SimpleQTP Reference Guide

Document Version: 2.0

08/01/2011

Weifeng Lu

1. Overview

SimpleQTP is an open source automation framework designed based on QTP. It is intended to make it easier to create a common platform to the automation engineer doing their job. SimpleQTP can help maximize reusability of automation components and to minimize the maintenance costs to increase the efficiency, productivity, and quality of your automation testing.

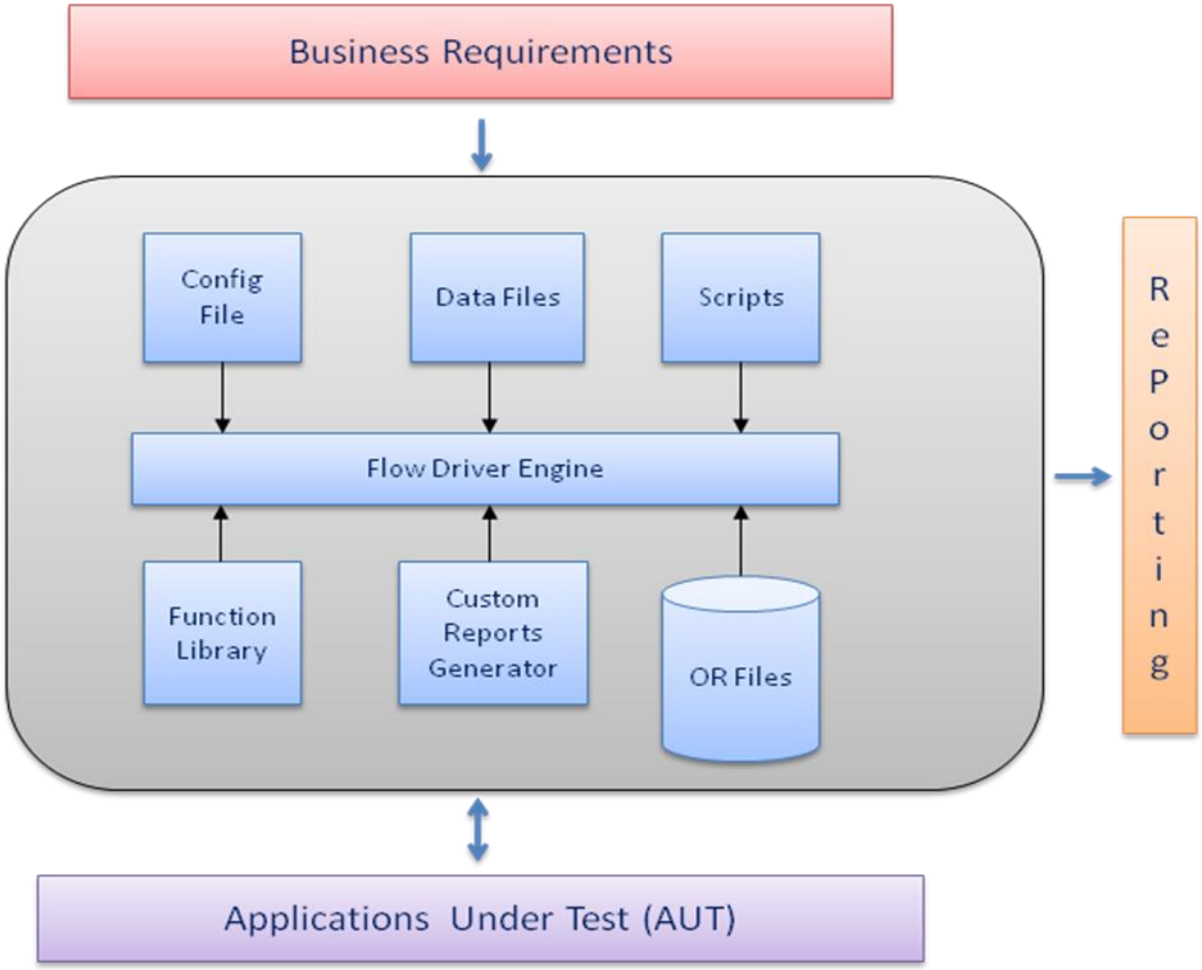
Here are the features contained in this automation framework.

- Provides a flow driver engine to make the scripts associated with test cases and test data, and determine which application/test suite/test case to run by Run Status flag.
- Each test case execution script is designed as GUI Layer (the extension of an OR with dictionary object) and Business Layer (Actions), which provides a highly effective way that enables to exit a test smoothly, preventing QTP from getting stuck when it fails to identify GUI objects during runtime.
- Provide a custom HTML report generator is designed based on XML+XSLT 1.0+CSS+Javascript technology which provides summary statistics and detail information(consists of clearly hierarchical tree – Test Suite layer, Test Case layer and Test Step layer, and each layer can be expanded and collapsed).
- Provide a rich, useful and generic function library (classes) to handle with such as database, excel, xml, FTP, registry, datetime, string.

This document introduces this automation framework in detail to let user understand SimpleQTP faster and better.

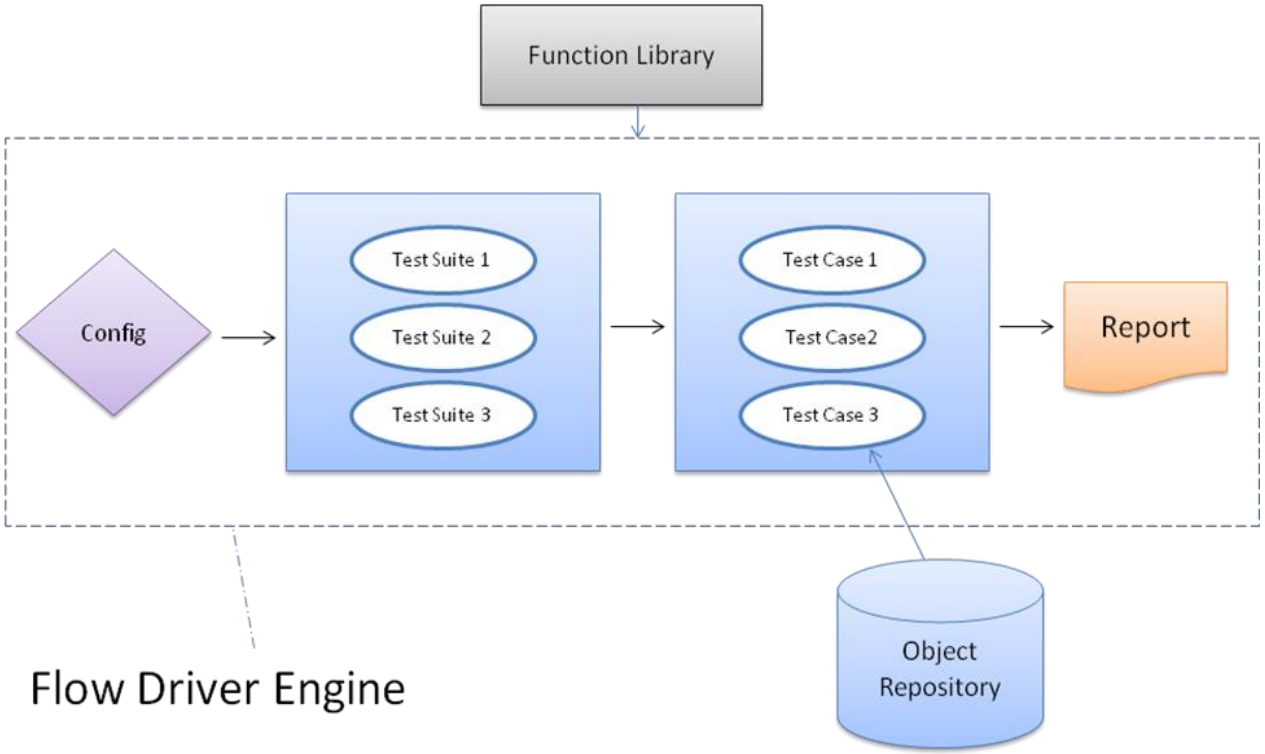
2. Test Architecture

SimpleQTP architecture is designed as following.



3. Test Workflow

SimpleQTP workflow is designed as following.



This flow driver engine will be executed as following orders.

1. Read PreExecutionSetup config file to decide which application to run by RunStatus flag.
2. Read test suite data file to decide which test suite to run by RunStatus flag.
3. Read test case data file to decide which test case to run by RunStatus flag.
4. Go into the GUI Layer (the extension of an OR with dictionary object), and Business Layer (Actions) to run the each test case.
5. Generate custom Html report (XML + XSL + CSS + JavaScript).

4. Test Components

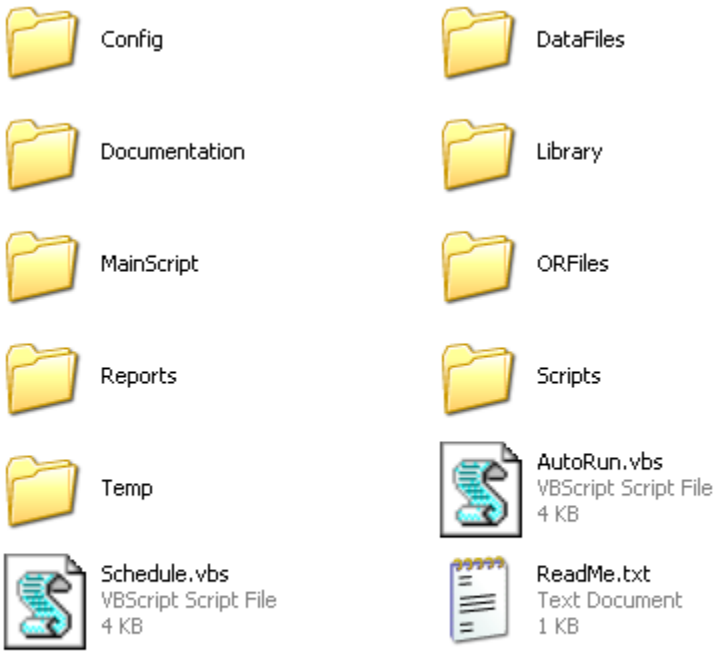
SimpleQTP consists of the following distinct components.

Component	Description
Flow driver Engine	Read config and data files to determine which application/test suite/test case to run by Run Status flag, and make the scripts associated with test cases and test data
Generic Function Library	Provide a rich, useful and generic function library (classes) to handle with such as database, excel, xml, FTP, registry, datetime, string.
Object Repository Files	Object Repository used for learning Application Objects and their Corresponding Property Values and Storing Object Descriptions in global Object Repository Files.
Test Case Execution Script	Design as GUI Layer (the extension of an OR with dictionary object) and Business Layer (Actions) to run the each test case
Custom Report Generator	Generate custom Html Reports with Statistics and Screen Shots based on XML + XSL + CSS + JavaScript.

5. Directory Structure

The directory structure consists of the following folder:

- MainScript
- Config
- DataFiles
- Library
- ORFiles
- Scripts
- Reports
- Documentation
- Temp



- The MainScript directory contains the flow driver engine which loads global Constants, loads functions library, Reads config and data files to determine which application/test suite/test case to run by RunStatus flag.

- The Config directory is used to configure the test environment.

GlobalConstants.vbs
VBScript Script File
18 KB

PreExecutionSetup.xls
Microsoft Office Excel 97-200...
19 KB

  - PreExecutionSetup – this is excel file, it contains RunStatus flag, application environment, URL/Location, and login information to load at startup.

	A	B	C	D	E	F	G	H
1	RunStatus	ApplicationName	Environment	URL/Location	Release	Build	User	Password
2	Y	Flight	A	C:\Program Files\Mercury I	4.0	4	9632	MERCURY
3	N	Flight	B	C:\Program Files\Mercury I	4.0	4	Demo	MERCURY

Note: Please make sure the application name is same with the corresponding test suite data file name.

  - GlobalConstants – Centralize all constant definitions loaded during starting of the scripts such as file/folder path, etc.

- The DataFiles directory contains test suite and test case data files which are excel files with columns and rows representing the data applicable to the test.

TestCases

TestSuites

  - The test suite data file contains RunStatus flag, test suite name.

	A	B	C
1	**RunStatus**	**TestSuiteName**	
2	Y	Flight Reservation	
3			

Note: Please make sure the test suite name is same with the corresponding test case data file name.

- The test case data file contains RunStatus flag, test case id, Case Name, Function Entry (i.e. Keyword), parameter values (fields).

RunStatus	CaseID	Case Name	Function Entry	Parameter1	Parameter2	Parameter3	Parameter4
Login							
Y	FR_LI_M_001	Login	Do_Login	URL	UserID	Password	
Order							
Y	FR_OD_C_001	Insert Order	Do_InsertOrder				
Y	FR_OD_H_002	Update Order	Do_UpdateOrder				
Y	FR_OD_L_003	Delete Order	Do_DeleteOrder				

The "Function Entry" column field represents the function name to be called in the scripts.
There are two types of parameter values:

- Variable, which passed from PreExecutionSetup file, such URL, UserID, Password.
- Constant.

- The Library directory contains generic functions library (Classes).

Library name	Purpose
ExcelLib	A library to work with excel file such as read and write operation with excel document.
XlsDictLib	A library to create a high-effective test data Dictionary based on excel file.
CompareTwoExcelLib	A library to compare two excels cell by cell and highlights the difference by red color.
FSOLib	A library to work with file/folder system such as create, read, find, edit, copy, delete and compare operate for file/folder.
DBLib	A library to work with database such as export query to 2D Array, export query to excel.
DateTimeLib	A library to work with DateTime such as Format date/time and Get relative date.
ArrayLib	A library to work with array such as get the length/dimension Of array, convert array to string/dictionary, and sort array.
RegLib	A library to work with registry such as create, modify, and delete operate with registry.
StringLib	A library to work with string
WordLib	A library to work with word file such as insert, find, replace, modify and convert operation with word document.

EmailLib	A library to send email using CDO/MAPI
FTPLib	A library to work with FTP/SFTP
ZipLib	A library to zip/unzip file
LoggerLib	A library to log debug information for application
ObjectGeneralLib	A library to work with general object such as Click a control by its text, Drag And Drop, and Highlight
BrowserLib	A library to work with Browser object such as Activate, Maximize, Minimize, and Close Browser object

- The ORFiles directory contains global object variables mapped to the Object Repository (OR).

```
''' ############################################################
''' <summary>
''' Centralize all Objects definitions mapped to Object Repository
''' </summary>
''' <remarks>First clear the objects instances in memory</remarks>
''' ############################################################


''' <summary>
''' Flight Reservation Login screen Objects Mapping
''' </summary>
Set FR_Login_Dialog_OR = nothing
Set FR_Login_Dialog_OR = Dialog("Login")
Set FR_AgentName_OR = Nothing
Set FR_AgentName_OR = Dialog("Login").WinEdit("Agent Name:")
Set FR_Password_OR = Nothing
Set FR_Password_OR = Dialog("Login").WinEdit("Password:")
Set FR_OK_OR = Nothing
Set FR_OK_OR = Dialog("Login").WinButton("OK")
```

It reduces the laborious work of changing object names when changes in application since being maintained in one central file. The assigned global object variables can be used either in scripts or in function library.

Note: This file might need to be re-loaded based on need during execution of scripts because some time some of the objects might be changed and cannot be recognized.

- The Scripts directory contains all test case execute scripts which are designed as GUI Layer (the extension of an OR with dictionary object) and Business Layer (Actions).

  This design pattern is published by Meir Bar-Tal on December 20, 2008 - Implementing a GUI Layer with Classes, the core concept of the design pattern is first checking if the correspondi ng GUI objects "exist" before doing actions, providing a highly effective way that enables to exit a test smoothly, preventing QTP from getting stuck when it fails to identify GUI objects during runtime, and locate details in the report. Effectively reduce maintenance and save au tomation testing time.

- The Reports directory contains user-defined html report to offer more flexibility in reviewing test results. Detail for customized report please refers to “Custom Report Reference Guide” document.