



Open2Test Test Automation Framework for Selenium Web Driver - Introduction

Version 3.1.0

March 2016

DISCLAIMER

Verbatim copying and distribution of this entire article is permitted worldwide, without royalty, in any medium, provided this notice is preserved.

TABLE OF CONTENTS

1. PREFACE	3
2. FRAMEWORK OVERVIEW	4
2.1. Introduction to Framework	4
2.2. Framework Features	4
2.3. Framework Benefits	5
3. FRAMEWORK ARCHITECTURE	7
3.1. Framework Architecture	7
3.1.1. Selenium.Java.....	8
3.1.2. Object Repository.....	8
3.1.3. Keywords.....	9
3.1.4. External Test Data.....	9
3.1.5. Global Variables.....	9
3.1.6. Reporting.....	9
4. FLOW DIAGRAM OF THE OPEN2TEST TEST AUTOMATION FRAMEWORK.....	10
5. CONCLUSION.....	11

1. Preface

Automation testing draws maximum benefits with minimum effort. The benefit of automation testing is its ability to increase the efficiency of resources, increase test coverage, and increase the quality and reliability of the software.

While there are several frameworks that provide support for automated software testing, this document introduces one particularly effective type: the **Open2Test Test Automation Framework**.

In the Open2Test Test Automation Framework, the discrete functional business events that make up an application are described using keywords. This approach fosters code reusability, optimum utilization of the tool, and greater productivity.

2. Framework Overview

2.1. Introduction to Framework

Automation testing requires a well-defined approach, based on a comprehensive framework, in order to reap maximum benefits.

A **framework** is a hierarchical directory that encapsulates shared resources, such as a dynamic shared library, image files, localized strings, header files, and reference documentation, in a single package.

There are various frameworks available for automation, such as:

- Test Script Modularity Framework
- Test Library Architecture Framework
- Data-Driven Automation Framework
- Hybrid Automation Framework
- Keyword-Driven Automation Framework

Keyword-based test design and test automation is formed on the idea that the discrete functional business events that make up any application can be described using short text description (keywords). By designing keywords that describe those discrete functional business events, testers begin to build a common library of keywords that can be used to create test scripts.

The Open2Test Test Automation Framework is a keyword-driven automation framework that works with Selenium Web Driver. This framework allows testers to develop test cases using Microsoft Excel using a list of keywords. When the test is executed, the framework processes the Excel workbook and calls functions associated with the keywords entered in the Excel spreadsheet. These keyword functions in turn perform specific actions against the application under test (AUT). The framework interprets the keyword and executes a set of statements in the program.

With this framework in place, applications can be automated without starting from scratch. Testers simply use the application-independent keywords and create extra application-specific keywords.

2.2. Framework Features

In addition to standard features such as performing operations and verifications on the objects, the framework includes other sophisticated features, such as:

1. **Use of variables:** Variables can be defined and used across the generated test script. This can be used to capture runtime values, which can be reused as input elsewhere during test execution.
2. **Conditional checking:** Conditional constructs such as 'if' can be implemented using keywords to handle different flows based on various conditions.
3. **Data-driven testing:** This framework supports data-driven testing by importing data from an external data sheet.

4. **Reports:** Customized reporter messages can also be used to perform effective analysis on execution reports. These reports can be customized to display the pass or fail condition of any functionality, even during the verification of any checkpoints.
5. **Calling functions and reusable actions:** Common functions or actions can be triggered through keywords. This framework supports a functional decomposition approach. This increases the reusability of functions, which in turn reduces the unnecessary repetition of steps.
6. **Exception handling:** Runtime errors can be effectively handled and reported using this framework.

2.3. Framework Benefits

Reusability:

The Open2Test Test Automation Framework is an application-independent framework that deals with all possible actions and verifications that can be performed on an object. Therefore, the code for the same object can be used across different applications.

Duplication of work is minimized at every level. For instance, a user might have to perform a certain action on an object of a similar class (e.g., clicking a button) repeatedly. This can be in the same test case or in a different application altogether. In both cases, the same code can be reused.

Optimum utilization of the tool:

The framework has the advantage of using keywords as the input for triggering an action. This well-built framework uses the features of the tool effectively. For instance, there is a shared object repository where all the objects required can be added and reused across the scripts for an application under test.

Less effort:

The effort involved in coding and reviewing is minimal when compared to other frameworks, since a good percentage of coding is done within the framework. The tester simply has to enter the keywords, reducing the time required for coding. Recording is also not required as the global repository is used. The amount of rework required for migrating from one application to the other on the same platform is reduced since the code remains the same.

Increased quality:

The scripts will be of uniform quality since they make use of the same code.

Greater productivity:

The Open2Test Test Automation Framework provides both qualitative and quantitative benefits for automation, and it is highly productive compared to any other framework. This framework also addresses the ongoing maintenance of the test scripts in a cost-effective manner.

Maintenance:

Simple modifications to the application can be easily handled. The changes will be done only in the external file containing the script and the code need not be changed. Hence it is easy to maintain the scripts and provide cost-effective solutions for the test automation.

No scripting skills required by the end user:

No coding skills are required to automate and review the scripts. The scripts are user-friendly with good readability. Scripts can be interpreted easily by a person who does not have complete knowledge of the tool.

Return on investment is high:

Although the initial effort for building the framework is high, in the long run, the return on investment will be high because of the reusability and optimum utilization of the tool.

3. Framework Architecture

3.1. Framework Architecture

Architecture forms the foundation of any software application. It should be robust enough to handle the desired functions efficiently and effectively. In this approach, the goal is to develop an application-independent reusable-keyword framework that can be used directly across any application without spending any extra time on it.

In order to make all the components of the system work in sync, it is important to define the components and its functionalities, as well as the binding relationship between them.

An Automation Framework Architecture comprises the following components:

- **Framework**

The framework consists of the following sub-components:

- Selenium.java

- **Abstract Layer**

The abstract layer consists of the following sub-components:

- Object Repository
- Keyword scripts

- **External Data**

External data consists of the following sub-components:

- Data Sheets
- Global Variables

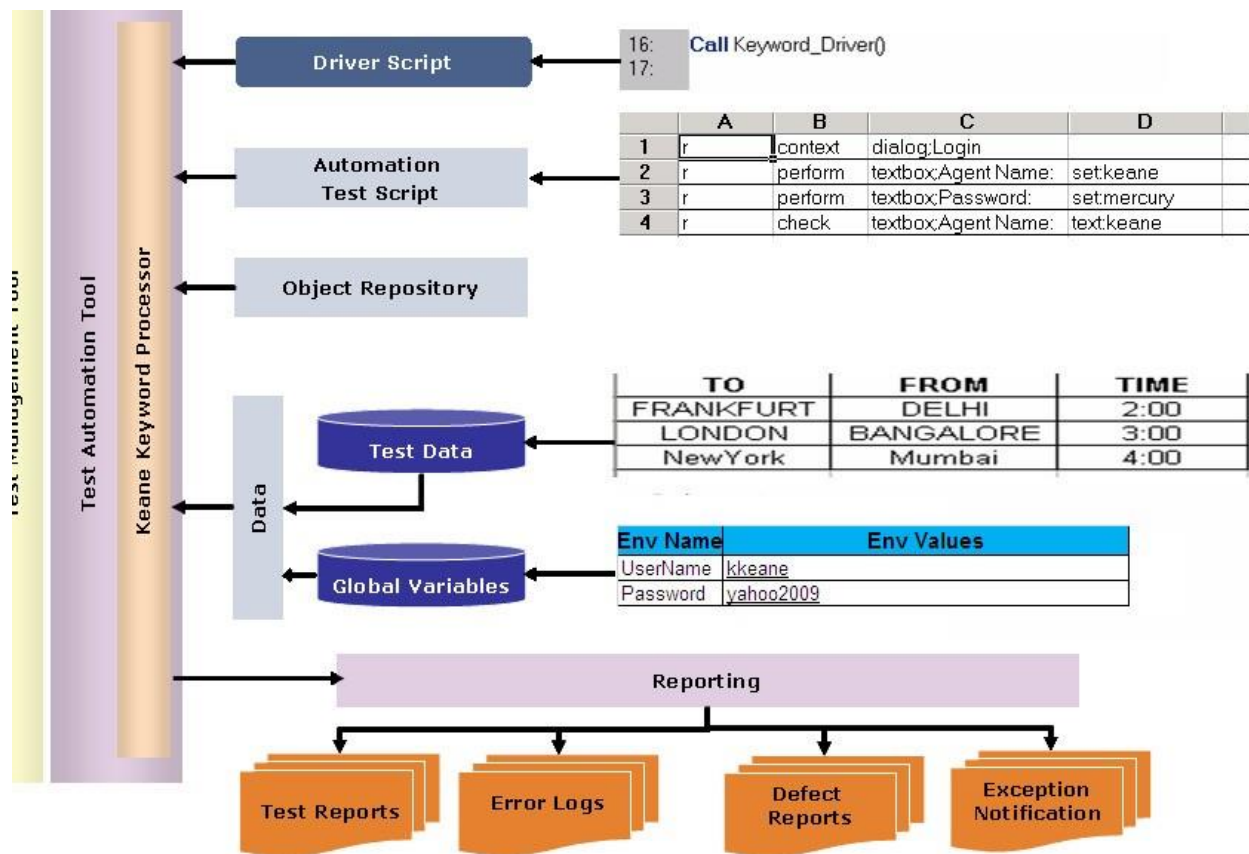


Figure 1: Framework Architecture

3.1.1. Selenium.Java

This is the main framework file that deals with the processing logic of the keyword scripts so that when the test is executed, the actions are performed on the application under test. It also interacts with the Selenium server to launch the browser.

This forms the backbone of the Automation Framework. It involves importing the test suite, test script, and object repository located in the form of Excel sheets. It holds the function for customizing the reports.

3.1.2. Object Repository

Selenium learns the interface of an application by learning the application's objects and their corresponding property values and objects descriptions. The object repository is the place where Selenium recognizes and stores information such as properties, values, etc.

With this object repository, the tool identifies the application and executes the business flows as per the functions in the Test script. If any one or more of the object's property values in the application differs from the property values stored in the object repository, the

tool will not identify the object and the script will fail. Therefore, it is necessary to change the property value in the object repository if it differs from the property value in the application.

3.1.3. Keywords

Keywords trigger specific functions in the framework to perform a specific operation on the desired object in the application. These keywords are fed as an input Excel sheet.

3.1.4. External Test Data

External test data is given as input to the test scripts to perform the same operations on the application using different set of data. This spreadsheet holds multiple combinations of inputs to be fed to test the application. The best practice here is to keep the data sheet in a common place, preferably in the test management tool.

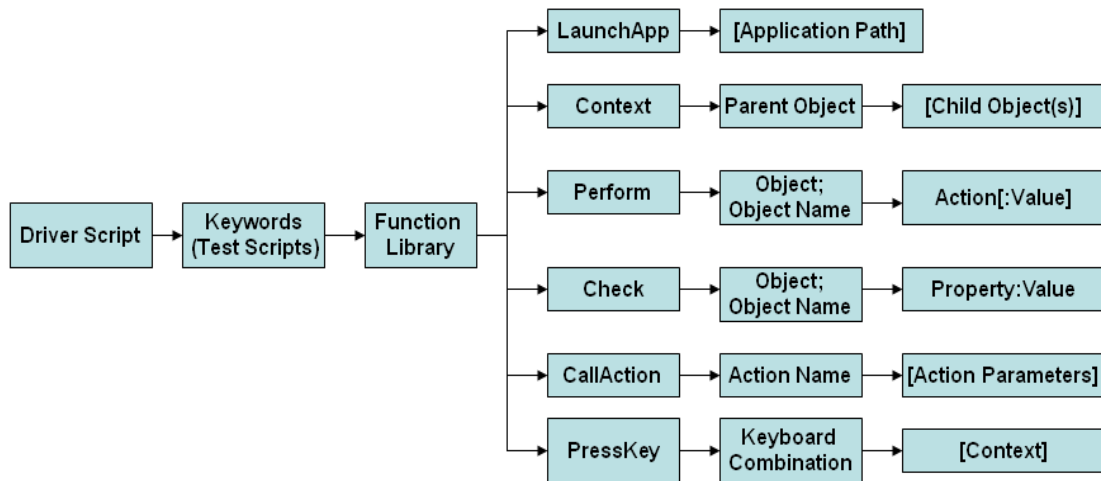
3.1.5. Global Variables

Global variables to be used across all the test cases should be defined and kept in a common place and should be provided as an input to the framework. After defining a variable to refer to a test object, you can use the variable instead of the entire object hierarchy in other statements. Using variables in this way makes the script easier to read and maintain. These global variables can also be used to store runtime values and used again in another script.

3.1.6. Reporting

After execution of the test script, it is necessary to get the results of the execution. The reports are customized in the framework such that the summary report and the detailed report are stored in html format. This provides report details for which test scripts have failed or passed while running a test suite. The summary report provides details of execution, duration, test start time and end time. The detailed reports describe exceptional cases handled, steps passed, and steps failed. This helps in performing effective analysis on the execution report.

4. Flow Diagram of the Open2Test Test Automation Framework



Note :

The values within [] are optional, usage depends on the scenario

Figure 2: Flow Diagram

5. Conclusion

Analysis is an important and time-consuming phase of automation testing. However, in the long run, the time spent will be useful during the regression phase. In order to keep up with the pace of product development and delivery, it is essential to implement effective, reusable test automation. The Open2Test Test Automation Framework provides a way to drive productivity and foster code reuse — ultimately enhancing the quality of resulting software.

COPYRIGHT

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.