewed using Excel, the 2nd rows starts the 1st row of data in the DataTable. UFT makes a data row enabled by marking the border of the row in

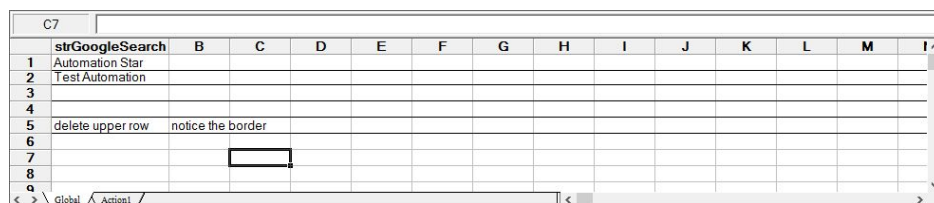
the actual spreadsheet. A row with no data but with marked borders is still considered as and enabled row by UFT. To delete and enabled row we must select the row and delete.

STEP3: Delete the column value but not the row

 *Navigate "DataTableTest" test in UFT | Provide "this is text in row3" in 3rd row | Provide "this is text in row 4" in 4th row | Provide "delete upper row" text in 5th row | Go to 3rd and 4th row and press Delete key | Save*

STEP4: Delete the Row that have no value but remain enables

 *Navigate to DataTable Sheet | Right click on enabled Row | Edit | Delete*




Concept of Exercise 5.2: Design Time DataTable


In STEP1, we can insert data in datatable. In STEP2, when we insert values in DataTable columns, these value also reflect in Default.xls sheet in test folder. This sheet known as Design Time DataTable. DataTable is not same as ME-Excel but more or less basic functionalities remain same. We explored the datatable at physical location in machine. We cannot duplicate column name in sheet but we can provide same data column in different sheets in same test. You need to provide values from first row and cannot skip empty rows. A DataTable is stored as "Default.xls" file in the test folder (location where you saved your test).

Exercise 5.3 Run Time DataTable

This exercise will explore the design time datatable.

STEP1: Code to set the value in DataTable

 *Open UFT | File | New | Test | Provide "RunTimeDataTableTest" in GUI Test | Create | Record | Navigate to google.co.uk | Search with "Krishna Wikipedia" | Enter | Click on Image link | Close IE | Stop | Save | Click Run* (or F5 key) | Insert the following code in script after Browser("Google").Page("Google").WebEdit("q").Set "Krishan Wikipedia" " line | Save | Run*

 *Navigate to "DataTableTest" test in UFT | File | Save As "RunTimeSheetTest" | Insert the following code in script after "Browser("Google").Page("Google").WebEdit("q").Set "UFT SuperStar" " line | Save | Run*

```
DataTable.Value("Secondcol","Global")="RunTimeValue"
```


STEP2: Create Column in DataTable



Navigate to "RunTimeSheetTest" test in UFT | Double click on column "A" in DataTable | Provide name "strGoogleSearch" | Provide value in two rows in first column "Automation Star" & "Test Automation" | Double click on column "B" in DataTable | Provide the new Column Name "Secondcol" | Provide value in two rows in second column "Designtimeval1" & "Designtimeval1" | Save

STEP3: Run Test



Navigate to "RunTimeSheetTest" test in UFT | Run

STEP4: Explore Run-Time sheet



Navigate to Test Folder" | open "Res<n>" folder inside test folder | open "Report" folder | Default.xls

	A	B
1	strGoogleSearch	Secondcol
2	Automation Star	RunTimeValue
3	Test Automation	RunTimeValue

Concept of Exercise 5.3 Run Time DataTable

In STEP1, we are saving new test from existing test. In STEP2, we modified test script to insert value in datatable at run-time. In STEP3, test executed. In STEP4, we explore the run-time datatable. DataTable reside in result folder for specific run. By default UFT create Res<n> folder and increment on every new run.

There are two Default.xls, Design-Time and Run-Time DataTable. Run-Time DataTable contains the DataTable and correspondent value at run-time of test while design-time DataTable contain the values which user provided during test design.

Key Learning of Exercise: Run-time datatable (RTD) and Design-time datatable (DTD) have same name "Default.xls". DTD resides in test folder while RTD resides in res (result) folder under test folder. If we capture data by output value or device our own logic to send data into datatable during run-time then it goes into RTD. When test execution finish UFT bring DTD back to us to design the data for our test.

Exercise 5.3 Run Time DataTable - END

Aadhya: hmm, Aric DataTable is great feature. But is it just limit with getting/setting values of data?

Aric: Do you notice "Value" in Datatable statement. Datatable is much more versatile and it provide many other method then "Value".

Exercise 5.4 Explore DataTable features

This exercise shows the datatable and MS Excel relationship.

STEP1: Create a Test



Open UFT | File | New | Test | Provide "ExploreDataTableTest" in GUI Test | Create | Record | Navigate to google.co.uk | Search with "UFT SuperStar" | Enter | Click on Image link | Close IE | Stop | Save | Click Run (or F5 key)

STEP2: Create Excel Sheet



Navigate to "C:\Test" | *Create "DataForTest.xls" | Provide the value as figure | Save | Close

	A	B	C
1	Country	City	Street
2	USA	Redwood	Dewit Rd
3	UK	London	Victoria Rd
4	India	Indore	MG Rd

Step3: Import the Excel in DataTable in Test



Navigate to "ExploreDataTableTest" | View | Data | Right Click "Sheet" | Import** | File | OK | Select file "C:\Test\ DataForTest.xls" | Provide the value as figure | Save | Close

	Country	City	Street
1	USA	Redwood	Dewit Rd
2	UK	London	Victoria Rd
3	India	Indore	MG Rd



***ConfuSense** – UFT11.52 supports .xls and .xlsx format. Prior to UFT 11.52, only .xls was supported format with UFT.

*User can also create a tab delimited .csv file instead of .xls file.

** Point to remember – If the DataTable exist with columns then first row of imported file should match the column name of UFT DataTable

If you insert value "01234" then DataTable will remove 0 and only "123" will appear in cell. We need to provide '0123 instead of 0123 to force cell to accept string start with 0.

1	123'0123
---	----------

Concept of Exercise 5.4 Explore DataTable features

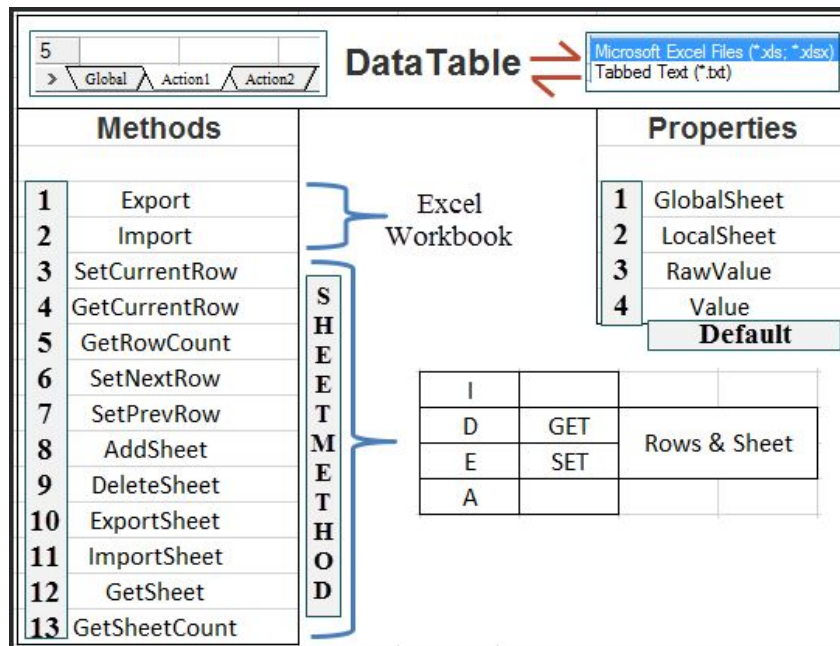
In STEP1, we created a test to search Google. In STEP2, MS Excel file created. In STEP3, we imported MS Excel file in UFT datatable. The first column of MS Excel will be the header of datatable.

DataTable is UFT feature. It looks same as MS-Excel but it is not MS-Excel. It can import and export excel file/sheet. There are similar excel features in datatable e.g. format, sort, encrypt, find and replace column, sort etc. We can programmatically use DataTable object that we will explore later.

Aadhya: Is Datatable features completed? I was expecting Mr. UFT little more versatile to provide more functionality for Datatable.

Aric: No Aadhya! There are many more things in datatable you can do. Let me tell you briefly about few points and rest we will discuss later.

Rule of - IDEA to get/set rows and sheet i.e. Import, Export of excel workbook or sheet, Deletion of sheet, Addition of sheet, Get, Set and count of rows and sheet are all methods applicable for Datatable. Also there are four properties and Value is default property of Datatable.



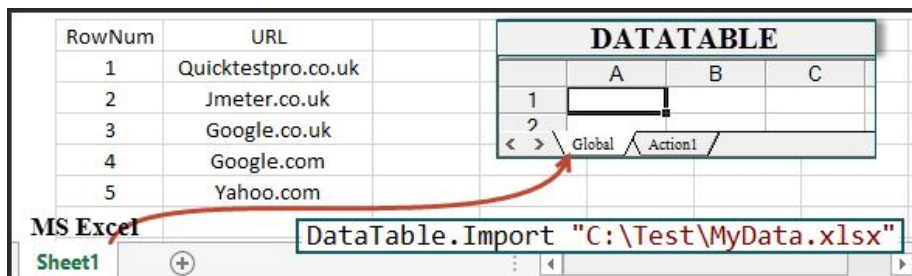
Key Learning of Exercise: Testing needs to manipulate a scenario with lots of data. We may require checking the boundary values, equivalence partitioning values, states, response etc. We can write logic on datatable. During run-time, if we need specific value then we can traverse with in datatable. We can do counting of rows, sheets, import /export, addition and deletion of datatable. Apart from show data clearly in tabular form, datatable is very great thinking of UFT to organize and segregate data from script. But remember this is not the only way to interact with data in script.

Concept of Exercise 5.4 Explore DataTable features - END

Exercise – Exploring Datatable methods

STEP1: Create Excel file

 *Navigate to "C:\Test" | Create Excel | Rename "MyData.xlsx" | Provide data as below*



STEP2: Write UFT program to explore Datatable method and property the sheet

[Navigate to UFT / File / New / Test / Provide "DataTableMethodTes" in GUI Test / Create / Code as given below in Action1 / Run / Stop after executing script](#)

```

DataTable.Import "C:\Test\MyData.xlsx" 1 2
intTotalRow = DataTable.GetSheet("Global").GetRowCount

If intTotalRow > 0 Then 3
    DataTable.GetSheet("Global").AddParameter "Site_Performance", ""
    DataTable.GetSheet("Global").AddParameter "Total_Links_Num", ""
    For i = 1 To intTotalRow
        SystemUtil.CloseProcessByName "iexplore.exe"
        startTime = Timer
        SystemUtil.Run "iexplore", DataTable.Value("URL")
        'We will discuss these two lines later
        Set objLink = Browser("title:=.*").Page("title:=.*").Object.Links
        totallinks = objLink.Length
        SystemUtil.CloseProcessByName "iexplore.exe"
        endTime = Timer
        DataTable.Value("Site_Performance")=(endTime - startTime)
        DataTable.Value("Total_Links_Num")=totallinks
        DataTable.SetNextRow 4
    Next
End If
wait 10000 5
        
```

1. Importing excel

2. Getting total rows

3. Adding column in DataTable and first row value (we gave empty i.e. "")

4. Setting next row in DataTable

5. Pausing. Run-time DataTable will vanish after test finish

RowNum	URL	Site_Performance	Total_Links_Num
1	Quicktestpro.co.uk	1.8125	25
2	Jmeter.co.uk	1.234375	0
3	Google.co.uk	1.34375	40
4	Google.com	0.8359375	40
5	Yahoo.com	3.648438	123

Concept of Exercise – Exploring Datatable methods

In STEP1, we are creating excel file with test data. In STEP2, we are creating program. Point-1 will import excel sheet. The first row of excel will become the column heading of datatable. Point-2 returns the total row counts of the datatable. Point-3 creates column in Global sheet. The program travers on first website (QuickTestPro.co.uk), counts all the links on webpage with the corresponding time taken in this process. These two values, link count and duration to count links, goes in column in datatable. Point-4 do the same logic for all the websites.

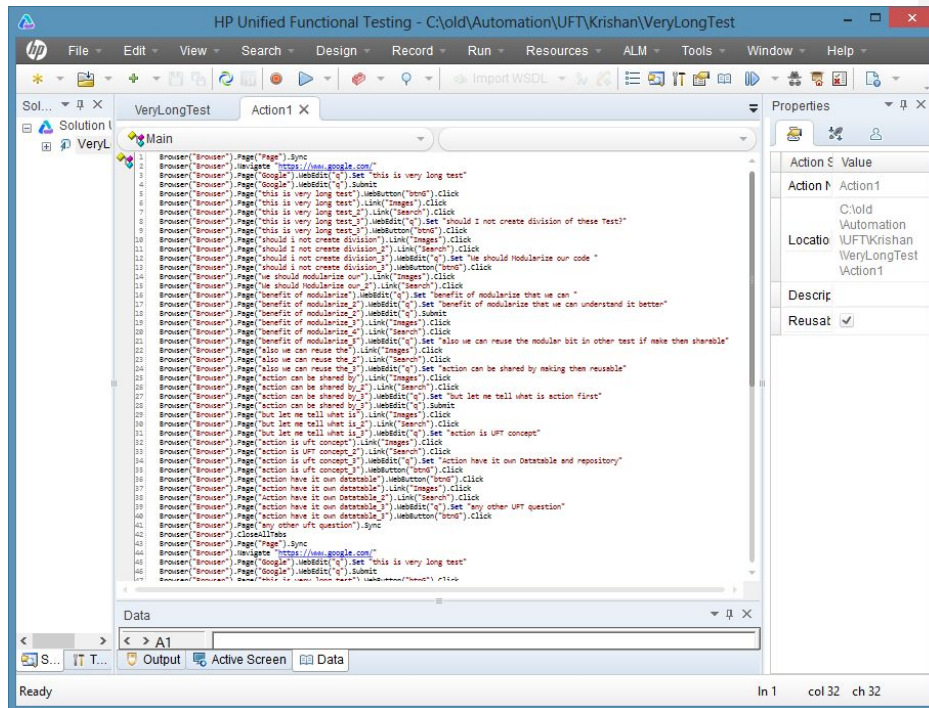
Key Learning of Exercise: Datatable can be manipulate by its methods. We can add sheet, column at run time. Value is default property.

A curious case of parenthesis in VBScript <revise>

Concept of Exercise – Exploring Datatable methods - END

Actions in Action

UFT records everything in actions. By default UFT records script in Action1. A test with hundred steps will create a very big and difficult-to-manage test script. Glance at the following Image, It is just code of n-number of lines. If something goes wrong than we need to debug whole program.



Aadhya: Aric! I need to test the following scenario

1. User open and login into web application (e.g. gmail.com)
2. Verify the first object on the page (First Email)
3. Logout from application
4. Login again in to web application
5. Verify the first object on page
6. Logout from application

I need to do the STEP 1 and STEP 6 i.e. login and logout every time for all 20 test cases as well. Should I need to repeat them each time in each test script?

TestCase 1	TestCase 2	TestCase 3	TestCase 4	...	TestCase-n
Login Steps	Login Steps	Login Steps	Login Steps	...	Login Steps
Testcase1 Logic	Testcase2 Logic	Testcase3 Logic	Testcase4 Logic	...	Testcase-n Logic
LogOut Steps	LogOut Steps	LogOut Steps	LogOut Steps	...	LogOut Steps

Aric: Aadhya, You should make modular code and reuse the modular code in various other tests e.g. login and logout need to create once and use in all scripts. You may create them in separate action and add them in your tests.

Now, suppose we need to automate Testcase1 with following scenario

1. Open IE and Search for "UFT SuperStar"
2. Sign In on google.co.uk
3. Provide your email ID and password
4. Navigate to www.gmail.com
5. Click on "Important" link on the page
6. Click on your user name and Logout
7. Close the browser

Now if do not want to repeat login and logout we can divide action in to three actions and can reuse login and logout action in other tests then we need not to repeat these steps in each script.

```

Main
1  'Open Browser and Navigate to Google Code
2  SystemUtil.Run "iexplore.exe"
3  Browser("Google").Navigate "http://google.co.uk/"
4
5  'Google Search Code
6  Browser("Google").Page("Google").WebEdit("q").Set "UFT SuperStar"
7  Browser("Google").Page("Google").WebEdit("q").Submit
8
9  'Sign -in Code
10 Browser("Google").Page("UFT SuperStar - Google").Link("Sign in").Click
11 Browser("Google").Page("Google Accounts_2").WebEdit("Passwd").SetSecure "51d1dd79afa0ffc5"
12 Browser("Google").Page("Google Accounts_2").WebEdit("Email").Set "KrishanShuklaUFT"
13 Browser("Google").Page("Google Accounts_2").WebButton("Sign in").Click
14 Browser("Google").Page("UFT SuperStar - Google_2").Sync
15
16 ' Navigate to Gmail Code
17 Browser("Google").Navigate "http://gmail.com/"
18 Browser("Google").Page("Inbox (7) - krishanshuklaft@g").Link("Important").Click
19 Browser("Google").Page("Important - krishanshuklaft@g").Link("Krishan Shukla").Click
20
21 'Logout Code
22 Browser("Google").Page("Important - krishanshuklaft@g").Link("Sign out").Click
23 Browser("Google").Page("Gmail: Email from Google").Sync
24 Browser("Google").CloseAllTabs

```

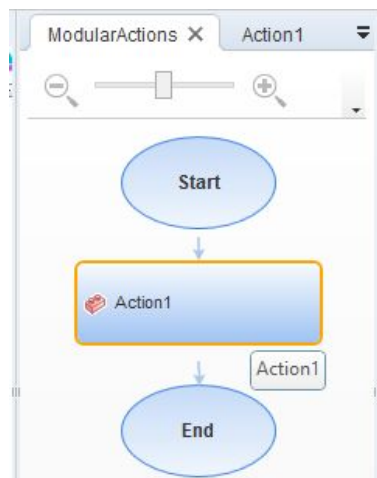
Plan the Modular Test

Exercise 5.5 of 15 STEPS – Create reusable actions

In the previous section we saw the advantages of modular program. Let's do one exercise to develop a modular script.

STEP1: Create a test "ModularAction"

 [Navigate to UFT | New | Test | GUI Test | ModularAction | Create](#)

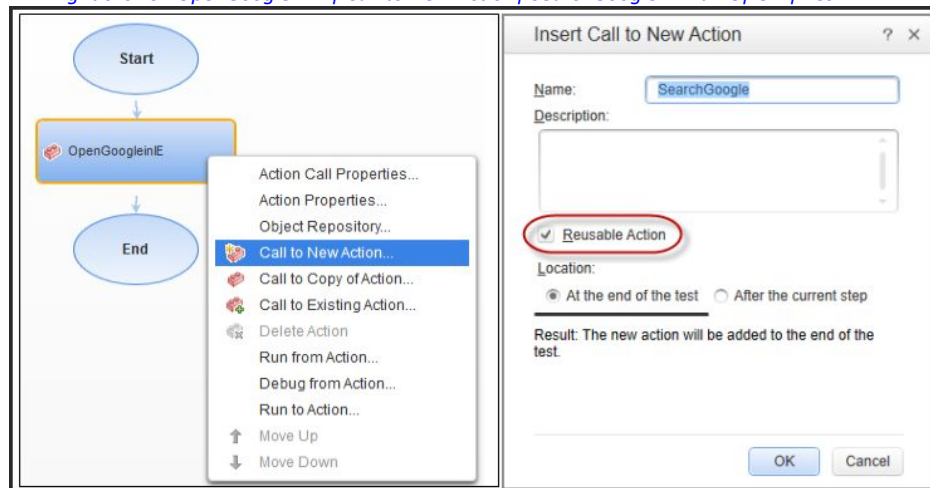


STEP2: Rename the Action1 with "OpenGoogleinIE" Action

 [Right click on Action1 | Action Properties | OpenGoogleinIE in Name | OK | Yes](#)

STEP3: Call New Action after OpenGoogleinIE Action

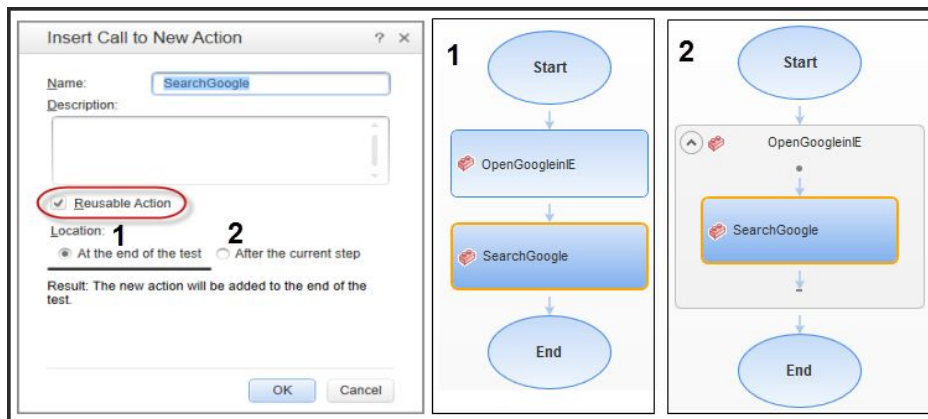
 [Right click on OpenGoogleinIE | Call to New Action | SearchGoogle in Name | OK | Yes](#)



STEP4: Call New Action after SearchGoogle Action



Right click on SearchGoogle | Call to New Action| SignIn in Name| OK | Yes



STEP5: Call New Action after SignIn Action



Right click on SignIn | Call to New Action| Login in Name| OK | Yes

STEP6: Call New Action after Login Action

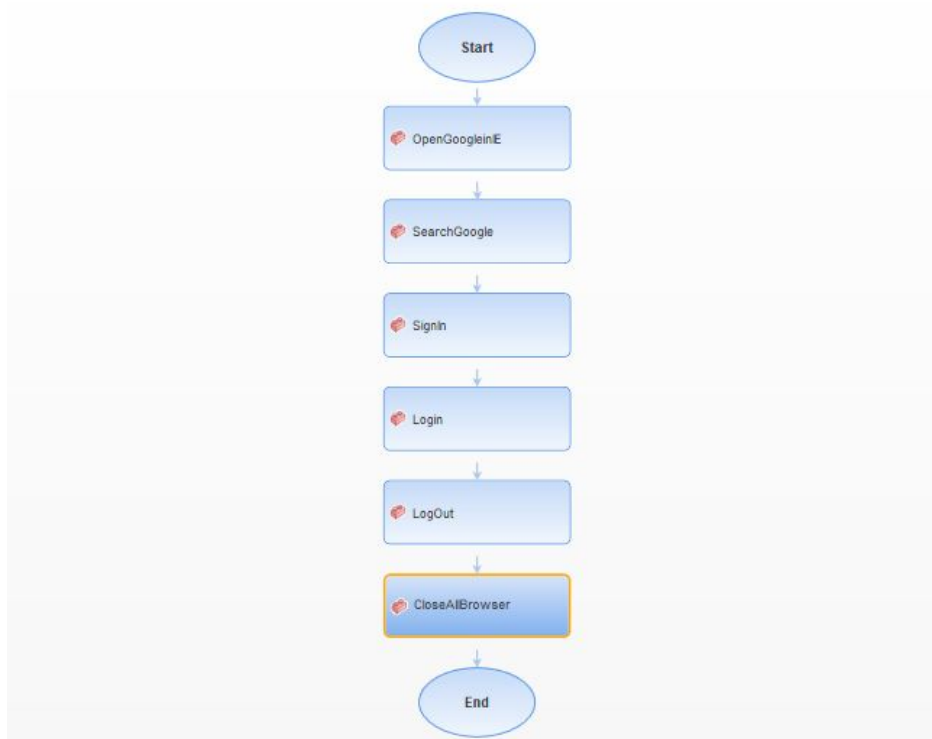


Right click on Login | Call to New Action| LogOut in Name| OK | Yes


STEP7: Call New Action after LogOut Action

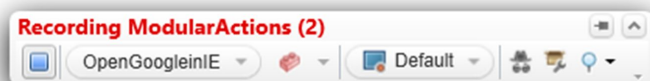


Right click on LogOut | Call to New Action| CloseAllBrowser in Name| OK | Yes



STEP8: Record OpenGoogleinIE Action

 Click on Record Button (or F6 Key) | Click on Internet Explorer| Google.com| Enter | Stop by Recording but do not close browser | Modify script as in figure



ModularActions*		OpenGoogleinIE X
Main		
1		
2	<i>'If you don not get Systemutil step in script then manually write it</i>	
3	SystemUtil.Run "iexplore.exe"	
4	Browser("Google").Navigate " https://www.google.com/ "	
5	Browser("Google").Page("Google_2").Sync	
6		
7	wait 1	

STEP9: Record SearchGoogle Action



Double Click on SearchGoogle in UFT | click on Record | Search “HP Unified Functional Test” in google | Click on Search button | Stop Recording but keep browser open

STEP 10: Record SignIn Action



Double Click on SignIn in UFT | click on Record | Click on SignIn Button | Stop Recording but keep browser open

STEP 11: Record Login Action



Double Click on Login in UFT | click on Record | Provide Email ID and Password | Click on SignIn button | Stop Recording but keep browser open

STEP 12: Record Logout Action



Double Click on LogOut in UFT | click on Record | Click on google user link and Logout | Stop Recording but keep browser open

STEP 13: Record Close All Browser Action



Double Click on CloseAllBrowser in UFT | click on Record | Close InternetExplorer | Stop Recording

STEP 14: Replay the Modular Action Test

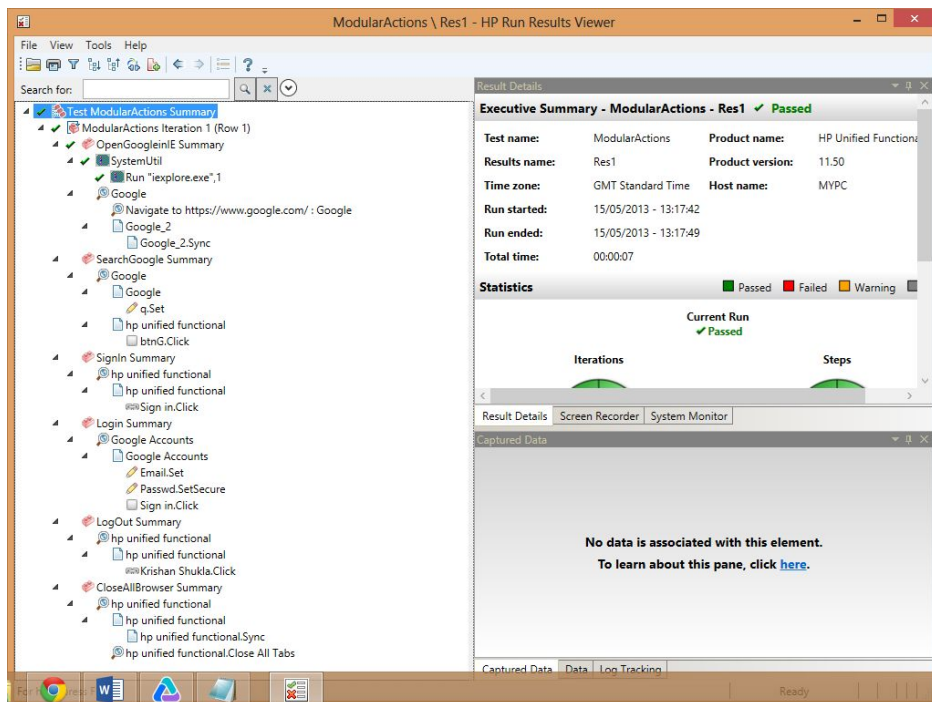


Go to File | SaveAll | Modular Action | Run (or F5 Key)

STEP 15: View the result



Go to View | Last Run Results | Click on result tree | View | Expand All



Concept of Exercise 5.5 – Create reusable actions

In STEP1, we are creating a test. Default test comes with Action1. In STEP2, we are renaming action1 to OpenGoogleinIE. In STEP3, we are calling a “SearchGoogle” action in “OpenGoogleinIE” action. This action is reusable action so it can be called in other tests as well. We can add action at the end of the test or after current step. In STEP4 to STEP-7, we are creating reusable actions same as in STEP3. In STEP8, we are recoding our logic in “OpenGoogleinIE” action. In STEP9 to STEP13, we are recording logic in their respective actions.

Key Learning of Exercise: Test can accommodate multiple actions. We can insert new or already existing actions in the test flow. Actions modularize the code, and they are UFT specific concept. Every action have its own object repository (i.e. local repository for each action).

Concept of Exercise 5.5 – Create reusable actions - END

Aadhya: Aric, You must be kidding. Are you doing time pass by telling all these techniques? Recording is straight forward and you unnecessary created so many actions.

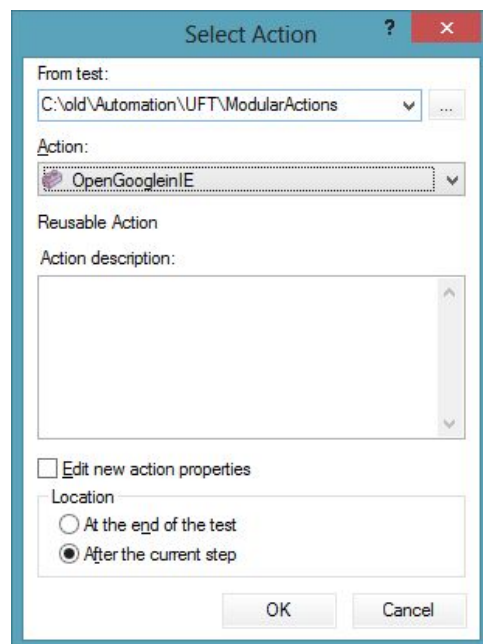
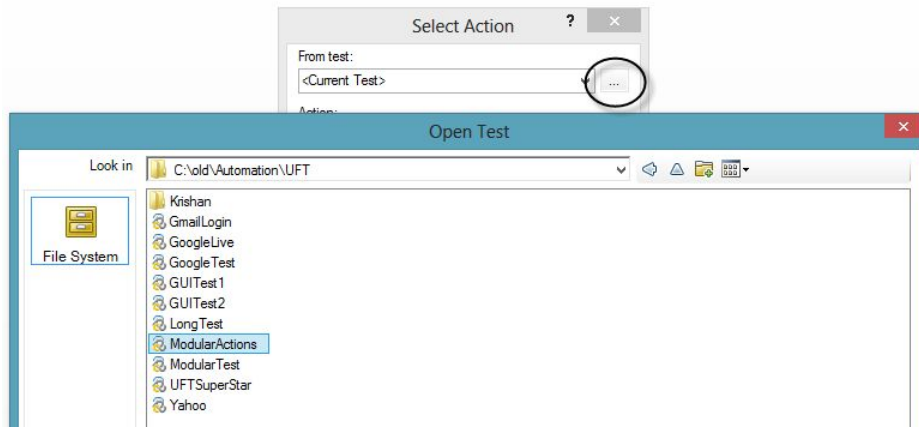
Aric: Hey Aadhya, Keep patience. Now let’s create a new test with name “GoogleLive”. In this test we will not search anything and just verify login and logout functionality.

Exercise 5.6 – Create a Test with Reusable actions

STEP1: Create "GoogleLive" Test



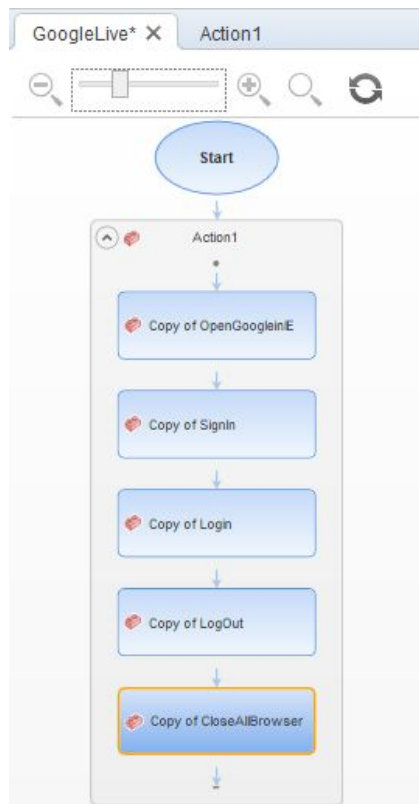
Right Click on GoogleLive in UFT | Call to Copy Action | Select "ModularAction" Test | Select "OpenGoogleinIE" | OK



2. Repeat Adding by copy Actions



Right Click on GoogleLive in UFT | Call to Copy Action | Select ModularAction Test | Select SingIn | OK | Repeat the step for all the actions in figure



3. Run the GoogleLive Test



[Go to File](#) | [SaveAll](#) | [GoogleLive Action](#) | [Run \(or F5 Key\)](#) | [Go to View](#) | [Last Run Results](#) | [Click on result tree](#) | [ALT + V + X](#) | [Close](#)

Concept of Exercise 5.6 – Create a Test with Reusable actions

In STEP1, We are calling reusable action “OpenGoogleinE” from “ModularAction” test. In STEP2, we are calling all the action as shown in figure. In STEP3, we are verifying the test run in UFT reports.

Key Learning of Exercise: Actions modularize the code. If we can design our whole automation test functionality in various actions then we need not to write the logic again. It also gives the flexibility to maintain the code at one place. So any new change in application will require change in code at few places instead of modifying each script separately.

Exercise 5.6 – Create a Test with Reusable actions - END

Aric: Have you noticed that this time we have not recorded or created any action. We reuse action and successfully create a new automated test. Also we have clear idea about the specific action so if

any error occurs during runtime then we can easily figure out in to particular action instead of debugging whole script.

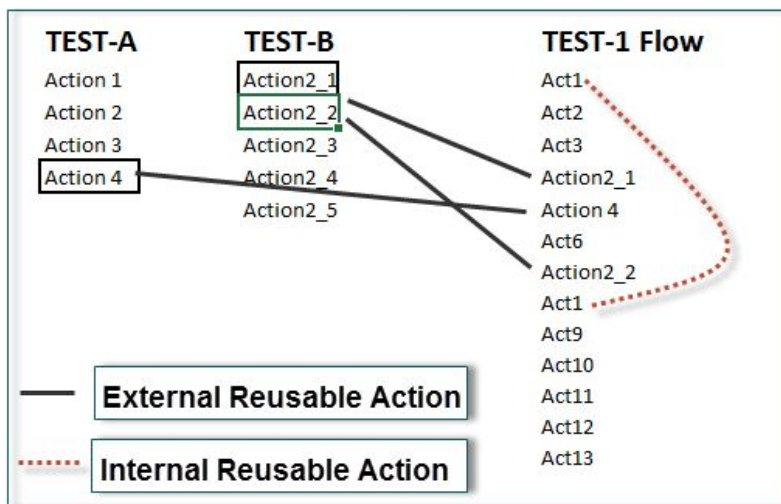
Aadhya: Ok that's great! So it is like to create an Action and make the copy of that action for n-number of times and use them in any tests. Can you tell me more about actions?

Types of Actions in UFT

We have explored the "Call to copy Action" and "Call to New Action" in last exercise. There are three types of actions:

1. Normal/Non-Reusable action – An action that can be called only in the test in which it resides and can be called only once. It make non-reusable action, uncheck the reusable action checkbox from action properties.
2. Reusable Action – An action that can be called multiple times by the test in which it resides and can also called by other tests
3. External reusable action – A reusable action stored in another test. External action are read only in the calling test, but we can choose to use a local, editable copy of the Data Table information for the external action.

Commented [ks1]: From 2nd edition



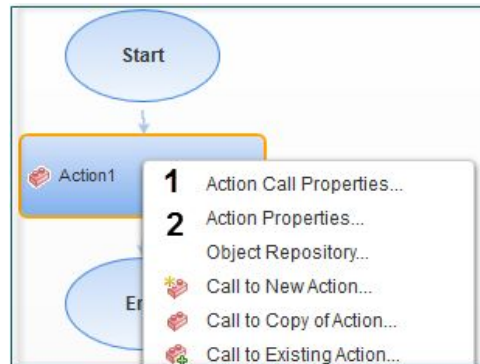
Inserting Calls to Action:

There are three types of actions calls that can be inserted

1. An "Insert Call to New..." will create a new action in the current test.
2. An "Insert Call to Existing" will allow to insertion of a call to reusable action present in the current test or in another test.
3. An "Insert Call to Copy..." inserts a copy of the specified action into the current test. If the copied action uses checkpoints and OR objects, these are also copied.

Action Properties in UFT


There are Action properties and Action Call properties. Action properties have attribute like action name, location, parameters and reusability. Action Call property deal with the number of iterations action need to run.

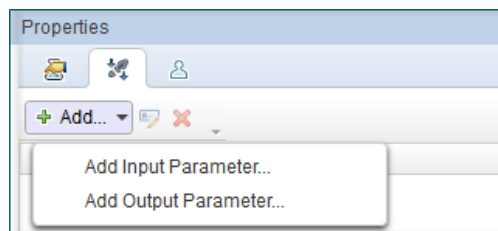


Exercise 5.7 – Action Properties and Parameterization

This exercise shows the action input and output parameters and action properties.

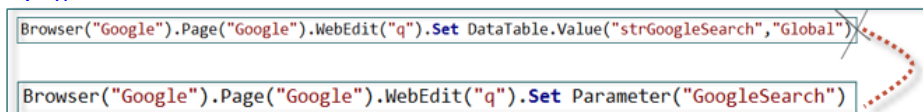
STEP 1: Create a Input Parameter in Action1

 Navigate to "DataTableTest" test in UFT | Save as "ActionParamTest" | View | Solution Explorer | Action1 | Right Click and select "Properties" | Click on Properties | Add | Add Input Parameter... | Select "GoogleSearch" in Name | Select String in Type | Provide "This Value is from Action Parameter" in Default Value | OK | Yes



STEP2: Access parameter in test script

 Navigate to "ActionParamTest" | Change the following line | Save | Run | Last Run Results | ALT + V + X



```
Browser("Google").Page("Google").WebEdit("q").Set DataTable.Value("strGoogleSearch", "Global")
Browser("Google").Page("Google").WebEdit("q").Set Parameter("GoogleSearch")
```

STEP3: Create Output parameter in Action 1 and Input parameter in Action2



Navigate to "ActionParamTest" | View | Solution Explorer | Right click on "ActionParamTest" | Add | Call New Action | Click OK to create Action 2 | Right click on Properties of Action 1 | Click on



in properties | Add | Add Output parameter | Provide "Action1_Output_For_Action2" in name | OK | Yes | View | Solution Explorer | Right click on "ActionParamTest" | Right click on



Properties of Action 1 | Click on in properties | Add | Add Input parameter | Provide "Action2_Input_Map_Action1_Output" in name | OK | Yes

STEP4: MAP Output parameter of Action1 with Input parameter of Action2



Navigate to "ActionParamTest" | View | Test Flow | Right click on Properties of Action 2 | Action Call properties | Parameter Values | Click on "Action2_Input_Map_Action1_Output" row | Click on Value Cell | Click on <#> | Select "Parameter" in Value Configuration Options | Select Output from previous Action | Select Action1 and Action1_Output_For_Action2 | OK | OK | Double Click on Action 2 | write following line | Save | run

```
Msgbox Parameter("Action2_Input_Map_Action1_Output")
```

STEP5: Provide value to Output parameter of Action1



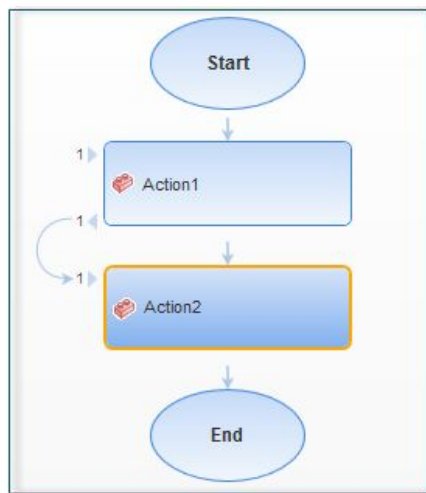
Navigate to "ActionParamTest" | View | Test Flow | Double click on Action 1 | insert following line before 7th line | run | Save

```
Parameter("Action1_Output_For_Action2")= "Value from Action1 for Action2"
```

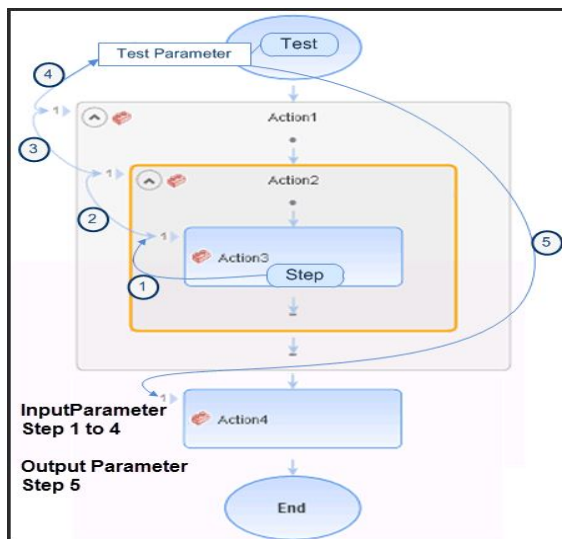
Concept of Exercise 5.7 – Action Properties and Parameterization

In STEP1, we are creating input parameter. In STEP2, we are taking the input parameter value in script. We can take Input parameter in script by parameter ("parametername"). This is another way to parameterize the test like datatable. In STEP3, we create an output parameter to send it to another action. In STEP4, we are sending the output parameter from first action into input parameter in second action. In STEP5, we are giving value in output parameter of action1.

We have explore that DataTable can be used for parameterization. Actions are like function they can take input parameter and return output parameter during runtime. Output parameter of action can be used by other actions as their input parameter. So, effectively make the proper chain of actions.



Input action parameter values can be used only in current action. If current action contains the other action the input parameter can be used in them as well.
 Output action parameter values can be retrieved from a previous action at the same hierarchical level.



Key Learning of Exercise: Actions are sibling of functions though they differ by having datatable and OR. Action can have input and output parameters. Input parameter can be access by using Parameter method.

Exercise 5.7 – Action Properties and Parameterization - END


Exercise 5.8 – Action Iterations and DataTable rows

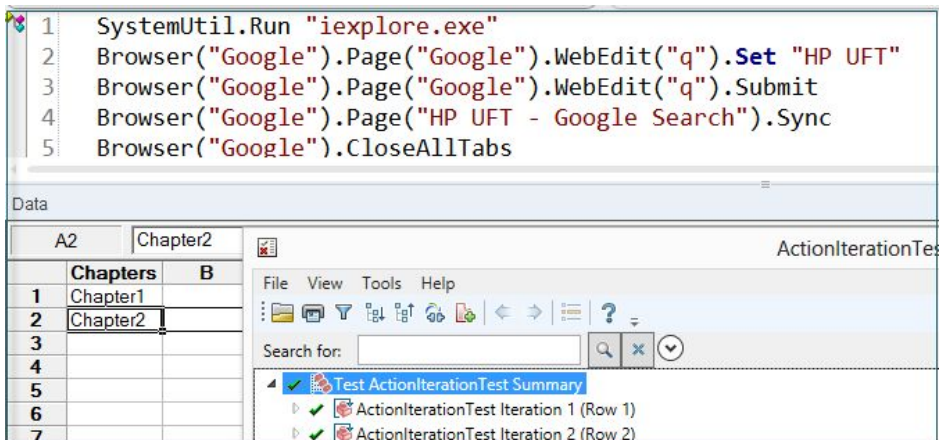
This exercise depicts the relationship between datatable and action iterations.

STEP1: Create a Test

 *Open UFT | File | New | Test | Provide "ActionIterationTest" in GUI Test | Create | Record | Navigate to google.co.uk | Search with "HP UFT" | Enter | Close IE | Stop | Save | Click Run (or F5 key)*

STEP2: Modify Global DataTable with two rows

 *Navigate "ActionIterationTest" test in UFT | View | Data | Global Sheet | Double click on column "A" | Provide "Chapters" | OK | Insert "Chapter1" in first row | Insert "Chapter2" in second row | Save | Run | View | Last Run Results | ALT + V + X*



The screenshot shows the UFT interface. The top pane displays five test steps:

- 1 SystemUtil.Run "iexplore.exe"
- 2 Browser("Google").Page("Google").WebEdit("q").Set "HP UFT"
- 3 Browser("Google").Page("Google").WebEdit("q").Submit
- 4 Browser("Google").Page("HP UFT - Google Search").Sync
- 5 Browser("Google").CloseAllTabs


The bottom pane shows a data table with the following structure:

Data	
A2	Chapter2
Chapters	B
1	Chapter1
2	Chapter2
3	
4	
5	
6	
7	

Below the data table, there is a summary of test iterations:

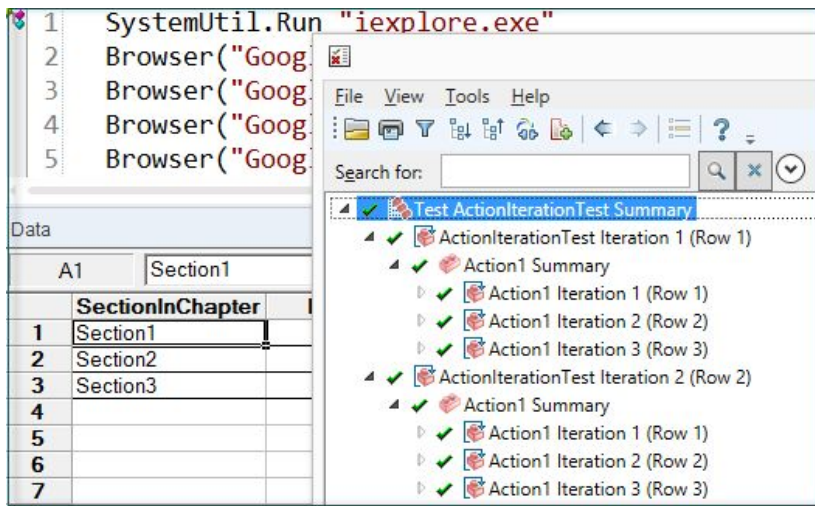
- Test ActionIterationTest Summary
- ▶ ActionIterationTest Iteration 1 (Row 1)
- ▶ ActionIterationTest Iteration 2 (Row 2)

STEP3: Modify Action DataTable with three rows

 *Navigate "ActionIterationTest" test in UFT | View | Data | Action1 Sheet | Double click on column "A" | Provide "SectionInChapter" | OK | Insert "Section1" in first row | Insert "Section2" in second row | Insert "Section3" in third row | Save | Run | View | Last Run Results | ALT + V + X*

STEP4: Modify Action Call property for DataTable

 *Navigate "ActionIterationTest" test in UFT | View | Test Flow | Right click on "Action1" | Action Call Properties | Select "Run on all rows" | OK | OK | Run | View | Last Run Results | ALT + V + X*



STEP5: Modify Action Call property for DataTable



Navigate "ActionIterationTest" test in UFT | View | Test Flow | Right click on "Action1" | Action Call Properties | Select "Run from row" | Provide 1 in first field | Provide 2 in second field | OK | OK | Run | View | Last Run Results | ALT + V + X

Concept of Exercise 5.8 – Action Iterations and DataTable rows

In STEP1, we are creating "ActionIterationTest" to search Google. In STEP2, we are writing two rows in Global datatable. In STEP3, we are modifying Action datatable with three rows. In STEP4, we are giving action iteration number (run on all rows) by setting iteration in Action call properties. So the test will run for Global (2) X Local iteration (3) = Total iteration (6). In STEP5, we are giving action iteration number from row 1 to row 2.

Action iteration can be control by DataTable Iteration setting. Every action has the "DataTable iteration" property in "Action call properties". Action default iteration is one but we can change to all datatable rows or the range of datatable rows. Point to notice is that we are discussing of Action specific datasheet and not global datasheet. Test run in following way:

GlobalSheet (row 1) X Action datatable (rows based on DataTable iteration setting)
 GlobalSheet (row 2) X Action datatable (rows based on DataTable iteration setting)
 ...
 GlobalSheet (row n) X Action datatable (rows based on DataTable iteration setting)


Key Learning of Exercise: We can manage the number of action iteration by setting datatable iteration in action call properties. We can manage this programmatically as well by using UFT object model.

Exercise 5.8 – Action Iterations and DataTable rows - END

Exercise 5.9 – Multiple Action Iterations and DataTable rows

STEP1: Create a Test

 Open "ActionIterationTest" test in UFT | File | "Save ActionIterationTest As" | "MultiActionIterationTest" | Save |

 STEP2: Create Action2 and set DataTable iterations

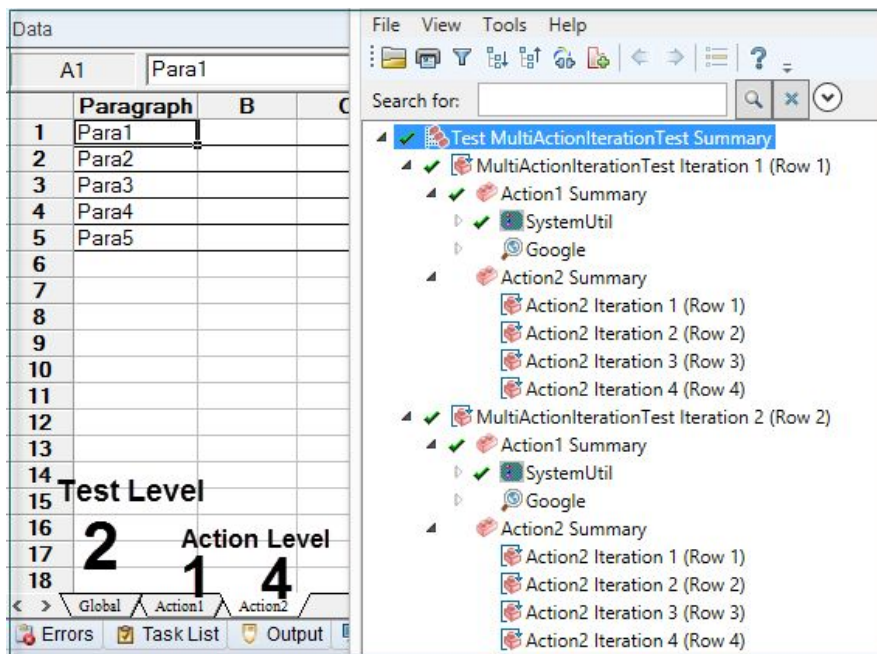
Navigate to "MultiActionIterationTest" test in UFT | View | Test Flow | right click on Action1 | Call to new Action | OK | View | Data | Action2 in DataTable | Double click on "A" | "Paragraph" | OK | "Para1" in 1st row | "Para2" in 2nd row | "Para3" in 3rd row | "Para4" in 4th row | "Para5" in 5th row | View | Test Flow | Right click on Action1 | "Action Call Properties" | Select "Run one iteration only" | OK | Right click on Action2 | "Action Call Properties" | Select "Run from row | Provide 1 to 4 row | OK | Save | Run | View | Last Run Results | ALT + V + X

Concept of Exercise 5.9 – Multiple Action Iterations and DataTable rows

In STEP1, we are saving as "ActionIterationTest". In STEP2, we are creating an Action2. It will give us an opportunity to have one action2 datatable. We set Action1 with one iteration and Action2 with 1 to 4 row. This time total number of iteration will be

Action1 Execution = Global (2) X Action1 (1) = Total iteration (2)

Action2 Execution = Global (2) X Action (4) = Total iteration (8)



The screenshot displays the UFT interface. On the left, a Data table is visible with columns A1, Paragraph, B, and C. The rows contain data from Para1 to Para5. Below the table, the 'Test Level' is set to 2 and the 'Action Level' is set to 4. On the right, the Test Flow tree shows the test structure. It includes 'Test MultiActionIterationTest Summary', 'MultiActionIterationTest Iteration 1 (Row 1)', 'Action1 Summary', 'SystemUtil', 'Google', 'Action2 Summary', 'Action2 Iteration 1 (Row 1)', 'Action2 Iteration 2 (Row 2)', 'Action2 Iteration 3 (Row 3)', 'Action2 Iteration 4 (Row 4)', 'MultiActionIterationTest Iteration 2 (Row 2)', 'Action1 Summary', 'SystemUtil', 'Google', 'Action2 Summary', 'Action2 Iteration 1 (Row 1)', 'Action2 Iteration 2 (Row 2)', 'Action2 Iteration 3 (Row 3)', and 'Action2 Iteration 4 (Row 4)'.

A1	Paragraph	B	C
1	Para1		
2	Para2		
3	Para3		
4	Para4		
5	Para5		
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			

Key Learning of Exercise: Every action has action call properties. These properties provide way to manage action iterations based on the rows in datatable. By default it runs for one iteration with "run one iteration only" option.

Exercise 5.9 – Multiple Action Iterations and DataTable rows - END

Exercise 5.10 – Action Vs Test Iterations

STEP1: Set Global Data Table Iteration to manage Test Iteration



Open “MultiActionIterationTest” test in UFT | File | “Save MultiActionIterationTest As” | “TestIterationTest” | Save |

STEP2: Set the Global sheet iteration to set whole test iteration



Navigate to “TestIterationTest” test in UFT | File | Settings | Run | Select Run one iteration only | OK

STEP3: Run the test



Navigate to “TestIterationTest” test in UFT | Run | View | Last Run Results | ALT + V + X

Data		Search for:
A2	Chapter2	
Chapters	B	
1	Chapter1	
2	Chapter2	
3		
4		
5		
6		

- Test TestIterationTest Summary
 - Action1 Summary
 - Action2 Summary
 - Action2 Iteration 1 (Row 1)
 - Action2 Iteration 2 (Row 2)
 - Action2 Iteration 3 (Row 3)
 - Action2 Iteration 4 (Row 4)

STEP4: Change the Global Data sheet iteration setting



Navigate to “TestIterationTest” test in UFT | File | Settings | Run | Select “Run from row” | Provide “1” to row “2” | OK

STEP5: Run test for the changed settings



Navigate to “TestIterationTest” test in UFT | Run | View | Last Run Results | ALT + V + X

Concept of Exercise 5.10 – Action vs. Test Iterations

In STEP1, we are “saving as” MultiActionIterationTest test as TestIterationTest. In STEP2, we are setting the test iteration to one by managing Global sheet iteration property. It is different from Action iteration. In STEP3, we are running test. This test runs only first row of the Global sheet. In STEP4, we are resetting the test iteration to run with row 1 to 2.

The number of action iteration manage by datatable iteration setting in “action call properties” while whole test iteration can be manage by the Global sheet iteration value. Global sheet iteration manage by the UFT Settings (File | Settings | Run)

Global Data Table

Data Table iterations

☐ Run one iteration only
☒ Run on all rows
☐ Run from row to row

File | Settings | run

Global DataTable	Action 1 DataTable	Action2 DataTable
Row1	Row1	Row1
	Row2	Row2

	Row-n	Rown
Row2	Row1	Row1
	Row2	Row2

	Row-n	Row-n
...		
Row-n	Row1	Row1
	Row2	Row2

	Row-n	Row-n

Action DataTable

Data Table iterations

☒ Run one iteration only
☐ Run on all rows
☐ Run from row to row

Action call properties

Key Learning of Exercise: Datatable is programmable, rich with features, compatible with MS Excel. It also helps to manage the test iterations. Iteration can be set for test level or action level. So, datatable not only modularize the data but also helps to manage the flow of test.

Exercise 5.10 – Action Vs Test Iterations - END

Troubleshooting and Limitations - Actions

Limitation1: If you make a copy of an existing test (either in the file system or in ALM), then you cannot insert a call to the same action from both of these tests into the same test.

Workaround1: Instead of creating a copy of the test, use Save As to create a duplicate of the test.

Limitation2: You cannot create a call to a new action called Global, since Global is the name reserved for the Global sheet in the Data pane. If you create an action called Global, you will not be able to select the local or global data sheet when parameterizing an identification property.

Workaround2: Do not create action with name "Global".

Limitation3: Calling an external action from a function library using the RunAction statement results in a run-time error.

Workaround3: Call the action using the LoadAndRunAction statement.

Limitation4: You cannot add a new action as a nested action to an external action.

Workaround4: Open the external action and add the call to the nested action directly.

Workaround5: The Action Template feature supported in QuickTest 11.00 is not yet supported in UFT.

Aadhya: Aric, Action feature is great feature in UFT. It is providing the reusability and modularization of code. Is there any other way to do the same things?

Aric: Well Functions, classes, sub-procedures can also segregate the logic from script navigation. They can reuse multiple times. Actions are the advance way to managing test reusability because it comprises its own datatable, input-output parameters and

configurable run iteration settings (programmable or by UFT GUI). Also it makes read-only sharing of action by using “call existing test” or let user creates its own copy by copying the reusable test.



Summary

In this chapter we learned about datatable to modularize the data from test and action to modularize the code from script. These are the two most important UFT features. Actions and datatable can be manage, update, create, edit, associate, read, delete and manipulate by programming. UFT has its own object model. This object model let us program UFT tasks and settings. Datatable supports various methods and properties to get/set data, sheets and it can communicate with MS excel (import/export). Actions can be reusable or non-reusable. Reusable action can be called in external test or internal test multiple times. The datatable iteration settings in action call properties can restrain the action iterations. So in this way you get the linkage between actions and datatables.