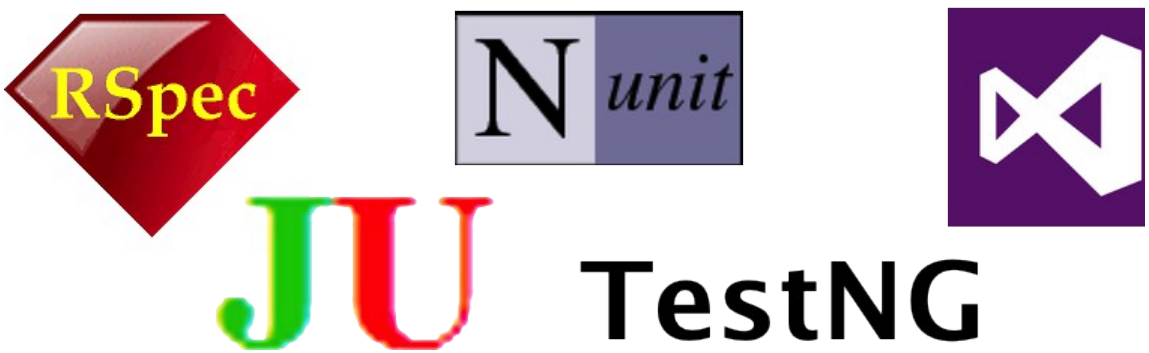
What is Selenium?

- Selenium automates browsers.
- It drives browsers natively.
- It works with Firefox, Chrome, IE and Opera.
- It is most commonly used to automate testing of web applications.




Selenium Setup

- You can run it locally or set it up on several servers to scale your testing.
- Selenium can be controlled manually or by most testing frameworks.



Writing Tests

- Tests can be written natively in Selenium.
- Or you can use the Selenium API to write in C#, Java, Python and Ruby.



Running Tests

- Selenium tests can run manually.
- Or through your build automation suite.

Keyword-Driven Testing

Selenium tests can be configured in an external file.

This means that a tester can configure a set of tests in an Excel sheet just by writing down keywords for each test case along with any necessary data without touching the test code.

Which in turn means that non-technical testers can configure new tests.

This is called **keyword-driven testing**.

Data-Driven Testing

A Selenium test suite can be written in such a way as to load data for the test from an external file or database.

This means that testers can control the data that is loaded without touching the test code.

Which in turn means that non-technical testers can create new sets of data that can result in new types of tests.

This is called **data-driven testing**.

Hybrid Testing

Keyword- and data-driven testing approaches can be combined.

This gives non-technical testers a lot of flexibility to create new tests that respond to changes in the software under test.

But it requires an experienced automation engineer to set up the test suite in a way that minimizes the amount of technical maintenance required.

Integration and Reports

Since Selenium is just an API for browser automation, it integrates with your integrated development environment or application lifecycle management tool like any other API and lets you log and report results natively in the IDE or ALM.

Test Results Summary

Test Details :

Run Date	Fri Jul 05 20:21:08 IST 2013
Run Environment	Window 7,IE8(Compatibility mode enabled)
Release	0.1.3

Report :

SUITE NAME	DESCRIPTION	EXECUTION RESULT
Individual	Individual window	FAIL
Group	Group window	FAIL
Add Document	Add Document window	PASS
Specific Communication	Specific Communication window	FAIL
Group Contact	Group Contact window	PASS

Group Contact Detailed Results

TCID	TSID	Description	Keyword	Object	Data	Result1
Group Contact	TS01	Open the browser	openBrowser		config|browserType	PASS
Group Contact	TS02	Navigate to the test site	navigate		config|testsiteBaseURL	PASS
Group Contact	TS03	Write User ID in user ID field	writeTextByID	userID	col|User ID	PASS
Group Contact	TS04	Write Password in the Password field	writeTextByID	pswrdID	col|Password	PASS
Group Contact	TS05	Click on Sign In button to login	clickByID	signInButtonID		PASS
Group Contact	TS06	Click on recipients in the left navigation bar	clickUsingJSByID	recipientID		PASS
Group Contact	TS07	click on the Groups tab in the Recipient window	clickByID	recipientGrptabID		PASS

Extensible

Selenium is an open source project. The Selenium community has created plugins and documented solutions that solve most problems.

Here are a few interesting drivers and extensions:

- Drivers for testing on Android and iOS-based browsers.
- Drivers for automating testing desktop applications with AutoIT.
- Drivers for testing Android and iOS apps with Selenium.

OptimusQA
Your Trusted QA Partner

Selenium Testing in the Wild

Case study examples of companies that are benefitting from Selenium automation (even though they thought it wasn't right for them).

Frequently Changing Ecommerce Site

Conventional wisdom holds that test automation isn't worth the investment if the user interface of your application changes frequently. And that's exactly why one electronics retailer hadn't bothered with Selenium automation for their ecommerce site.

The truth has long since moved on from conventional wisdom.

Using Selenium and Java we generated GET and POST requests through HTTP, parsed the responses and then used Selenium's native methods for validating test results thereby skipping the user interface entirely.

We set up a series of Excel spreadsheets with login, credit card, items to purchase and other details so that testers can change the automated tests without having to edit the test code.

By automating one level above the user interface and providing an Excel-based interface for testers, the retailer was able to automate testing while minimizing maintenance in a way that did not require frequent maintenance by an automation engineer.

SaaS Web Application Testing with Selenium

A software as a service web application provider wanted to automate their regression test suite to reduce the friction involved in testing and deploying changes to their product.

The effort involved in manually running their regression testing meant that not all of their builds were being thoroughly tested before being pushed out. They hadn't had any major bugs slip into production, but realized that it could happen at any time.

The company wasn't willing to commit the resources to hire a full-time automation engineer to maintain the test suite either.

A hybrid keyword- and data-driven testing framework was clearly in order. But even a hybrid automated solution was going to require too much maintenance by an automation engineer to justify the expense.

In addition to pulling test scenario and test data definitions from Excel sheets in Java-based test code, we also abstracted page object definitions to an external configuration file.

This allowed testers to update existing tests when user interface elements changed, and even create some new tests.

Detailed and summary test reports were generated automatically giving testers, developers and managers the information they need to act on the results.

Resources

- **Selenium**: WebDriver Documentation
- **Google Code**: Selenium Project Home
- **Google TechTalks**: Automating Your Browser Based Testing Using WebDriver