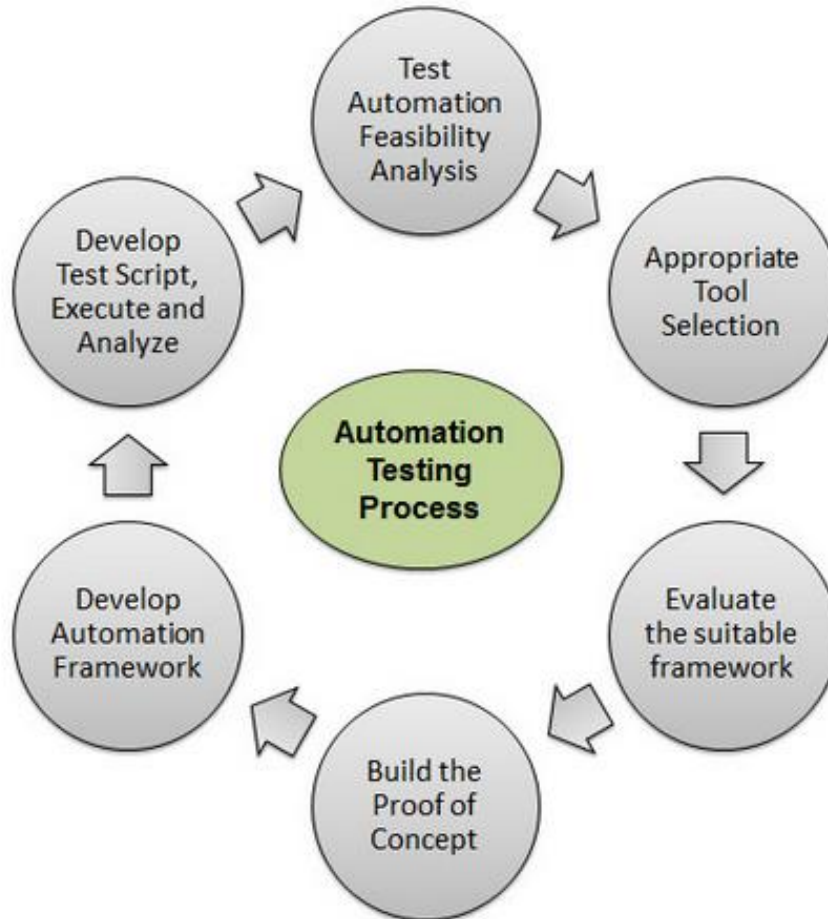


# UNIFIED FUNCTIONAL TESTING BASICS

## Automated Testing Process:

For any automated tool implementation, the following are the phases/stages of it. Each one of the stages corresponds to a particular activity and each phase has a definite outcome.



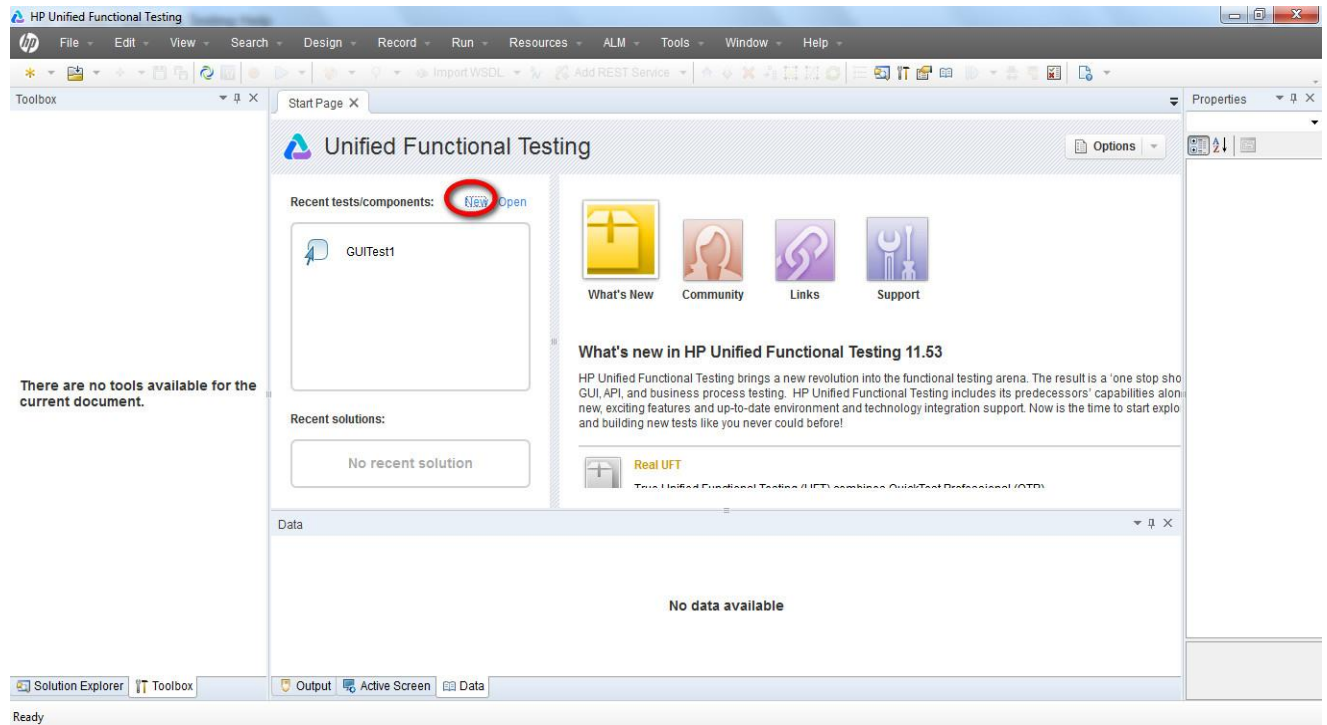
1. **Test Automation Feasibility Analysis** - First step is to check if the application can be automated or not. Not all applications can be automated due to its limitations.
2. **Appropriate Tool Selection** - The Next most important step is the selection of tools. It depends on the technology in which the application is built, its features and usage.
3. **Evaluate the suitable framework** - Upon selecting the tool the next activity is to select a suitable framework. There are various kinds of frameworks and each framework has its own significance. We will deal with frameworks in detail later this chapter.
4. **Build the Proof of Concept** - Proof of Concept(POC) is developed with an end to end scenario to evaluate if the tool can support the automation of the application. As it is performed with an end to end scenario which will ensure that the major functionalities can be automated.
5. **Develop Automation Framework** - After building the POC, framework development is carried out which is the crucial step for the success of any test automation project. Framework should be build after diligent analysis of the technology used by the application and also its key features.
6. **Develop Test Script, Execute and Analyze** - Once Script development is completed, the scripts are executed, results are analyzed and defects are logged, if any. The Test Scripts are usually version controlled.

# Record and Playback

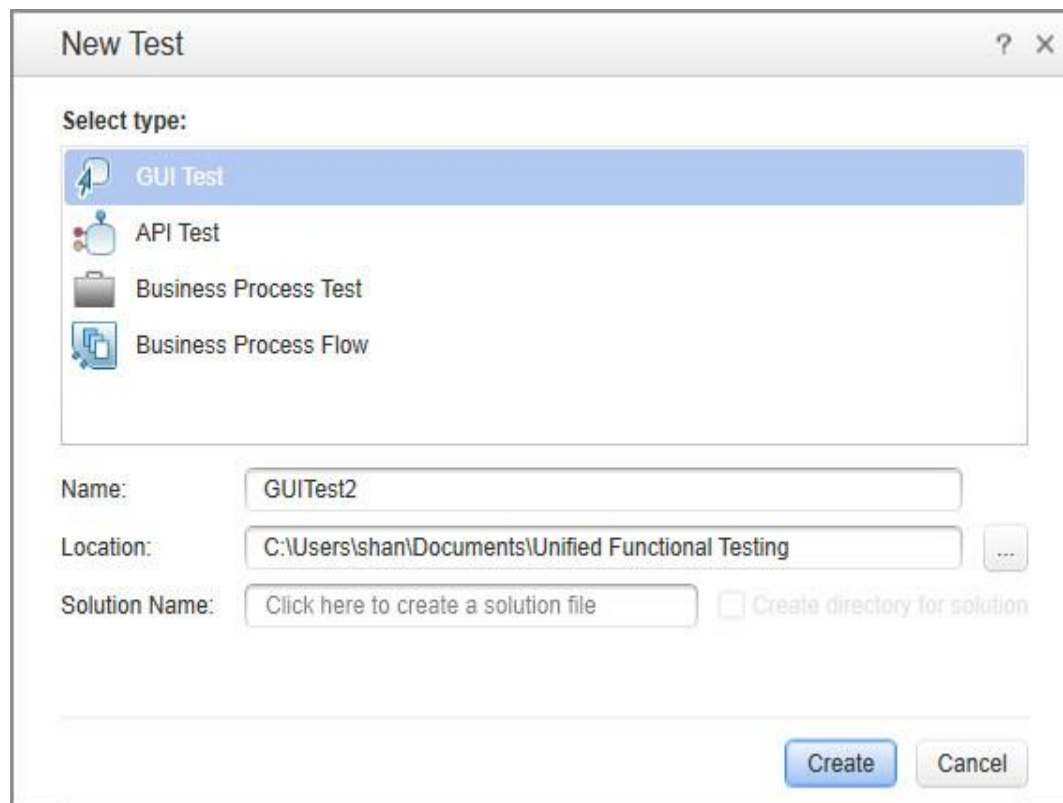
Recording a test corresponds to recording the user actions of the application under test so that UFT automatically generates the scripts that can be played back. Record and Playback can give us the first impression if the tool can support the technology or NOT if the initial settings are done correctly.

Steps for Record and Playback is as follows:

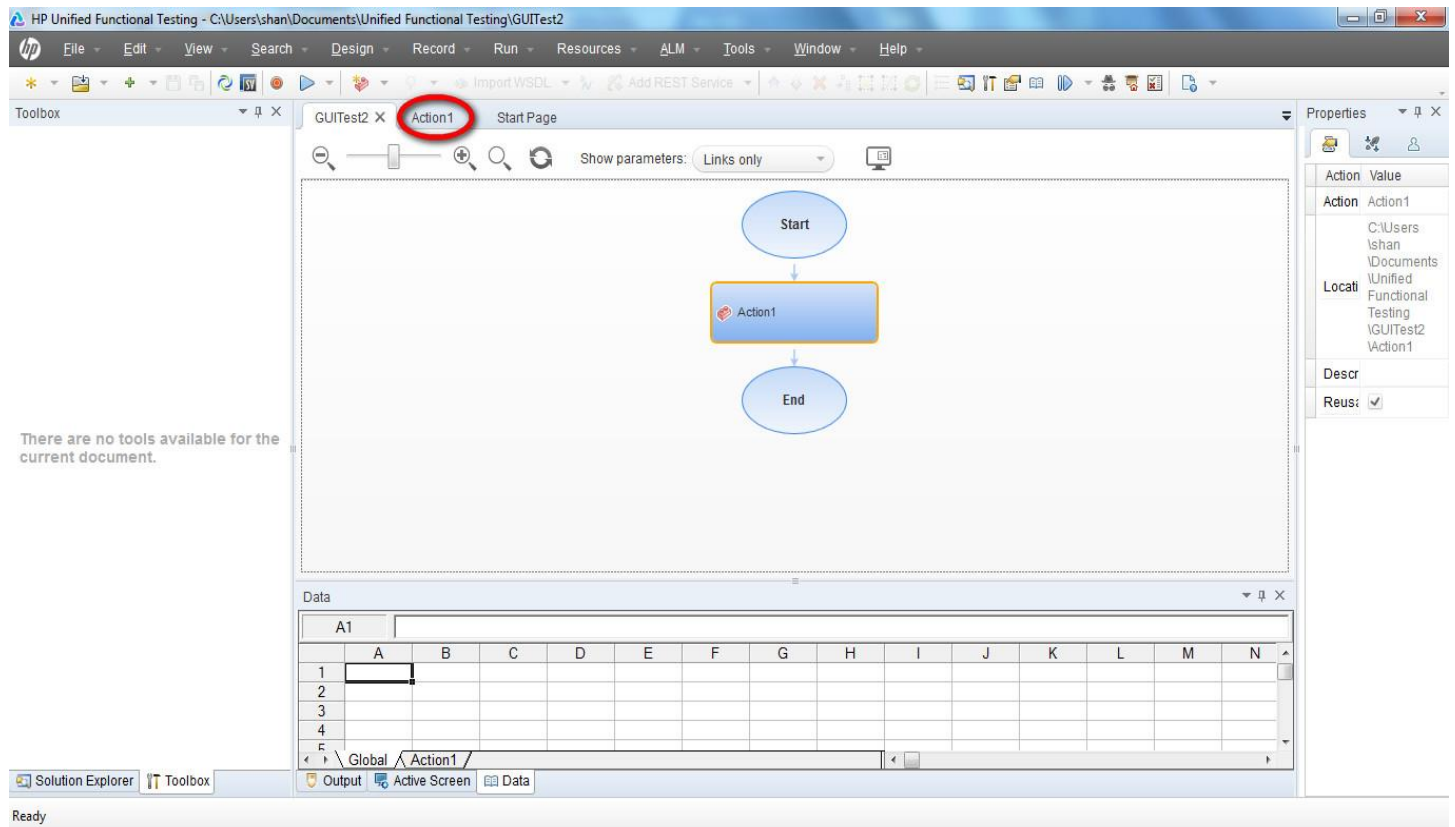
**Step 1:** Click on "New" test from the Start Page as shown below:



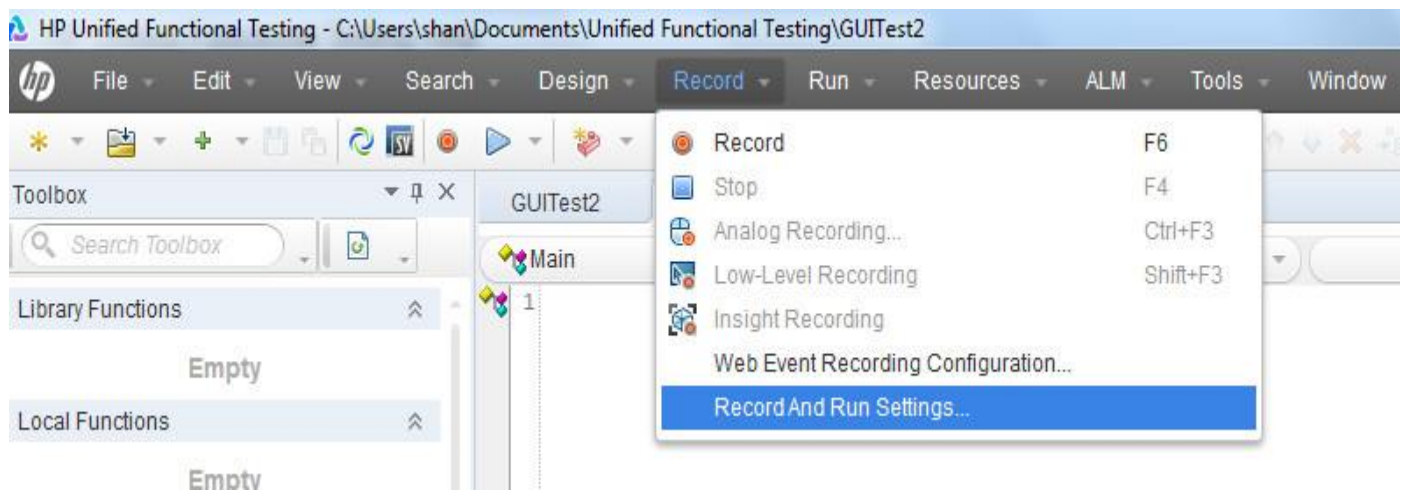
**Step 2:** Upon Clicking, "New" Link, the new test window opens and the user need to select the test type. Select "GUI Test", give a name for the test and also the location where it needs to be saved.



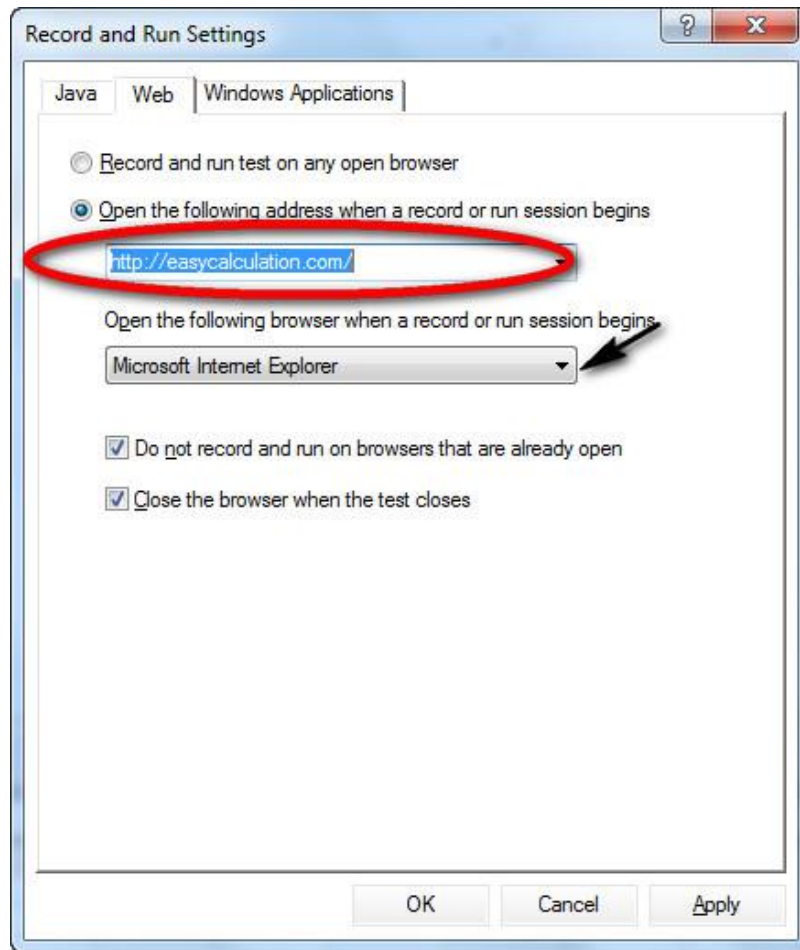
**Step 3.** Once a New test is created, the new test screen opens as shown below and click on "Action1" Tab which is created with 1 action by default.



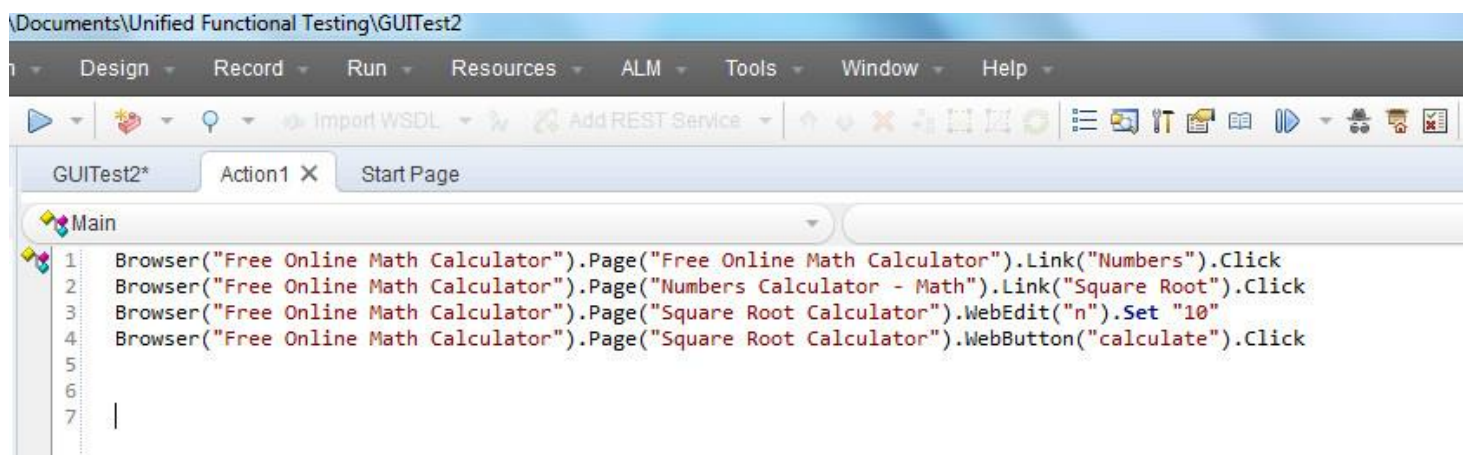
**Step 4.** Click on "Record" Menu and select "Record and Run Settings" as shown below:



**Step 5.** The Record and Run Settings Dialog opens and based on the type of application, one can select i.e Web, Java, Windows Applications. For Example, We will record a Web Based Application (<http://easycalculation.com/>)



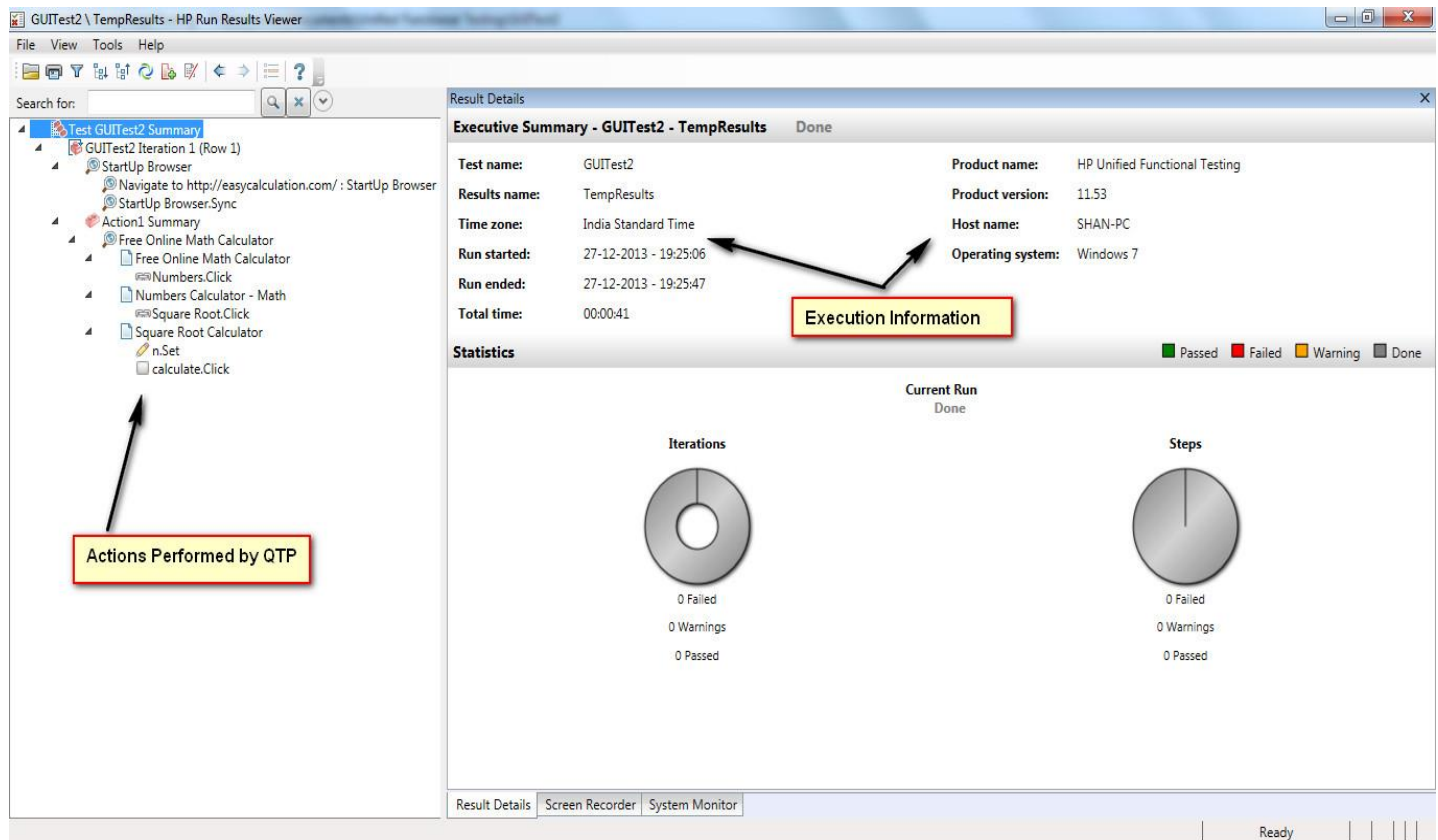
**Step 6.** Click Record Button, the Internet Explorer opens automatically with the web address <http://easycalculation.com/> as per the settings. Click "Numbers" link under "Algebra" and key in a number and hit "calculate". Upon completion of the action click "Stop" button in the record panel. You will notice that the script is generated as shown below:



**Step 7.** Now playback the script by clicking on the playback button. The Script replays and result is displayed.



**Step 8.** The result window is opened by default which exactly shows the timestamp of execution, pass and failed steps.



## Significance of Record and Playback:

1. It is used as the preliminary investigation method to verify if UFT can support the technology/application.



2. Used to create a test a basic functionality of an application or feature that does not require long-term maintenance.
3. It can be used for recording both mouse movements and keyboard inputs.

## Modes of Recording: (Interview Question)

1. **Normal Recording** : This is the default Recording mode that records the objects and the operations performed on the application under test.
2. **Analog Recording** : This records not only the keyboard actions but also the mouse movements relative to the screen or the application window.
3. **Low-Level Recording** : This records the exact co-ordinates of the objects independent of the fact whether UFT recognizes the object or NOT. It just records the co-ordinates, hence does NOT record mouse movements.
4. **Insight Recording** : UFT records operation based on its appearance and NOT based on its native properties.

## Object Repository:

Object Repository is a collection of object and properties with which QTP will be able to recognize the objects and act on it. When a user records a test, the objects and its properties are captured by default. Without understanding objects and its properties, QTP will NOT be able to play back the scripts.

Click on each one of the below topics to know more about Object Repository and its associated features.

Topic	Description
Object Spy and its Features	To Understand the usage of object Spy and its associated functionalities.
Working with Object Repository	Adding,Editing, Deleting Objects from a Object Repository and its associated functionalities.
Types of Object Repository	Deals with Shared Object and Local Object Repository and their context with respect to scripting.
User-defined Objects	Deals with the circumstances to use the User-Defined Objects.
Object Repository in XML	Deals with coverting OR's to XML and how to use the object Repository as XML.
Comparing and Merging OR	Operations such as Compare OR', Merge OR's to effectively work with Object Repository.
Ordinal Identifiers	Circumstances when the ordinal identifiers are used and their advantages.
Child Objects	Using Child Objects for effective scripting

## Actions:

Actions helps testers to divide scripts into groups of QTP statements called actions. Actions are similar to functions in VBScript, however there are few differences. By Default QTP creates a test with 1 action.

Actions	Functions
Actions are inbuilt feature of QTP.	VBScript Functions are supported by both VBScript and QTP.
Actions parameters are passed byvalue only.	Function parameters are passed either by byvalue or byref.
Actions have extension .mts	Functions are saved as .vbs or .qfl
Actions may or maynot be reusable.	Functions are always reusable.

The properties of the action can be accessed by Right Clicking on the Script Editor Window and Selecting "Properties".

## Action properties contains following information

- Action Name
- Location
- Reusable Flag
- Input Parameters
- Output Parameters

## Types of Actions: (Interview Question)

There are **three** types of actions:

- **Non-reusable action** - An action that can be called only in that specific test in which it has been designed and can be called only once
- **Reusable action** - An action that can be called multiple times any test in which it resides and can also be used by any other tests
- **External Reusable action** - It is a reusable action stored in another test. External actions are read-only in the calling test, but it can be used locally with the editable copy of the Data Table information for the external action

## Working with Actions:

There are three options to insert an action. Click on each one of those to know more about the selected type of action.

Action Type	Description
-------------	-------------

<b>Insert Call to New Action</b>	Inserts a New Action from the existing action
<b>Insert Call to Copy of Action</b>	Inserts a copy of an existing action
<b>Insert Call to Existing Action</b>	Inserts a call to existing re-usable action

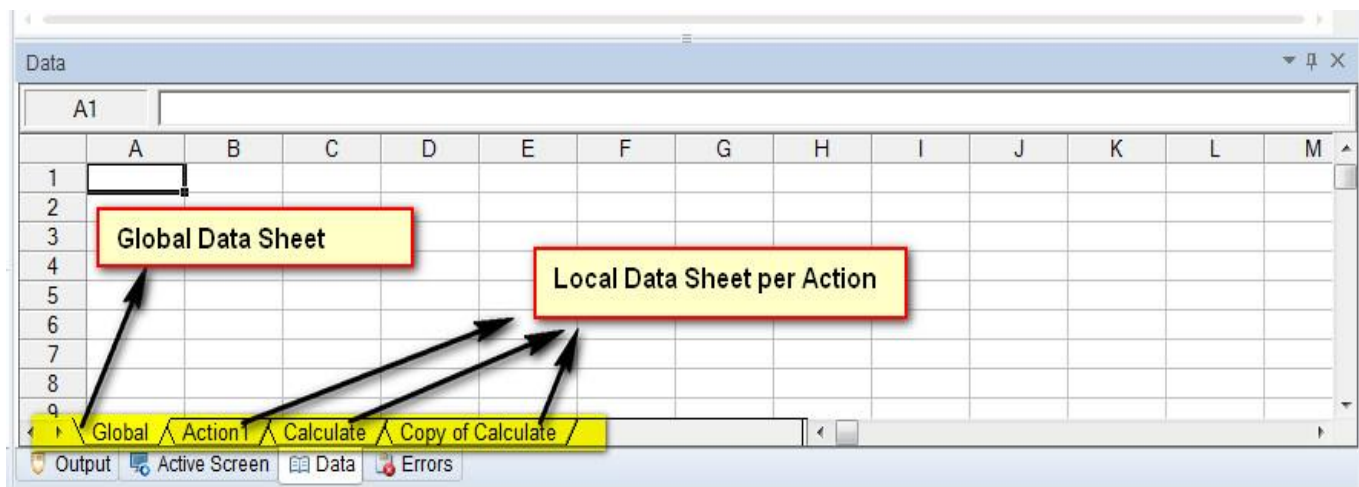
## Data Tables:

A DataTable, similar to Microsoft Excel helps testers to create data driven test cases that can be used to run an Action multiple times.

**There are two types of Data Tables. (Interview Question)**

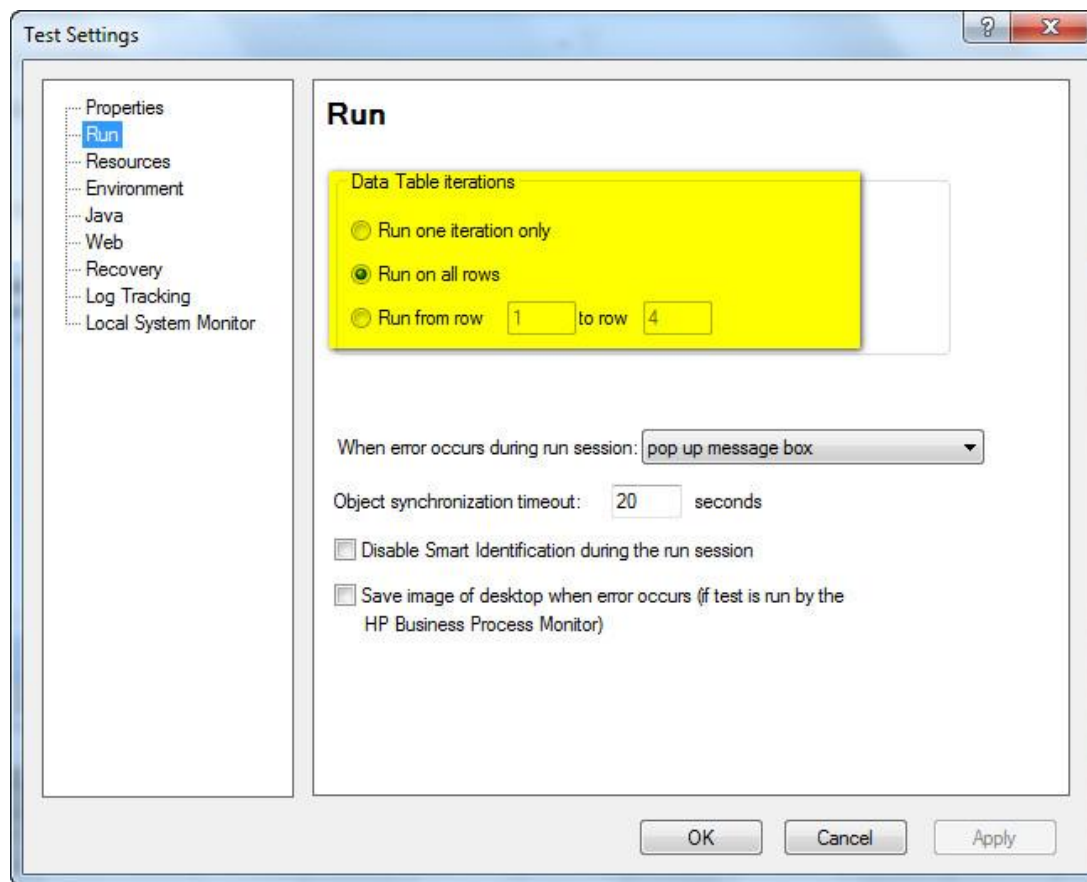
- **Local Data Table** - Each action has its own private data table also known as local data table which is can also be accessed across actions.
- **Global Data Table** - Each test has one global data sheet that is accessible across actions.

The Data sheet can be accessed from the "Data" Tab of QTP as shown below:



To execute a test case for some specified number of iterations, one can set the iterations of global data table in the Test Settings dialog, that can be accessed using File -> Settings -> Run(Tab) as shown below:





## Example:

For Instance, if the user wants to parameterize "compound Interest" of "http://easycalculation.com/" that can be accessed using "http://easycalculation.com/compound-interest.php". The Parameters can be created as shown below. Most of the functionalities of Excel can be used in Data table as well.

Data			
	A3	1322	
	Principal	Rate	Time
1	232	3.26	4
2	2556	7%	5
3	1322	6.50%	6
4	32.21	3.32%	4.4
5			
Global / Action1 / Calculate			

# Data Table Operations:

There are **three** types of objects to access Data Table. Data Table Operations can be well understood by traversing through the below link:

Object Type	Description
Data Table Methods	Gives Detailed information about the data table methods.
DTPParameter Object Methods	Gives Detailed information about the DTPParameter methods.
DTSheet Object Methods	Gives Detailed information about the DTSheet methods.

# What are CheckPoints? (Interview Question)

Checkpoints, as the name says it all, it refers to a validation point that compares the current value for specified properties or current state of an object with the expected value which can be inserted at any point of time in the script.

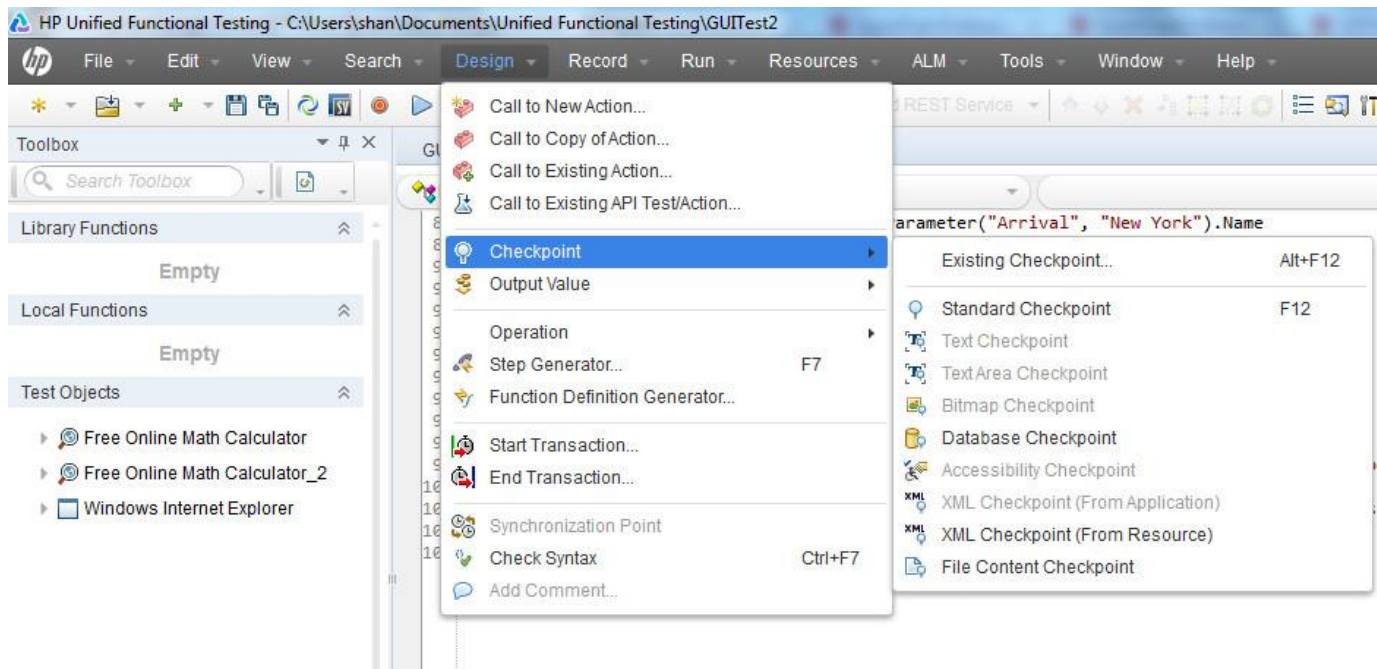
## Types:

Type	Description
Standard Checkpoint	Verifies the property values of an object in application under test and supported by all add-in environments.
Bitmap Checkpoint	Verifies an area of your application as a bitmap
File Content Checkpoint	Verifies the text in a dynamically generated or accessed file such as .txt,.pdf
Table Checkpoint	Verifies the information within a table. Not all environments are supported.
Text Checkpoint	Verify if the text that is displayed within a defined area in a Windows-based application, according to specified criteria.
Text Area Checkpoint	Verifies if the text string is displayed within a defined area in a Windows-based application, according to specified criteria.
Accessibility Checkpoint	Verifies the page and reports the areas of the Web site that may not conform to the World Wide Web Consortium (W3C) Web Content Accessibility Guidelines
Page Checkpoint	Verifies the characteristics of a Web page. It can also check for broken links.
Database Checkpoint	Verifies the contents of a database accessed by the application under test.
XML Checkpoint	Verifies the content of the .xml documents or .xml documents in Web pages and frames.

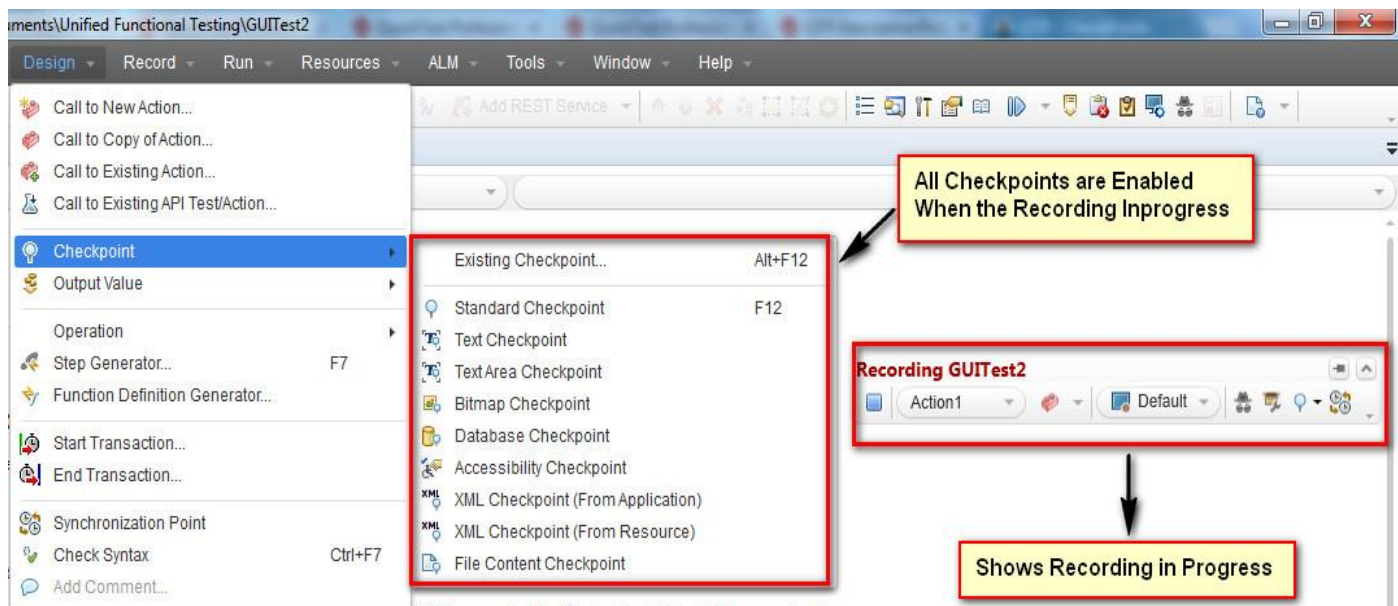
# Inserting CheckPoint

When the user wants to insert a checkpoint, one has to ensure that most of the checkpoints are supported during the recording sessions only. Once the user stops recording, checkpoints are NOT enabled.

Below is the checkpoint menu, when the user is **NOT in the recording mode**.



Below is the checkpoint menu, when the user **is in the recording mode**.



## What is Synchronization? (Interview Question)

Synchronization point is the time interface between Tool and Application under test. Synchronization point is a feature to specify delay time between one step and another of the test script.

For Example, clicking on a link may load the page in 1 second, sometimes 5 seconds or even it might take 10 seconds to load it completely. It depends on various factors such as the application server response time, network bandwidth, client system capabilities etc.

If the time is varying then the script will fail unless the tester handles these time differences intelligently.

## Ways to Insert Sync Point: (Interview Question)

- WaitProperty
- Exist
- Wait
- Sync(only for web based apps)
- Inserting QTP Inbuilt Synchronization points.

## Smart Identification: (Interview Question)

**\*\*This is a completely useless [and sometimes dangerous] feature of UFT/QTP that should never be used! What happens is this:**

- You tell UFT to find an object labeled "x1"
- UFT is not able to find "x1" for some reason
- When Smart Identification is enabled, UFT will look for "x1", and when it's not found, it will find **another** object that has similar properties and say that it found the object you were initially looking for. (It will find "x2" and say that its properties match closely with "x1")
- As a Test Engineer, if you want to perform an action on "x1" you need UFT to tell you if "x1" is found or not because that may be a defect. You don't want UFT to find the next closest object and use that!

Sometimes, QTP is unable to find any object that matches the recognized object description or it may find more than one object that fits the description, then QTP ignores the recognized description and uses the Smart Identification mechanism to recognize the object.

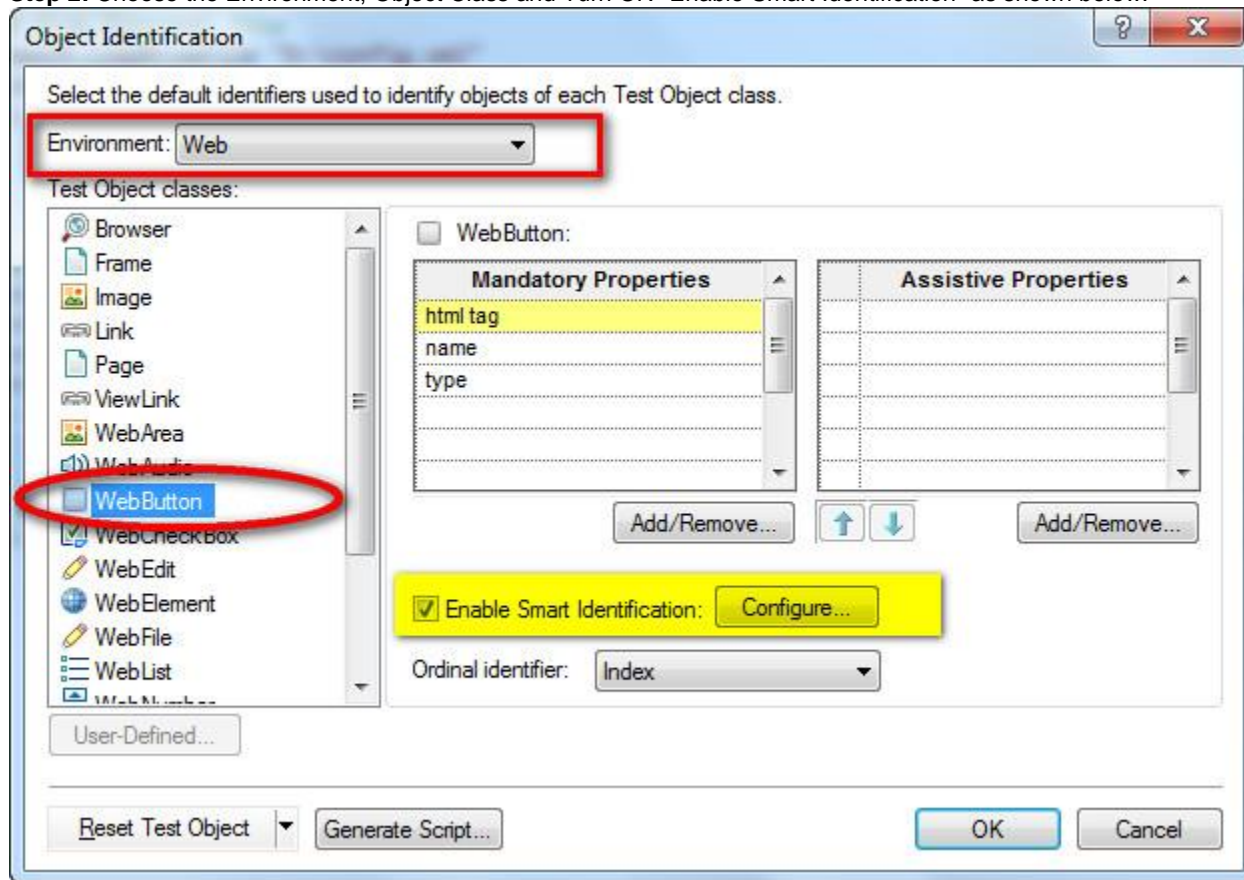
QTP's Smart Identification uses **two types of properties**: (Interview Question)

- **Base Filter Properties** - The basic properties of a particular test object class whose values cannot be changed without changing the essence of the original object.
- **Optional Filter Properties** - Other properties also assist in identifying the objects of a particular class whose properties are unlikely to change often but can be ignored if they are no longer applicable.

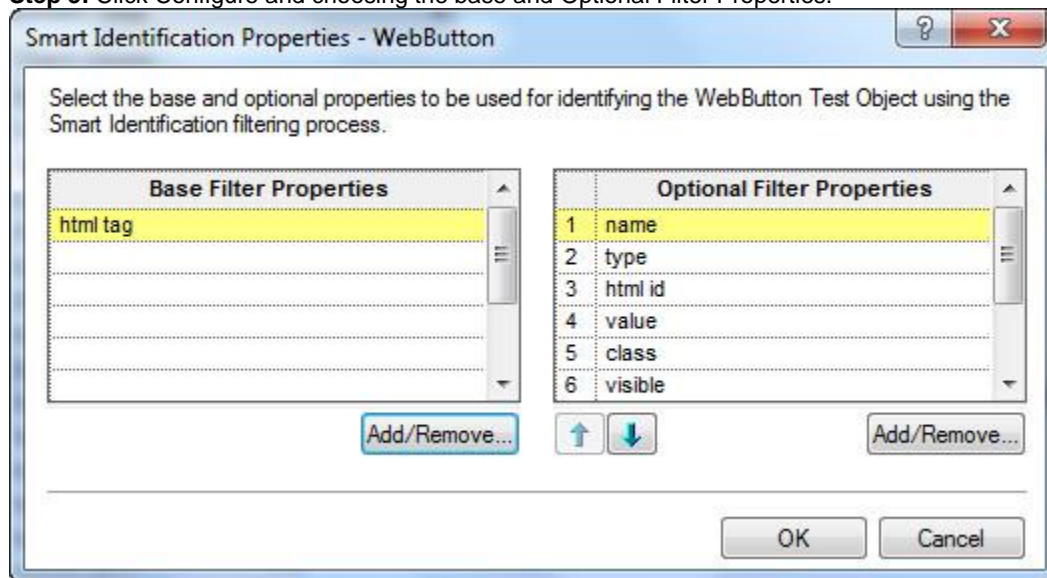
## Enabling Smart identification for an object:

**Step 1:** Navigate to "Tools" -> "Object Identification". Object Identification Dialog Opens.

**Step 2:** Choose the Environment, Object Class and Turn ON "Enable Smart Identification" as shown below:

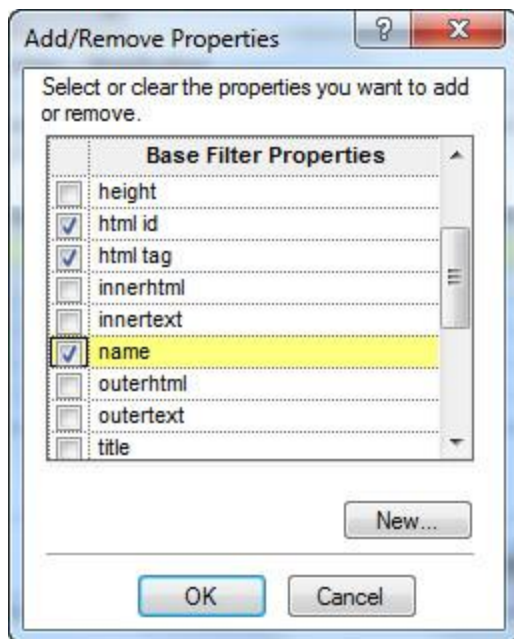


**Step 3:** Click Configure and choosing the base and Optional Filter Properties.

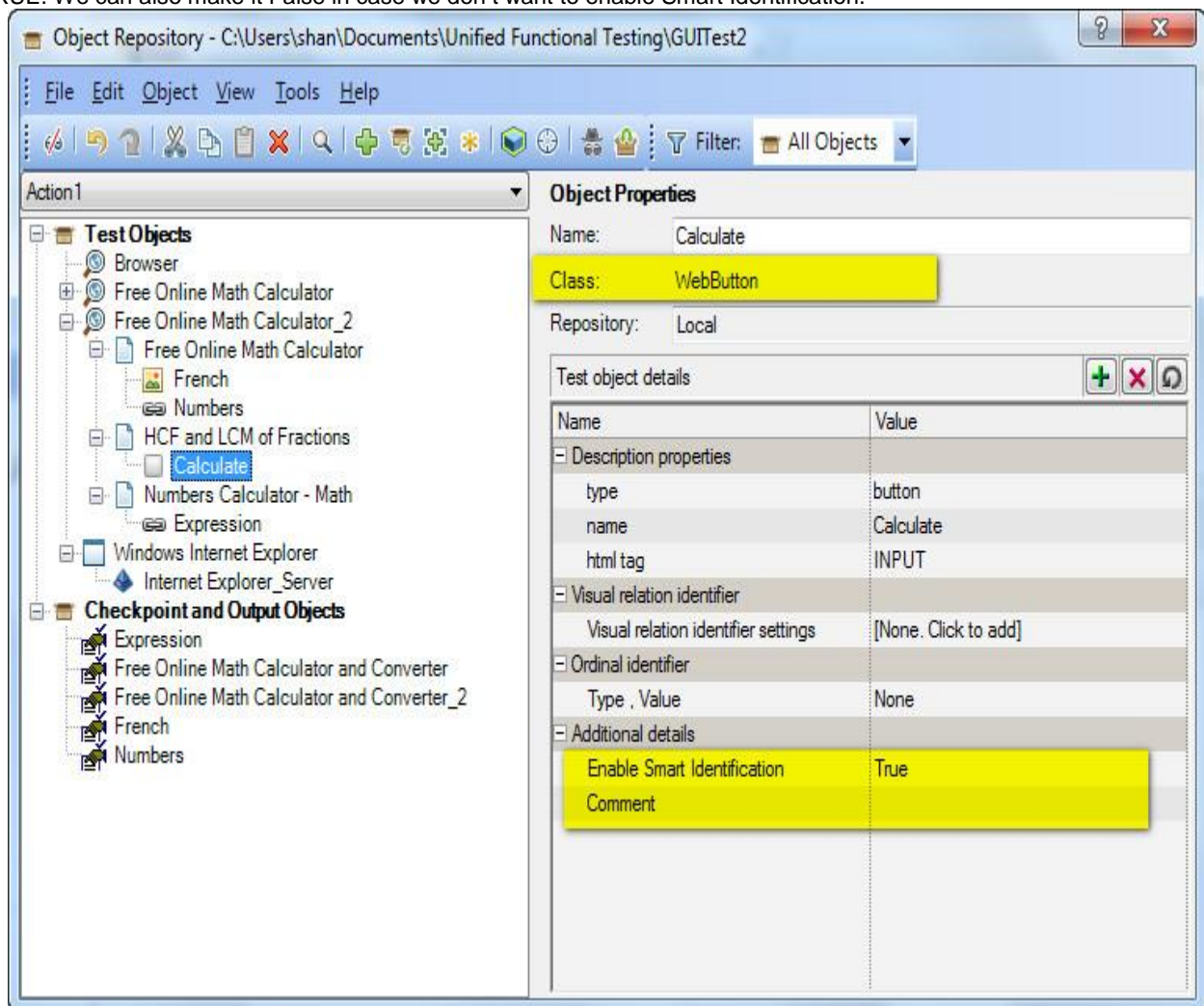


**Step 4:** Add Properties in Base Properties apart from the Default one and also add/remove Optional Filter Properties. Please Note that Same properties cannot be part of both Mandatory and Assistive Properties and click "OK".

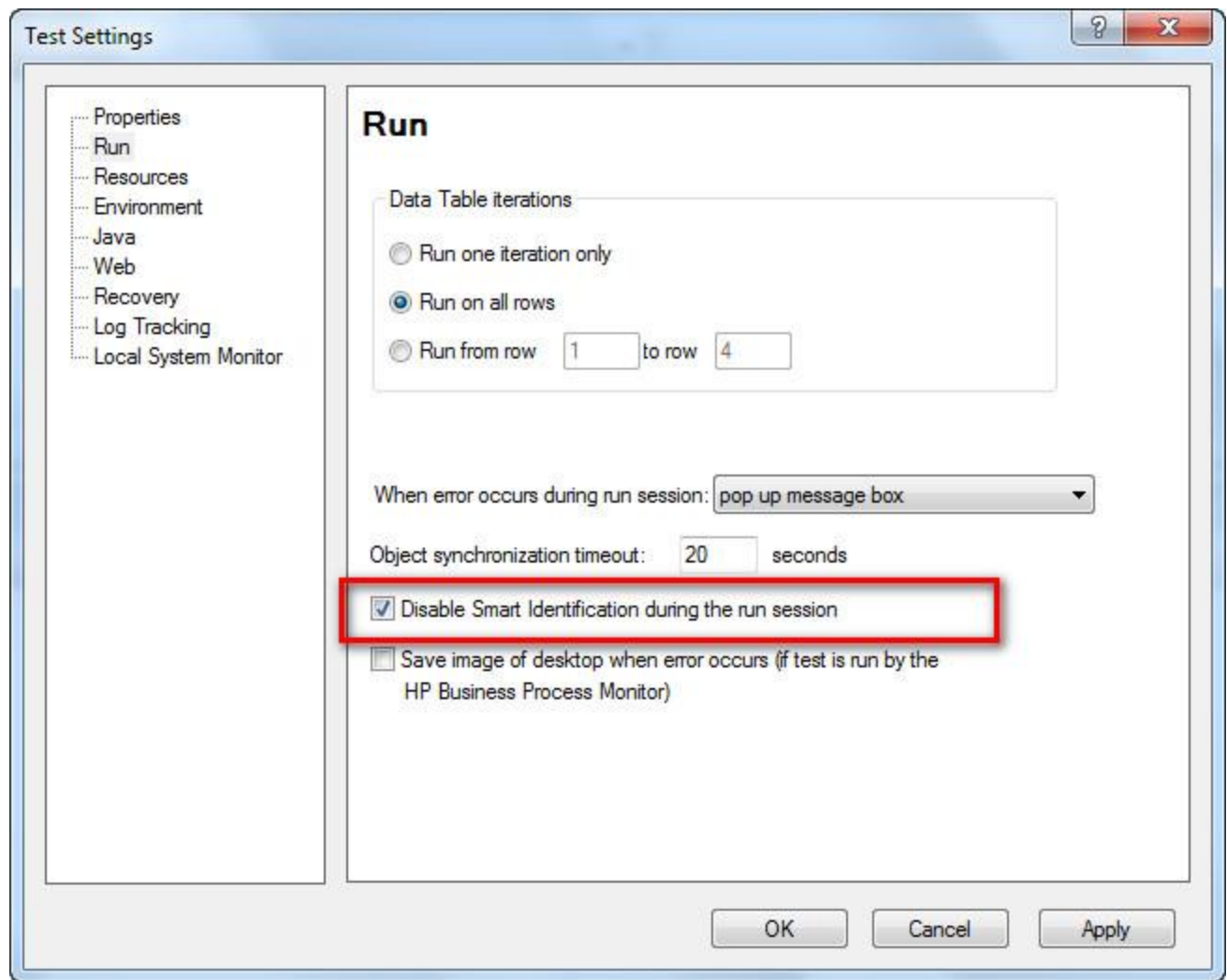




**Step 5:** Verifying if the Smart Identification is enabled after Adding object of that type in the Object Repository. Smart Identification is set to TRUE. We can also make it False in case we don't want to enable Smart Identification.



**Step 6:** We can even disable a test Level by applying at test script level under "Settings" of "File" Menu as shown below:



**Step 7:** If the Smart Identification is disabled as per Step# 6 then it won't apply smart identification for any object during script execution.

**Step 8:** In case objects are added with Smart Identification as Off, QTP won't use Smart Identification for recognizing in future, even though we have enabled it afterwards.

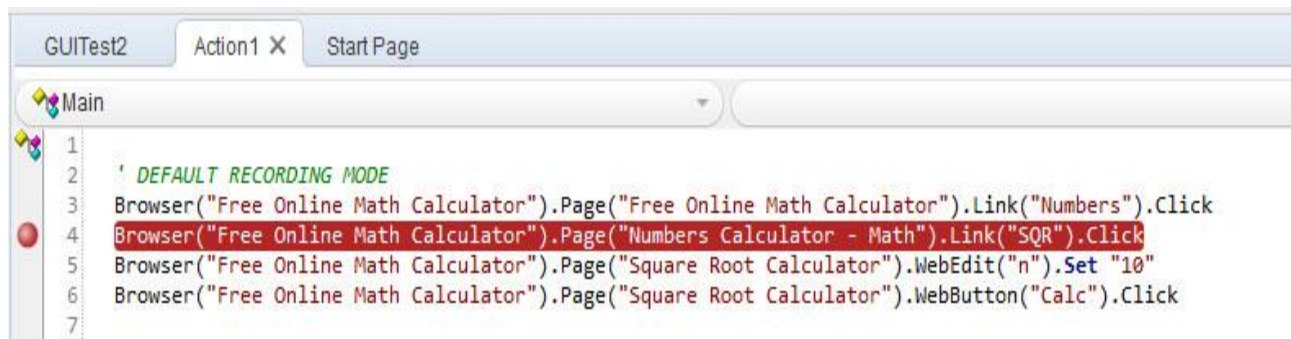
**Debugging:** (*your VBScript, not the application*)

Debugging, in automation testing context, is a systematic process of spotting and fixing the coding issues in the automation scripts so that the script will be more robust and can spot the defects in the application.

There are various ways to perform debugging using break points in QTP.

Break Points can be inserted just by pressing "F9" or by using the Menu option "Run" -> "Inserting/Removing Break Point".

After Inserting the Break point the Red Colored Dot and the line will be highlighted in RED as shown below:



Method	Short Cut	Description
Step Into	F11	Used to execute each and every Step. Steps into the Function/Action and executes line by line. It pauses on each line after execution.
Step Over	F10	Used to Step over the Function. Step Over runs only the current step in the active document.
Step Out	Shift+F11	After Step Into the function, you can use the Step Out command. Step Out continues the run to the end of the function and then pauses the run session at the next line.

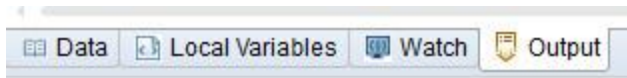
## Options in Break Point:

Various Options in Break Point can be accessed by Navigating 'Run' Menu.

Short Cut	Description
F9	Insert/Remove BreakPoint
Ctrl+F9	Enable/Disable BreakPoint
Ctrl+Shift+F9	Clear All BreakPoint
Use Only Menu	Enable/Disable All BreakPoints

## Debugging Pane:

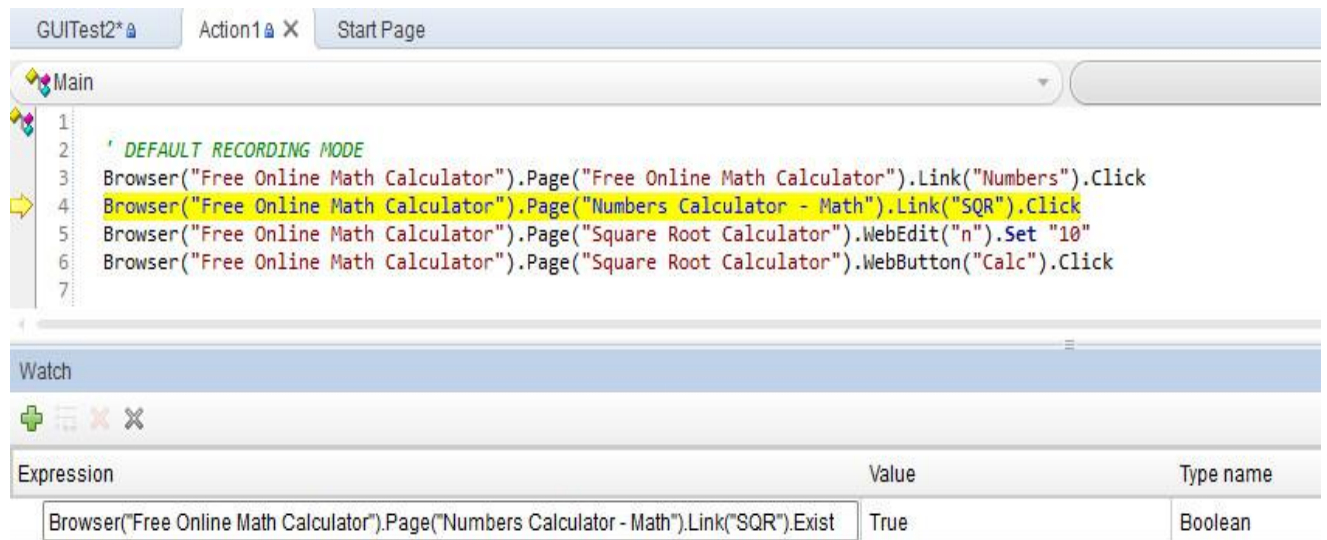
The Following are the panes in the debugging window:



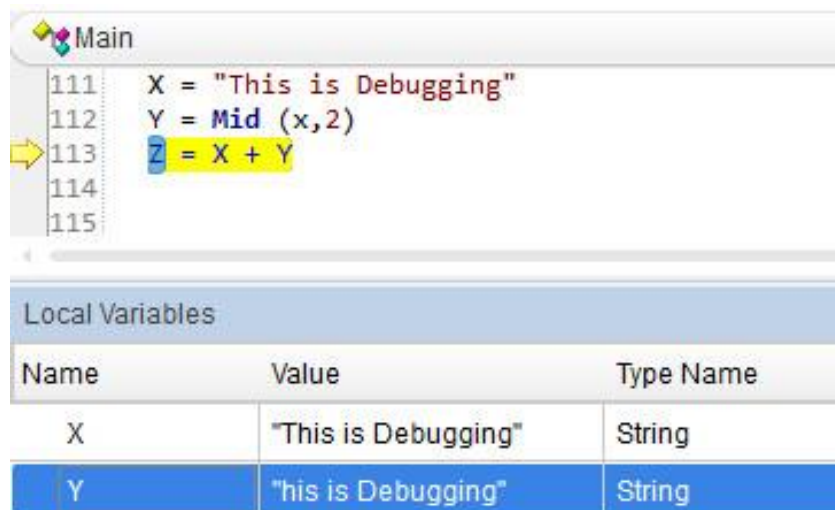
- **Output** - This Tab displays all the Output of the Print Statements.
- **Watch** - This Tab displays the Boolean output of the Given Expression.
- **LocalVariables** - This Tab displays the Output of the Local Variables.

## Example:

The Watch Pane shows the output expression as shown below:



The Local Variables Pane shows the values held by the local variables as shown below:



## What is Error Handling?

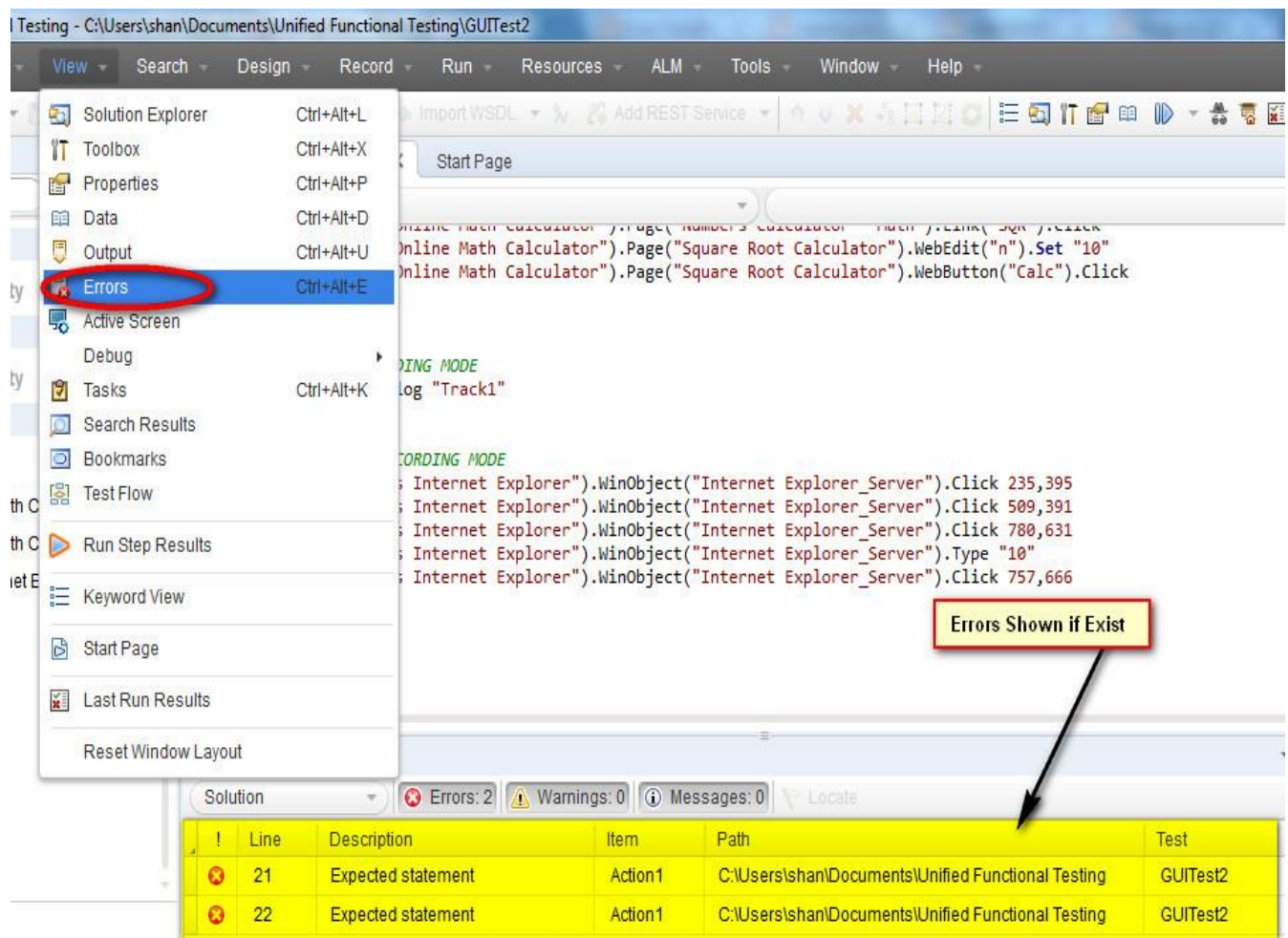
There are various ways on handling errors in QTP. There are three possible kinds of error type one would encounter while working with QTP.

- Syntax Errors
- Logical Errors
- Run Time Errors

## Error Types:

### SYNTAX ERRORS:

Syntax errors are the **typos** or a piece of the code that does not confirm with the VBScripting language grammar. Syntax errors occur at the time of compilation of code and cannot be executed until the errors are fixed. To verify the syntax one use the keyboard shortcut as Ctrl+F7 and the result is displayed as shown below. If the window is NOT displayed one can navigate to "View" -> "Errors".



### LOGICAL ERRORS:

If the **script is syntactically correct but it produces unexpected results**. Logical error usually does not interrupt the execution but produces incorrect results. Logical errors could occur due to variety of reasons, i.e. wrong assumptions or misunderstanding of the requirement and sometimes incorrect program logics(using do-while instead of do-Until) or Infinite Loops.

One of the ways to detect a logical error is to perform peer reviews and also verifying the QTP output file/result file to ensure the tool has performed what it has intended to do.



# RUNTIME ERRORS:

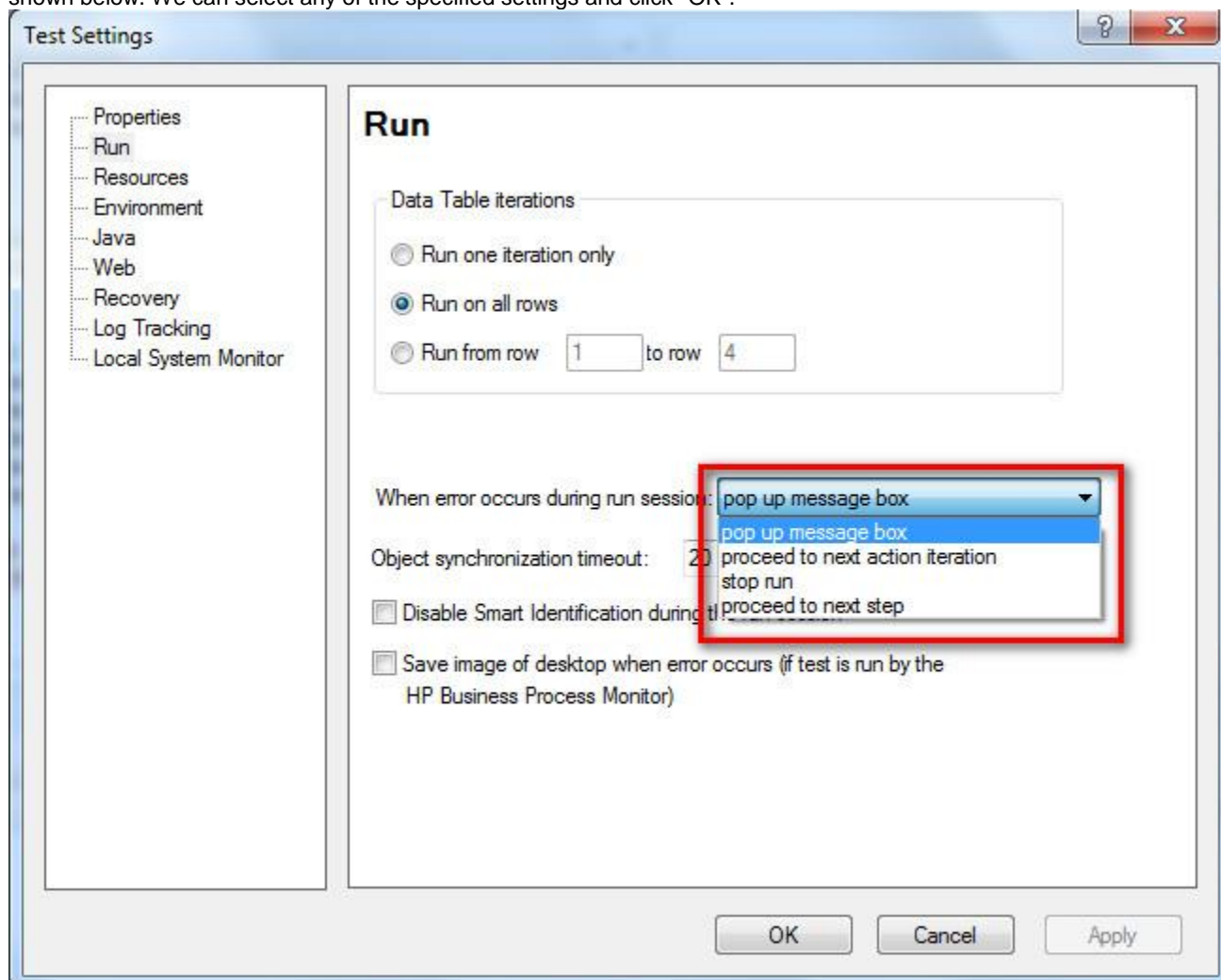
As The name states, this kind of **Error happens during Run Time**. The reason for such kind of errors is that the script trying to perform something but it is unable to do so and the script usually stops as it is unable to continue with the execution. Classic Examples for Run Time Errors are,

1. **File NOT found but the script trying to read the file.**
2. **Object NOT found but scripts is trying to act on that particular object.**
3. **Dividing a number by Zero.**
4. **Array Index out of bounds while accessing array elements.**

## Handling Run-Time Errors:

There are various ways to handle errors in the code.

**1. Using Test Settings** - Error handling can be defined the Test Settings by Navigating to "File" >> "Settings" >> "Run" Tab as shown below. We can select any of the specified settings and click "OK".



**2. Using On Error Statement** - On Error statement is used to notify the VBScript engine of intentions to handle the run-time errors by tester, rather than allowing the VBScript engine to display error messages that are not user friendly.

- On Error Resume Next - On Error Resume Next informs the VBScript engine to proceed executing the next line of code when an error is encountered.
- On error Goto 0 - This helps the testers to turn off the error handling.
-

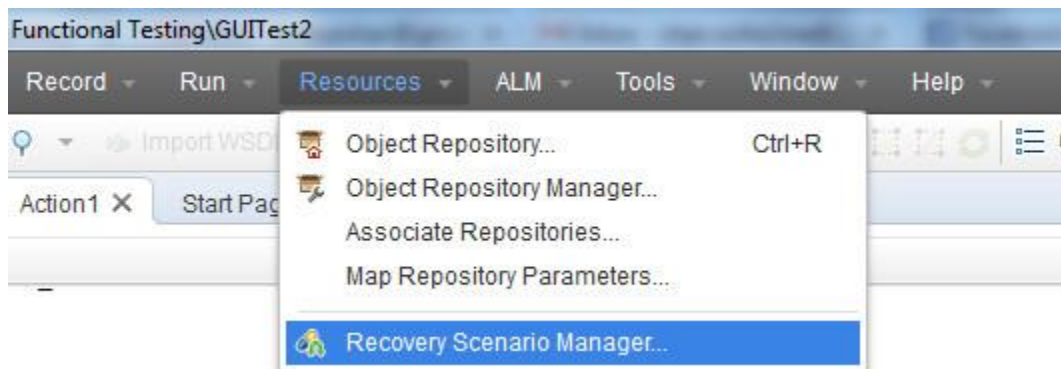


**3. Using Err Object** - Error object is an inbuilt object within VBScript that captures the run time error number and error description with which we will be able to debug the code easily.

- **Err.Number** - The Number property returns or Sets a numeric value specifying an error. If Err.Number value is 0 then No error had occurred.
- **Err.Description** - The Description property returns or sets a brief description about an error.
- **Err.Clear** - The Clear method resets the Err object and clears all the previous values associated with it.

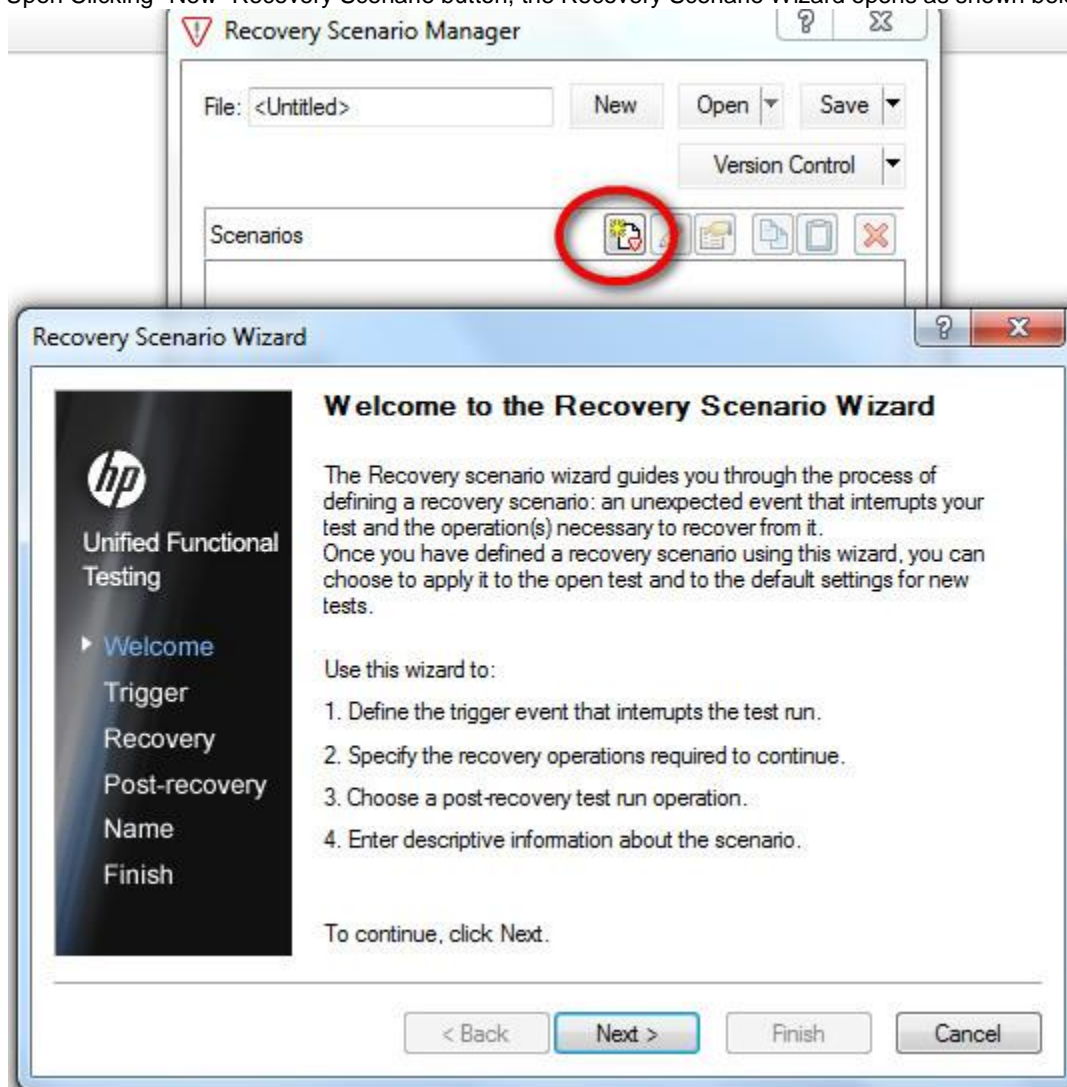
## Recovery Scenarios

While executing the QTP scripts, we might get some **unexpected errors**. In order to recover the test and continue executing the rest of the script from these unexpected errors, Recovery Scenarios are used. The Recovery Scenario Manager can be accessed by Navigating to "Resources" -> Recovery Scenario Manager as shown below:



## Steps to create Recovery Scenario:

**Step 1 :** Upon Clicking "New" Recovery Scenario button, the Recovery Scenario Wizard opens as shown below:

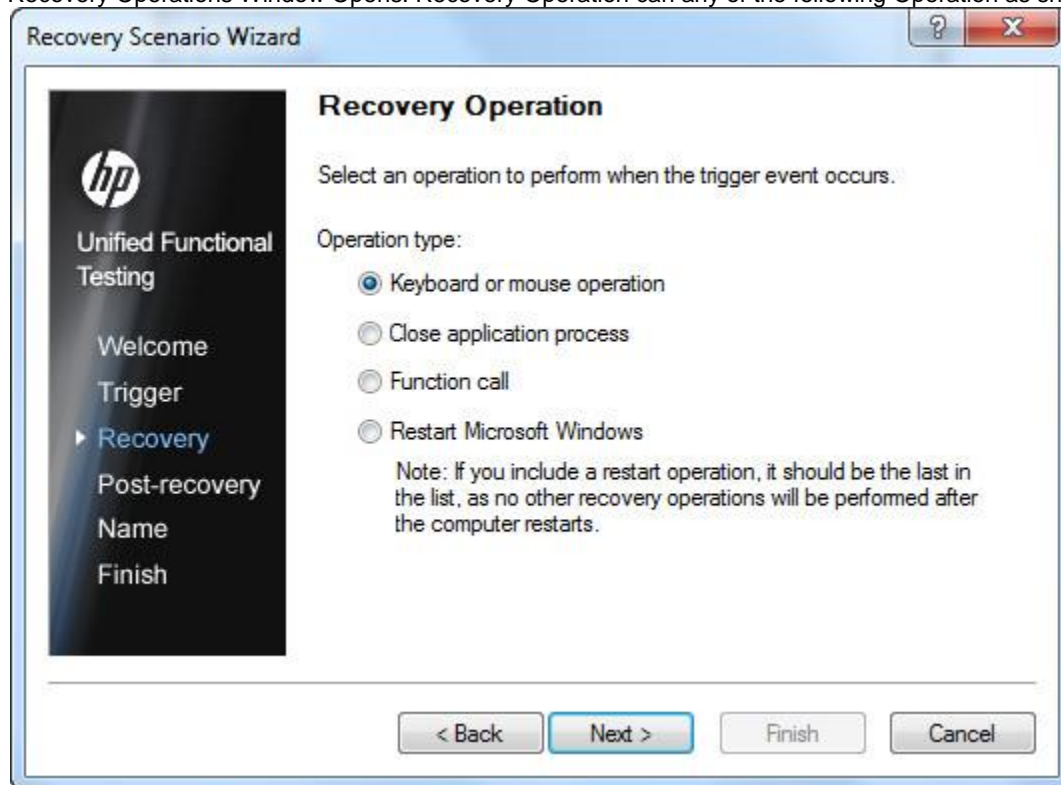


**Step 2 :** Next Step is to choose the Trigger Event. It corresponds to event which arises It can be any of the below four events.

- **Pop-Up Window**
- **Object State**
- **Test Run Error**

## Application Crash

**Step 3 :** Recovery Operations Window Opens. Recovery Operation can any of the following Operation as shown below:



**Step 4 :** Upon Specifying the appropriate Recovery Operation, we need to specify the Post Recovery Operation as well as shown below:

Recovery Scenario Wizard

hp  
Unified Functional Testing

Welcome  
Trigger  
Recovery  
► Post-recovery  
Name  
Finish

### Post-Recovery Test Run Options

Select the test run operation you want to perform when the recovery operation is complete.

Test run options:

- ☒ Repeat current step and continue
- ☐ Proceed to next step
- ☐ Proceed to next action or component iteration
- ☐ Proceed to next test iteration
- ☐ Restart current test run
- ☐ Stop the test run

< Back   Next >   Finish   Cancel

**Step 5 :** Upon Specifying the Post Recovery Operation, the recovery scenario should be named and added to the Test so that it will be activated.

Recovery Scenario Wizard

hp  
Unified Functional Testing

Welcome  
Trigger  
Recovery  
Post-recovery  
► Name  
Finish

### Name and Description

Provide a name and description for your recovery scenario.

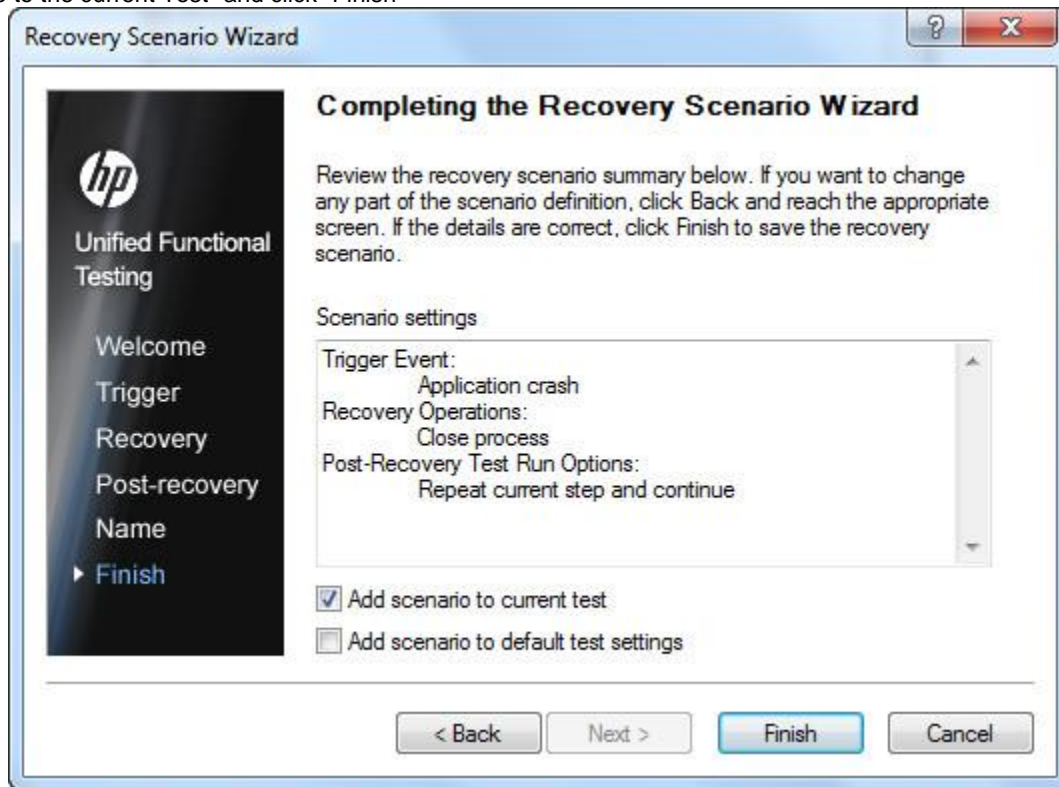
Recovery file: <New Recovery Scenario File>

Scenario name: Recover\_Application\_Crash

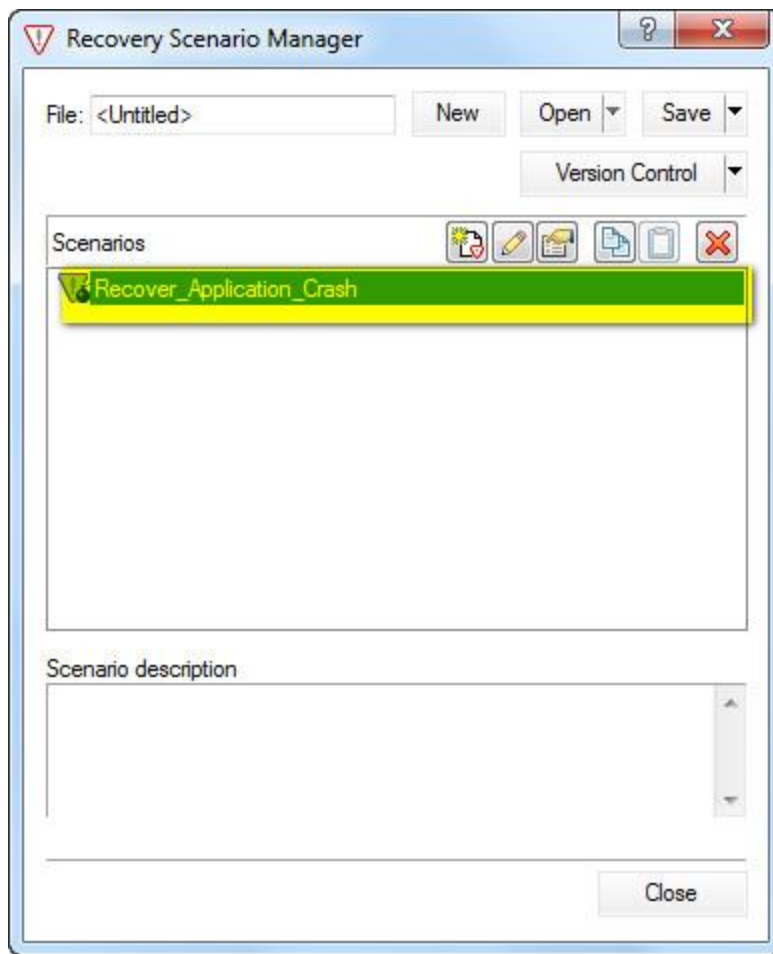
Description:

< Back   Next >   Finish   Cancel

**Step 6 :** The Recovery Scenario creation is complete and needs to be mapped to the current Test by checking the option "Add Scenario to the current Test" and click "Finish"

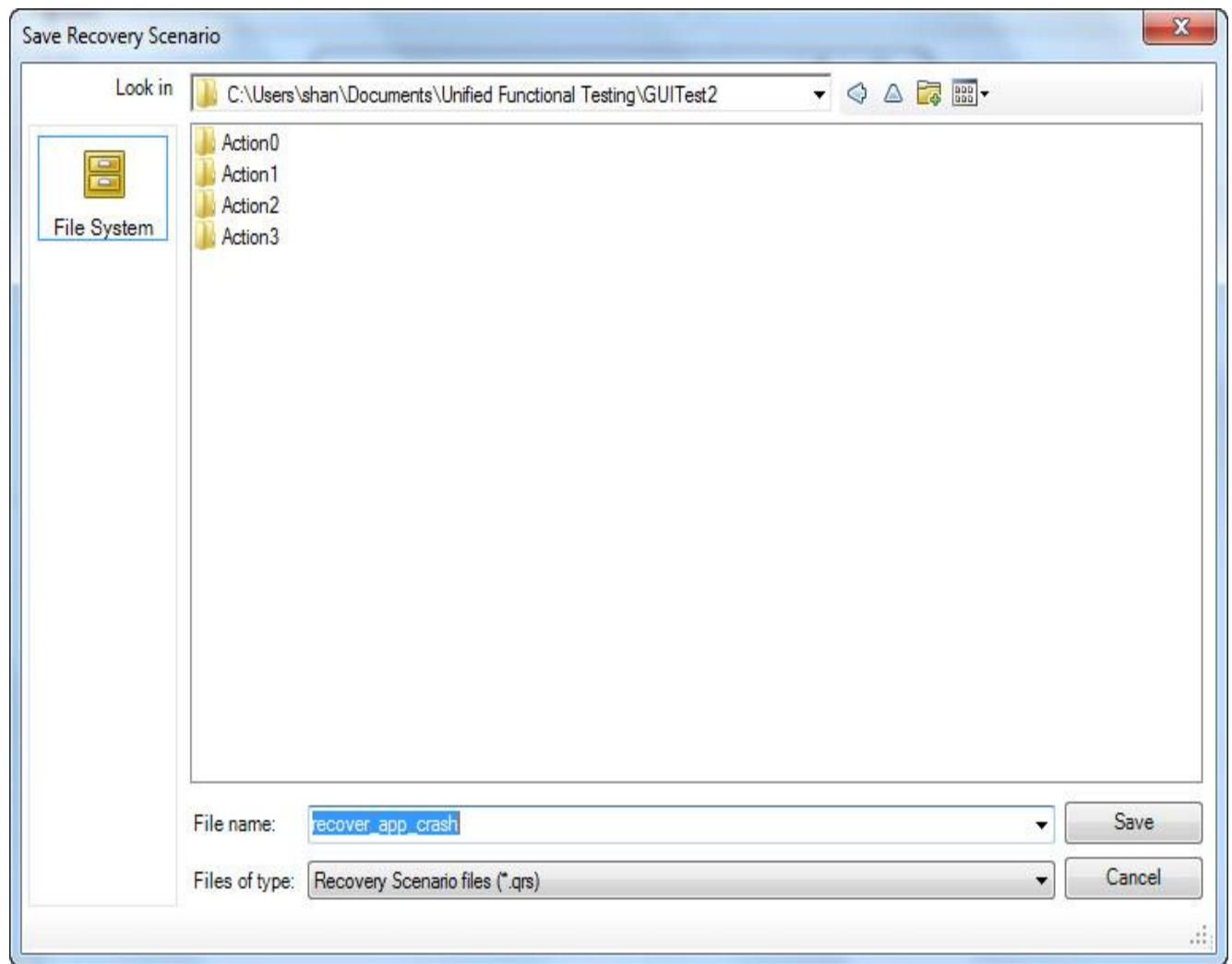


**Step 7 :** The Added Recovery Scenario will be as shown below and click on "Close" Button to continue.



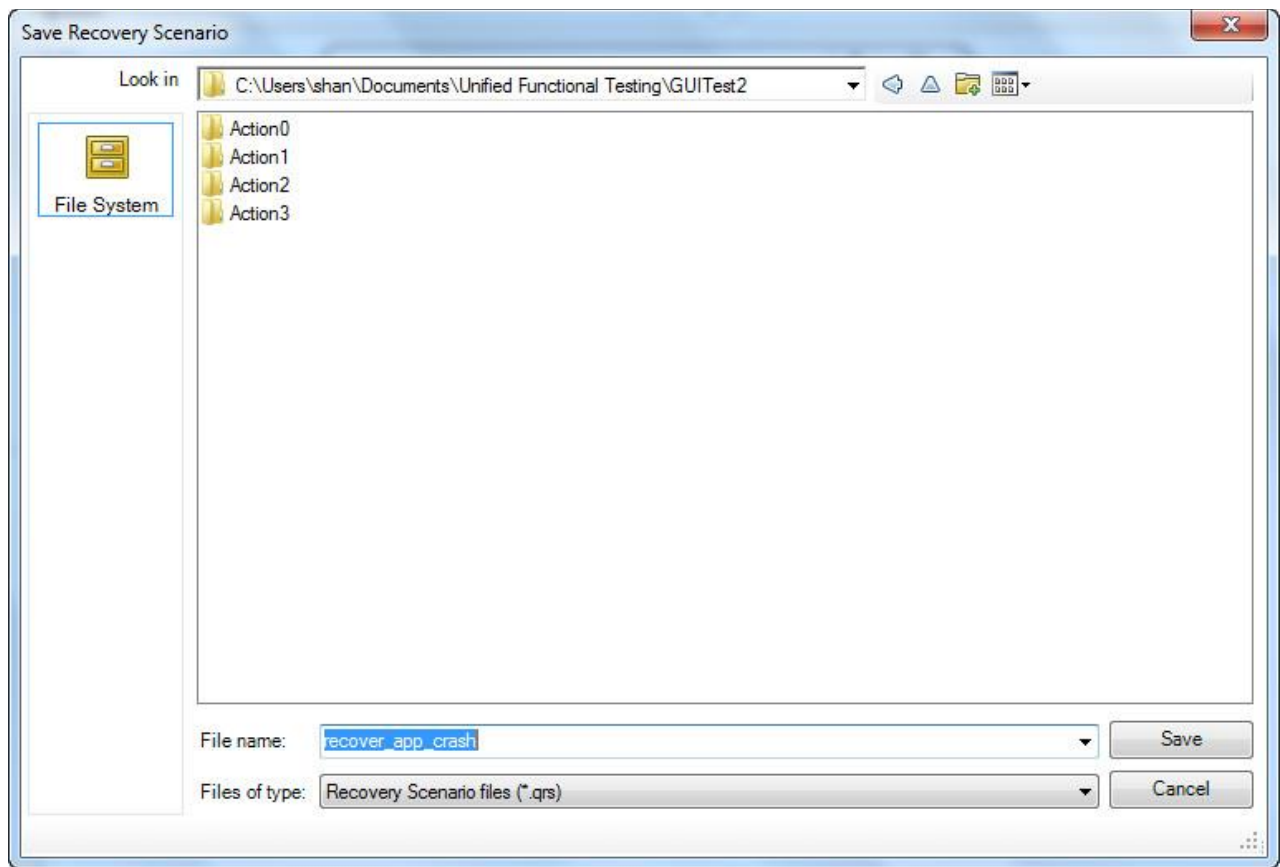
**Step 8 :** Upon Clicking on Close Button, QTP would Pop up user to Save the created Recovery Scenario. It will be saved with the extension .qrs and the wizard would be closed.





## Verification:

The Created Recovery Scenario should be part of the test now and can be verified by navigating to "File" -> "Settings" -> "Recovery" Tab.

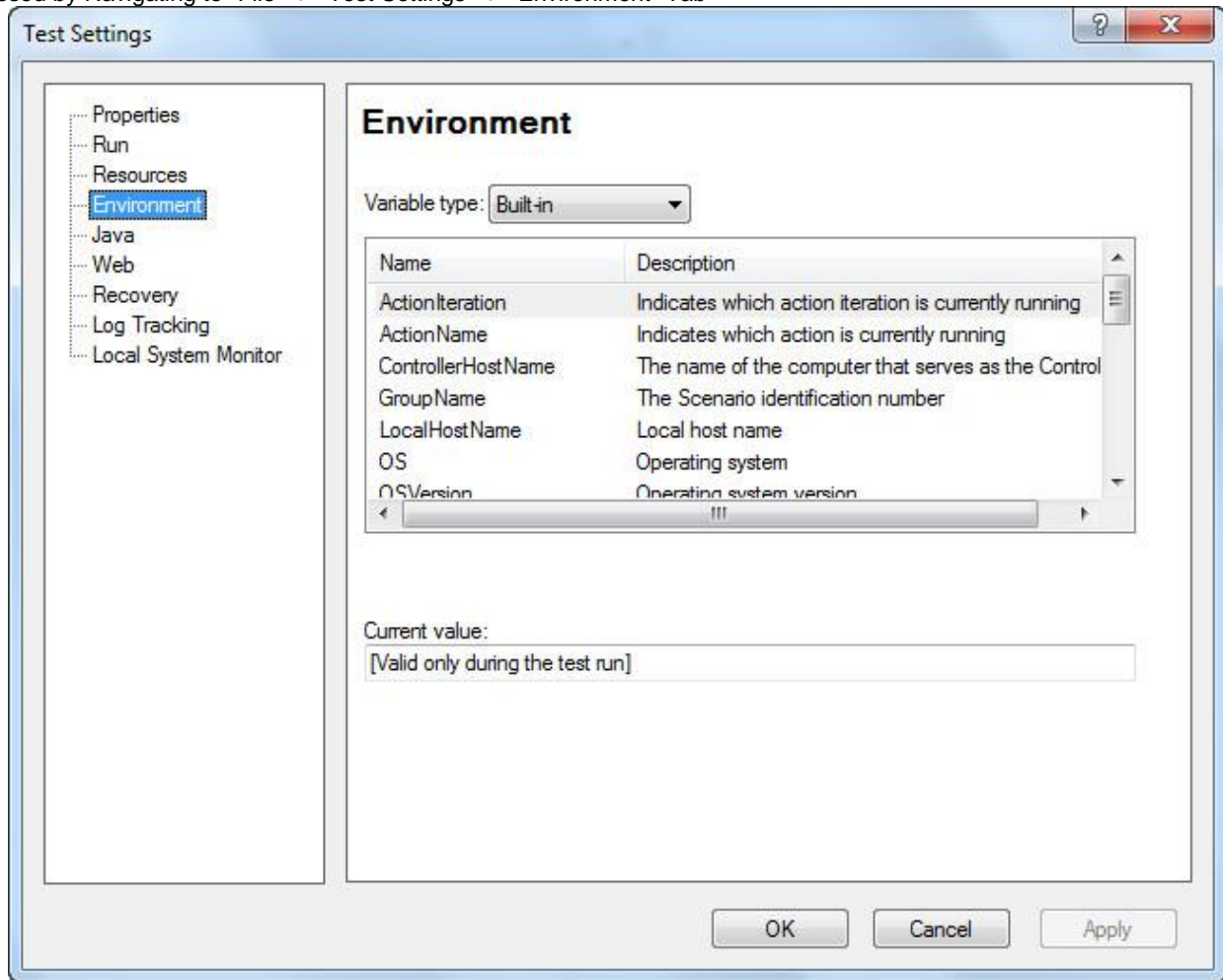


## Environment Variables

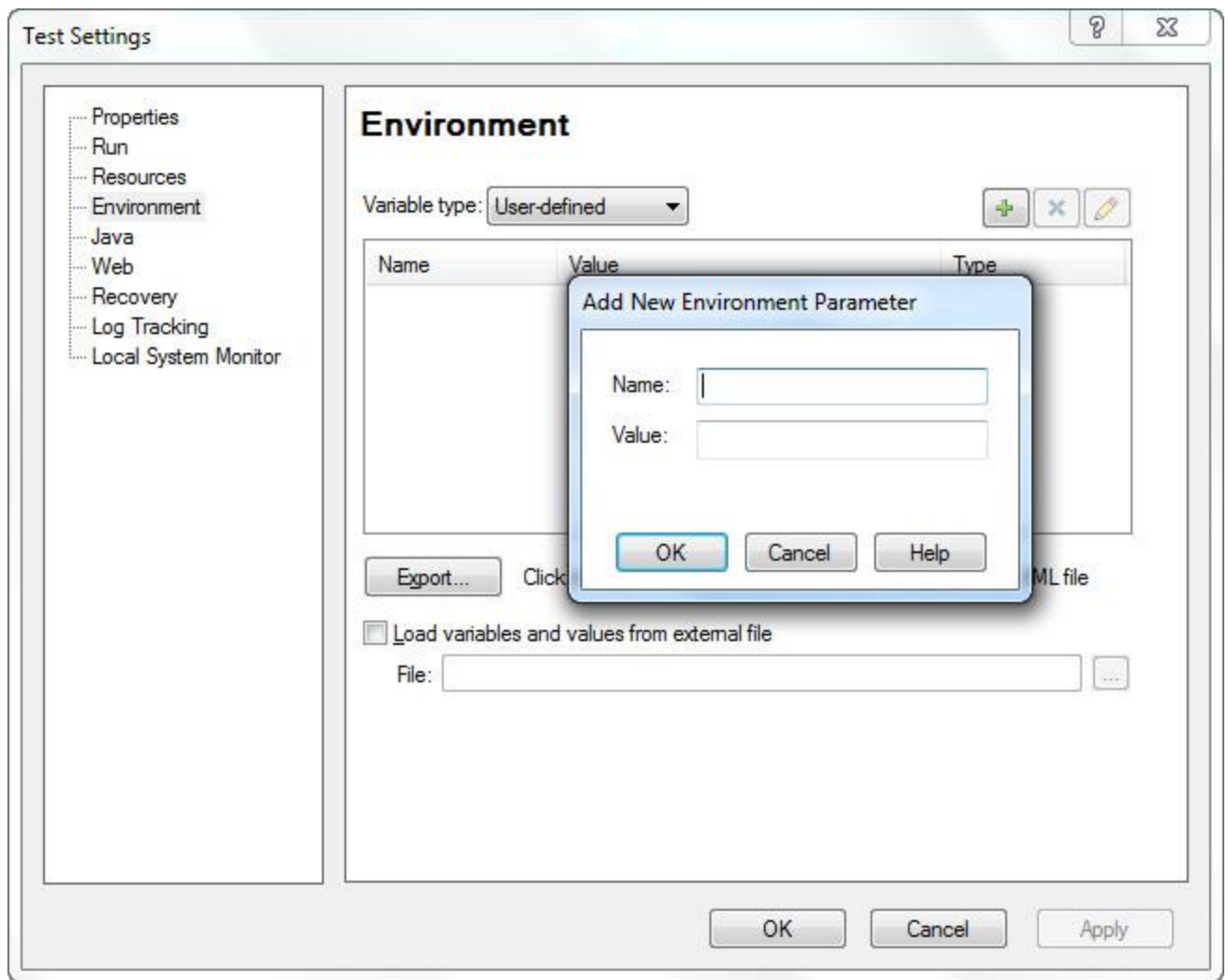
QTP environment variables are special types of variables that can be accessed by all actions, function libraries and recovery Scenarios. There are inbuilt environment variables for Windows that are available to all the applications running on that particular system where as QTP environment variables are only available to that test script during run-time.

### Types of Environment Variables

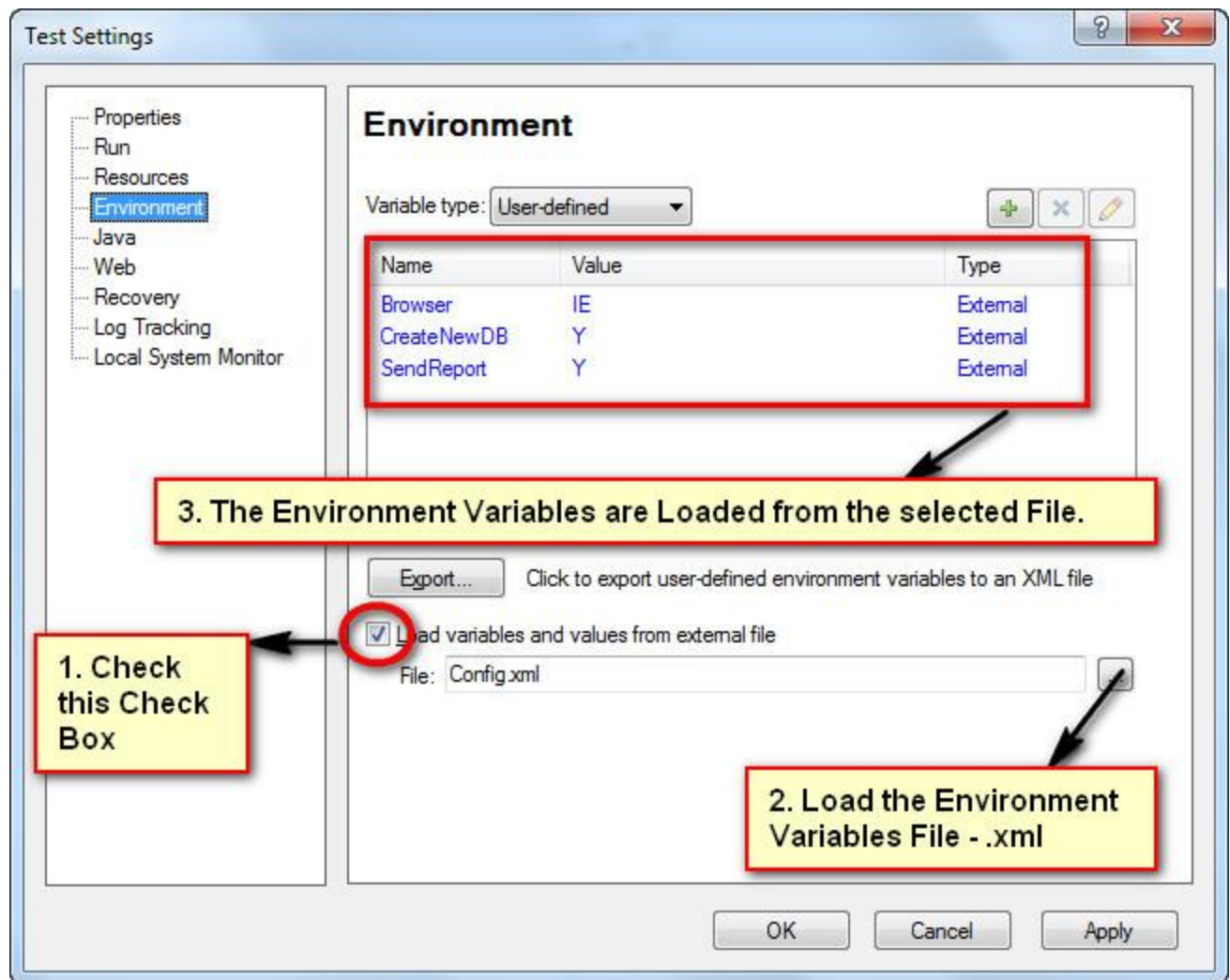
**Built-in Environment Variables** - provides a range of environment parameters that can provide information such as test name, action name, the test path, local host name, the operating system name, type and its version. The Environment Variable names can be accessed by Navigating to "File" -> "Test Settings" -> "Environment" Tab



**User defined Internal** - User Defined Variables can be saved by Selecting "User Defined" in the Environment Tab Window. The "+" button is Clicked to enter Parameter Name and Value as shown below:



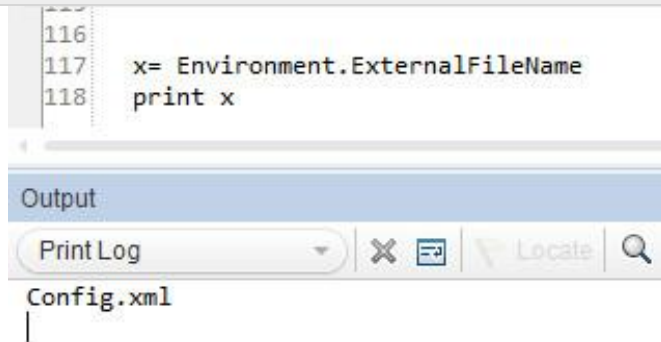
**User Defined External** - User Defined Variables can be stored in a external file as an .xml and can be loaded into the test as shown in the below figure or can also be loaded dynamically during run-time as explained below in one of the examples.



**Environment Variables - Supported Methods:**

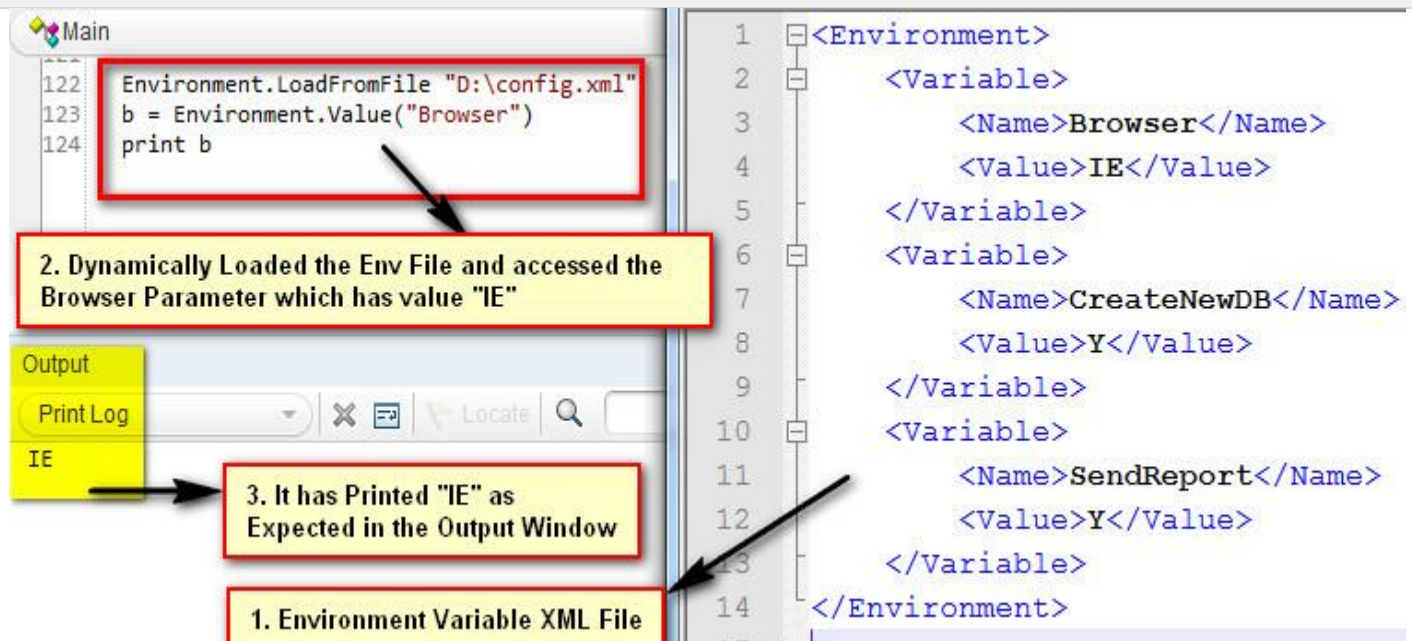
**1. ExternalFileName Property** - Returns the name of the loaded external environment variable file specified in the Environment tab of the Test Settings dialog box. If no external environment variable file is loaded, this property returns an empty string

```
x= Environment.ExternalFileName  
print x
```



**2. LoadFromFile Method** - Loads the specified environment variable file(.xml) dynamically during run time. When using this method, the environment variables need NOT be added manually into the Environment Tab.

```
Environment.LoadFromFile "D:\config.xml"  
b = Environment.Value("Browser")  
print b
```



**3. Value Property** - Retrieves the value of environment variables. We can also set the value of user-defined internal environment variables using this property.

```
' Get the Value of the InBuilt Environment Variables  
a = Environment.Value("OS")  
print a
```



```
b = Environment.Value("ActionName")
print b

'Loaded from External File
Environment.LoadFromFile "D:\config.xml"
c = Environment.Value("Browser")
print c
```

```
120 ' Get the Value of the InBuilt Environment Variables
121 a = Environment.Value("OS")
122 print a
123 b = Environment.Value("ActionName")
124 print b
125
126 'Loaded from External File
127 Environment.LoadFromFile "D:\config.xml"
128 c = Environment.Value("Browser")
129 print c|
```

Output

Print Log



Locate



Microsoft Windows 7 Workstation

Action1

IE

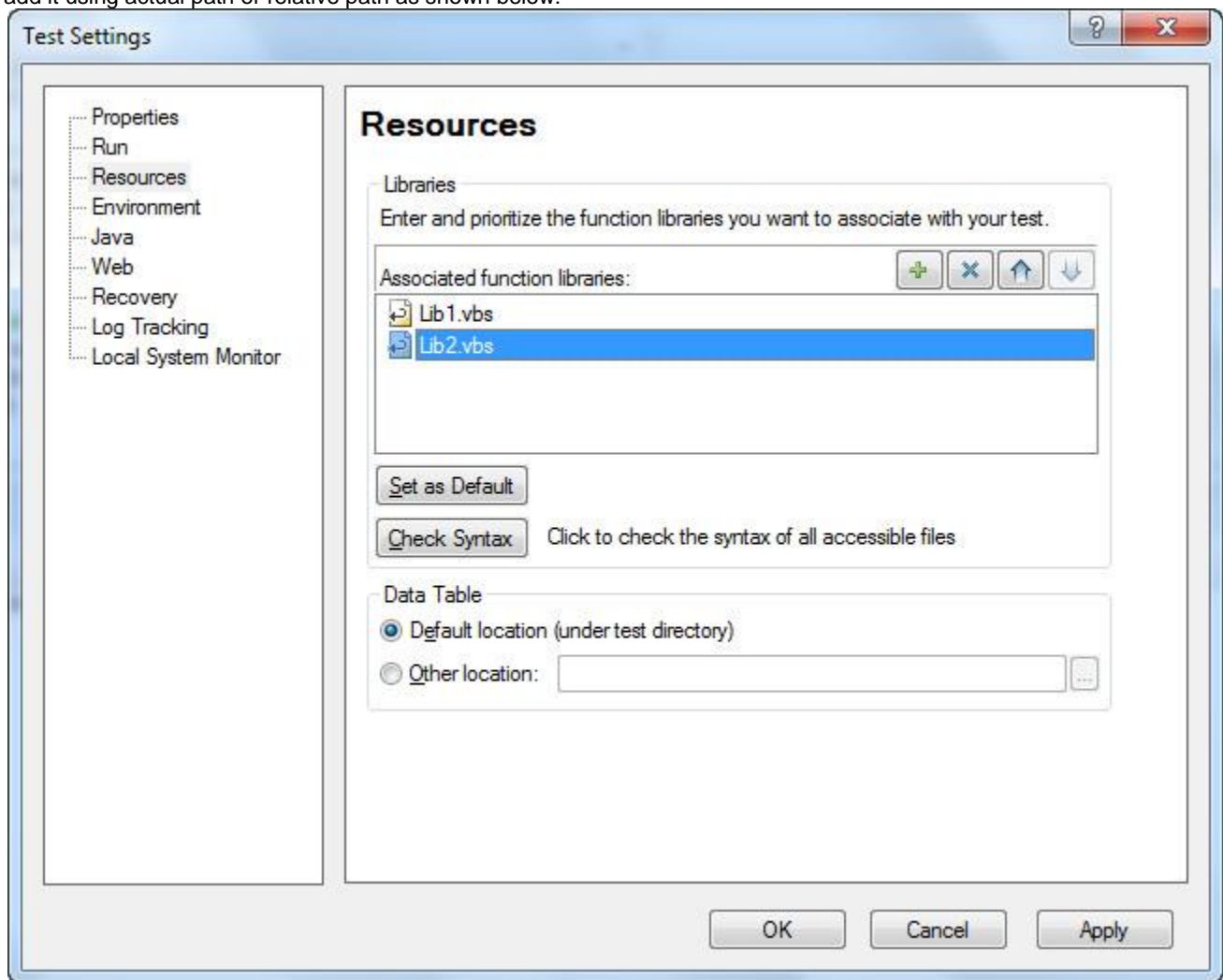
## Library Files

In order to modularize the script, library Files are added to the QTP Script. It contains variable declaration, Functions, Classes etc. They enable reusability that can be shared across test scripts. They are saved with an extension .vbs or .qfl

A New Library File can be Created by Navigating to "File" >> "Function Library"

## Associating Function Libraries

**Method#1 :** By using "File" > "Settings" > Resources > Associate Function Library option. Click on "+" Button to Add Function Library File and add it using actual path or relative path as shown below:



**Method#2 :** Using **ExecuteFile** method.

```
'Syntax : ExecuteFile(Filepath)
ExecuteFile "C:\lib1.vbs"
ExecuteFile "C:\lib2.vbs"
```

**Method#3 :** Using **LoadFunctionLibrary** Method.

```
'Syntax : LoadFunctionLibrary(Filepath)
LoadFunctionLibrary "C:\lib1.vbs"
LoadFunctionLibrary "C:\lib2.vbs"
```

**Method#4 : Automation Object Model(AOM)** - It is a mechanism using which we can control various QTP operations outside QTP. Using AOM, we can launch QTP, Open the Test, Associate Function Libraries etc. The Following Vbscript should be saved with Extension .vbs and upon executing the same, QTP will be launched and test would start executing. AOM will be discussed in detail in the later chapters.

```
'Launch QTP
```

```
Set objQTP = CreateObject("QuickTest.Application")
objQTP.Launch
objQTP.Visible = True

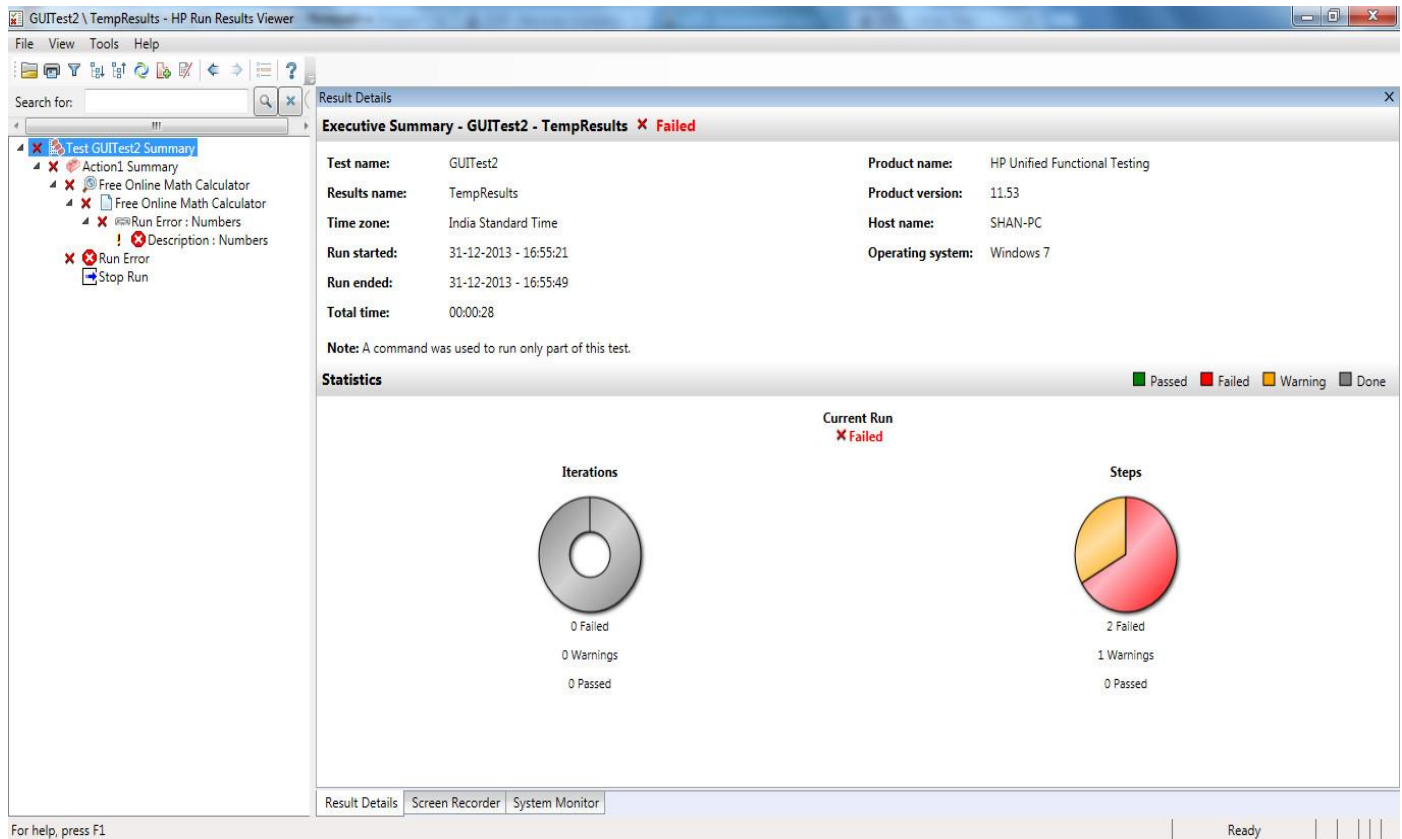
'Open the test
objQTP.Open "D:\GUITest2", False, False
Set objLib = objQTP.Test.Settings.Resources.Libraries

'Associate Function Library if NOT associated already.
If objLib.Find("C:\lib1.vbs") = -1 Then
    objLib.Add "C:\lib1.vbs", 1
End
```

## Test Results:

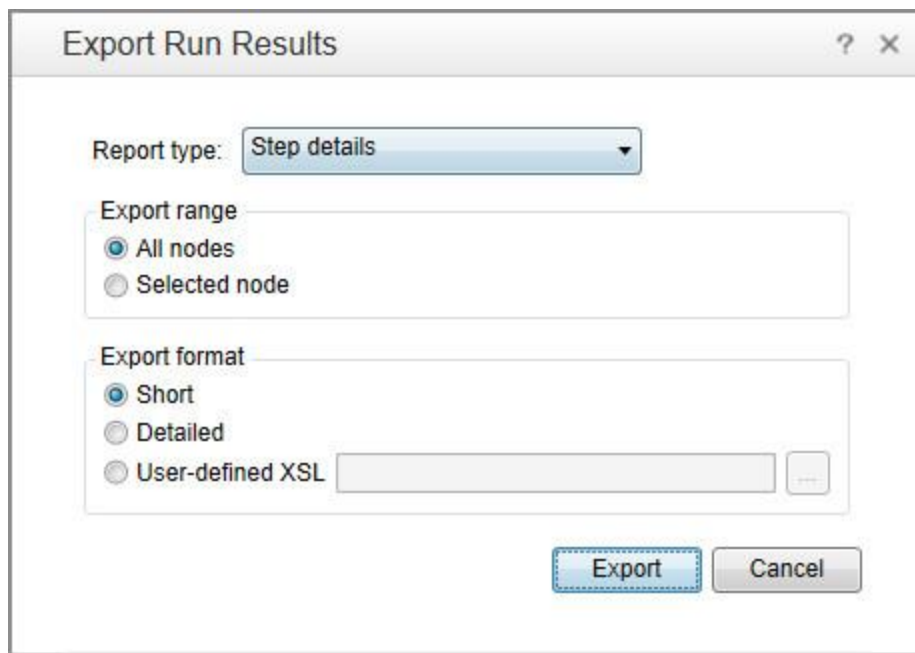
The Test Results Window gives us sufficient information to show the steps passed, failed etc. Results window opens automatically after execution of the test (as per default settings).

- Steps Passed
- Steps Failed
- Environment Parameters
- Graphical Statistics



## Operations performed in Test Results: CONVERTING RESULTS TO HTML

In the Results Viewer window, Navigate to "File" -> "Export to File", Export Run Results dialog box opens as shown below:

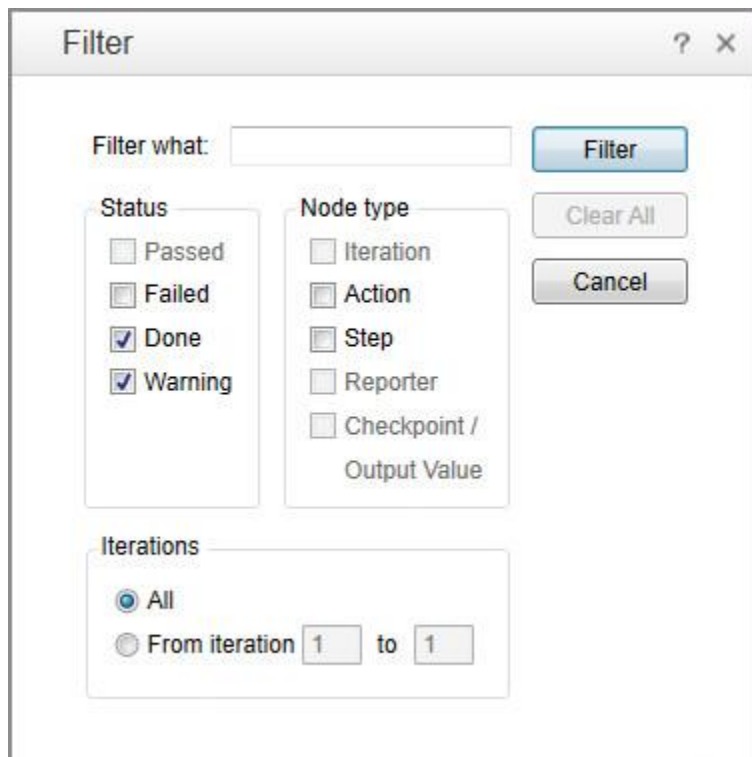


The "Export Run Results" dialog box has a title bar with a question mark and a close button. It contains three main sections: "Report type" with a dropdown menu set to "Step details"; "Export range" with radio buttons for "All nodes" (selected) and "Selected node"; and "Export format" with radio buttons for "Short" (selected), "Detailed", and "User-defined XSL" (which has an adjacent text field and a browse button). At the bottom right are "Export" and "Cancel" buttons.

We can choose what type of report to be exported. It can be short results, Detailed Results or even we can select nodes. Upon Selecting the File Name and exporting it, the file would be saved as .HTML File

## FILTERING THE RESULTS:

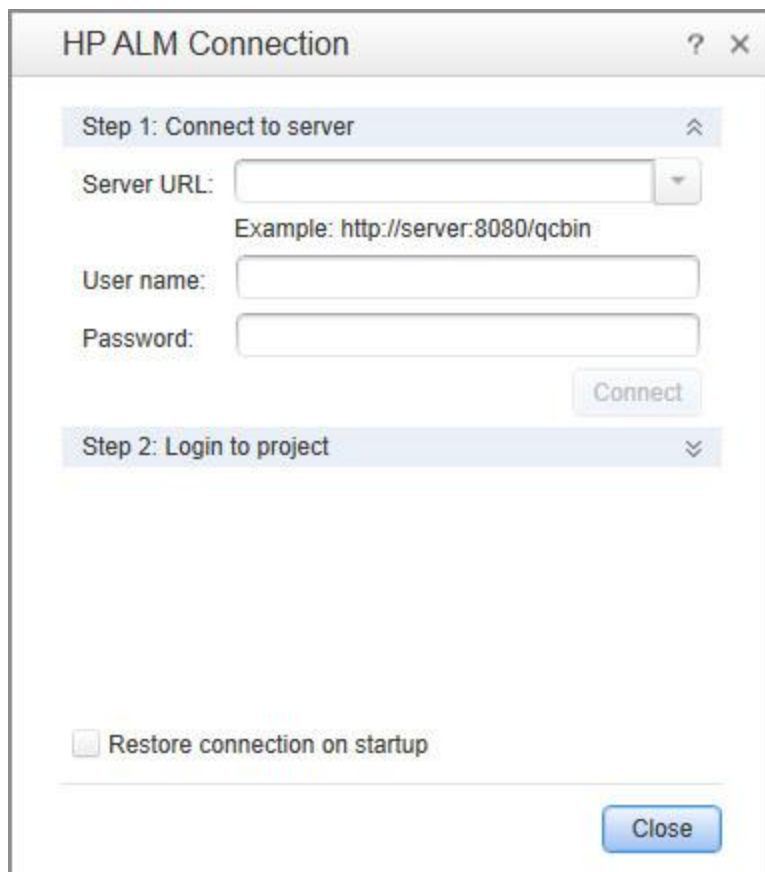
Results can be filtered based Status, Node Type, Iterations. It can be accessed by using Filter Button in the "Test Results Window"



The "Filter" dialog box has a title bar with a question mark and a close button. It features a "Filter what:" text field at the top left, followed by a "Filter" button. Below this are three filter categories: "Status" with checkboxes for "Passed", "Failed", "Done" (checked), and "Warning" (checked); "Node type" with checkboxes for "Iteration", "Action", "Step", "Reporter", and "Checkpoint / Output Value"; and "Iterations" with radio buttons for "All" (selected) and "From iteration" (with input fields for "1" and "1"). To the right of the filter categories are "Clear All" and "Cancel" buttons.

## RAISING DEFECTS

Defects can be logged into QC directly from the Test Results Window pane by accessing "Tools" -> "Add Defect" which open's connection to ALM as shown below:



The image shows a dialog box titled "HP ALM Connection" with a standard Windows-style title bar containing a question mark and a close button. The dialog is divided into two main sections by expandable/collapsible headers. The first section, "Step 1: Connect to server", is currently expanded and contains three input fields: "Server URL:" with a dropdown arrow, "User name:", and "Password:". Below the "Server URL" field is an example text: "Example: http://server:8080/qcbin". To the right of the "Password" field is a "Connect" button. The second section, "Step 2: Login to project", is currently collapsed. At the bottom of the dialog, there is a checkbox labeled "Restore connection on startup" which is currently unchecked. A "Close" button is located at the bottom right of the dialog.

HP ALM Connection ? X

Step 1: Connect to server ^

Server URL:  ▼  
Example: http://server:8080/qcbin

User name:

Password:

Connect

Step 2: Login to project v

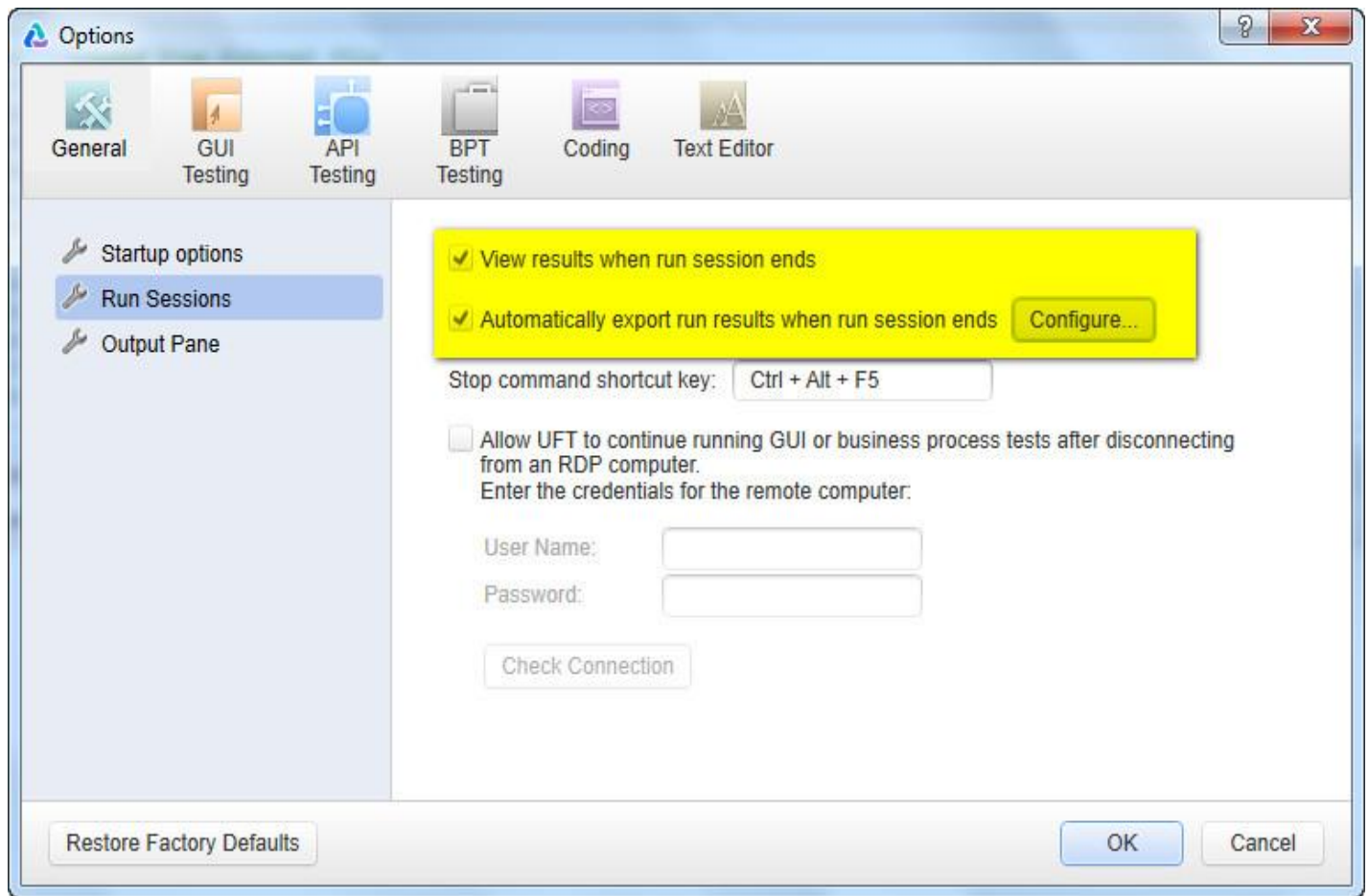
☐ Restore connection on startup

Close

## Test Results:

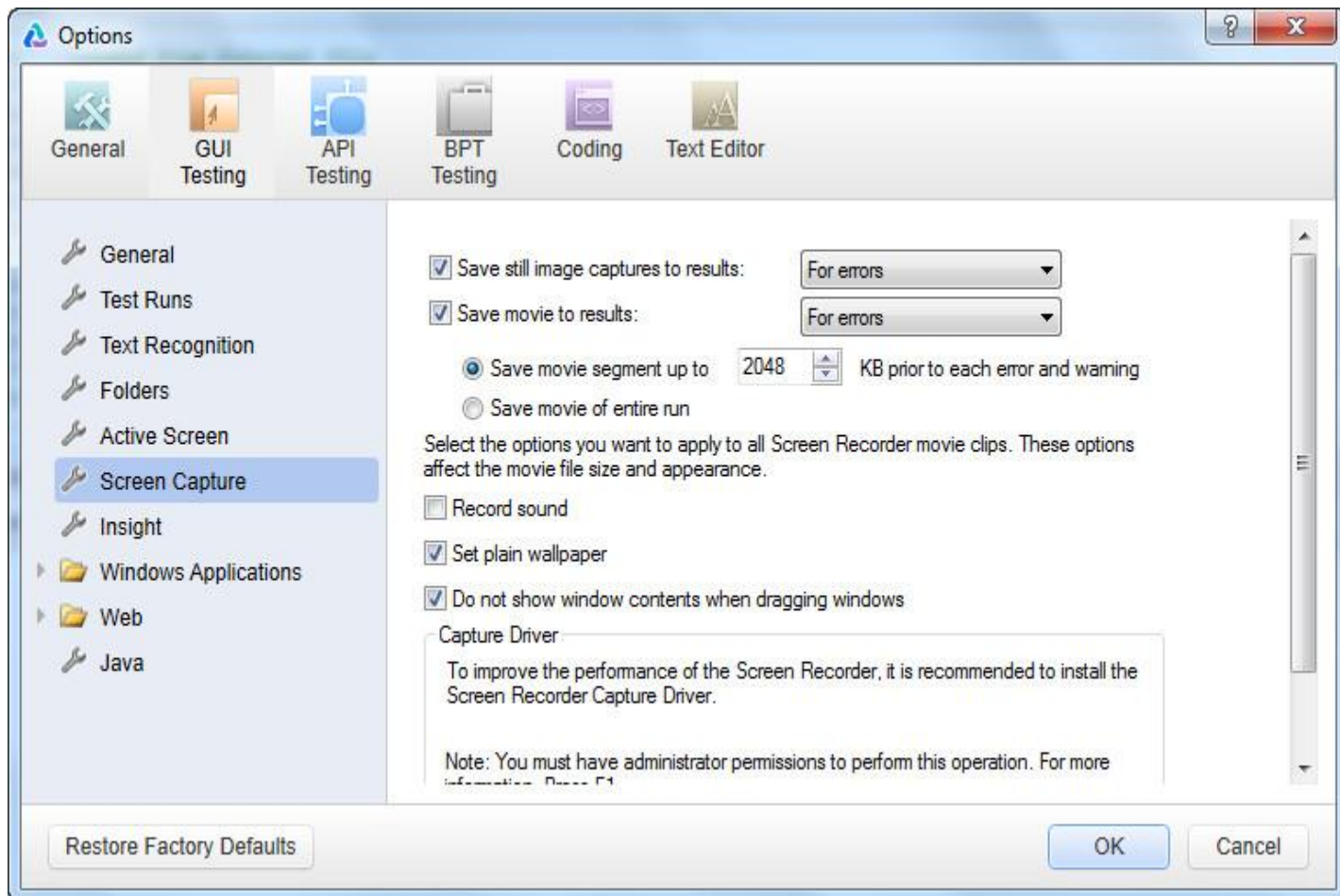
The Automatic Test Results Window can be configured under "Tools" -> "Options" -> "Run Sessions" Tab. We can turn it off if required and also we can switch ON "Automatically Export Results when session Ends"





Upon Errors, the screenshot or the movie can be recorded based on the settings. The same can be configured under "Tools" -> "Options" -> "Screen Capture" Tab. We can save the screenshot or movies based on **3 conditions**.

- **For Errors**
- **Always**
- **For Errors and Warnings**



# Working with GUI Objects:

There are various GUI objects with which QTP interacts during the script execution. Hence it is important to know the basic methods for the key GUI objects using which we will be able to work on it effectively.

## Working with Text Box

Below are the **methods** using which we access text box during Run Time.

- **Set** - Helps the tester to Set Values into the Text Box
- **Click** - Clicks on the Text Box
- **SetSecure** - Used to set the text in the password boxes securely.
- **WaitProperty** - Waits Till the Property value becomes true.
- **Exist** - Checks for the existence of the Text Box
- **GetROProperty("text")** - Gets the Value of the Text Box
- **GetROProperty("Visible")** - Returns a Boolean value if visible.

## EXAMPLE:

```
Browser("Math Calculator").Sync
Set Obj = Browser("Math Calculator").Page("SQR Calc").WebEdit("n")

'Clicks on the Text Box
Obj.Click

'Verify if the Object Exist - Returns Boolean value
a= obj.Exist
print a

'Set the value
obj.Set "10000" : wait(2)

'Get the Runtime Object Property - Value of the Text Box
val = obj.GetROProperty("value")
print val

'Get the Run Time Object Property - Visibility - Returns Boolean Value
x= Obj.GetROProperty("visible")
print x
```

# Working with Check Box

Following are some of the key methods with which one can work with Check Box.

- **Set** - Helps the tester to Set the checkbox value "ON" or "OFF"
- **Click** - Clicks on the check Box. Even checks ON or OFF but user won't be sure about the status.
- **WaitProperty** - Waits Till the Property value becomes true.
- **Exist** - Checks for the existence of the Check Box
- **GetROProperty("name")** - Gets the Name of the check Box
- **GetROProperty("Visible")** - Returns a Boolean value if visible

## EXAMPLE:

```
'To Check the Check Box
Set Obj = Browser("Calculator").Page("Gmail").WebCheckBox("PersistentCookie")
Obj.Set "ON"

'To UnCheck the Check Box
Obj.Set "OFF"

'Verifies the Existence of the Check box and returns Boolean Value
val = Obj.Exist
print val

'Fetches the Name of the CheckBox
a= Obj.GetROProperty("name")
print a

'Verifies the visible property and returns the boolean value.
x = Obj.GetROProperty("visible")
print x
```

# Working with Radio Button

Following are some of the key methods with which one can work with Radio Button.

- **Select**(RadioButtonName) - Helps the tester to Set the Radio Box "ON"
- **Click** - Clicks on the Radio Button. Even Radio Button ON or OFF but tester can't get the status.
- **WaitProperty** - Waits Till the Property value becomes true.
- **Exist** - Checks for the existance of the Radio Button
- **GetROProperty**("name") - Gets the Name of the Radio Button
- **GetROProperty**("Visible") - Returns a Boolean value if visible

## EXAMPLE:

```
'Select the Radio Button by name "YES"
Set Obj = Browser("Calculator").Page("Forms").WebRadioGroup("group1")
Obj.Select("Yes")

'Verifies the Existence of the Radio Button and returns Boolean Value
val = Obj.Exist
print val

'Returns the Outerhtml of the Radio Button
txt = Obj.GetROProperty("outerhtml")
print txt

'Returns the boolean value if Radio button is Visible.
vis = Obj.GetROProperty("visible")
print vis
```

# Working with Combo Box

Following are some of the key methods with which one can work with Combo Box.

- **Select**(Value) - Helps the tester to Select the value from the ComboBox
- **Click** - Clicks on the object.
- **WaitProperty** - Waits Till the Property value becomes true.
- **Exist** - Checks for the existence of the Combo Box
- **GetROProperty**("Text") - Gets the Selected Value of the Combo Box
- **GetROProperty**("all items") - Returns all the items in the combo Box
- **GetROProperty**("items count") - Returns the number of items in the combo Box

## EXAMPLE:

```
'Get the List of all the Items from the ComboBox
Set ObjList = Browser("Math Calculator").Page("Statistics").WebList("class")
x = ObjList.GetROProperty("all items")
print x

'Get the Number of Items from the Combo Box
y = ObjList.GetROProperty("items count")
print y

'Get the text value of the Selected Item
z = ObjList.GetROProperty("text")
print z
```



# Working with Buttons

Following are some of the key methods with which one can work with Buttons.

- **Click** - Clicks on the Button.
- **WaitProperty** - Waits Till the Property value becomes true.
- **Exist** - Checks for the existence of the Button
- **GetROProperty**("Name") - Gets the Name of the Button
- **GetROProperty**("Disabled") - Returns a boolean value if enabled/disabled

## EXAMPLE:

```
'To Perform a Click on the Button
Set obj_Button = Browser("Math Calculator").Page("SQR").WebButton("Calc")
obj_Button.Click

'To Perform a Middle Click on the Button
obj_Button.MiddleClick

'To check if the button is enabled or disabled.Returns Boolean Value
x = obj_Button.GetROProperty("disabled")
print x

'To fetch the Name of the Button
y = obj_Button.GetROProperty("name")
print y
```

# Working with webTables

In Today's web based application, webtables have become very common and testers need to understand how Web Tables work and how to perform an action on web Tables. This Topic will help you to work with the web Tables Effectively.

Statement	Description
if statement	An if statement consists of a boolean expression followed by one or more statements.
if..else statement	An if else statement consists of a boolean expression followed by one or more statements. If the condition is True, the statements under If statements are executed. If the condition is false, Else part of the script is Executed
if...elseif..else statement	An if statement followed by one or more Elself Statements, that consists of boolean expressions and then followed by an optional else statement, which executes when all the condition becomes false.
nested if statements	An if or elseif statement inside another if or elseif statement(s).
switch statement	A switch statement allows a variable to be tested for equality against a list of values.

- **html id** - If the table has an id tag then it is best to make use of this property.
- **innerText** - Heading of the Table.
- **sourceIndex** - Fetches the Source Index of the Table
- **ChildItemCount** - Gets the number of ChildItems present in specified Row
- **RowCount** - Gets the number of Rows in the Table
- **ColumnCount** - Gets the number of Columns in the Table
- **GetCellData** - Gets the Value of the Cell based on the column and Row Index

## EXAMPLE:

```
Browser("Tutorials Point").Sync
' WebTable
Obj = Browser("Tutorials Point").Page("VBScript Decisions").WebTable("Statement")
' Fetch RowCount
x = Obj.RowCount
print x

' Fetch ColumnCount
y = Obj.ColumnCount(1)
print y

' Print the Cell Data of the Table
For i = 1 To x Step 1
    For j = 1 To y Step 1
        z = Obj.GetCellData(i,j)
        print "Row ID : " & i & " Column ID : " & j & " Value : " & z
    Next
Next

'Fetch the Child Item count of Type Link in a particular Cell
z = Obj.ChildItemCount(2,1,"Link")
print z
```

# What are Virtual Objects? (Interview Question)

Sometimes, application under test may contain standard window object but are NOT recognized by QTP. Under these circumstances objects can be defined as virtual object(VO) of type button, link etc so that user actions can be simulated on the virtual objects during execution.

## Example

Let us say we are automating a scenario in Microsoft Word. I activated MS word application and I click on any icon in the ribbon. For example, In the Insert Ribbon, User clicks on "Picture" button. A Button is recognized as WinObject hence importance of virtual objects is pronounced.

```
Window("Microsoft Word").WinObject("Ribbon").Click 145,45  
Window("Microsoft Word").WinObject("Ribbon").WinObject("Picture...").Click 170,104
```

## Virtual Object Limitations

- QTP doesn't support virtual objects for analog or low-level recording.
- Checkpoints cannot be added on Virtual Objects.
- Virtual Object are NOT controlled by Object Repository.
- Though we map an object to a particular class (button or List), all the methods of the native objects are not supported by Virtual objects.
- Object Spy cannot be used on Virtual Object.
- The test execution will fail if the screen resolution changes as the co-ordinates change.
- Application Window should be of same screen size so that Virtual objects are captured correctly.

# Accessing Databases:

As such QTP doesn't provide any built-in support to connect to database, however using VBScript testers will be able to connect and interact with databases using ADODB objects

ADODB has 4 properties or methods with which we will be able to work with the databases.

- **ADODB.Connection** - Used to establish a connection to the Database
- **ADODB.Command** - Used to execute a SQL command(Queries or Stored Procedures)
- **ADODB.Fields** - Used to fetch a particular column from a record set after executing a query/stored proc
- **ADODB.Recordset** - Used to fetch data from a database

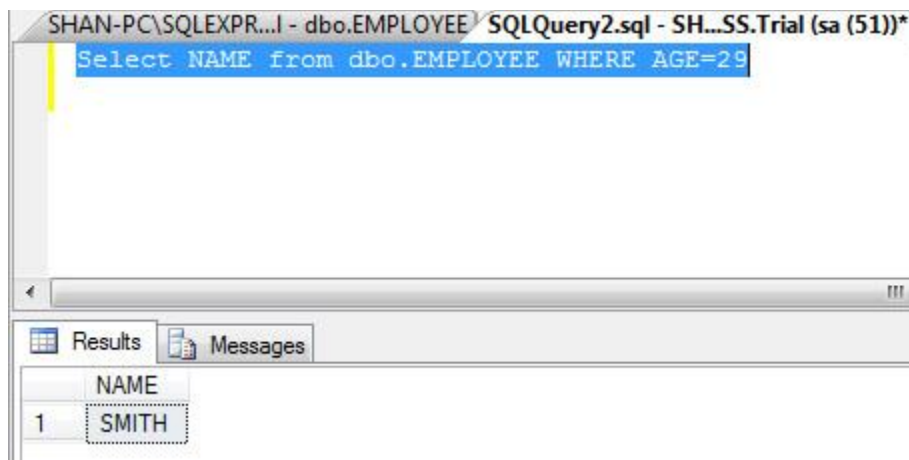
## How to connect to Database? (Interview Question)

Databases can be connected using Connection strings. Each database differ the way we connect to the them, however the connection strings can be build with the help of <http://www.connectionstrings.com/>

Now Let us see how to connect to the database with the following parameters.

- **Database Type** - MSSQL SERVER
- **Server Name** - SQLEXPRESS
- **Database Name** - Trial
- **User Id** - sa
- **password** - Password123

The Output of the Query is shown in the SQL Server Management Studio as follows:



```
Dim objConnection
'Set Adodb Connection Object
Set objConnection = CreateObject("ADODB.Connection")
Dim objRecordSet

'Create RecordSet Object
Set objRecordSet = CreateObject("ADODB.Recordset")

Dim DBQuery 'Query to be Executed
DBQuery = "Select NAME from dbo.EMPLOYEE where AGE = 29"

'Connecting using SQL OLEDB Driver
objConnection.Open "Provider=sqloledb.1;Server=.\SQLEXPRESS;User
Id=sa;Password=Password123;Database=Trial"

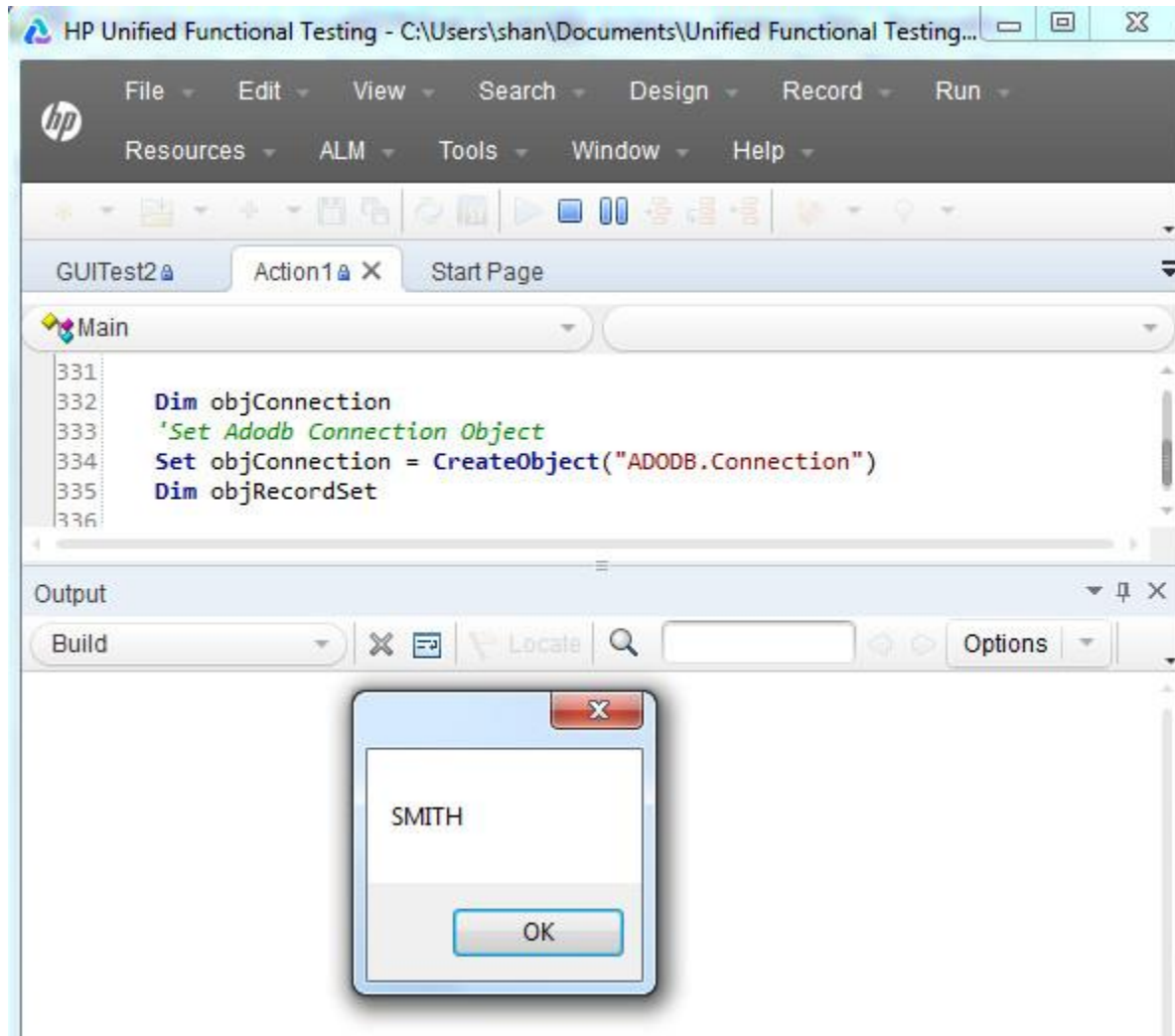
'Execute the Query
objRecordSet.Open DBQuery,objConnection

'Return the Result Set
Value = objRecordSet.fields.item(0)
msgbox Value
```

```
' Release the Resources  
objRecordSet.Close  
objConnection.Close  
  
Set objConnection = Nothing  
Set objRecordSet = Nothing
```

## RESULT

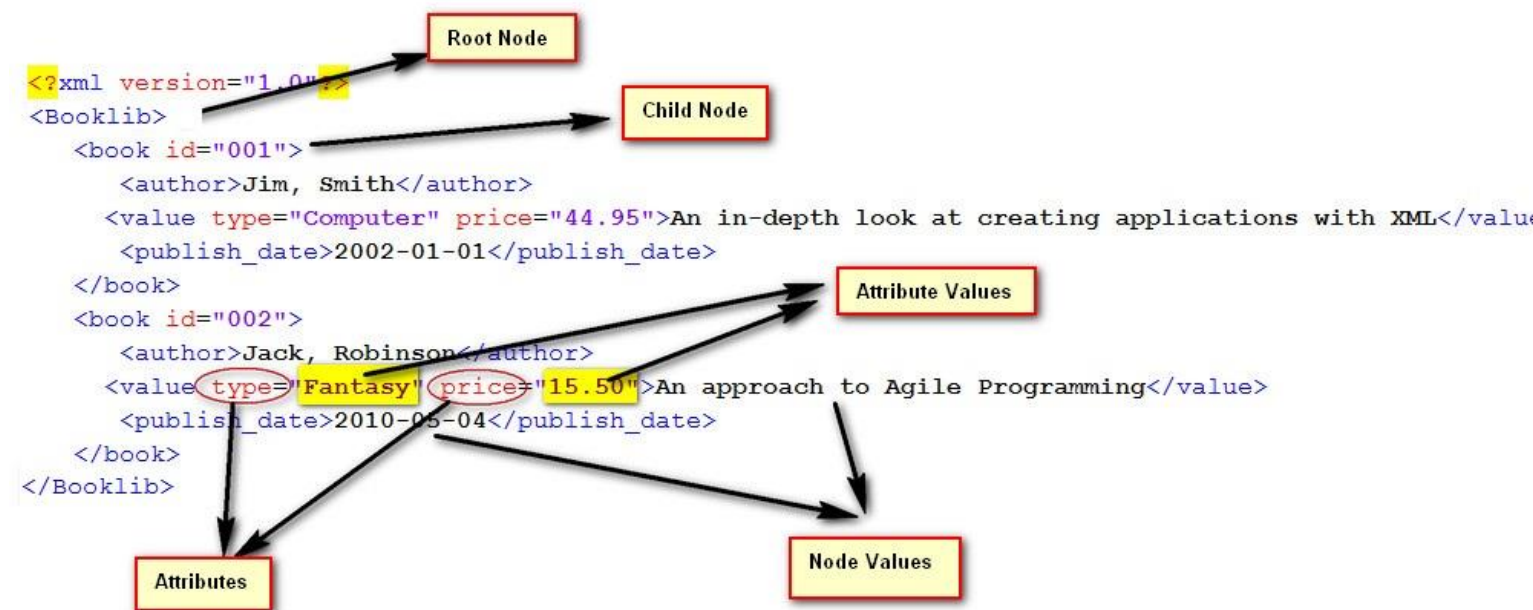
Upon Executing the above script the output is shown in the message box as shown below:



# What is an XML?

XML is a markup language **designed for how to store data that is in the form that both human and machine readable format.** Using XML, data can also be easily exchanged between computer and database systems.

Sample XML and their key elements are represented below:



## Accessing XML

```
Const XMLDataFile = "C:\TestData.xml"
Set xmlDoc = CreateObject("Microsoft.XMLDOM")
xmlDoc.Async = False
xmlDoc.Load(XMLDataFile)

' Getting the number of Nodes (books)
Set nodes = xmlDoc.SelectNodes("/bookstore/book")
Print "Total books: " & nodes.Length      ' Displays 2

' get all titles
Set nodes = xmlDoc.SelectNodes("/Booklib/book/value/text()")

' get their values
For i = 0 To (nodes.Length - 1)
    Title = nodes(i).NodeValue
    Print "Title is" & (i + 1) & ": " & Title
Next
```

## Comparing XML

We can compare Two given xml's.

```
Dim xmlDoc1
Dim xmlDoc2

' Load the XML Files
Set xmlDoc1 = XMLUtil.CreateXMLFromFile ("C:\File1.xml")
Set xmlDoc2 = XMLUtil.CreateXMLFromFile ("C:\File2.xml")

'Use the compare method of the XML to check if they are equivalent
Comp = xmlDoc1.Compare (xmlDoc1, xmlDoc2)

'Returns 1 if the two files are the same
If Comp = 1 Then
    MsgBox "XML Files are the Same"
```



```
Else  
    MsgBox "XML Files are Different"  
End if
```

## Descriptive Programming:

QTP scripts can execute only if the objects are present in the Object Repository. If the descriptions of the Objects are created using Descriptive programming when testers want to perform an operation on an object that is not present in the object repository.

- When objects in the application are very dynamic in nature.
- When the Object Repository grows big which will result in poor Performance as the size of the Object Repository increases.
- When the framework is built such that it has been decided not to use Object Repository at all.
- When testers want to perform an action on the application at run-time without having the knowledge of object's unique properties.

## Syntax

There are two ways to script using Descriptive Programming technique. They are

1. **Description Objects**
2. **Description Strings**

## Description Objects

Script is developed using description Objects that depends upon the properties used and their corresponding values. Then these descriptions are used to build the script.

```
'Creating a description object
Set btnCalc = Description.Create()

'Add descriptions and properties
btnCalc("type").value = "Button"
btnCalc("name").value = "calculate"
btnCalc("html tag").value = "INPUT"

' Use the same to script it
Browser("Math Calc").Page("Num Calculator").WebButton(btnCalc).Click
```

## Description Strings

The description of the objects are developed using the properties and values as strings as shown below.

```
Browser("Math
Calc").Page("NumCalculator").WebButton("htmltag:=INPUT", "type:=Button", "name:=calculate").Click
```

## Child Objects

QTP provides the ChildObjects method which enables us to create a collection of objects. The parent objects precedes ChildObjects.

```
Dim oDesc
Set oDesc = Description.Create
```

```

oDesc("micclass").value = "Link"

'Find all the Links
Set obj = Browser("Math Calc").Page("Math Calc").ChildObjects(oDesc)

Dim i
'obj.Count value has the number of links in the page
For i = 0 to obj.Count - 1
    'get the name of all the links in the page
    x = obj(i).GetROProperty("innerHTML")
    print x
Next

```

## Ordinal Identifiers

Descriptive programming is used to script based on ordinal identifiers which will enable QTP to act on those objects when two or more objects have same properties.

```

' Using Location
Dim Obj
Set Obj = Browser("title:=.*google.*").Page("micclass:=Page")
Obj.WebEdit("name:=Test", "location:=0").Set "ABC"
Obj.WebEdit("name:=Test", "location:=1").Set "123"

' Index
Obj.WebEdit("name:=Test", "index:=0").Set "1123"
Obj.WebEdit("name:=Test", "index:=1").Set "2222"

' Creation Time
Browser("creationtime:=0").Sync
Browser("creationtime:=1").Sync
Browser("creationtime:=2").Sync

```

## Automation Object Model: (Interview Question)

QTP itself can be automated using the COM interface that is provided by Hp-QTP. Automation object model is a set of objects, methods, and properties that helps testers to control the configuration settings and execute the scripts using the QTP interface. The Key Configurations/actions that can be controlled are listed below but not limited to

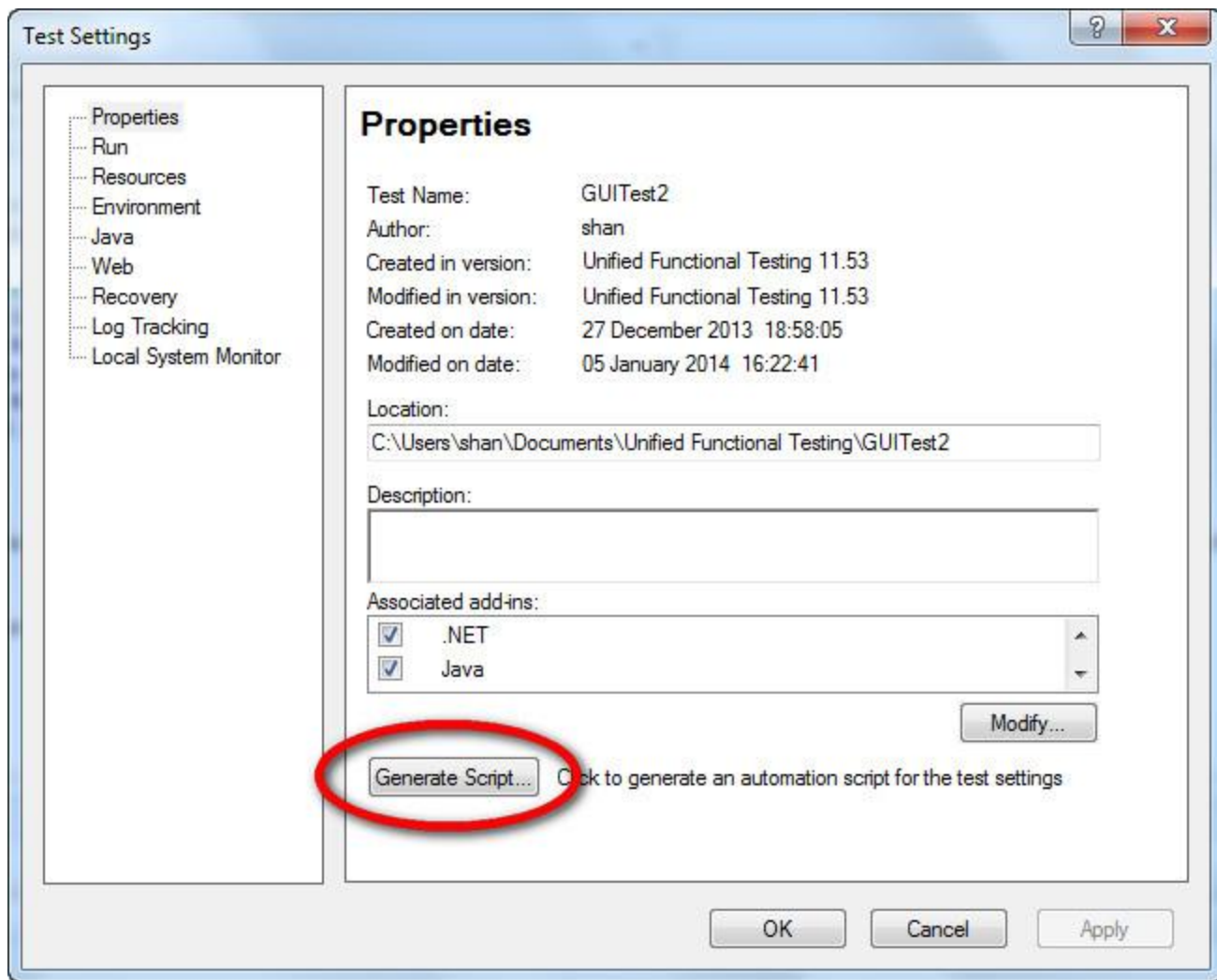
- Loads all the required add-ins for a test

- Makes QTP visible while execution
- Opens the Test using the specified location
- Associates Function Libraries
- Specifies the Common Object Sync Time out
- Start and End Iteration
- Enable/Disable Smart Identification
- On Error Settings
- Data Table Path
- Recovery Scenario Settings
- Log Tracking Settings

QTP 11.5x provides an exclusive documentation on Automation Object model that can be referred by navigating to "Start" >> "All Programs" >> "HP Software" >> "HP Unified Functional Testing" >> "Documentation" >> "Unified Functional Testing Automation Reference"

## Generate AOM Script:

Tester Can generate AOM Script from QTP itself using "Generate Script" Option. Navigate to "Run" >> "Settings" >> "Properties" Tab >> "Generate Script" as shown below:



## Example

```
' A Sample Script to Demonstrate AOM
Dim App 'As Application
Set App = CreateObject("QuickTest.Application")
App.Launch
App.Visible = True

App.Test.Settings.Launchers("Web").Active = False
App.Test.Settings.Launchers("Web").Browser = "IE"
App.Test.Settings.Launchers("Web").Address = "http://easycalculation.com/"
App.Test.Settings.Launchers("Web").CloseOnExit = True

App.Test.Settings.Launchers("Windows Applications").Active = False
App.Test.Settings.Launchers("Windows Applications").Applications.RemoveAll
App.Test.Settings.Launchers("Windows Applications").RecordOnQTDendants = True
App.Test.Settings.Launchers("Windows Applications").RecordOnExplorerDendants = False
App.Test.Settings.Launchers("Windows Applications").RecordOnSpecifiedApplications = True

App.Test.Settings.Run.IterationMode = "rngAll"
App.Test.Settings.Run.StartIteration = 1
App.Test.Settings.Run.EndIteration = 1
App.Test.Settings.Run.ObjectSyncTimeOut = 20000
App.Test.Settings.Run.DisableSmartIdentification = False
App.Test.Settings.Run.OnError = "Dialog"

App.Test.Settings.Resources.DataTablePath = ""
App.Test.Settings.Resources.Libraries.RemoveAll

App.Test.Settings.Web.BrowserNavigationTimeout = 60000
App.Test.Settings.Web.ActiveScreenAccess.UserName = ""
App.Test.Settings.Web.ActiveScreenAccess.Password = ""
```

```

App.Test.Settings.Recovery.Enabled = True
App.Test.Settings.Recovery.SetActivationMode "OnError"
App.Test.Settings.Recovery.Add "D:\GUITest2\recover_app_crash.qrs", "Recover_Application_Crash", 1
App.Test.Settings.Recovery.Item(1).Enabled = True

' .....
' System Local Monitoring settings
' .....
App.Test.Settings.LocalSystemMonitor.Enable = false
' .....
' Log Tracking settings
' .....
With App.Test.Settings.LogTracking
    .IncludeInResults = False
    .Port = 18081
    .IP = "127.0.0.1"
    .MinTriggerLevel = "ERROR"
    .EnableAutoConfig = False
    .RecoverConfigAfterRun = False
    .ConfigFile = ""
    .MinConfigLevel = "WARN"
End With

```

## What is a Software Framework? (Interview Question)

A Framework defines a set of guidelines/best practices that enforces a set of standards which makes it easy to use for the end users to work with. There are different types of automation frameworks and the most common ones are listed below:

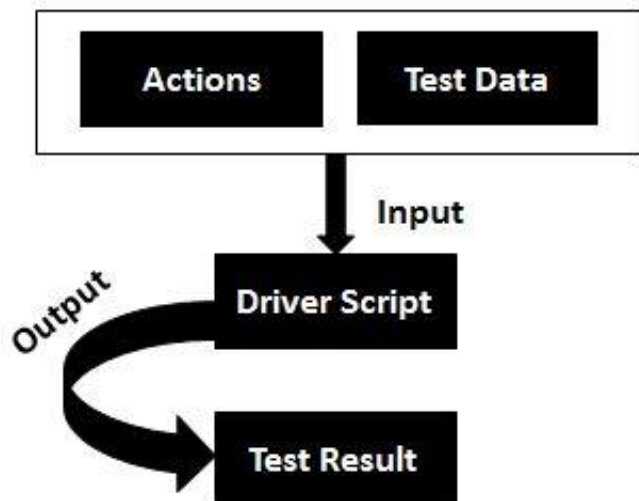
1. **Keyword-Driven Framework**
2. **Data-Driven Framework**



## Keyword-Driven Framework

Keyword driven testing is a type of functional automation testing framework which also known as table-driven testing or action word based testing.

In Keyword-driven testing we use a table format, usually a spreadsheet, to define keywords or action words for each function that we would like to execute.



### ADVANTAGES:

- It is best suited for novice or a non-technical tester.
- Enables writing tests in a more abstract manner using this approach.
- Keyword driven testing allows automation to be started earlier in the SDLC even before a stable build is delivered for testing.
- There is a high degree of reusability.

### DISADVANTAGES:

- Initial investment in developing the keywords and its related functionalities might take longer.
- It might act as a restriction to the technically abled testers.

## Data Driven Framework

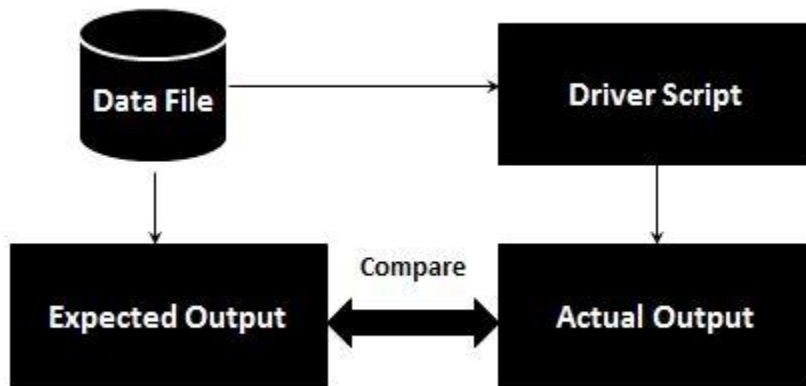
Data-driven testing is creation of test scripts where test data and/or output values are read from data files instead of using the same hard-coded values each time the test runs. This way tester can test how the application handles various inputs effectively. It can be any of the below data files.

- DataPools
- Excel files
- ADO objects

- CSV files
- ODBC sources

## FLOW DIAGRAM:

Data Driven Testing can be best understood by the following diagram:



## ADVANTAGES:

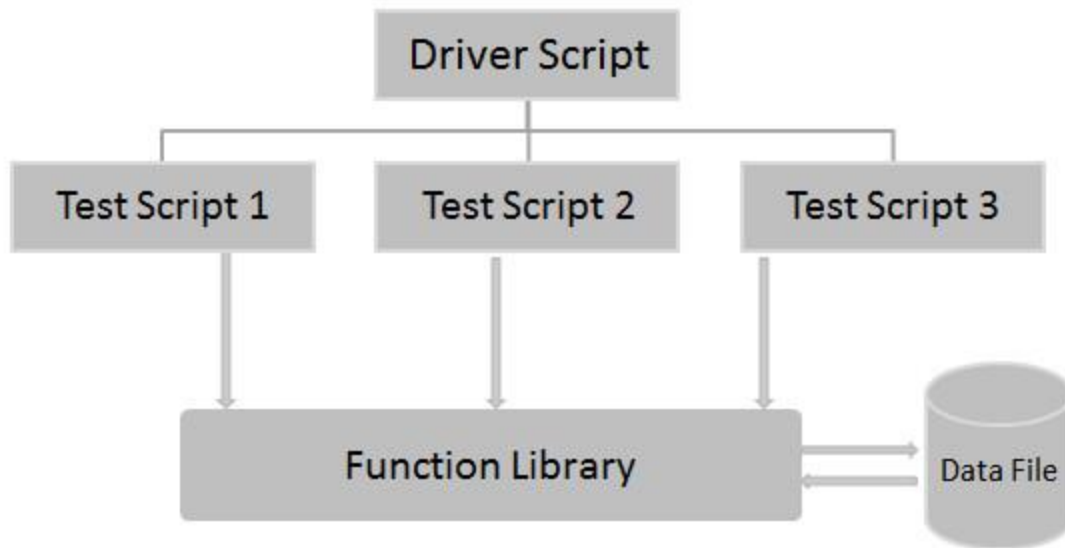
- Data driven framework results in less amount of code.
- Offers greater flexibility for maintaining and fixing the scripting issues.
- Test Data can be developed

## DISADVANTAGES:

- Each script needs to be different to understand different sets of data.

## Hybrid Framework

Hybrid Framework is a combination of Keyword driven and data Driven framework that can be best described using the following flow diagram.



## Affecting Factors

Following are the parameters one should take into account while developing the framework. The affects factors are listed below

- Framework Files Should Support Versioning Controlling Software such as SVN, CVS, MS Source Control
- Framework should support executing the scripts in different environments viz- QA, SAT, DEV
- Upon Object Changes, Scripts should execute with minimal changes.
- Framework should configure itself and take care of prerequisite such as creating folders/databases.
- Framework Should have Robust Reporting Structure so that issues in the script/application can be easily spotted
- Framework Should have greater flexibility so that it should be easy to use
- Framework Should follow coding standards so that files, functions and history of changes are maintained correctly.

# QTP/UFT INTERVIEW QUESTIONS

### **1. What is the difference between an Action and a Function?**

- a. **Action** is an **activity** specific to QTP while functions are a generic thing that is a feature of VB Scripting.
- b. Action can have an object repository associated with it while a function can't.
- c. **Function** is just **lines of code** with some / none parameters and a single return value.
- d. Action can have more than one output parameters.

### **2. What are the various events available in the Recovery Scenario Manager?**

- a. Application Crash: This event is useful in handling crashed applications at runtime.
- b. Pop Up Window: This event is useful in managing various unwanted application windows, which get built-up at runtime.
- c. Test Run Error: This event is useful in handling VBScript statement errors at runtime.
- d. Object State: This event is useful in handling object related errors at runtime.

### **3. What are the Elements of Recovery Scenario?**

#### **a. Steps to handle the exceptions are**

- i. Trigger Event: Is an unexpected event like appearance of a Pop-up window, object state, test run error causing application crash or interruption in our running session.
- ii. Recovery Steps: Constitutes a series of steps required to be performed to enable QTP to proceed further with the process of test after some trigger event has interrupted the run session.
  1. Examples of a recovery operation can be
    - a. 1) A keyboard or mouse Operation like a Click over the "OK" button in the Pop-up window
    - b. 2) Close Application Process
    - c. 3) Function Call
    - d. 4) Restarting the OS etc.
- iii. Post-Recovery Test Run: Are a set of instructions designed to provide further instructions to QTP on how to proceed further with the test after some recovery operation has been carried out.
  1. Examples of Post Recovery actions can be repeating the complete test from the beginning or some steps may be skipped altogether & continuing with the remaining steps in the test.

#### **4. When to use a Recovery Scenario and When to use “on error resume next”?**

- a. Recovery scenarios are useful when it is difficult to predict at which step the errors can come or when we are confident that the error will not come in the QTP script, whereas it can be anywhere outside the QTP Script.
  - i. For Example:
    - 1. Pop-up message of “out of paper”, as caused by the printer device driver.
    - 2. “On error resume next” is preferred when we are sure that the error is expected one and wish to perform some other actions.

#### **5. What are various types of properties when using Object Identification in QTP?**

- a. QTP uses three types of properties when identifying an object
  - i. Mandatory Properties: Always learn these properties for the object
  - ii. Assistive Properties: Learn in case Mandatory properties are not enough to identify the object uniquely
  - iii. Ordinal Identifiers: Learn in case both mandatory and assistive properties are not able to recognize the objects correctly

#### **6. What is the use of Parameterization in QTP?**

- a. Parameterization is helpful in aspects like:
  - i. Parameterization allows us to pick different values at run time.
  - ii. Reduces Time and Effort.
  - iii. Usage of data drivers allows us to use the same data for various input boxes.
  - iv. Parameterization can also be done for checkpoints.

#### **7. What are the Environment Variables?**

- a. Environment variables are global variables available to all Actions

- b. They can be used to run a test case on different environment
- c. To add a new Environment variable go to Test -> Settings...->Environment (Tab)
- d. Environment variables are of **two types**:
  - i. Built in Environment Variables: These provide information about the system and the current test
  - ii. User-Defined Environment Variables: These are added in the Environment tab of Test Settings. These are Read-only during the test run

## 8. What is Descriptive Programming?

- a. Descriptive Programming is an alternate way of writing test cases without having objects in object repository
- b. Descriptive programming can be done in **two ways**:
  - i. Using Object Description
  - ii. Using String Description
- c. In Descriptive Programming objects are identified by describing all the identification properties

## 9. After creating the test, what is the purpose of running them?

- a. To Check the Application: The test starts running from the first line in our test and stops at the end of the test. While running, QTP connects to our application and performs each operation in our test, including any checkpoints, such as checking any text strings, objects, tables, and so forth. If we had parameterized our test with Data Table parameters, QTP repeats the test (or specific actions in your test) for each set of data values we had defined.
- b. To Debug the Application: We can control our run session to help us identify and eliminate defects in our test. We can use the Step Into, Step Over, and Step Out commands to run our test step by step. We can begin our run session from a specific step in our test, or run the test until a specific step is reached. We can also set breakpoints to pause our test at predetermined points. We can view the value of variables in our test each time it stops at a breakpoint in the Debug Viewer.
- c. To Update the Application: We can run our test using Update Run Mode to update the property sets used for test object descriptions, the expected checkpoint values, the data available to retrieve in output values, and the Active Screen images and values. We can run our test using Maintenance Run Mode when we know that our application has changed, and we therefore expect that QTP will not be able to identify the objects in our test. When we run tests in Maintenance Run Mode, a wizard opens for steps that fail because an object

could not be found in the application. The wizard then guides us through the steps of resolving the issue, and, after we resolve the issue, the run continues.

#### **10. How can we do the Analysis of Results in QTP?**

- a. After we run our test, we can view the results.
  - i. View the results in the Test Results window: After we run our test, we can view the results of the run in the Test Results window. We can view a summary of our results as well as a detailed report. If we had captured still images or movies of our application during the run, we can view these from the Test Results window.
  - ii. Report defects detected during a run session: If we have access to Quality Center, the HP centralized quality solution; we can report the defects we discover to the project database.
  - iii. We can instruct QTP to automatically report each failed step in our test, or we can report them manually from the Test Results window.

#### **11. What is the method used by QTP to learn objects?**

- a. QTP “looks” at the object being learned and stores it as a test object, determining in which test object class it fits. In the same way, Bob immediately checked whether the item was a person, animal, plant, or inanimate object. QTP might classify the test object as a standard Windows dialog box, a Web button, or a Visual Basic scroll bar object, for example.
- b. Then, for each test object class, QTP has a list of mandatory properties that it always learns; similar to the list of characteristics that Bob planned to learn before seeing the picture.
- c. When QTP learns an object, it always learns these default property values, and then “looks” at the rest of the objects on the page, dialog box, or other parent object to check whether this description is enough to uniquely identify the object.
- d. If it is not, QTP adds assistive properties, one by one, to the description, until it has compiled a unique description.
- e. If no assistive properties are available, or if those available are not sufficient to create a unique description, QTP adds a special ordinal identifier, such as the object’s location on the page or in the source code, to create a unique description.

#### **12. What are the Test Object Properties in QTP?**



- a. Test object properties are the properties whose values are captured from the objects in our application when QTP learns the object. QTP uses the values of these properties to identify run-time objects in our application during a run session.
- b. Property values of objects in our application may change dynamically each time our application opens, or based on certain conditions. We may need to modify the test object property values to match the run-time object property values. We can modify test object properties manually while designing our test, or use SetTOProperty statements during a run session.

### **13. How to decide on whether to save the objects in Local or Shared Object Repositories?**

- a. **Local object repository** is easiest to use when we are **creating simple tests**, especially under the following conditions:
  - i. We have only one, or very few, tests that correspond to a given application interface, or set of objects.
  - ii. We do not expect to frequently modify object properties.
  - iii. We generally create single-action tests.
- b. Shared object repository is the preferred option when:
  - i. We are creating tests using keyword-driven methodologies & not by recording.
  - ii. We have several tests that test elements of the same application, interface, or set of objects.
  - iii. We expect the object properties in our application to change from time to time and we regularly need to update or modify object properties.
  - iv. We often work with multi-action tests and regularly use the Insert Copy of Action and Insert Call to Action options.

### **14. What is the use of Ordinal Identifiers in QTP?**

- a. 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 for objects having the same values for all properties.
- b. This ordered value provides a backup mechanism that enables QTP to create a unique description to recognize an object when the defined properties are not sufficient to do so.
- c. We can specify the ordinal identifier for test objects in the local object repository using the Object Repository window or Object Properties dialog box, and for test objects in the shared object repository using the Object Repository

Manager.

**15.How do you Import data from a ".xls" file to the in-built Data table during Runtime.**

- a. Datatable.Import "...XLS file name..."
- b. DataTable.ImportSheet(FileName, SheetSource, SheetDest)
- c. DataTable.ImportSheet "C:\name.xls" ,1 ,"name"

**16.An action has both shared and local OR associated to it and both have the same object in them. In the test which one will be considered?**

- a. If a local OR and Shared OR have an object with the same name, the action will consider the object in its local OR.

**17.What is the QTP Testing process?** (From a Record & Run Perspective)

- a. Quick Test Professional's testing process consists of 7 main phases:
  - i. Create your test plan - This is preparatory phase where you identify the exact test steps, test data and expected results for you automated test. You also identify the environment and system configurations required to create and run your QTP Tests.
  - ii. Recording a session on your application - During this phase, you will execute test steps one by one on your AUT, and QTP will automatically record corresponding VB script statements for each step performed.
  - iii. Enhancing your test - In this stage you will insert checkpoints, output values, parameterization, programming logic like *if...else* loops to enhance the logic of your test script.
  - iv. Replay & Debug - After enhancements you will replay the script to check whether it's working properly and debug if necessary.
  - v. Run your Tests - In this phase you will perform the actual execution of your Test Script.
  - vi. Analyzing the test results - Once test run is complete, you will analyze the results in the Test Fusion report generated.
  - vii. Reporting defects - Any incidents identified needs to be reported. If you are using Quality Center, defects can be automatically raised for failed tests in QTP.

**18.What is the difference between Checkpoint and Output value.**

- a. Checkpoint is a verification point that compares a current value for a specified property with the expected value for that property. Based on this comparison, it will generate a PASS or FAIL status.
- b. Output value is a value captured during the test run and can be stored in a specified location like the Databse or even a variable. Unlike Checkpoints, no PASS/FAIL status is generated.

**19.What is the extension for a function library?**

- a. The extension is '.QFL'

**20.How does QTP identify objects?**

- a. HP QTP identifies any GUI Object based on its corresponding properties. While recording, QTP identifies and store peculiar properties (as defined in the Object Identification settings) in the object repository of the GUI object. At run-time, QTP will compare the stored property values with the on-screen properties, to uniquely identify the GUI object.

**21.How will you call from one action to another action**

- a. HP QTP allows calling an action in 2 ways:
  - i. Call to copy of Action - The original action is copied in its entirety, including checkpoints, parameterization, the corresponding action tab in the Data Table, any defined action parameters, local object repository. The action is inserted into the test as an independent, non-reusable action (even if the original action was reusable). After the action is copied into your test, you can add to, delete from, or modify the action just as you would with any other non-reusable action. Any changes you make to this action after you insert it affect only this action, and changes you make to the original action do not affect the copied action.
    - 1. To view: Right-click & select Action > Insert Call to Copy.
  - ii. Call to Existing an Action - In this approach, a link is created to the called Action. User can view the steps of the action in the action view, but you cannot modify them. The called action's local object repository (if it has one) is also read-only. If the called external action has data in the Data Table, however, you can choose whether you want the data from the action's data sheet to be imported as a local, editable copy, or whether you want to use the (read- only) data from the original action. (Columns and data from the called action's global data sheet are always imported into the calling test as a local, editable copy.) To modify a called, external

action, you must open the test with which the action is stored and make your modifications there. The modifications apply to all tests that call that action. If you chose to use the original action's data when you call an external action, then changes to the original action's data are applied as well.

1. To view: Right-click & select Action > Insert Call to Existing Action.

## **22.What is the file extension of the script file and OR in QTP**

- a. In QuickTest, File extension is as follows –
  - i. Per test Object Repository: filename.mtr
  - ii. Shared Object Repository: filename.tsr
  - iii. Code file extension id: script.mts

## **23.What is automation framework and the different types of Test Automation**

### **Frameworks**

- a. **Automation Framework** is a **set of guidelines** like coding standards, test-data handling, object repository treatment etc. It **increases code re-usage, higher portability, reduced script maintenance cost etc.** These are **just guidelines and not rules**; they are not mandatory and you can still script without following the guidelines. But you will miss out on the advantages of having a Framework. **A test automation framework is a set of assumptions, concepts, and practices that provide support for automated software testing.**
- b. Type of Framework:
  - i. Linear Scripting: The **simplest one**, also known as "**Record & Playback**". In this approach, Tester manually records each step (Navigation and User Inputs), Inserts Checkpoints (Validation Steps) then, Plays back the recorded script in the subsequent rounds.
    1. Advantages
      - a. Fastest way to generate script
      - b. Automation expertise not required
      - c. Easiest way to learn the features of the Testing Tool
    2. Disadvantages
      - a. Little reuse of scripts
      - b. Test data is hard coded into the script
      - c. Maintenance Nightmare
  - ii. Data-Driven Testing Framework: In this approach, while **Test case logic resides in Test Scripts**, the **Test Data is separated and kept outside the Test Scripts**. Test Data is read from the external files (**Excel Files, Text**

Files) and are loaded into the variables inside the Test Script. Variables are used both for Input values and for Verification values. Test Scripts themselves are prepared either using Linear Scripting or Test Library Framework.

1. Advantages

- a. Changes to the Test Scripts do not affect the Test Data
- b. Test Cases can be executed with multiple Sets of Data
- c. A Variety of Test Scenarios can be executed by just varying the Test Data in the External Data File

2. Disadvantages

- a. More time is needed to plan and prepare both Test Scripts and Test Data

iii. Keyword-Driven Framework: In this approach, keywords are developed, independent of the test automation tool used to execute them. Tests can be designed with or without the Application. The functionality of the application-under-test is documented in a table as well as in step-by-step instructions for each test. There are 3 basic components of a Keyword Driven Framework viz. Keyword, Application Map, Component Function.

- 1. **Keyword** - Keyword is an Action that can be performed on a GUI Component. Example: For GUI Component Textbox some Keywords (Action) would be InputText, VerifyValue, VerifyProperty and so on.
- 2. **Application Map**- An Application Map Provides Named References for GUI Components. Application Maps are nothing but “Object Repository”.
- 3. **Component Function** - Component Functions are those functions that actively manipulate or interrogate GUI component. Example: A function would be click on web button with all error handling, enter data in a Web Edit with all error handling. Component functions could be application dependent or independent.

a. Advantages

- i. Provides high code re-usability
- ii. Test Tool Independent
- iii. Independent of Application Under Test, same script works for AUT (with some limitations)
- iv. Tests can be designed with or without AUT

b. Disadvantages

- i. Initial investment being pretty high, the benefits of this can only be realized if the application is considerably

big and the test scripts are to be maintained for quite a few years.

- ii. High Automation expertise is required to create the Keyword Driven Framework.

## **24.How does QTP identify an object?**

- a. QTP has a predetermined set of properties that it learns/stores for every class of object it identifies.
  - i. There are **3 aspects to this**:
    1. Mandatory properties: This is the list of properties for a certain class that QTP always stores. We could say that this is the object description. It also checks this in conjunction with the parent object to see if the description is sufficient to identify the object uniquely.
    2. Assistive properties: In case the description of mandatory properties is insufficient to identify the Object a set of Non-Mandatory properties will be added to the description one after the other until there is enough data to identify the object.
    3. Ordinal Identifier: If the assistive properties also do not result in unique identification of an object a special ordinal identifier is added by QTP, such as the object's location on the page or in the source code.

## **25.What is difference between Run time object and Test object?**

- a. The difference between Run time Object and Test object are:
  - i. **Run time object** are actual object in the application whereas test object are reference of the actual object.
  - ii. **Run time object** always have same name whereas test object name varies in different environment.
  - iii. **Test object** are used to identify the actual object in the application which is run time objects.
  - iv. **Run time object** resides in the application whereas test object resides in the object repository.

## **26.Can we create a QTP test from QC?**

- a. **Yes** we can create QTP test from QC but we must first make sure QTP has the ability to execute tests from Quality Center. Please ensure the following option in QTP Run Settings is enabled:
- i. When QTP is enabled, follow the below steps to schedule and execute tests from Quality Center:
    1. Login to Quality Center and Navigate to Test Lab Module
    2. After selecting the correct Test Set, Click the Execution Flow Tab.
    3. Right-click on the test that requires configuration of Time Dependency and click Test Run Schedule.
    4. In the Run Schedule window, select the Time Dependency tab. The time and date of execution can be configured.
    5. Time dependency will be added to the relevant test.
    6. After time dependency has been added, navigate back to the Execution Grid pane. From the Execution Grid, select the tests to be run at the designated date and time.
    7. From the Automatic Runner dialog, click Run All.
    8. Once Run All is clicked from the Automatic Runner dialog, the test status will change to Waiting and QC will fire the tests to be run at the scheduled date and time.
    9. QC will fire the tests in the sequence configured in the Execution Flow pane. The Test Run Scheduler will show all the tests that were selected and are executing and are to be executed.