Selenium Open Source Test Automation Framework FAQ

Version 1.0

September 2009

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this document is to answer frequently asked questions about the (Selenium) Open Source Test Automation Framework.

2. Frequently Asked Questions

2.1. Open Source Test Automation

The table below lists FAQs with respective answers related to the Selenium Open Source Test Automation Framework.

1.	Is the Open Source Test Automation Framework application-dependent? **No, it is not application-dependent because the Open Source Test Automation Framework is designed to perform operations for all standard object types (WebEdit, Weblist, Webcheckbox, etc.) of technology commonly used to build a Web application.**
2.	Is Open Source Test Automation Framework technology-dependent? **Yes, it is technology-dependent.**
3.	What are the benefits of using this framework? **The Open Source Test Automation Framework provides the following benefits:** **1. The testers can automate test cases without the help of programmers or programming background.** **2. They can run automated test cases more reliably.** **3. The Open Source Test Automation Framework reduces maintenance and increases productivity.** **4. Keywords are application-independent.**
4.	Does the Open Source Test Automation Framework support a data-driven framework? **Yes.**
5.	Does it support a function modular framework? **Yes.**
6.	What are the prerequisites for using the Open Source Test Automation Framework? **The prerequisites for using the Open Source Test Automation Framework are:** **1. Knowledge of the keywords.** **2. Basic knowledge of the Selenium tool.** **3. All the necessary files should be associated.**
7.	Does the Open Source Test Automation Framework support all versions of Selenium? **Yes.**
8.	What are the various components of the Open Source Test Automation Framework? **The components include: Main.rb, functionlibrary.rb, selenium.rb, object repository, keywords, external test data, and global variables.**
9.	What are the different components needed to run a Selenium script using the Open Source Test Automation Framework? **Place the keyword script and test data (if used) in the location specified in the framework. Place the Object Repository Excel in the Specified location.** **Save the Open Source Test Automation Framework to the system and then right click on the main.rb and select 'Edit'.** **Make the Selenium Server up and running. Press F5 to run the script.** **Or** **Launch command prompt**

	Go to path main.rb **Run the below command** **Ruby main.rb**
10.	Does the Open Source Test Automation Framework support external functions? **Yes, it supports external functions.**
11.	Does the Open Source Test Automation Framework support using an xml file for test data? **No.**
12.	How does the Open Source Test Automation Framework work? **Using the Open Source Test Automation Framework, testers can develop test cases using Microsoft Excel and the available list of keywords. When the test is executed, the framework processes the Excel workbook and calls the functions associated with the keywords entered in the Excel spreadsheet. These keyword functions in turn perform specific actions against the Application Under Test (AUT).**
13.	What are the benefits of the Open Source Test Automation Framework? **Reusability, greater test productivity, optimum utilization of the tool through keyword support, and minimum effort needed to build scripts.**
14.	What are the features of the Open Source Test Automation Framework? The following are the features of the Open Source Test Automation Framework: **Performing operations and verifications on the objects** **Usage of variables** **Conditional checking** **Data-driven testing** **Reports** **Exception handling** **Handling Web objects**
15.	How reliable is this framework as compared to a linear script? **The framework provides standardized, tested code. It is typically much more reliable and more thoroughly tested than recorded scripts. It also provides uniformity across automation scripts and ensures standard procedures are followed for coding.**
16.	What is the advantage of building scripts using the Open Source Test Automation Framework over writing code in Ruby? **Being a keyword-driven framework, one does not need to know Ruby programming. Testers can develop scripts without learning the underlying automation tool. The tests are easier to understand and maintain, and they provide maximum code reuse.**
17.	How do I decide whether I can automate an application using the Open Source Test Automation Framework? **Currently the Selenium Open Source Test Automation Framework supports automation of web applications.**
18.	What is the Open Source Test Automation Framework? **This framework is built for several of the leading test automation tools. It allows scripting of test cases using a set of keywords provided as part of the framework. This is often referred to as keyword-driven testing or action-based testing.**
19.	What is the use of Selenium.rb? **Selenium.rb is the ruby file that holds the built-in functions of Selenium and it has to be present in the framework code to perform those actions on the application.**

20.	How do we integrate AutoIt with Selenium?
	Before starting the integration of AutoIt, we need to install AutoIt in the machine. Once the installation is complete, we can use the following method to access the command from Ruby.
	
	AutoIt command can be accessed from Ruby by creating an object of AutoIt in the Ruby script using Win32OLE. Win32OLE extension library provides an interface to OLE Automation from Ruby.
	
	`require 'win32ole'`
	`application = WIN32OLE.new('AutoItX3.Control')`
	
	Now using Win32OLE a new object-named application is created for accessing AutoIt commands. This will include all the commands of AutoIt.

21.	How does Selenium handle popup windows? **Using Multithreading concept of Ruby and AutoIt, we can handle all the popups that are present in any web-based application during the execution of any step.**

22.	How does the Multithreading concept works with Selenium?
	The below code will explain the implementation,
	
	`application = WIN32OLE.new('AutoItX3.Control')`
	`@@t1 = Thread.new {@selenium.click <btnname>}`
	`sleep 5`
	`Thread.kill(@@t1)`
	`application.ControlClick("<windowname>","","<buttonname>")`
	
	Usually when we get a windows popup upon clicking any object in the application, Selenium RC will not get any response from the server. This will cause the execution to expire without proceeding to the next step.
	
	The above code will handle this issue. If there is a button, which when clicked gives us a windows popup, then we click that button in a separate thread. That way, this thread will be killed after clicking the button and will not wait for any server response. The main thread will continue the execution. Now the popup is active, so using the command "ControlClick" of AutoIt we can click on either the "OK" or "Cancel" buttons in any popup.

23.	Is it possible to type a value in the edit box of a popup? **The keyword below is used to type any value in an edit box of a popup.** **<Value> is the value that needs to be typed.** **Details of <Window Name> should be specified in Object Repository.** • **r | popup | <window name>;presskey| <Value>**

2.2. Keyword Scripts

The table below lists the FAQs with respective answers related to keyword scripts.

1.	What are the various keywords available in the Open Source Test Automation Framework? **Refer to the Keyword Reference Document for the list of keywords that are available in the Open Source Test Automation Framework. The document covers all the keywords along with their syntax and usage.**
2.	Does the Open Source Test Automation Framework support a mechanism for providing multiple sets of data to the same parameter? **Yes. Use the Loop keyword.**
3.	Does the Open Source Test Automation Framework support if-else loops? **Yes. Use the Condition keyword.**
4.	Does the Open Source Test Automation Framework support reusable actions? **Yes. Please refer to the Keyword Reference document for the keyword to be used.**