

STeP-IN SUMMIT 2007

International Conference On Software Testing

QTP™ Script Executor

by
Rajeev J and Raghu Gooty
First Indian Corporation

rajeev@firstam.com and raghu@firstam.com

Copyright: STeP-IN Forum and Quality Solutions for Information Technology Pvt. Ltd.

Published with permission for restricted use in STeP-IN SUMMIT 2007 in agreement with full copyrights from owner(s) / author(s) of material. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without the prior consent of the owner(s) / author(s). This edition is manufactured in India and is authorized for distribution only during STeP-IN SUMMIT 2007 as per the applicable conditions.

Practices Experience Knowledge Automation

Produced By

STeP-IN
Forum

www.stepinforum.org

Hosted By



www.qsitglobal.com

Abstract:

A **requisite** to execute cluster of test scripts from a test suite, which require changes during runtime poses a challenge to be done within a short time; this situational demand was often experienced.

This paper '*QTP™ Script Executor*' is to present, a utility that provides an extended functionality using inbuilt capabilities of QTP™ Professional (QTP™). Script Executor (**SE**) is an endeavor to fill in the **shortcomings** faced during batch execution of automation script. The rich power of this is **harnessed** while executing script batches with different parameters and setting's. SE can be implemented across any **applications/systems** that are automated using QTP™.

Practitioners who are currently automating using QTP™ will be able to **directly** implement this and those in the process of planning automation using QTP™ will get a broad perceptive and advantages of SE.

Brief description of the Script Executor:

SCRIPT EXECUTOR is developed using **Object Model Reference** in QTP™. This helps in automating QTP™ operations using objects, methods and properties exposed by QTP™ automation object model. **SCRIPT EXECUTOR** helps in **configuring** QTP™ options, run tests or business components instead of performing these operations manually using the QTP™ interface.

SCRIPT EXECUTOR core use is to perform the same tasks **multiple times** or on multiple tests/components, quickly configuring QTP according to your needs for a particular execution of automated script.

Advantages of using Script Executor:

100% Flexibility in test script execution. The user is completely flexible in selecting the test script required to execute for a particular run. A test management tool like Quality control would partially support this kind of execution. **SCRIPT EXECUTOR** helps in achieving around **80% of time reduction** and gives more **flexibility** and functionality than QTP™ batch runner gives.

- Driver script drives the automation.
- Conditional execution of scripts is possible without any test management tool.
- Any new feature in QTP™ can be effortlessly implemented to all existing scripts.
- Component based scripts.
- Run time settings configured through excel.
- Flexibility of passing environmental values to QTP™ .

- Maintain the same configuration setting across every test resources.
- Uniform directory structure.

Major Issues faced and resolution:

- While passing parameter through the excel sheet, SCRIPT EXECUTOR was not able to handle more than **26** script in a single run. This was handled by changing the parameter into environmental variable.
- Stopping the script during execution was not possible, by sending a status from the script to the excel sheet this was achieved.

Introduction:

In today's world of emerging complex applications, which are built and maintained with cutting edge technologies, a need arises to adopt well-defined and sophisticated approach in SDLC (Software Development Life cycle). As testing plays a major role in SDLC, we should refine our way of doing testing continuously. Automation of testing is one of the approach, which implemented successfully delivers a high quality product and greater customer satisfaction.

The challenge here is selecting and implementing the best tools available in market, which suits our customer needs and technology, adopted to build software.

We in our testing strategy have selected QTP™ as a tool to automate software systems. Our automation testing process involves building scripts for sanity, regression and functional testing. After any deployment, the IT team would need to execute a set of scripts in order to make sure that build is deployed properly. The normal procedure involves QTP™ specialist to select targeted scripts from a test suite and running them according to needs. This may sometimes involve changing the parameter values, setting the result directory, changing the object repository file, setting the iterations for the scripts, selecting the required Add-in for the script. This change in the configuration of the script needs a QTP™ specialist to be present in person whenever scripts are to be executed.

The dependency and the time of a QTP™ specialist posed a challenge every time the script needs to be executed. As a part of avoiding the dependency, the following alternatives were considered.

(a) **Batch Runner** : QTP™ come with utility called Batch Runner which could facilitate the execution of batch script, but again a QTP™ specialist was needed in order to select the script and in case any changes in the script needs to be done there was a dependency on a QTP™ specialist.

(b) **Quality center™**: provides the option of select the necessary script and executed them, but the challenge here was any changes to the script needs to be done within the script and this required a QTP™ specialist and on top of this the person in charge of executing the script should also had to know to use Quality Center. This also would require additional Quality Center license.

Since the alternatives could not provide an effective solution, we created a user-friendly Interface called as “**Script Executor**”.

The above alternatives could not be used to completely overcome the issues that were faced, also for the option (b) it required additional license for Quality Center™. As per our finding to overcome the above situation, QTP™ Automation Object Model can be customized and used according to our need.

Script Executor and QTP™ Automation Object Model:

Script Executor sits above the QTP™ Automation Object Model and interacts with QTP™ engine. The figure below depicts the interaction between script executor and QTP Automation object model, which in turn interacts with QTP™ Engine.

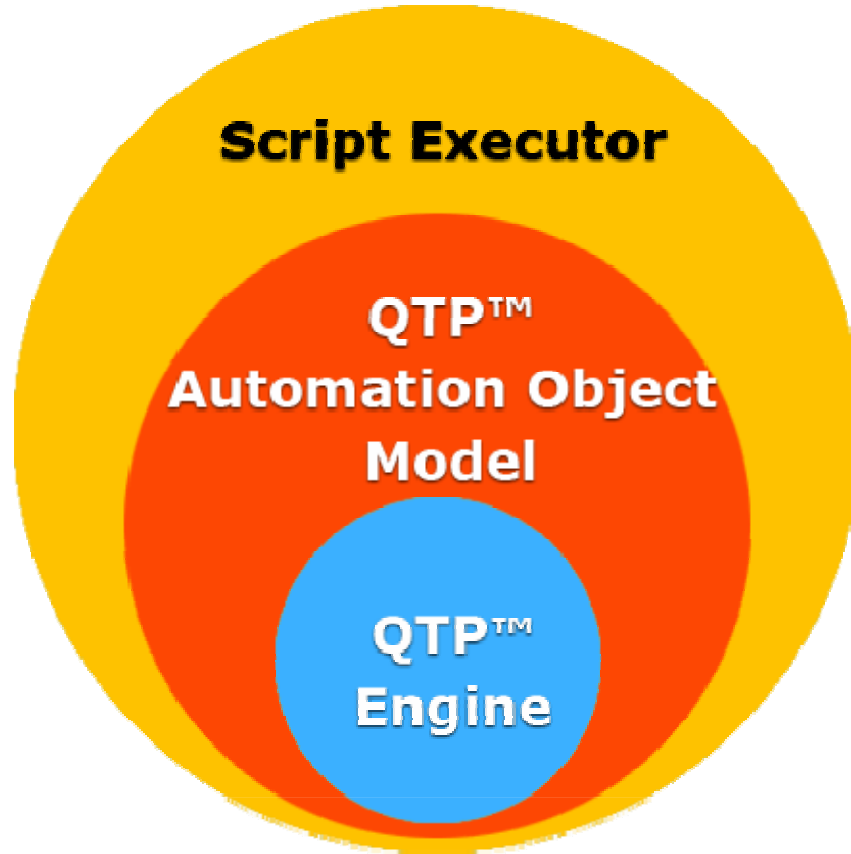


Figure 1

About QTP Automation Object Model [1]:

Just as QTP™ is used to automate the testing of applications, QTP™™ automation object model is used to automate QTP™™ operations. Using the objects, methods, and properties exposed by QTP™™'s automation object model, programs can be written to configure QTP™™ options and run tests or business components instead of performing these operations manually using the QTP™™ interface.

Automation programs are especially useful for performing the same tasks multiple times or on multiple tests or components, or quickly configuring QTP™™ according to a particular environment or application.

Essentially all configuration and run functionality provided via the QTP™ interface is in some way represented in the QTP™ automation object model via objects, methods, and properties. Although a one-

on-one comparison cannot always be made, most dialog boxes in QTP™ have a corresponding automation object, most options in dialog boxes can be set and/or retrieved using the corresponding object property, and most menu commands and other operations have corresponding automation methods.

Objects, methods, and properties exposed by the QTP™ automation object model can be used along with standard programming elements such as loops and conditional statements to design the programs.

Example 1: Create and run an automation program from Microsoft Visual Basic that loads the required add-ins for a test, starts QTP™ in visible mode, opens the test, configures settings that correspond to those in the Options, Test Settings or Business Component Settings, and Record and Run Settings dialog boxes, runs the test, and saves the test.

This can be enhanced by adding a simple loop to the program so that a single program can perform the operations described above for a number of tests.

Example 2: Create an initialization program that opens QTP™ with specific configuration settings. Instruct all of testers involved to open QTP™ using the created initialization automation program to ensure that all testers are always working with the same configuration.

Into Script Executor:

Script Executor is derived from QTP™ Automation object model. Script Executor is build using Visual basic scripting and executed using browser. The script that needs to be executed with necessary parameters for execution can be set in excel sheet; **driver script** executes the scripts with the given parameter.

Architecture of Script Executor:

Main Components of Script Executor are:

1. Driver Script
2. Configuration file (Excel sheet)

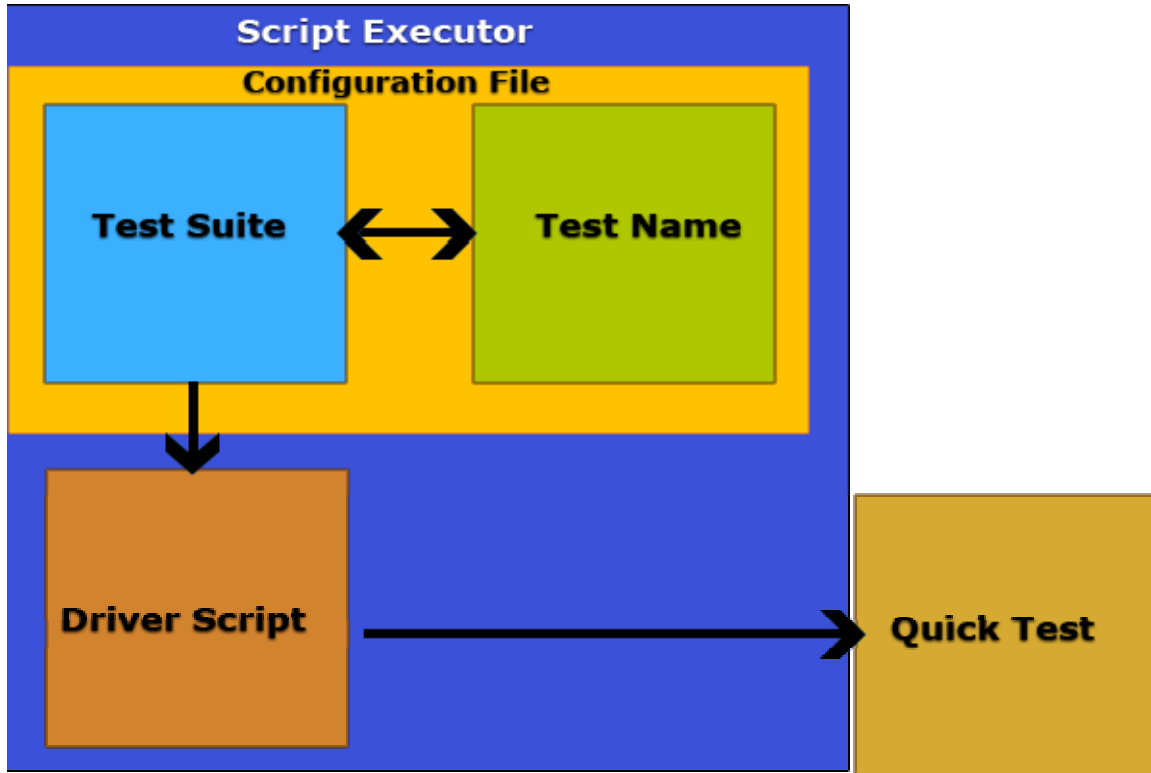


Figure 2

Driver Script:

Driver script drives automation that invokes QTP™ and starts execution of scripts. Driver script can be configured such that it can run the script with QTP™ visible to the user or QTP™ running in the background. Other technical details on the functions present in the Driver scripts and how the flow of the Driver script is not covered here.

Configuration file:

This is the data file which the driver script would be reading, this file consist of what script needs to be executed for each run. This is an excel file which consist of 2 sheets. Sheet 1 is “Test Suite” where Test Name is mentioned along with the script that needs to be executed for a particular run.

	A	B	C	D	E	F	G	H	I	J	K	L
1												
2	TestName	TestSuite										
3	Sanity	1										
4	Functional	2										
5	Regression	1,2,3										
6	Specific testcases	4										
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												
21												
22												
23												
24												
25												
26												
27												
28												
29												
30												

Figure 3

Figure 3: is a snapshot of the sheet “**Test Suite**”, here Test Set named “**Functional**” is represented by a number 1. This means that for the test named “**Functional**” it needs to execute test’s marked with number 1. Test Suite could clearly identify the test set name and identify the corresponding test cases needs to be executed for this test set.

The next sheet in this configuration file is the “**Test Case**” where the actual test cases and also the corresponding parameter/setting for the script is mentioned . At this point of time these are the following parameters that can be passed to the Driver Script.

- Action
- Number of iterations
- Object repository file
- Environmental variables
- Environmental Name
- Test Result

No	Action	Iterations	TsrFile	Status	Environment Name	Environment Value	TestResult
1	gTest	1			User1/Password	test/test1	E:\Backup\QTPFrame\google1\Actions\gTest\Res101
2	gTest	3					
3	gTest	1			Login/Password	testuser/password	
4	gTest	2					
5	gTest	1					
6	gTest	4					
7	yTest	1					
8	yTest	4					
9	yTest	1			search	ownername	
10	yTest	6					
11	yTest	1			Search	Address	E:\Backup\QTPFrame\google1\Actions\gTest\Res
12	yTest	1					
13	yTest	1					
14	yTest	1					
15	yTest	1					
16	yTest1	5					E:\Backup\QTPFrame\google1\Actions\gTest\Res1
17	yTest1	1					
18	yTest	1					
19	yTest	4					
20	yTest	1					E:\Backup\QTPFrame\google1\Actions\gTest\Res2
21	yTest	1					
22	yTest	3					
23	yTest	1					
24							
25							
26							

Figure 4

The figure 4 shows a snapshot of the “Testcase” sheet in the configuration file. Here apart from the above mentioned parameters these can also be specified

- Test Number
- Status

Let’s go into the details of what each column in this sheet means

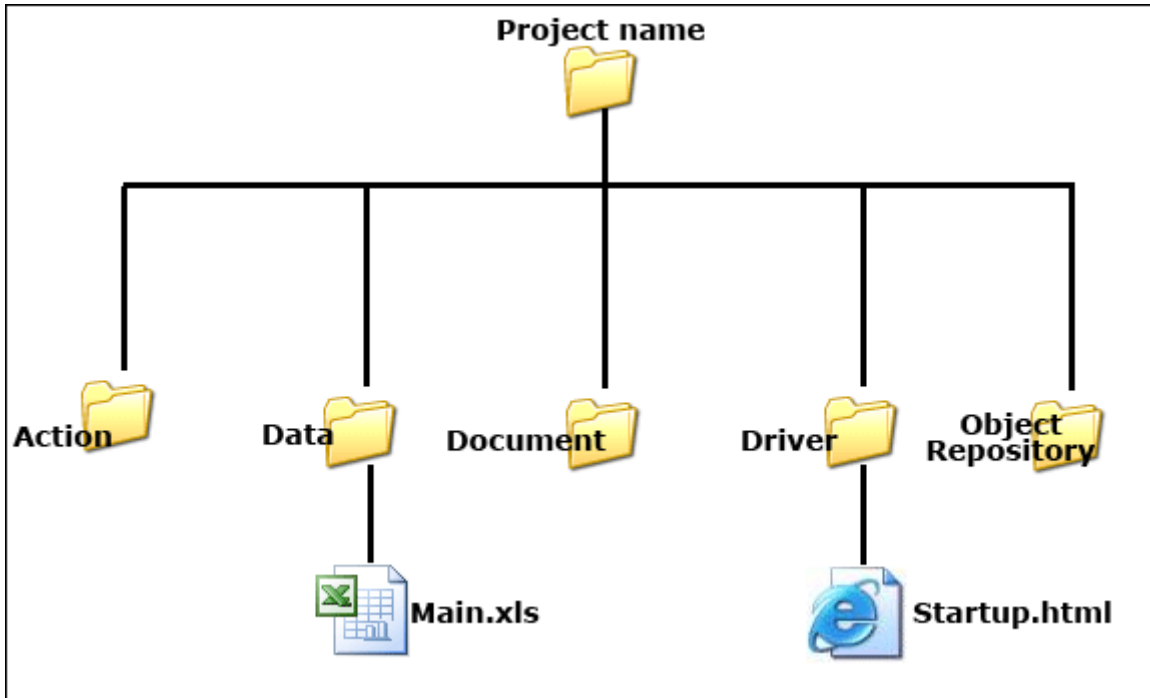
- **Test Number:** Here is the place where a number is specified to a test case and this number would be useful in the “Test Suite” sheet where we can specify a particular test name and also specify which test case needs to be executed for this test name. As seen in the figure 3 there are 2 numbers specified (i.e 1 and 2) and execution can be done for test with numbered 1 or test with numbered 2 or both for any particular run. Imagine a situation where only few test case needs to be executed for the regression test and a few for functional testing only or a need arises or executing selective test cases during some iterations. At this point a specific number to all the test scripts that needs to be executed is assigned and the same is mentioned in the “Test Suite” tab and the Driver script

would make sure that it would execute only test script which is assigned with the specified number.

- **Action:** Specifying any particular action or test script here would mean that the driver script would execute that particular action provided the condition of the number for the test script is the same number that is present in the “Test Suite” file.
- **Iterations:** If any particular script needs to be executed for different amount of time, it can be specified here. Instead of changing the number iteration inside every scripts it could easily be done here and it is effected during execution of the script. This avoids the overhead of getting into each and every script and changing the iteration.
- **TsrFile:** Test scripts requires object repository to identify and execute the scripts. Object repository needs to be associated with the test script can be specified here. In case any change required to repository file this can be set in this column instead of doing it with in QTP
- **Status:** This column contains the status of the script execution. Once the script is executed from QTP, Driver Script gets back the confirmation from the script for the successful execution and updates this column in the excel sheet. Based on this result the next set of test script can be executed or halted.
- **Environmental Name and Variable:** When it is required to send some values externally into the script, in such case using this column called Environmental Name and Environmental Variable it cab be achieved by specifying the variable name and value to the variable in this column. If this column is left blank no values will be passed.
- **Test Result:** The path in which the result file should be stored, and result file needs to be directed to that specified location is specified here.

Common platform for all the team members

Another advantage of using Script Executor is that team members would be on the same platform and be able to use this. For implementing Script Executor the team needs to follow a predefined folder structure, this is because Driver Script needs such folder structure in order to execute. The folder structure that needs to be followed is given below.



- **Action:** This folder should contain all the scripts or action of QTP
- **Data:** This folder contains the configuration file. As mentioned above configuration file consist of an excel sheet with the following sheets “Test Suite” and “Test Name”
- **Document:** This folder contains the documentation for the QTP script.
- **Driver:** This folder contains Driver script and this script need not be changed by users, this script is pre coded. Apart from the Driver Script any custom VBScript can be placed in this folder.
- **ObjectRepository:** Common Object repository can be placed in this folder and QTP scripts can use this in their scripts apart from this Script Executor also provides a feature where user can specify from where the scripts can get the object repository.

Summing up Script Executor:

1. Any changes in the script such as iteration, object repository can be handled outside the script in the excel sheet
2. Selective execution of QTP™ script can be easily handled. The QA Engineer involved in creating the script need not be present during execution, those who understand the flow of execution can select the necessary scripts and execute them.
3. Provides a common folder structure and all QA engineers can be in same line.
4. Focuses on Component based script.

Further development on Script Executor:

Moving forward Script Executor can be also be designed in such a way that it can also include few more parameter to QTP™ such as

- Add-in selection to QTP
- Adding library script to QTP
- Recovery Management

And all those object which QTP™ automation object model exposes. By this creating own customized executor that would suite particular needs would be easy.

Where can Script Executor be implemented:



Given below are few examples that could give an idea of when Script Executor would be useful.

- **Initialization programs**—A program that automatically starts QTP™ and configures the options and the settings required for recording on a specific environment.
- **Maintaining test or component database**—A program that iterates over entire test or component database to accomplish certain goal. For example:
 - Updating values—opening each test or component with the proper add-ins, running it in update run mode against an updated application..
 - Applying new options to existing test or components— If a new version of QTP™ offers certain options which needs to be applied in the existing, a program can be written that opens each existing test or component, sets values for the new options, then saves and closes it.
- **Calling QTP™ from other applications**—Designing own applications with options or controls that run QTP™ automation programs. For example, Creating a Web form or simple Windows interface from which a product manager could schedule QTP™ runs, even if the manager is not familiar with QTP™.

Reference:

- [1] AutomationObjectModel.chm which comes along with QTP™.

Author Biography

Photograph	Biography
	<p>Rajeev J is an engineering graduate having 7+ years experience in Software QA/QC. He is a certified CSQA from QAI and a Green Belt certified from General Electric. From past 4 years he is been working as a Technical Lead QA in First Indian Corporation implementing effective Methods/process in manual testing and automation. He is currently leading a team of 10.</p>
	<p>Raghu Gooty is an engineering graduate having 6 years experience in software QA/QC. Raghu has varied experience working in different domains testing complex systems. He is currently working as Senior software engineer in First India Corporation where in has immensely contributed in achieving greater customer satisfaction through controlled process structures.</p>