QTP Tutorial

Absolute Beginners
ABOUT THE TUTORIAL

QTP Tutorial

HP QuickTest Professional (QTP), an automated functional testing tool that helps testers to perform automated regression testing in order to identify any gaps, errors/defects in contrary to the actual/desired results of the application under test.

This tutorial will give you an indepth understanding on HP QuickTest Professional, it's way of usage, record and play back of tests, object repository, actions, checkpoints, sync points, debugging, test results etc and other related terminologies.

Audience

This tutorial is designed for Software Testing Professionals with a need to understand the QTP in enough detail along with its simple overview, and practical examples. This tutorial will give you enough ingredients to start with QTP from where you can take yourself at higher level of expertise.

Prerequisites

Before proceeding with this tutorial you should have a basic understanding of software development life cycle (SDLC). A basic understanding of VBScript is also required. You can also go through the basics of VBScript here.

Copyright & Disclaimer Notice

©All the content and graphics on this tutorial are the property of tutorialspoint.com. Any content from tutorialspoint.com or this tutorial may not be redistributed or reproduced in any way, shape, or form without the written permission of tutorialspoint.com. Failure to do so is a violation of copyright laws.

This tutorial may contain inaccuracies or errors and tutorialspoint provides no guarantee regarding the accuracy of the site or its contents including this tutorial. If you discover that the tutorialspoint.com site or this tutorial content contains some errors, please contact us at webmaster@tutorialspoint.com
# Table of Content

QTP Tutorial .......................................................... 2
Audience ........................................................................ 2
Prerequisites .................................................................. 2
Copyright & Disclaimer Notice ....................................... 2
QTP-Introduction .......................................................... 7
Testing Tools: .................................................................. 7
CLASSIFICATION OF TOOLS ........................................ 7
TYPES OF TOOLS: .......................................................... 7
  WHERE QTP FITS IN? .................................................. 8
  QTP HISTORY AND EVOLUTION: ............................ 8
  VERSION HISTORY: ................................................. 8
  ADVANTAGES: .......................................................... 8
  DISADVANTAGES: ....................................................... 9
Test Automation ........................................................... 10
Environment Setup ........................................................ 12
Installation Procedure : .................................................. 12
Launching UFT and Addins Page: ..................................... 23
Record and Playback ....................................................... 27
Significance of Record and Playback: ............................... 33
Modes of Recording: ....................................................... 34
How to Choose Recording Modes: .................................... 34
Object Repository .......................................................... 36
Merging Object Repositories ............................................. 50
Steps to Merge OR's ....................................................... 51
Index: ........................................................................... 53
Location: ........................................................................ 53
Creation Time: .............................................................. 54
Actions ........................................................................... 56
Types of Actions: ........................................................... 57
Working with Actions: ..................................................... 58
Datatables ........................................................................ 67
Example: .......................................................................... 68
Data Table Operations: ................................................... 69
Example: .......................................................................... 72
Checkpoints ..................................................................... 73
Types: ............................................................................. 73
Inserting CheckPoint ....................................................... 74
Example: ................................................................. 75
Viewing Checkpoint Properties ........................................ 75
Synchronization ....................................................... 78
Ways to Insert Sync Point: ........................................... 78
METHOD 1: WAITPROPERTY ........................................ 78
METHOD 2: EXIST .................................................. 79
METHOD 3: WAIT .................................................. 79
METHOD 4: SYNC METHOD ........................................ 79
METHOD 5 : INSERTING QTP INBUILT SYNCHRONIZATION POINTS: 79
Default Synchronization: .......................................... 80
Smart Identification .................................................. 82
Enabling Smart identification for an object: ....................... 82
Debugging .................................................................. 87
Options in Break Point: ............................................. 87
Debugging Pane: ..................................................... 88
Example: .................................................................. 88
Error Handling ......................................................... 90
Error Types: .................................................................. 90
  SYNTAX ERRORS: .................................................. 90
  LOGICAL ERRORS: ................................................ 91
  RUNTIME ERRORS: ............................................... 91
  HANDLING RUN-TIME ERRORS: .............................. 92
  EXAMPLE: .................................................................. 93
Recovery Scenarios ....................................................... 94
Steps to create Recovery Scenario: ................................. 94
Verification: .................................................................. 100
Environment Variables ................................................. 102
Types of Environment Variables .................................... 102
Environment Variables - Supported Methods: .................. 105
Library Files ............................................................. 108
Associating Function Libraries ....................................... 108
Test Results .............................................................. 111
Operations performed in Test Results: ............................ 113
CONVERTING RESULTS TO HTML .............................. 113
  FILTERING THE RESULTS ....................................... 113
  RAISING DEFECTS ................................................ 114
Test Results: ............................................................ 115
Working with GUI Objects ............................................ 118
  WORKING WITH TEXT BOX ..................................... 118
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISADVANTAGES</td>
<td>140</td>
</tr>
<tr>
<td>Hybrid Framework</td>
<td>140</td>
</tr>
<tr>
<td>Affecting Factors</td>
<td>141</td>
</tr>
<tr>
<td>Designing a Framework</td>
<td>141</td>
</tr>
<tr>
<td>- MASTER DRIVER SCRIPT</td>
<td>142</td>
</tr>
<tr>
<td>- LIBRARY FILES</td>
<td>144</td>
</tr>
<tr>
<td>- OBJECT REPOSITORY:</td>
<td>148</td>
</tr>
<tr>
<td>- DATA TABLE</td>
<td>149</td>
</tr>
<tr>
<td>- THE EXECUTION LOG:</td>
<td>150</td>
</tr>
</tbody>
</table>
QTP-Introduction

QTP stands for QuickTest Professional, a product of Hewlett Packard (HP). This tool helps testers to perform an automated functional testing seamlessly without monitoring once script development is complete. HP QTP uses Visual Basic Scripting (VBScript) for automating the applications. The Scripting Engine need not be installed exclusively as it is available part of the Windows OS. The Current version of VBScript is 5.8 which is available as part of Win 7. VBScript is NOT a object oriented language but a object based language.

Testing Tools:

Tools from a software testing context, can be defined as a product that supports one or more test activities right from planning, requirements, creating a build, test execution, defect logging and test analysis.

CLASSIFICATION OF TOOLS

Tools can be classified based on several parameters. It includes,

- The purpose of the tool
- The Activities that are supported within the tool
- The Type/level of testing it supports.
- The Kind of licensing (open source, freeware, commercial)
- The technology used

TYPES OF TOOLS:

<table>
<thead>
<tr>
<th>S.No#</th>
<th>Tool Type</th>
<th>Used for</th>
<th>Used by</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Test Management Tool</td>
<td>Test Managing, scheduling, defect logging, tracking and analysis.</td>
<td>testers</td>
</tr>
<tr>
<td>2.</td>
<td>Configuration management tool</td>
<td>For Implementation, execution, tracking changes</td>
<td>All Team members</td>
</tr>
<tr>
<td>3.</td>
<td>Static Analysis Tools</td>
<td>Static Testing</td>
<td>Developers</td>
</tr>
</tbody>
</table>
WHERE QTP FITS IN?

QTP is a Functional testing tool which is best suited for regression testing of the applications. QTP is a licensed/commercial tool owned by HP which is one of the most popular tools available in the market. It compares the actual and expected result and reports the results in the execution summary.

QTP HISTORY AND EVOLUTION:

HP Quick Test Professional was originally owned by Mercury Interactive and it was acquired by Hp. Its original name was Astra Quick Test and later named as Quick Test Professional but the latest version is known as Unified Functional Tester (UFT).

VERSION HISTORY:

Now let us take a look at the version history of QTP.

<table>
<thead>
<tr>
<th>Versions</th>
<th>Timelines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Astra Quick Test v1.0 to v5.5 - Mercury Interactive</td>
<td>May 1998 to Aug 2001</td>
</tr>
<tr>
<td>QuickTest Professional v6.5 to v9.0 - Mercury Interactive</td>
<td>Sep 2003 to Apr 2006</td>
</tr>
<tr>
<td>Hp-QuickTest Professional v9.1 to v11.0 - Acquired and Released by HP</td>
<td>Feb 2007 to Sep 2010</td>
</tr>
<tr>
<td>Hp-Unified Functional Testing v11.5 to v11.53</td>
<td>2012 to Nov 2013</td>
</tr>
</tbody>
</table>

ADVANTAGES:

- Developing automated tests using VBScript doesn't require a highly skilled coder and relatively easy when compared other object oriented programming languages.
- Easy to use, ease of navigation, results validation and Report generation.
- Readily Integrated with Test Management Tool (Hp-Quality Center) which enables easy scheduling and Monitoring.
- Can also be used for Mobile Application Testing.
• Since it is a Hp product, the full support is provided by HP and by its forums for addressing technical issues.

DISADVANTAGES:

• Unlike Selenium, QTP works in Windows operating system only.

• Not all versions of Browsers are supported and the testers need to wait for the patch to be released for each one of the major versions.

• Having said that it is a commercial tool, the licensing cost is very high.

• Even though scripting time is less, the execution time is relatively higher as it puts load on CPU & RAM.
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 a 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.
Environment Setup

QTP Installation

QTP is a commercial tool and trial version can be downloaded from HP site directly. Only the current version which is Unified functional testing (11.5x) is available for download. Below is the URL from where the trial version can be downloaded.


Installation Procedure:

Steps 1 - Click "Trials and Demos" link and select "Hp Unified Functional Testing 11.50 CC English SW E-Media Evaluation" as shown below:
Step 2 - Upon Selecting "Hp Unified Functional Testing 11.50", the download wizard opens. Fill in the Personal details and click next
Step 3 - Read the terms of use and click "NEXT".
Step 4 - Download window opens. Now, click on "Download" Button.
Step 5 - The downloaded file will be of the format .RAR. Now you need to unzip the archive and the folder contents would be as shown below and execute the Setup.exe.
**Step 6** - Upon Executing the Setup File, inorder to install, select "Unified Functional Testing Set up" from the list as shown below:
Step 7 - Then Click Next to Continue.
Step 8 - In the Custom Setup Window, select the plugins that are required for your automation. i.e. You Should select the plugins based on the technology of your application under test. For Example, If your application is based on .NET then you should ensure that you select .NET.
Step 9 - Upon selecting the required plugins for Installation, Click Next and upon completion of the installation you will end up with a Finish button Window.
Step 10 - Once you complete your installation, the “Additional Installation Requirements” Dialog box opens. Select everything in the list other than “Run License Installation Wizard” and click “RUN”. We Need NOT select “Run License Installation Wizard” because we are installing the trial version which by default gives a license for 30 days.
Step 11 - Upon completion of Additional Installation Requirements, a tick mark is shown which in turn states that the components are installed successfully. Now, click close:
Launching UFT and Addins Page:

**Step 1** - After Installation, application can be launched from the Start Menu as shown in the figure.
Step 2 - The License page appears. You can click on continue as we have installed the trial license.

Step 3 - The Addins Dialog box opens for the user to select the required addins' DONOT load all the addin's but just the required addins and click "Ok" button.
Step 4 - Upon loading the required addins the UFT 11.5 tool opens for the user and the first glimpse of the UFT looks as shown below:
There are no tools available for the current document.

What's new in HP Unified Functional Testing 11.53

HP Unified Functional Testing brings a new resolution into the functional testing area. The result is a 'one stop shop' for GUI, API, and business process testing. HP Unified Functional Testing includes its predecessors' capabilities along with new exciting features and up-to-date environment and technology integration support. Now is the time to start exploring and building new tests like you've never could before!

No data available
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 are as follows:

Step 1: Click on "New" test from the Start Page as shown below:
Step 2: Upon clicking the "New" link, the new test window opens and the user needs 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. 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 webaddress 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:

```
Browser("Free Online Math Calculator").Page("Free Online Math Calculator").link("Numbers").Click
Browser("Free Online Math Calculator").Page("Numbers Calculator - Math").link("Square Root").Click
Browser("Free Online Math Calculator").Page("Square Root Calculator").webEdit("n").set("10")
Browser("Free Online Math Calculator").Page("Square Root Calculator").webButton("calculate").Click
```

**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:
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.

How to Choose Recording Modes:

Upon clicking on Recording button, user can choose the recording mode from the recording pane that appears on the screen once recording starts. The selection can be made from any the ones that has been discussed above.

![Recording GUI](image)

The Below Scenario is recorded in all the modes and see how the same action has been recorded under various circumstances.

2. Click "Numbers" under "Algebra"
3. Click "Square Root" link
4. Enter a value to calculate the square root. Let us say 10
5. Hit Calculate

Script Recorded under Default, Analog and Low Level Recording Mode.

```csharp
' DEFAULT RECORDING MODE
Browser("Free Online Math Calculator").Page("Free Online Math Calculator").Link("Numbers").Click
Browser("Free Online Math Calculator").Page("Numbers Calculator - Math").Link("Square Root").Click
Browser("Free Online Math Calculator").Page("Square Root Calculator").WebEdit("n").Set "10"
Browser("Free Online Math Calculator").Page("Square Root Calculator").WebButton("calculate").Click
```
' ANALOG RECORDING MODE
Desktop.RunAnalog "Track1"

' LOW LEVEL RECORDING MODE
Window("Windows Internet Explorer").WinObject("Internet Explorer_Server").Click 235,395
Window("Windows Internet Explorer").WinObject("Internet Explorer_Server").Click 509,391
Window("Windows Internet Explorer").WinObject("Internet Explorer_Server").Click 780,631
Window("Windows Internet Explorer").WinObject("Internet Explorer_Server").Type "10"
Window("Windows Internet Explorer").WinObject("Internet Explorer_Server").Click 757,666

The Recordings using insight recording mode will be as shown below:

' INSIGHT RECORDING MODE
Browser("Free Online Math Calculator").InsightObject( ).Click
Browser("Free Online Math Calculator").InsightObject( ).Click
Browser("Free Online Math Calculator").InsightObject( ).Click
Browser("Free Online Math Calculator").InsightObject( ).Click
Window("Windows Internet Explorer").WinObject("Internet Explorer_Server").Type "10"
Browser("Free Online Math Calculator").InsightObject( ).Click
Browser("Free Online Math Calculator").InsightObject( ).Click
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.

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

Object Spy and its Features

Object Spy is a utility/option within QTP to add objects to the Object Repository. Object Spy can be accessed from the tool bar as shown below:

1. Upon Clicking on the Object Spy icon, the Object Spy Dialog box opens. The Objects can be added to the repository upon clicking the pointing hand.
2. After spying the object, the object hierarchy will be shown. Let us say, we are spying the "Numbers" link at "http://easycalculation.com/". The Object properties will be as shown below.
3. After Spying an object, we can click on "Highlight" option to highlight the object in the application.

4. For adding the object into the Object Repository, one should click on "Add objects" button in the Object spy dialog.

5. The properties and its values are displayed for the selected object in the dialog box which should be Unique for QTP to recognize the objects while the script executes.

6. The supported operations on the object can be retrieved by clicking on the operation tab. Operations such as "click" for a button, "Set" for a text box are retrieved from the "operations" tab as shown below:
Working with Object Repository

Adding Objects into OR:

After Spying the object, adding the objects into Repository is the first step. The Script can execute successfully if and only if the objects are added into the Object Repository. Upon Clicking "Add Objects to OR", the objects are added into Object repository.

Even when a user does a recording, the objects and its properties are captured automatically. Hence we are able to replay the script successfully.

Object Repository - Features:

1. After adding objects to the OR, we can verify by navigating to "Resources" -> "Object Repository". The Object Repository Window opens and we can locate the added object in the Repository as shown below:
2. One can add the properties additionally apart from the default ones by clicking on "+" button and remove it using "x" button. If we want to restore to defaults we can click on the "circular arrow" button.
3. One can change the object's Name. Here the name of the object is "Numbers" that can be renamed to "num" which will NOT have any effect on identifying the object uniquely. If there is a change in logical name, the same name should be used while scripting. Only the Object's Name can be changed and NOT its Properties.
4. Properties of any object must be unique so that QTP will be able to recognize the objects and act on it. If the object properties are same for two or more objects then during execution an error would be thrown that "More than one object is matched for the specified properties".

5. After adding the objects, the same can be used in script by simple drag-drop as shown in the figure. When the object is dragged and dropped, the default operation is set. Eg: Click for a button, Set for a Text Box etc.
Types of Object Repository

Based on Context, the Object Repository is of two types.

**Local Object Repository** - As the Name Suggests, the Object Repository is applicable only for that Action. As we know that QTP creates a New Test with 1 action by default. Local Object Repository can be opened by traversing to Resources -> gt:Object Repository. This is the default OR in QTP.
Shared Object Repository - The Object Repository is shared across actions/modules which would be mapped for two or more actions. Local objects can be exported to be saved into Shared Object Repository by using the option "Export Local Objects" Options. Shared Object Repository can be opened by traversing to Resource ->gt; Object Repository Manager
Below are the major difference between Local

<table>
<thead>
<tr>
<th>Local Object Repository (LOR)</th>
<th>Shared Object Repository (SOR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>This Object Repository is available one for each action.</td>
<td>This type of OR is available for multiple tests and for multiple actions.</td>
</tr>
<tr>
<td>This is the OR that is available for each tests by default.</td>
<td>This type of OR is usually used in frameworks considering reusability and</td>
</tr>
</tbody>
</table>
Local Object Repository is Editable in Object Repository.

Shared Object Repository is read only by default but can be edited in Object Repository Manager.

It is NOT a standalone file that can edited.

SOR is a standalone file that can be edited easily.

It is Saved with an Extension .bdb

SOR is saved with an extension .tsr

It Should be used when not many tests are working on the same screens of the application under test.

SOR Should be used when there are different scripts interacting with the same type of the object.

**User-defined Objects**

Sometimes, All Objects are NOT recognizable by QTP if incase the application does NOT use Standard Windows Classes. QTP uses Class Name to find the type of Object. Sometimes, the object is expected to behave like a button or Combo Box etc.

When we try to add such kind of button, it might recognize as Winobject.

Hence we can map that WinObject to behave like "CheckBox" Object type by Navigating to Tools -> Object Identification, and select Environment as "Standard Windows" and click "User Defined" button. Please Note this option won't work against any other environment.
In the below example, an object of a specific class is made to recognize as an object of Type "button". Hence this object inherits all properties of a button and we can use the objects that are supported by button.
Object Repository in XML

Object Repository can be saved in XML format so that size of the Object repository is reduced. The OR can be saved as XML by “Exporting as XML” from Shared object Repository Window as shown below:

The Same object repository can be imported from XML and can be edited/deleted and exported back to XML.

Comparing and Merging OR

Comparing Object Repositories:

Many a times, we might be in a position to compare two object repositories to spot the difference and merge it if incase objects are missing in the main repository.

These tools help us to a very great extent to spot the differences in the Object Repository

Steps to Compare OR's

Step 1: Navigate to "Resources" >> "Object Repository Manager"
Step 2: Goto “Tools” >> “Object Repository Comparison Tool”
Step 3: The Object Repository comparison Window opens and the user need to select the two Object Repository files to be compared.

Step 4: It performs the Comparison and displays the differences one by one as shown below:
Step 5: Even one can filter based on 3 parameters viz - unique objects, identical objects, partial match objects.

Merging Object Repositories

Some times there are circumstances when Object repository needs to merged as maintaining two or more object repository adds additional overheads.

Merging Option is a great boon for testers who need to merge two object repository without loosing the object hierarchy.
Steps to Merge OR's

**Step 1**: Navigate to "Resources" >> "Object Repository Manager"

**Step 2**: Goto "Tools" >> "Object Repository Comparison Tool"

**Step 3**: The Object Repository Merge Window opens and the user need to select the two Object Repository files to be merged.

**Step 4**: Upon merging, the statistics are shown to the user.
Step 5: Even one can filter based on 2 parameters viz - show all objects, show only conflicting objects.

Ordinal Identifiers

What are Ordinal Identifiers?

Sometimes there are series of objects with same class name and properties. Lets us say in a window there are series of checkboxes with the same set of properties. If we want to act on those objects we need to uniquely identify them so that QTP will be able to act on it.

An Ordinal Identifier assigns a numerical value to the test objects which indicate its location or order relative to its group. The Ordered value enables QTP to recognize it uniquely when the inbuilt properties are NOT sufficient to do so.

There are 3 Ordinal Identifiers in QTP that can be used in different context:
- Index
- Location
- Creation Time

**Index:**

An object appearing first in the page/Window will have a smaller Index value when compared to another object that comes later in the same page/Window.

The Value of index for the group of text boxes will be as follows:

<table>
<thead>
<tr>
<th>Object Name</th>
<th>Index Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TextBox 1</td>
<td>0</td>
</tr>
<tr>
<td>TextBox 4</td>
<td>1</td>
</tr>
<tr>
<td>TextBox 2</td>
<td>2</td>
</tr>
<tr>
<td>TextBox 5</td>
<td>3</td>
</tr>
<tr>
<td>TextBox 3</td>
<td>4</td>
</tr>
<tr>
<td>TextBox 6</td>
<td>5</td>
</tr>
</tbody>
</table>

**Location:**

The Location property works vertically from top to bottom and from left to right. Hence for the same case, the value of location for the group of text boxes will be as follows:

<table>
<thead>
<tr>
<th>Object Name</th>
<th>Index Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TextBox 1</td>
<td>0</td>
</tr>
<tr>
<td>TextBox 2</td>
<td>1</td>
</tr>
</tbody>
</table>
Creation Time:

The Creation Time property holds good only for web based application. When we open 2 browser sessions of the same website, QTP will not be able to recognize the window as both the window will have the same set of properties. Hence we can use creation time with which QTP will be able to act on the window.

| TextBox 3 | 2 |
| TextBox 4 | 3 |
| TextBox 5 | 4 |
| TextBox 6 | 5 |

Will have CreationTime value = 0
SystemUtil.Run "iexplore.exe", "http://www.google.com"

Will have CreationTime value = 1
SystemUtil.Run "iexplore.exe", "http://www.yahoo.com"

Will have CreationTime value = 2

Will have CreationTime value = 3

Hence to work on a specific browser, we need to explicitly mention the Creation time in OR or we can use description of objects which we will see in detail in descriptive programming section.

'Sync's www.google.com
Browser("creationtime:="").Sync

'Gets the RO text property of www.yahoo.com
Browser("creationtime:=1").GetROProperty("text")

'Highlights microsoft.com
Browser("creationtime:=2").Highlight

Child Objects

What are Child Objects?

The objects (text box, combo box, links) contained in the frame or Window is known as child objects. Sometimes we would be in a situation to get the properties of all the links in a webpage or to get the values of all radio buttons in a window.

In these circumstances if we want to work on the child objects, we need to use description of objects which we will be able to work on all the objects in a particular window/page. Descriptive programming will be dealt in detail in the upcoming chapter but the significance of this chapter is to understand child objects and its usage.

The below Script gets the all the name of the links from the website "http://easycalculations.com/"

Dim oDesc
Set oDesc = Description.Create
oDesc("micclass").value = "Link"

'Find all the Links
Set obj = Browser("Math Calculator").Page("Math Calculator").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

The Result is printed in the output window as shown below:
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.

<table>
<thead>
<tr>
<th>Actions</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actions are inbuild feature of QTP.</td>
<td>VBScript Functions are supported by both VBScript and QTP.</td>
</tr>
<tr>
<td>Actions parameters are passed byvalue only.</td>
<td>Function parameters are passed either by byvalue or byref.</td>
</tr>
<tr>
<td>Actions have extension .mts</td>
<td>Functions are saved as .vbs or .qfl</td>
</tr>
<tr>
<td>Actions may or maynot be reusable.</td>
<td>Functions are always reusable.</td>
</tr>
</tbody>
</table>

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:

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.

<table>
<thead>
<tr>
<th>Action Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert Call to New Action</td>
<td>Inserts a New Action from the existing action</td>
</tr>
<tr>
<td>Insert Call to Copy of Action</td>
<td>Inserts a copy of an existing action</td>
</tr>
<tr>
<td>Insert Call to Existing Action</td>
<td>Inserts a call to existing re-usable action</td>
</tr>
</tbody>
</table>

Insert Call to New Action

Testers can insert a new action at any point of the script by performing the following steps:

**Step: 1** - Right Click on the Scripting area and select "Call to New Action"

**Step: 2** - In the "Insert Call to New Action" Window, give the test name, description, and also specify if it is a reusuable action or not. Most importantly select the location the action to be inserted.
Step 3 - User can check the changes graphically in the test Name Tab as shown below:
We can also use QTP commands to call the Action at any point in the script.

```
RunAction "Calculate", oneIteration  'Executes Calculate Action for one iteration.
```

Action can be called with Parameters as shown below:

```
'Input to the action
num1 = 5
num2 = 10
Dim value1

'Run the action with parameters
OutputValue = RunAction("Calculate", oneIteration, num1, num2, value1)

'Display the output
print OutputValue
```

Insert Call to Copy of Action

Testers can insert a copy of an existing action at any point of the script by performing the following steps:

**Step : 1** - Right Click on the Scripting area and select "Call to Copy of Action"
Step : 2 - In the "Insert Call to Copy of Action" Window, Select the "Test Name", "Action Name" and also select the location the action to be inserted.
Step : 3 - Immediately script is autogenerated to show that the copy of an action is inserted.

```
RunAction "Copy of Calculate", oneIteration
```

Step : 4 - User Can check the changes graphically in the test Name Tab as shown below:
Insert Call to Existing Action

Testers can insert an Existing action at any point of the script by performing the following steps:

**Step : 1** - Right Click on the Scripting area and select "Call to Existing Action"
Step 2: In the "Select Action" Window, give the test name, Action name, description and also specify the location of the action to be inserted.
Step 3 - Once inserted, the below script is generated exactly in the location where the action was inserted.

```
RunAction "Action1", oneIteration
```

Step 4 - User can check the changes graphically in the test Name Tab as shown below:
Datatables

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 Datatables.

- **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 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 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:

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Table Methods</td>
<td>Gives Detailed information about the data table methods.</td>
</tr>
<tr>
<td>DTParameter Object Methods</td>
<td>Gives Detailed information about the DTParameter methods.</td>
</tr>
<tr>
<td>DTSheet Object Methods</td>
<td>Gives Detailed information about the DTSheet methods.</td>
</tr>
</tbody>
</table>

Data Table Methods

Data Table Object Methods:

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddSheet</td>
<td>Adds the specified sheet to the run-time data table</td>
</tr>
<tr>
<td>DeleteSheet</td>
<td>Deletes the specified sheet from the run-time data table</td>
</tr>
<tr>
<td>Export</td>
<td>Exports the Datatable to a new file in the specified location</td>
</tr>
<tr>
<td>ExportSheet</td>
<td>Exports a Specific Sheet of the Datatable in run-time</td>
</tr>
<tr>
<td>GetCurrentRow</td>
<td>Returns the active row of the run-time data table of global sheet</td>
</tr>
<tr>
<td>GetParameterCount</td>
<td>Returns the number of columns in the run-time data Table of Global Sheet</td>
</tr>
<tr>
<td>GetRowCount</td>
<td>Returns the number of rows in the run-time data table of Global Sheet</td>
</tr>
<tr>
<td>GetSheet</td>
<td>Returns the specified sheet from the run-time data table.</td>
</tr>
<tr>
<td>GetSheetCount</td>
<td>Returns the total number of sheets in the run-time data table.</td>
</tr>
<tr>
<td>Import</td>
<td>Imports a specific external Excel file to the run-time data table.</td>
</tr>
<tr>
<td>ImportSheet</td>
<td>Imports the specified sheet of the specific excel file to the destination sheet.</td>
</tr>
</tbody>
</table>
SetCurrentRow | Sets the Focus of the Current row to the Specified Row Number | DataTablesetCurrentRow(RowNumber)
SetNextRow | Sets the focus of the next row in the run-time data table | DataTable.SetNextRow
SetPreviousRow | Sets the focus of the previous row in the run-time data Table | DataTable.SetPrevRow

**Data Table Object Properties:**

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlobalSheet</td>
<td>Returns the first sheet of the run-time data table.</td>
<td>DataTable.GlobalSheet</td>
</tr>
<tr>
<td>LocalSheet</td>
<td>Returns the Active local sheet of the run-time data table.</td>
<td>DataTable.LocalSheet</td>
</tr>
<tr>
<td>RawValue</td>
<td>Retrieves the raw value of the cell</td>
<td>DataTable.RawValue(ParameterID, [SheetID])</td>
</tr>
<tr>
<td>Value</td>
<td>Retrieves the value of the cell in the specified parameter.</td>
<td>DataTable.Value(ParameterID, [SheetID])</td>
</tr>
</tbody>
</table>

**Example:**

Consider the below Data Table:

```
<table>
<thead>
<tr>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>A3</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Principal</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>
```

'Accessing Datatable to get Row Count and Column Count
rowcount = DataTable.GetSheet("Global").GetRowCount
msgbox rowcount  ' Displays 4

colcount = DataTable.GetSheet("Global").GetParameterCount
msgbox colcount  ' Displays 3

DataTable.SetCurrentRow(2)
val_rate = DataTable.Value("Rate","Global")
print val_rate  ' Displays 7%

val_ppl = DataTable.Value("Principal","Global")
print val_ppl  ' Displays 2556

val_Time = DataTable.Value("Time","Global")
print val_Time  ' Displays 5
DTParameter Object Methods

DTParameter Object Properties:

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Returns the name of the parameter in the run-time data table.</td>
<td>DTParameter.Name</td>
</tr>
<tr>
<td>RawValue</td>
<td>Returns the raw value of the cell in the current row of the run-time data table.</td>
<td>DTParameter.RawValue</td>
</tr>
<tr>
<td>Value</td>
<td>Retrieves or sets the value of the cell in the Active row of the parameter in the run-time data table.</td>
<td>DTParameter.Value</td>
</tr>
<tr>
<td>ValueByRow</td>
<td>Retrieves the value of the cell in the specified row of the parameter in the run-time data table.</td>
<td>DTParameter.ValueByRow(RowNum)</td>
</tr>
</tbody>
</table>

Example:

Consider the below Data Table:

```
<table>
<thead>
<tr>
<th>Principal</th>
<th>Rate</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>232</td>
<td>3.26</td>
</tr>
<tr>
<td>2</td>
<td>2556</td>
<td>7%</td>
</tr>
<tr>
<td>3</td>
<td>1322</td>
<td>6.50%</td>
</tr>
<tr>
<td>4</td>
<td>32.21</td>
<td>3.32%</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Val = DataTable.GetSheet("Global").GetParameter("Principal").ValueByRow(2)
print Val            ' Val Displays 2556

DataTablesetCurrentRow(1)
Val1 = DataTable.GetSheet("Global").GetParameter("Principal").Value
print Val1           ' Val1 displays 232

DTSheet Object Methods

DTSheet Methods:

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddParameter</td>
<td>Adds the specified column to the sheet in the run-time data table.</td>
<td>DTSheet.AddParameter(ParameterName, Value)</td>
</tr>
<tr>
<td>DeleteParameter</td>
<td>Deletes the specified parameter from the run-time data table.</td>
<td>DTSheet.DeleteParameter(ParameterID)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td>Code Example</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetCurrentRow</td>
<td>Returns the row number of the active row in the run-time Data Table.</td>
<td>DTSheet.GetCurrentRow</td>
</tr>
<tr>
<td>GetParameter</td>
<td>Returns the specified parameter from the run-time Data Table.</td>
<td>DTSheet.GetParameter(ParameterID)</td>
</tr>
<tr>
<td>GetParameterCount</td>
<td>Returns the total number of Columns in the run-time Data Table.</td>
<td>DTSheet.GetParameterCount</td>
</tr>
<tr>
<td>GetRowCount</td>
<td>Returns the total number of rows in the run-time Data Table.</td>
<td>DTSheet.GetRowCount</td>
</tr>
<tr>
<td>SetCurrentRow</td>
<td>Sets the Focus on the specified Row of the Data Table.</td>
<td>DTSheet.SetCurrentRow(RowNumber)</td>
</tr>
<tr>
<td>SetNextRow</td>
<td>Shifts the Focus to the next Row of the Data Table.</td>
<td>DTSheet.SetNextRow</td>
</tr>
<tr>
<td>SetPrevRow</td>
<td>Shifts the Focus to the Previous Row of the Data Table.</td>
<td>DTSheet.SetPrevRow</td>
</tr>
</tbody>
</table>

Example:

```vbnet
'Accessing Datatable to get Row Count and Column Count
rowcount = DataTable.GetSheet("Global").GetRowCount
msgbox rowcount           ' Displays 4

colcount = DataTable.GetSheet("Global").GetParameterCount
msgbox colcount          ' Displays 3

DataTable.SetCurrentRow(2)
val_rate = DataTable.Value("Rate","Global")
print val_rate           ' Displays 7%

val_ppl = DataTable.Value("Principal","Global")
print val_ppl            ' Displays 2556

val_Time = DataTable.Value("Time","Global")
print val_Time           ' Displays 5
```
Checkpoints

What are Checkpoints?

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:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Checkpoint</td>
<td>Verifies the property values of an object in application under test and supported by all add-in environments.</td>
</tr>
<tr>
<td>Bitmap Checkpoint</td>
<td>Verifies an area of your application as a bitmap</td>
</tr>
<tr>
<td>File Content Checkpoint</td>
<td>Verifies the text in a dynamically generated or accessed file such as .txt,.pdf</td>
</tr>
<tr>
<td>Table Checkpoint</td>
<td>Verifies the information within a table. Not all environments are supported.</td>
</tr>
<tr>
<td>Text Checkpoint</td>
<td>Verify if the text that is displayed within a defined area in a Windows-based application, according to specified criteria.</td>
</tr>
<tr>
<td>Text Area Checkpoint</td>
<td>Verifies if the text string is displayed within a defined area in a Windows-based application, according to specified criteria.</td>
</tr>
<tr>
<td>Accessibility Checkpoint</td>
<td>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</td>
</tr>
<tr>
<td>Page Checkpoint</td>
<td>Verifies the characteristics of a Web page. It can also check for broken links.</td>
</tr>
<tr>
<td>Database Checkpoint</td>
<td>Verifies the contents of a database accessed by the application under test.</td>
</tr>
<tr>
<td>XML Checkpoint</td>
<td>Verifies the content of the .xml documents or .xml documents in Web pages and frames.</td>
</tr>
</tbody>
</table>
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.
Example:

The checkpoints are added for the application under test - "http://easycalculation.com/"

```vbnet
' 1. Inserted Standard Checkpoint
Status = Browser("Math Calculator").Page("Math Calculator").Link("Numbers").Check
Checkpoint("Numbers")

If Status Then
    print "Checkpoint Passed"
Else
    Print "Checkpoint Failed"
End if

' 2. Inserted BitMap Checkpoint
imgchkpoint = Browser("Math Calculator").Page("Math Calculator").Image("French").Check
Checkpoint("French")

If imgchkpoint Then
    print "Checkpoint Passed"
Else
    Print "Checkpoint Failed"
End if
```

Viewing Checkpoint Properties

After Inserting, in case a tester wants to change the values, we can do so by performing right click on the keyword 'checkpoint' of the script and navigating to "Checkpoint Properties" as shown below:
You can locate the same checkpoints in object repository as well as shown below. It exactly shows what type of checkpoint and what are the expected values, time out values.
Synchronization

What is Synchronization?

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 is 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:

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

Let us say we need to insert a sync point between clicking on "numbers" link and clicking on "simple Interest" calculator of in "http://easycalculation.com/". We will now take a look at all the 5 ways to insert sync point for the above scenario.

METHOD 1: WAITPROPERTY

WaitProperty is a method that takes the property name, Value and Timeout value as input to perform the sync. It is a dynamic wait and hence this option is encouraged.

' Method 1 - WaitProperty with 25 seconds
Dim obj
Set obj = Browser("Math Calculator").Page("Math Calculator")
**METHOD 2: EXIST**

Exist is a method that takes the Timeout value as input to perform the sync. Again it is a dynamic wait and hence this option is encouraged.

```vbscript
' Method 2 : Exist Timeout - 30 Seconds
Dim obj
Set obj = Browser("Math Calculator").Page("Math Calculator")
obj.Link("Numbers").Click
If obj.Link("Simple Interest").Exist(30) Then
    obj.Link("Simple Interest").Click
Else
    Print "Link NOT Available"
End IF
```

**METHOD 3: WAIT**

Wait is a hardcoded sync point which waits independent of the event happened or NOT. Hence usage of Wait is discouraged and can be used for shorter wait time such as 1 or 2 seconds.

```vbscript
' Method 3 : Wait Timeout - 30 Seconds
Dim obj
Set obj = Browser("Math Calculator").Page("Math Calculator")
obj.Link("Numbers").Click
wait(30)
Browser("Math Calculator").Page("Math Calculator").Link("Simple Interest").Click
```

**METHOD 4: SYNC METHOD**

Sync Method can be used only for web applications where there is always a lag between page loads.

```vbscript
' Method 4 :
Dim obj
Set obj = Browser("Math Calculator").Page("Math Calculator")
obj.Link("Numbers").Click
Browser("Math Calculator").Sync
Browser("Math Calculator").Page("Math Calculator").Link("Simple Interest").Click
```

**METHOD 5 : INSERTING QTP INBUILT SYNCHRONIZATION POINTS:**

*Step 1:* Get into Recording Mode. This Option Would be Disabled if the user is NOT in Recording Mode.
*Step 2:* Goto "Design" -> "Synchronization Point".
*Step 3:* We need to Select the object which we want to be the Sync Point. After Selecting the object, object window opens as shown below:
Step 4: Click Ok, the "Add Synchronization Window" Opens up. Select the Property, Value and Time out value and click ok as shown below:

Step 5: The Script would be generated as shown below which is the same as that of the WaitProperty(Method 1) that we had already discussed:

Browser("Math Calculator").Page("Math Calculator").Link("Numbers").Click

Default Synchronization:

When user hasn't used any of the above sync methods, still QTP has inbuild Object synchronization timeout which can be adjusted by the user.

Navigate to "File" >> "Settings" >> Run Tab >> Object Synchronization Time out as shown below.
Smart Identification

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:

- **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 3: 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”.

TUTORIALS POINT
Simply Easy Learning
Step 4: 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 5: We can even disable a test Level by applying at test script level under "Settings" of "File" Menu as shown below:
Step 6: If the Smart Identification is disabled as per Step# 6 then it won't apply smart identification for any object during script execution.

Step 7: 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

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 Coloured” Dot and the line will be highlighted in RED as shown below:

<table>
<thead>
<tr>
<th>Method</th>
<th>Shortcut</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step Into</td>
<td>F11</td>
<td>Used to execute each and every Step. Steps into the Function/Action and executes line by line. It pauses on each line after execution.</td>
</tr>
<tr>
<td>Step Over</td>
<td>F10</td>
<td>Used to Step over the Function. Step Over runs only the current step in the active document.</td>
</tr>
<tr>
<td>Step Out</td>
<td>Shift+F11</td>
<td>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.</td>
</tr>
</tbody>
</table>

Options in Break Point:

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

<table>
<thead>
<tr>
<th>Shortcut</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9</td>
<td>Insert/Remove BreakPoint</td>
</tr>
</tbody>
</table>
### 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.
- **Local Variables** - This Tab displays the Output of the Local Variables.

**Example:**

The Watch Pane shows the output expression as shown below:

```
Browser("Free Online Math Calculator").Page("Free Online Math Calculator").Link("Numbers").Click
Browser("Free Online Math Calculator").Page("Numbers Calculator - Math").Link("SQR").Click
Browser("Free Online Math Calculator").Page("Square Root Calculator").WebEdit("n").Set "10"
Browser("Free Online Math Calculator").Page("Square Root Calculator").WebButton("Calc").Click
```

The Local Variables Pane shows the values held by the local variables as shown below:
X = "This is Debugging"
Y = Mid (X, 2)
Z = X + Y

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Type Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>&quot;This is Debugging&quot;</td>
<td>String</td>
</tr>
<tr>
<td>Y</td>
<td>&quot;his is Debugging&quot;</td>
<td>String</td>
</tr>
</tbody>
</table>
Error Handling

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, viz- 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 script 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 in the Test Settings by Navigating to "File" >> "Settings" >> "Run" Tab as shown below. We can select any of the specified settings and click "OK".

   ![Test Settings](image)

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 process 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.

EXAMPLE:

```
'Call the function to Add two Numbers
Call Addition(num1,num2)

Function Addition(a,b)

    On error resume next
    If NOT IsNumeric(a) or IsNumeric(b) Then
        Print "Error number is " & err.number & " and description is : " & err.description
        Err.Clear
        Exit Function
    End If

    Addition = a+b

' disables error handling
On Error Goto 0

End function
```

Using Exit Statement - Exit Statements can be used along with Err object to exit from a test or action or iteration based on the Err.Number value. Let us see each one of those Exit statements in detail.

• ExitTest - Exits from the entire QTP test no matter what the run-time iteration settings are.

• ExitAction - Exits the current action.

• ExitActionIteration - Exits the current iteration of the action.

• ExitTestIteration - Exits the current iteration of the QTP test and proceeds to the next iteration.

5. Recovery Scenarios - Upon encountering an error, recovery scenarios are triggered based on certain conditions and it is dealt in detail in a separate chapter.

6. Reporter Object - Reporter Object helps us to report an event to the run results. It helps us to identify if the concerned action/step is pass/fail.

```

' Example
Reporter.ReportEvent micFail, "Login", "User is unable to Login."
```
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 the 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 perform 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:
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.

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 an 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
   ```

   **Output:**
   ```
   Config.xml
   ```
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.

```python
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.

```python
' 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
```
' 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

Output
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"


'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 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:

![Export Run Results Dialog Box]

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 and Iterations. It can be accessed by using Filter Button in the "Test Results Window"
RAISING DEFECTS

Defects can be logged into QC directly from the Test Results Window pane by accessing “Tools” -> “Add Defect” which opens a connection to ALM as shown below:
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 existance of the Text Box
- **GetROProperty("text")** - Gets the Value of the Text Box
- **GetROProperty("Visible")** - Returns a Boolean value if visible.

**EXAMPLE:**

```plaintext
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)
```

TUTORIALS POINT
Simply Easy Learning
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 existance of the Check Box
- **GetROProperty(“name”)** - Gets the Name of the check Box
- **GetROProperty(“Visible”)** - Returns a Boolean value if visible

**EXAMPLE:**

```vbscript
'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 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 Existance 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 existance 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

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>If statement</td>
<td>An if statement consists of a boolean expression followed by one or more statements.</td>
</tr>
<tr>
<td>if..else statement</td>
<td>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.</td>
</tr>
<tr>
<td>if...elseif..else statement</td>
<td>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.</td>
</tr>
<tr>
<td>nested if statements</td>
<td>An if or elseif statement inside another if or elseif statement(s).</td>
</tr>
<tr>
<td>switch statement</td>
<td>A switch statement allows a variable to be tested for equality against a list of values.</td>
</tr>
</tbody>
</table>

- **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:

```vbnet
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
```
Virtual Objects

What are Virtual Objects?

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 vitural objects is pronounced.

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

Creating a Virtual Object

**Step 1:** In such scenarios, virtual Objects are created using Virtual Object Manager or New Virtual Object from "Tools" >>"Virtual Object" >> "New Virtual Object" and click "Next" Button
Step 2: Map the Object against the Class Type and click "Next".

Step 3: Click "Mark Object" Button, cross hair cursor would appear and mark the object that you would like to map and click "Next".
Step 4: Selecting the parent of the Virtual object and click "Next".

Step 5: Name the collection in which you would like to store the virtual object and click "Finish".
Virtual Object Manager

Virtual object Manager manages the collections of Virtual objects. Testers can add or Delete the Virtual Objects from the Virtual object manager.

Navigation to Virtual object Manager: "Tools" >> "Virtual Object Manager" as shown below:

Using Virtual Objects

After creating the Virtual Objects the created object can be used as shown below:
Virtual Object Limitations

- QTP doesn't support virtual objects for analog or low-level recording.
- Checkpoints cannot be added on Virtual Objects.
- Virtual Object is 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?**

Databases can be connected using Connection strings. Each database differs the way we connect to them, however the connection strings can be build with the help of [http://www.connectionstrings.com/](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:
Dim objConnection
' Set Adodb Connection Object
Set objConnection = CreateObject("ADODB.Connection")
Dim objRecordSet

SMITH

OK
Working with XML

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:

```xml
<xml version="1.0" encoding="UTF-8">
  <Booklib>
    <book id="001">
      <author>Jim, Smith</author>
      <value type="Computer" price="44.95">An in-depth look at creating applications with XML</value>
      <publish_date>2002-01-01</publish_date>
    </book>
    <book id="002">
      <author>Jack, Robinson</author>
      <value type="Fantasy" price="15.50">An approach to Agile Programming</value>
      <publish_date>2010-10-04</publish_date>
    </book>
  </Booklib>
</xml>
```
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 description 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, it results 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.

```plaintext
'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
```
Description Strings
The description of the objects are developed using the properties and values as strings as shown below.

```vbscript
Browser("Math Calc").Page("Num Calculator").WebButton(btnCalc).Click
```

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

```vbscript
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.

```vbscript
' 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

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:

1. Loads all the required add-ins for a test
2. Makes QTP visible while execution
3. Opens the Test using the specified location
4. Associates Function Libraries
5. Specifies the Common Object Sync Time out
6. Start and End Iteration
7. Enable/Disable Smart Identification
8. On Error Settings
9. Data Table Path
10. Recovery Scenario Settings
11. 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 Demostrate 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").CloseOnExit = True

App.Test.Settings.Launchers("Windows Applications").Active = False
App.Test.Settings.Launchers("Windows Applications").RecordOnQTDescendants = True
App.Test.Settings.Launchers("Windows Applications").RecordOnExplorerDescendants = False
<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>App.Test.Settings.Run.IterationMode</td>
<td>&quot;rngAll&quot;</td>
</tr>
<tr>
<td>App.Test.Settings.Resources.DataTablePath</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>App.Test.Settings.Recovery.Add</td>
<td>&quot;D:\GUITest2\recover_app_crash.qrs&quot;, &quot;Recover_Application_Crash&quot;, 1</td>
</tr>
</tbody>
</table>

'Re 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
Frameworks

What is a Software Framework?

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:

- Keyword-Driven Framework
- Data-Driven Framework
- Hybrid 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 testers 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.

Designing a Framework

Let us design a simple framework by taking a sample application. We will automate few scenarios of the application under test and write reusable functions.

Learn more about designing frameworks:

The sample application under test is "Calculator", a default application that is available as part of Windows. Let us now create different components of a framework. Here, we will develop a hybrid framework and use Object Repository as it is fairly a simple application. However, this framework can be scaled to support a complex application as well.

The Folder Structure of the Framework is as follows. The Explanation of the folder structure is explained below:
1. Master Driver Script - The Script that drives the entire execution. It performs prerequisite and initial settings that are required for the execution.

2. Library Files - The Associated Functions that forms the Function Library

3. Data Table - The Test Data that is required for the Execution.

4. Object Repository - The Objects and its properties that enable QTP to recognize the objects seamlessly.

5. Execution Logs - The Folder contains the execution log file with user functions and function execution history.

MASTER DRIVER SCRIPT

```vbnet
Option Explicit
Public ExecDrive

' Get the Root folder of the Test so that we can make use of relative paths.
Dim x : x=Instr(Environment.Value("TestDir"),"Driver")-2
ExecDrive = mid(Environment.Value("TestDir"),1,x)

' Get the path of Libraries using relative to the current Drive
Dim LibPath : LibPath = ExecDrive+"\Libraries\"

' Dynamically Load the Function Libraries
LoadFunctionLibrary LibPath + "Calculator.qfl", LibPath + "common_utils.qfl"

' Capturing the Start Time
' clscommon is the class object created in common.qfl library file
clscommon.StartTime = Time()
```
' Launching the Application
SystemUtil.Run "C:\Windows\System32\Calc.exe" : wait (2)

' Initialize the Data Table Path
Dim FileName : FileName = ExecDrive+"\TestData\Calculator.xls"
Dim SheetSource : SheetSource = "Calc_test"
Dim SheetDest : SheetDest = "Global"

' Import the DataTable into the QTP Script
DataTable.ImportSheet FileName , SheetSource , SheetDest

' Object Repository Path
Dim RepPath : RepPath = ExecDrive+"\Object_Repository\Calc.tsr"
RepositoriesCollection.RemoveAll()
RepositoriesCollection.Add(RepPath)

' To Keep a Count on iteration
Dim InttestIteration
Dim InttestRows : InttestRows = datatable.GetRowCount

' Fetching Date-Time Stamp which will be unique for Naming the Execution Log File
clscommon.StrDateFormatted = day(date()) & "_" & MonthName(Month(date()),true) & "_" & YEAR(date()) & "_"&hour(now) & "_"&minute(now)

' Name the LogFile
clscommon.StrLogFile = ExecDrive & "\Execution Logs\" & clscommon.StrDateFormatted & ".txt"

' Create the Execution LogFile which captures the result
clscommon.Fn_FileCreate(clscommon.StrLogFile)

' Initialize the Parameters and all the relevant Test Details
Call Fn_InitializeLogFile()

' Kill all the previous calculator process
Call fn_Kill_Process("calc.exe")

For InttestIteration=1 to InttestRows
    datatable.SetCurrentRow InttestIteration
    Dim StrExecute : StrExecute = UCase(Trim(datatable.Value("Run","Global")))
    If StrExecute = "Y" Then
        clscommon.Number1   = Trim(datatable.Value("Number_1","Global"))
        clscommon.Number2   = Trim(datatable.Value("Number_2","Global"))
        clscommon.Number3   = Trim(datatable.Value("Number_3","Global"))
        clscommon.Number4   = Trim(datatable.Value("Number_4","Global"))
        clscommon.Number5   = Trim(datatable.Value("Number_5","Global"))
        clscommon.Number6   = Trim(datatable.Value("Number_6","Global"))
        clscommon.Test_Case_ID = Trim(datatable.Value("Test_Case_ID","Global"))
    End If
Next InttestIteration
clscommon.tcScenario = 
Trim(datatable.Value("Scenario","Global"))'
classcommon.LogWrite "The Test Case Data is Located at :: " & tcDataPath

Dim Expected_Val : Expected_Val = 
Trim(datatable.Value("Expected_Val","Global"))'
classcommon.LogWrite "The Test Case Data is Located at :: " & tcDataPath

Select case clscommon.tcScenario
Case "Add"
classcommon.LogWrite "=== Inside the Test Set :: " &
clscommon.tcScenario & " ==="
Call fnCalculate("+",Expected_Val)

Case "Subtract"
classcommon.LogWrite "=== Inside the Test Set :: " &
clscommon.tcScenario & " ==="
Call fnCalculate("-",Expected_Val)

Case "Multiply"
classcommon.LogWrite "=== Inside the Test Set :: " &
clscommon.tcScenario & " ==="
Call fnCalculate("*",Expected_Val)

Case "Divide"
classcommon.LogWrite "=== Inside the Test Set :: " &
clscommon.tcScenario & " ==="
Call fnCalculate("/",Expected_Val)

Case "Sqrt"
classcommon.LogWrite "=== Inside the Test Set :: " &
clscommon.tcScenario & " ==="
Call fnCalculate("sqrt",Expected_Val)

End Select
End If
Next

' Calling the End Test to Add the result Footer in exec log file.
Call fn_End_test()
' **************************** End of Master Driver Script ****************************

LIBRARY FILES

The Calculator Functions are written in a separate function file saved with the extension .qfl or .vbs. These functions are reusable across actions.

'  Calculator. Qfl File :: Associated Function Library for Calculator Master Driver
'= = = = = = = = = = = = = = = = = = = = = = = = = = = =
' FUNCTION NAME : Fn_InitializeLogFile
' DESCRIPTION : Function to Write the Initial Values in the Log File
' INPUT PARAMETERS : varExecDrive,StrDB,StrUID,StrPwd,StrNewDB
' OUTPUT PARAMETERS : NIL
' RETURN VALUE : Pass or Fail message

TUTORIALS POINT
Simply Easy Learning
Public Function Fn_InitializeLogFile()
    clscommon.LogWrite "******************************************************
    clscommon.LogWrite "Calc Automation Started"
End Function

Function fnCalculate(operator, Expected_Val)
    clscommon.LogWrite "Executing the Function 'fnCalculate' "
    Window("Calculator").Activate
    If Trim(clscommon.Number1) <> "" Then
        Window("Calculator").WinButton(clscommon.Number1).Click
    End If
    If Trim(clscommon.Number2) <> "" Then
        Window("Calculator").WinButton(clscommon.Number2).Click
    End If
    If Trim(clscommon.Number3) <> "" Then
        Window("Calculator").WinButton(clscommon.Number3).Click
    End If
    Window("Calculator").WinButton(operator).Click
    If Trim(clscommon.Number4) <> "" Then
        Window("Calculator").WinButton(clscommon.Number4).Click
    End If
    If Trim(clscommon.Number5) <> "" Then
        Window("Calculator").WinButton(clscommon.Number5).Click
    End If
    If Trim(clscommon.Number6) <> "" Then
        Window("Calculator").WinButton(clscommon.Number6).Click
    End If
    Window("Calculator").WinButton("= ").Click
    Dim ActualVal : ActualVal = Window("Calculator").WinEdit("Edit").GetROProperty("text")
    clscommon.LogWrite "The Actual Value after the Math Operation is " & ActualVal
    If Trim(ActualVal) = Trim(Expected_Val) Then
        clscommon.WriteResult "Pass", clscommon.Test_Case_ID, clscommon.tcScenario, " Expected Value matches with Actual Value :: " & ActualVal
    Else
        clscommon.WriteResult "Fail", clscommon.Test_Case_ID, clscommon.tcScenario, " Expected Value - " & Expected_Val & " Does NOT matches with Actual Value :: " & ActualVal
    End If
    Window("Calculator").WinButton("C").Click
    If Err.Number <> 0 Then
        clscommon.LogWrite "Execution Error : The Error Number is :: " & Err.Number & " The Error Description is " & Err.Description
        Err.Clear
    End If
    clscommon.LogWrite "Exiting the Function 'fnCalculate' "
End Function
FUNCTION NAME: fn_Kill_Process
DESCRIPTION: Function to Kill the process by name
INPUT PARAMETERS: Application name to be killed
OUTPUT PARAMETERS: NIL
RETURN VALUE: NIL
DATE CREATED: 30-Dec-2013

Function fn_Kill_Process(process)
Dim strComputer, strProcessToKill, objWMIService, colProcess
strComputer = "."
strProcessToKill = process
Set objWMIService = GetObject("winmgmts:__\" _
    & \"{impersonationLevel=impersonate}!\\" _
    & strComputer & \"\"\"_cimv2")
Set colProcess = objWMIService.ExecQuery(_
    "Select * from Win32_Process Where Name = \"" & strProcessToKill & 
    \"\"\"")
count = 0
For Each objProcess in colProcess
    objProcess.Terminate()
count = count + 1
Next
End Function

FUNCTION NAME: fn_End_test
DESCRIPTION: Function to finish the test Execution process
INPUT PARAMETERS: Application name to be killed
OUTPUT PARAMETERS: NIL
RETURN VALUE: NIL
DATE CREATED: 20/Dec/2013

Function fn_End_test()
clscommon.LogWrite "Status Message - Executing the Function 'fn_End_test'"
Window("Calculator").Close
On Error Resume Next
clscommon.StopTime = Time()
clscommon.ElapsedTime = DateDiff("n", clscommon.StartTime, clscommon.StopTime)
Dim Totaltests
Totaltests = clscommon.gintPassCount + clscommon.gintFailCount
clscommon.LogWrite "## The Execution Start Time :: " & clscommon.StartTime
clscommon.LogWrite "## The Execution End Time :: " & clscommon.StopTime
clscommon.LogWrite "## The Time Elapsed :: " & clscommon.ElapsedTime & " Minutes"
clscommon.LogWrite "## The OS :: " & Environment.Value("OS")
clscommon.LogWrite "## The Total No of Test Cases Executed :: " & Totaltests
clscommon.LogWrite "## The No. of Test Case Passed :: " & clscommon.gintPassCount
clscommon.LogWrite "## The No. of Test Case Failed :: " & clscommon.gintFailCount
The other library file which is 'common_utils.qfl' that contains the functions which enables us to write the output to a text file.

```plaintext
Set clscommon = New OS_clsUtils

'Creating a class file to handle global variables.
Class OS_clsUtils
    Dim StrLogFile
    Dim StrDateFormatted
    Dim Result
    Dim Number1, Number2, Number3
    Dim Number4, Number5, Number6
    Dim Test_Case_ID, tcScenario
    Dim StartTime, StopTime, ElapsedTime
    Dim gintPassCount, gintFailCount, gintWarningCount, gintdoneCount, gintinfoCount

    Function Fn_FileCreate(strFileName)
        Dim objFSO: Set objFSO = CreateObject("Scripting.FileSystemObject")
        On Error Resume Next
        Dim objTextFile : Set objTextFile = objFSO.CreateTextFile(strFileName)
        objTextFile.Close
        Set objTextFile = Nothing
        Set objFSO = Nothing
    End Function

    Function LogWrite(sMsg)
        Const ForAppending = 8
        Dim objFSO : Set objFSO = CreateObject("scripting.FileSystemObject")
        Dim objTextFile : Set objTextFile = objFSO.OpenTextFile(clscommon.StrLogFile, ForAppending, True)
        objTextFile.WriteLine day(date()) & "/" & MonthName(Month(date()),true) & "/" & YEAR(date()) & ": " & sMsg
        objTextFile.Close
        Set objTextFile = Nothing
        Set objFSO = Nothing
    End Function

    Function WriteResult(strStatus, functionName, functionDescription, Result)
        Const ForAppending = 8
        Dim objFSO : Set objFSO = CreateObject("scripting.FileSystemObject")
        Dim objTextFile : Set objTextFile = objFSO.OpenTextFile(clscommon.StrLogFile, ForAppending, True)
        objTextFile.WriteLine day(date()) & "/" & MonthName(Month(date()),true) & "/" & YEAR(date()) & ": " & strStatus & " Test status :: " & functionName
        objTextFile.WriteLine day(date()) & "/" & MonthName(Month(date()),true) & "/" & YEAR(date()) & ": " & strStatus & " Test status :: " & functionName
    End Function
```

The other library file which is 'common_utils.qfl' that contains the functions which enables us to write the output to a text file.
OBJECT REPOSITORY:

Object Repository has got all the objects that the user would be acting upon. The below image shows the list of all objects added into the repository with the name calc.tsr
TUTORIALS POINT
Simply Easy Learning

DATA TABLE

Data Table contains the keywords which drive the tests and also Test data with which QTP will act on the objects.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Y</td>
<td>TC_001</td>
<td>Add</td>
<td>5</td>
<td>9</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>949.</td>
</tr>
<tr>
<td>3</td>
<td>Y</td>
<td>TC_002</td>
<td>Subtract</td>
<td>5</td>
<td>9</td>
<td>6</td>
<td>1</td>
<td>8</td>
<td>1</td>
<td>415.</td>
</tr>
<tr>
<td>4</td>
<td>Y</td>
<td>TC_003</td>
<td>Multiply</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>8</td>
<td>9</td>
<td>1</td>
<td>278883.</td>
</tr>
<tr>
<td>5</td>
<td>Y</td>
<td>TC_004</td>
<td>Divide</td>
<td>9</td>
<td>9</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td></td>
<td>3.</td>
</tr>
<tr>
<td>6</td>
<td>Y</td>
<td>TC_005</td>
<td>Sort</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td></td>
<td>10.</td>
</tr>
</tbody>
</table>
THE EXECUTION LOG:

The Execution log file or output file contains user actions and function logs which will enable the testers to debug upon script failures.

```
8/Jan/2014 5:09:16 PM: ********************
8/Jan/2014 5:09:16 PM: Calc Automation Started
8/Jan/2014 5:09:16 PM: === Inside the Test Set :: Add ===
8/Jan/2014 5:09:16 PM: Executing the Function 'fnCalculate'
8/Jan/2014 5:09:17 PM: * * * * * * Test Case Exec Details * * * *
8/Jan/2014 5:09:17 PM: Test Staus :: Pass
8/Jan/2014 5:09:17 PM: Test Description :: Add
8/Jan/2014 5:09:17 PM: Test Result Details :: Expected Value matches with Actual Value :: 949.
8/Jan/2014 5:09:17 PM: * * * * * *
8/Jan/2014 5:09:17 PM: Executing the Function 'fnCalculate'
8/Jan/2014 5:09:17 PM: === Inside the Test Set :: Subtract ===
8/Jan/2014 5:09:17 PM: Executing the Function 'fnCalculate'
8/Jan/2014 5:09:17 PM: * * * * * * Test Case Exec Details * * * *
8/Jan/2014 5:09:17 PM: Test Staus :: Pass
8/Jan/2014 5:09:17 PM: Test Description :: Subtract
8/Jan/2014 5:09:17 PM: Test Result Details :: Expected Value matches with Actual Value :: 415.
8/Jan/2014 5:09:17 PM: * * * * * *
8/Jan/2014 5:09:17 PM: Exiting the Function 'fnCalculate'
8/Jan/2014 5:09:17 PM: === Inside the Test Set :: Multiply ===
8/Jan/2014 5:09:17 PM: Executing the Function 'fnCalculate'
8/Jan/2014 5:09:18 PM: * * * * * * Test Case Exec Details * * * *
8/Jan/2014 5:09:18 PM: Test Staus :: Pass
8/Jan/2014 5:09:18 PM: Test Description :: Multiply
8/Jan/2014 5:09:18 PM: Test Result Details :: Expected Value matches with Actual Value :: 278883.
8/Jan/2014 5:09:18 PM: * * * * * *
8/Jan/2014 5:09:18 PM: Exiting the Function 'fnCalculate'
8/Jan/2014 5:09:18 PM: === Inside the Test Set :: Divide ===
8/Jan/2014 5:09:18 PM: Executing the Function 'fnCalculate'
```

TUTORIALS POINT
Simply Easy Learning
8/Jan/2014 5:09:19 PM: Test status :: Pass
8/Jan/2014 5:09:19 PM: Test ID :: TC_004
8/Jan/2014 5:09:19 PM: Test Description :: Divide
8/Jan/2014 5:09:19 PM: Test Result Details :: Expected Value matches with Actual Value :: 3.
8/Jan/2014 5:09:19 PM: * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
8/Jan/2014 5:09:19 PM: Exiting the Function 'fnCalculate'
8/Jan/2014 5:09:19 PM: === Inside the Test Set :: Sqrt ===
8/Jan/2014 5:09:19 PM: Executing the Function 'fnCalculate'
8/Jan/2014 5:09:20 PM: * * * * * * Test Case Exec Details * * * * *
8/Jan/2014 5:09:20 PM: Test status :: Pass
8/Jan/2014 5:09:20 PM: Test ID :: TC_005
8/Jan/2014 5:09:20 PM: Test Description :: Sqrt
8/Jan/2014 5:09:20 PM: * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
8/Jan/2014 5:09:20 PM: Exiting the Function 'fnCalculate'
8/Jan/2014 5:09:20 PM: Status Message - Executing the Function 'fn_Finish_test'
8/Jan/2014 5:09:20 PM: # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
8/Jan/2014 5:09:20 PM: # The Execution Start Time :: 5:09:14 PM
8/Jan/2014 5:09:20 PM: # The Execution End Time :: 5:09:20 PM
8/Jan/2014 5:09:20 PM: # The Time Elapsed :: 0 Minutes
8/Jan/2014 5:09:20 PM: # The OS :: Microsoft Windows Vista Server
8/Jan/2014 5:09:20 PM: # The Total No of Test Cases Executed :: 25
8/Jan/2014 5:09:20 PM: # The No. of Test Cases Passed :: 25
8/Jan/2014 5:09:20 PM: # The No. of Test Case Failed ::
8/Jan/2014 5:09:20 PM: # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #