Quality Assurance Automation

On-boarding

Draft Version 1.1

Table of Contents

[1. Introduction 4](#_Toc356470318)

[2. Developer Tools/Packages 4](#_Toc356470319)

[2.1. Overview of Tools 4](#_Toc356470320)

[2.2. Installation 5](#_Toc356470321)

[2.2.1. Java 5](#_Toc356470322)

[2.2.2. Eclipse 5](#_Toc356470323)

[2.2.3. Install Java Packages 5](#_Toc356470324)

[2.2.4. Web Driver 7](#_Toc356470325)

[2.2.4.1. IEDriverServer 7](#_Toc356470326)

[2.2.4.2. Chromedriver 7](#_Toc356470327)

[2.2.5. Subversion 7](#_Toc356470328)

[2.2.5.1. Tortoise SVN 7](#_Toc356470329)

[2.2.5.2. Excel Add-Ins 7](#_Toc356470330)

[2.2.6. Selenium IDE 7](#_Toc356470331)

[2.2.7. Firebug 9](#_Toc356470332)

[3. Setting project (workspace) in Eclipse IDE 10](#_Toc356470333)

[3.1. Pre-Defined Environments (Projects) 11](#_Toc356470334)

[3.1.1. Create a new local Project (My Project) 11](#_Toc356470335)

[3.1.2. Sandbox – Project environment is defined and build path is set up 14](#_Toc356470336)

[3.1.3. Project Specific (ready only to public, owned by Automation group) 15](#_Toc356470337)

[4. Framework – Galaxy demo project 16](#_Toc356470338)

[4.1. Description 16](#_Toc356470339)

[4.2. Architecture 16](#_Toc356470340)

[4.3. Framework Structure 17](#_Toc356470341)

[4.4. Installation 17](#_Toc356470342)

[4.5. Folder Structure 17](#_Toc356470343)

[4.6. Excel Templates 18](#_Toc356470344)

[4.7. Custom Utilities (Java Classes & Methods used for this framework) 18](#_Toc356470345)

[4.8. Test Execution 18](#_Toc356470346)

[5. Version Control – subversion 19](#_Toc356470347)

[5.1. Checkout Test Data File 19](#_Toc356470348)

[5.2. Folder Structure 20](#_Toc356470349)

[6. Test Data Development Processes 20](#_Toc356470350)

[7. Automation Development Processes 20](#_Toc356470351)

[8. Roles and Responsibilities 21](#_Toc356470352)

[8.1. Automation Developer (AD) 21](#_Toc356470353)

[8.2. Functional Tester (FT) 21](#_Toc356470354)

[9. Selenium IDE 21](#_Toc356470355)

[10. Firebug 23](#_Toc356470356)

**Version History**

|  |  |  |
| --- | --- | --- |
| Version | Author | Description |
| 0.1 | Jeff Lai | Initial Draft; Incorporated comments Ketan, Syed, Hakim. |
| 2.0 |  |  |

# Introduction

This document explains the how developer tools, framework and processes needed for starting test automation for any Web Applications

# Developer Tools/Packages

# Overview of Tools

Download copy: \\eclipse\workspace\automation\Software\

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Tools/Packages | Purpose | Version | Require/Optional |
| 1 | Java | For Eclipse | 1.7 u17 |  |
| 2 | Eclipse | Java project IDE | Juno |  |
| 3 | Selenium Client | Web Driver Java language bindings | 2.32.0 |  |
| 4 | Selenium Server | Use to run Selenium RC style scripts or Remote Selenium Webdriver. Also use in a Grid configuration | 2.32.0 |  |
| 5 | JUnit | JUnit is a test framework which uses annotations to identify methods and provide assertions for testing expected results. | 4.11 |  |
| 6 | Apache-Ant | Java build tool to write a build.xml file. Run with JUnit framework and generate HTML/XML reports. | 1.9.0 |  |
| 7 | Apache-Poi | Java API for Microsoft Documents. Use to read/update Excel Workbook. To support Excel 2003, 2007 and 2010 | 3.9 |  |
| 8 | JDBC for DB2 | JDBC driver for IBM DB2, use to run SQL to fetch data directly to database | V10.1 | Optional |
| 9 | JDBC for Oracle | JDBC driver for Oracle, use to run SQL to fetch data directly to database |  | Optional |
| 10 | IEDriverServer | Internet Explorer driver for Selenium Webdriver |  |  |
| 11 | Chromedriver | Chrome driver for Selenium Webdriver |  |  |
| 12 | Microsoft Excel | Input source for test set, test data, result log and repository for WebElements catalog | Office 2007 and 2010 |  |
| 13 | Tortoise SVN | Client tools for committing changes to subversion, version control | 1.7.4 – x64  1.7.3 – win32 |  |
| 14 | msofficesvn | Excel Add-in for committing changes to subversion, | 140 |  |
| 15 | Selenium IDE | Firefox plugin does record-and-playback of interactions with browser, use this to either create simple scripts, assist in exploratory testing. | 2.0.0 |  |
| 16 | Firebug | Tools for debugging web application’s HTML | 1.11.3 |  |
|  |  |  |  |  |

# Installation

# Java

Java is needed for running Eclipse and ANT migration tool (if applicable). The minimum required version for ANT migration tool is 1.6.x. If you don’t have Java in your system, then see below to install Java 1.7.x version.

**File Name:** jdk1.7u17-x64

**Version: Java Version:** 1.7u17 64-bit

Steps:

1. Unzip the file to C:\Program Files\Java folder on your desktop
2. Open Window Explorer , right click on Computer and select Properties
3. Click on Advanced System Setting
4. Click on Environment Variables
5. In the System Variables section, click on New button.
6. In the Variable name field enter JAVA\_HOME
7. In the Variable value field, enter C:\Program Files\Java\ jdk1.7u17-x64
8. In the System Variables section, check if CLASSPATH variable exists. If it doesn’t exist, create by clicking on New button
9. Select CLASSPATH variable and click on Edit button
10. In the Variable value field, go to the end of the value, put “;” and enter %JAVA\_HOME%\bin and click on OK
11. In the User Variables section, check is PATH variable exists. Create one if it doesn’t exist
12. In the Variable value field of PATH, enter %JAVA\_HOME%\lib

# Eclipse

**File Name:** eclipse-jee-juno-SR1-win32-x86\_64

**Version**: juno

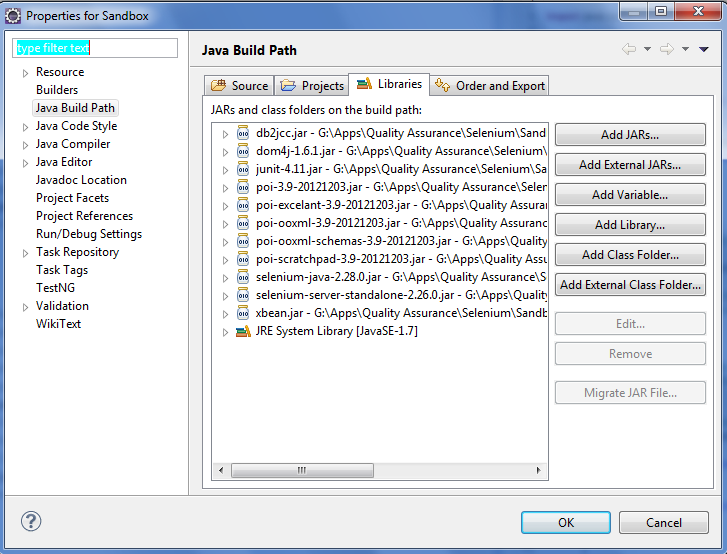
Steps:

1. Download the zip and extract it to C:\eclipse
2. Set Default workspace to c:\eclipse\workspace
3. Run InstallRefLib.bat to copy Referenced Libraries files to local machine

# Install Java Packages

Add required packages into project build path. You can either setup the build path locally or map to shared referenced libraries.

* 1. Selenium Client
  2. Selenium Server
  3. JUnit
  4. Apache–Poi
  5. JDBC for DB2
  6. JDBC for Oracle



* 1. Apache-Ant

Apache Ant: Used to compile, Run and to create a test report of overall pass/fail results of the executed scripts.

Version: ant-1.9.0-rhi-04.00.000.00

File Name: ant-1.9.0-rhi-04.00.000.00.zip

Steps:

Unzip the file to C:\APPS folder on your desktop

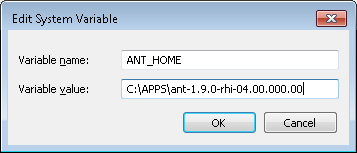
Set following system variables:

ANT\_HOME=C:\APPS\ant-1.9.0-rhi-04.00.000.00

JAVA\_HOME= C:\APPS\jdk1.7u17-x64

Add following to the end of existing PATH system variable:

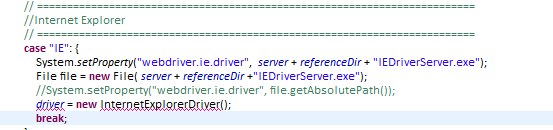
%ANT\_HOME%\bin; %JAVA\_HOME%\bin



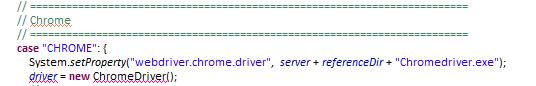


# Web Driver

# IEDriverServer



# Chromedriver



# Subversion

# Tortoise SVN

This is required for committing the code and test data from your desktop. An alternative would be to utilize Eclipse’ subversion plug-in to commit the source code/test data file

**64-bit:**

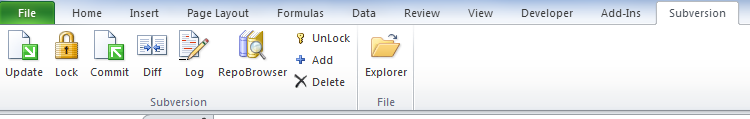
\\na\corpshared\apps\quality assurance\selenium\sandbox\referenced library\ TortoiseSVN-1.7.4.22459-x64-svn-1.7.2.msi

**32-bit:**

\\na\corpshared\apps\quality assurance\selenium\sandbox\referenced library\TortoiseSVN-1.7.3.22386-win32-svn-1.7.2.msi

# Excel Add-Ins

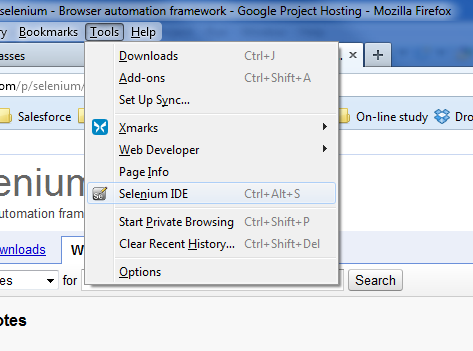
This Add-In is to manage check in and commit test data file to Subversion.

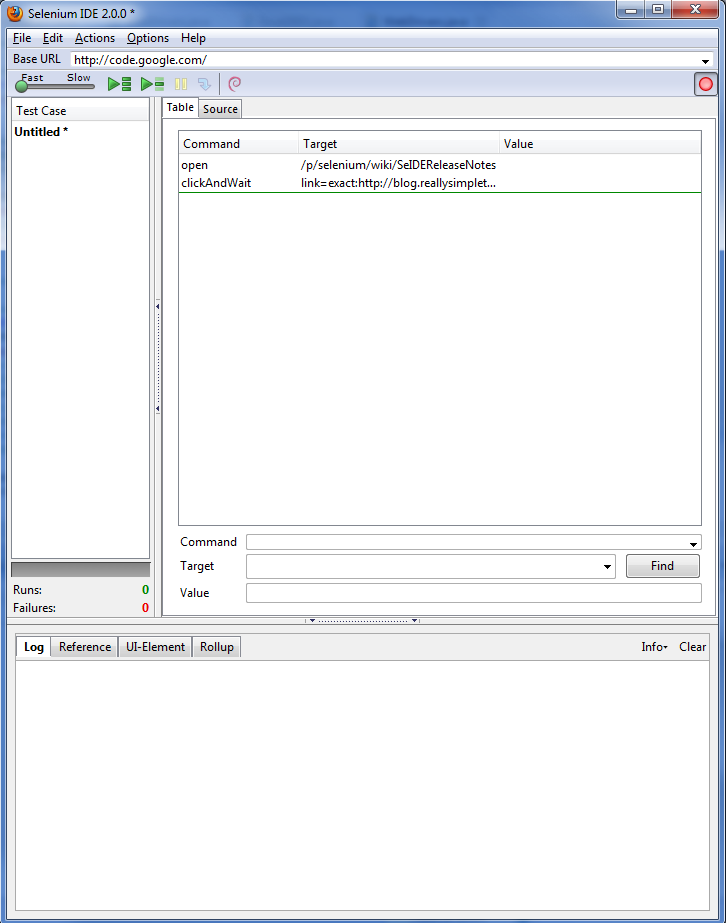


# Selenium IDE

Download site: <http://docs.seleniumhq.org/download/>

Selenium IDE installed under Tools





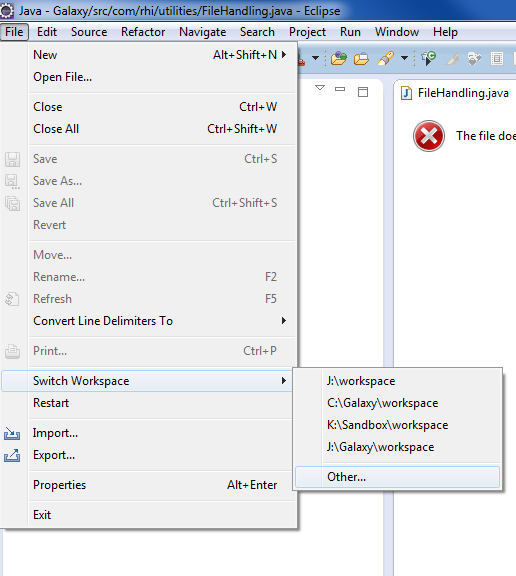
# Firebug

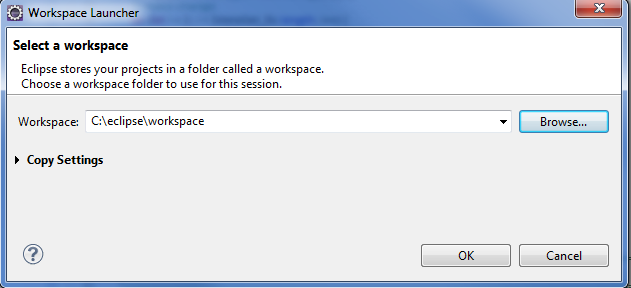
Firebug is add-on to Firefox that will help determine Web Element properties.

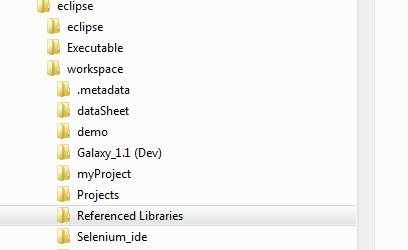
Firefox add-on can be installed by searching the Firefox Add-on page for the Firebug tool. https://addons.mozilla.org

# Setting project (workspace) in Eclipse IDE

Set default workspace to: c:\eclipse\workspace

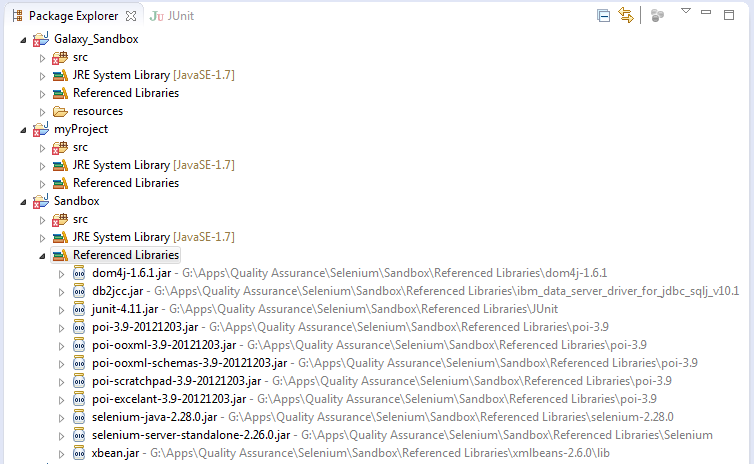




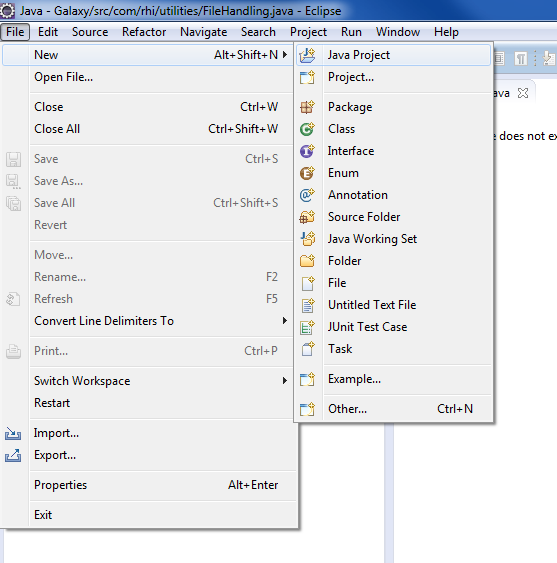
* Use Windows Explorer to create “Referenced Libraries” folder in workspace folder. The folder is used as local shared repository for tools and Java packages.
* 

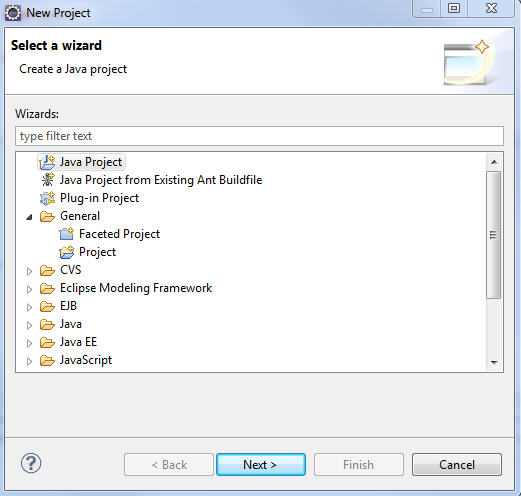
# Pre-Defined Environments (Projects)

1. My Project – local (your own development project)
2. Project specific – development environment for specific project (read only and owned by Automation group)
3. Sandbox – bare bone setup(public to everybody, place to try new codes and new packages)
4. Galaxy demo – Framework demo

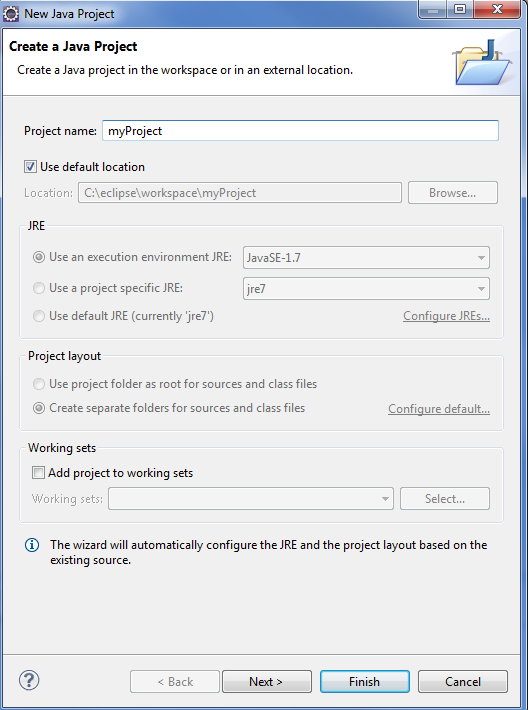


# Create a new local Project (My Project)

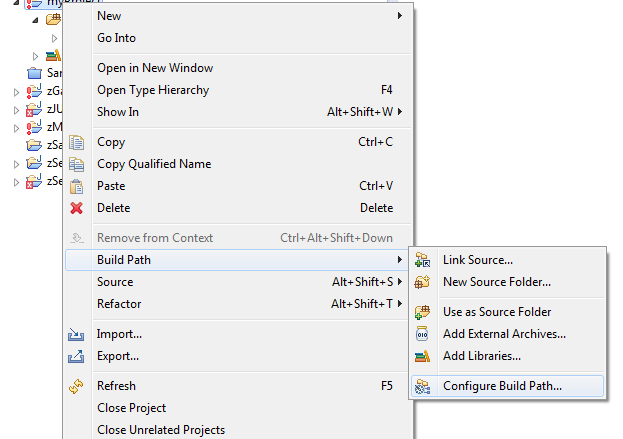


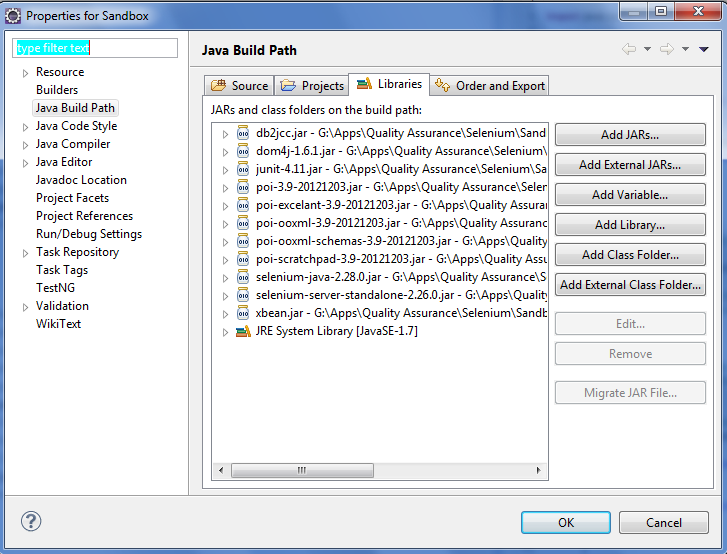


* **Check Use default location**

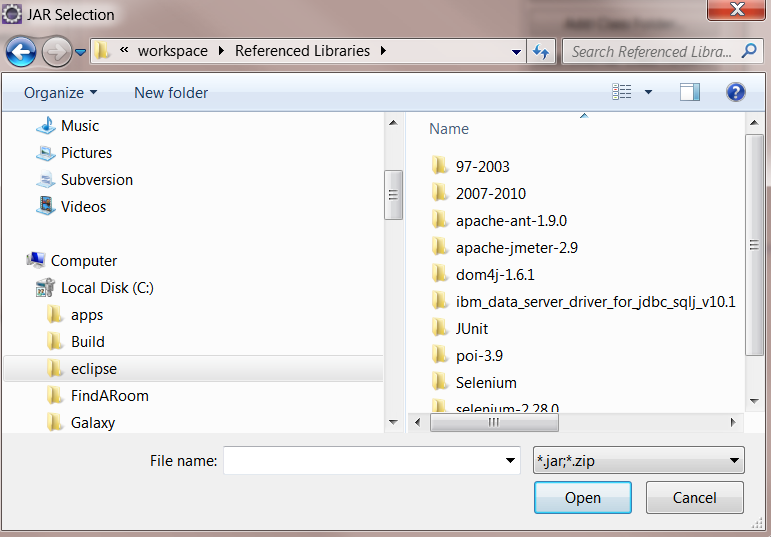


* **Set up Build Path: right click on [my Project] -> Select Configure Build Path -> Libraries tab**



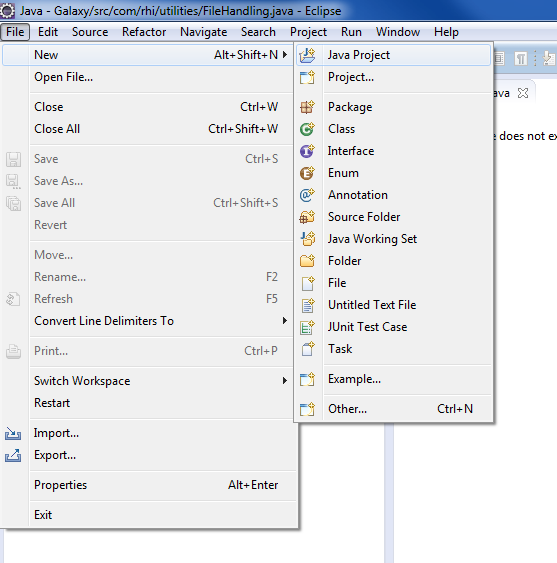


* **Click Add External JARs - c:\eclipse\Referenced Libraries**
  + Selenium - …\Selenium\selenium-server-standalone-2.26.0.jar
  + Selenium-java - … \selenium-2.28.0\ selenium-java-2.28.0.jar
  + JUnit - …\JUnit\junit-4.11.jar
  + Apache–Poi - …\poi-3.9\ poi-3.9-20121203.jar; poi-excelant-3.9-20121203.jar; poi-ooxml-3.9-20121203.jar; poi-scratchpad-3.9-20121203.jar
  + DOM - …\dom4j-1.6.1\ dom4j-1.6.1.jar
  + xmlbeans - …\ xmlbeans-2.6.0\lib\xbean.jar
  + JDBC for DB2 - …\ ibm\_data\_server\_driver\_for\_jdbc\_sqlj\_v10.1\ db2jcc.jar

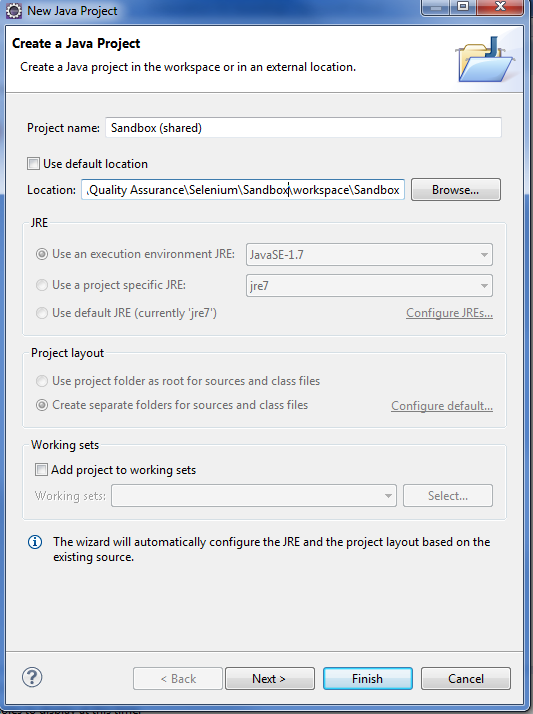


# Sandbox – Project environment is defined and build path is set up

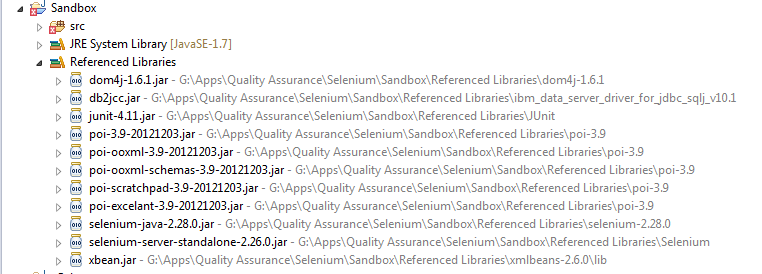
Create a new project and map to the predefined sandbox environment.



* **Uncheck Use default location**
* **Location: \\na\corpshared\Apps\QualityAssurance\Selenium\Sandbox\workspace\Sandbox**



**Build Path**



# Project Specific (ready only to public, owned by Automation group)

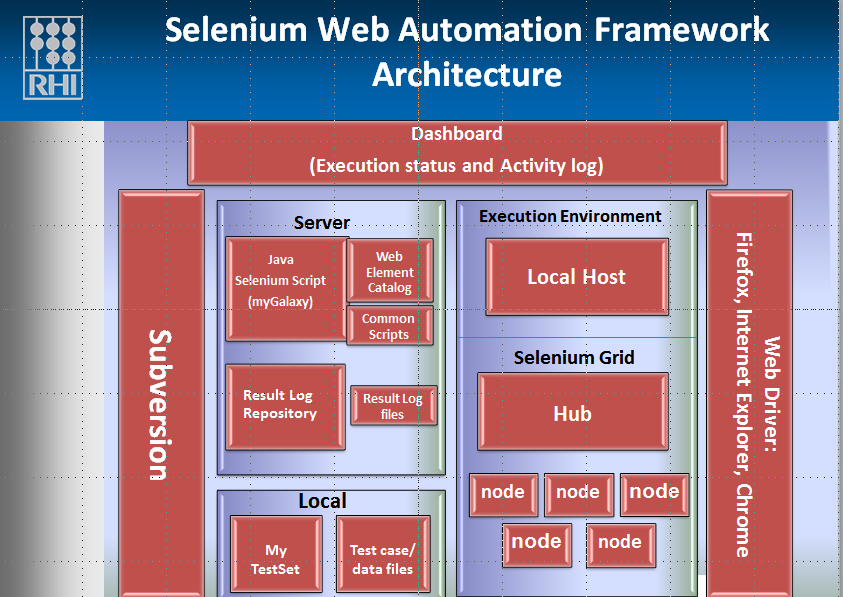
* 1. Create a new project
  2. Uncheck Use default location
  3. Map to specific project folder

# JSAT Framework – myGalaxy demo project

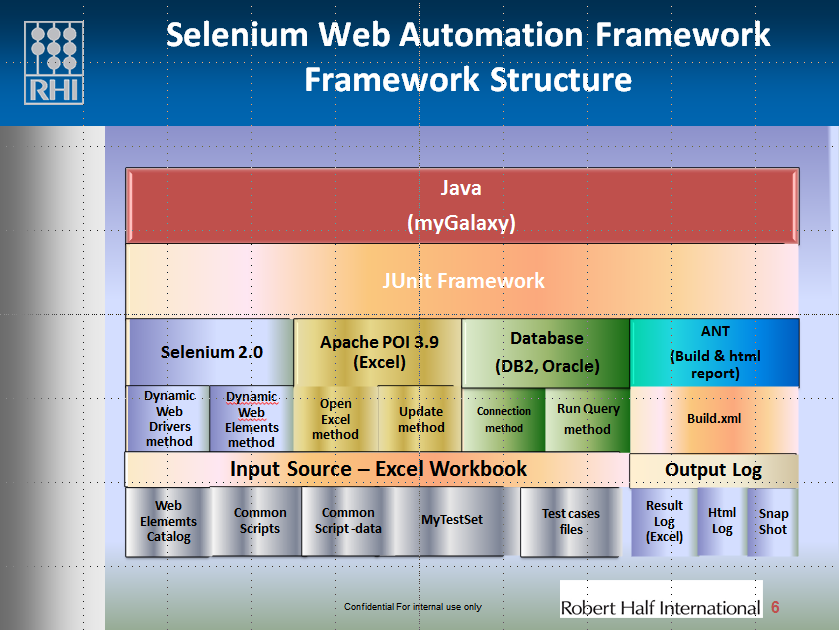
# Description

The objective of this framework is to standardize the automation approach, and to make testing automation more effective and efficient. The approach is to create main Java program using Selenium packages and JUnit framework for any Web Applications. Excel workbook is used as input source for testing data and execution steps (real-time find web elements generated thru dynamic method).

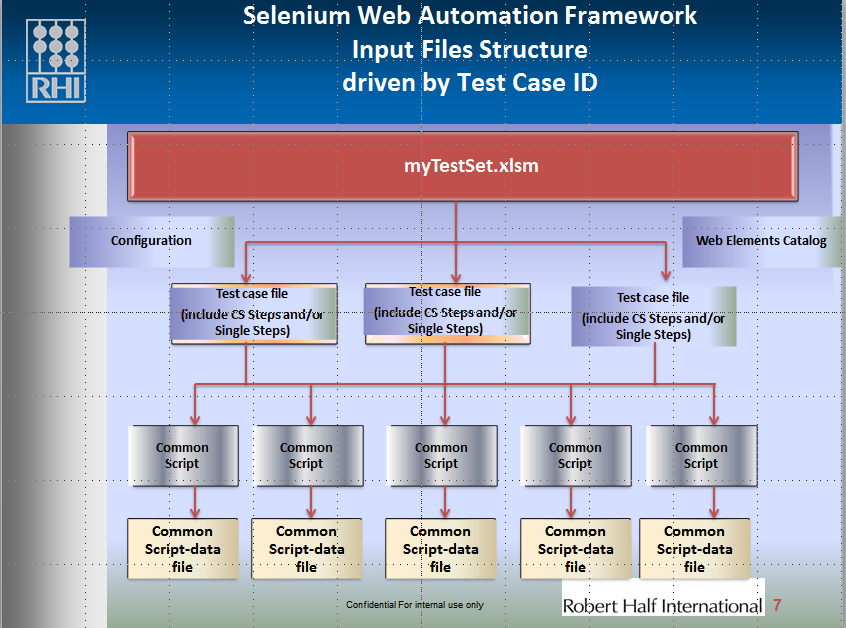
# Architecture



# Framework Structure



# Input Files Structure



# Web Elements and Identifier supported

**Standard Web Element Type**

1. WebEdit
2. WebButton
3. WebFile
4. WebRadioButton
5. WebSelect(Dropdown)
6. WebLink
7. WebCheckedBox
8. WebImage ()
9. URL ()
10. Windows ()

**Standard Web Element identifier**

1. id
2. xpath
3. name
4. linkText
5. partialLinkText
6. cssSelector

&linkText&

**Salesforce specific method**

WEC Identifier=: “SF”

WEC Type:-

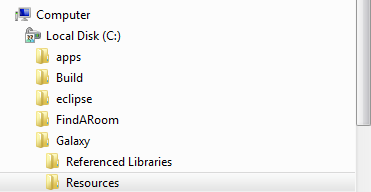
1. [WebLookup\_SF]=: SF\_WebLookup(driver)
2. [UpLoadDoc\_S\_SF]=: SF\_UploadDoc\_S(driver)
3. [UpLoadDoc\_E\_SF]=: SF\_UpLoadDoc\_E(driver)
4. [SelectBox\_SF]=: SF\_SelectBox(driver)
5. [LookupLink\_SF]=: SF\_LookupLink(driver)
6. [WebButton\_SF]=: SF\_WebButton(driver)
7. [MultiSelectPickListRow\_SF]=: SF\_MultiSelectPickListRow(driver)
8. [MultiSelectPickListCell\_SF]=:SF\_MultiSelectPickListCell(driver)
9. [WebTableLinks\_SF=]: SF\_WebTableLinks(driver, runTestStep);
10. [WebTable\_SF=]: SF\_WebTable(driver, runTestStep);
11. [WTElement\_SF]=:SF\_GetWebTableElement(driver,runTestStep)
12. [URL\_SF]=:SF\_URL(driver)

# Installation

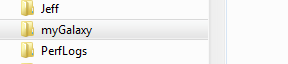
1. Send a request to Automation group ([jeff.lai@rhi.com](mailto:jeff.lai@rhi.com)) to create a personal development project.
2. Run Install bat file (myGalaxy.bat) to copy sample templates to local machine, folder is created.
   * myGalaxy – myTestSet and Test case files reside
3. Create a project in Eclipse and map to your personal project or application project.

# Folder Structure

1. **Galaxy**



1. **2) myGalaxy**



# Excel Templates

* Test Set (myTestSet)
* Test cases (Test cases/Scenarios)
* Common Script
* Test result Log (Execution log)
* Web Elements Catalog (Application specific)

# Custom Utilities (Java Classes & Methods used for this framework)

* WebDrivers
* WebElements
* Checkpoints
* FileHandling
* ResultLog
* SnapShot

# Test Execution

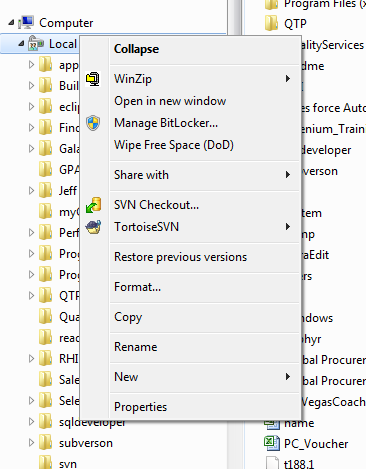
Executed from Excel Add-Ins Menu (To be released)

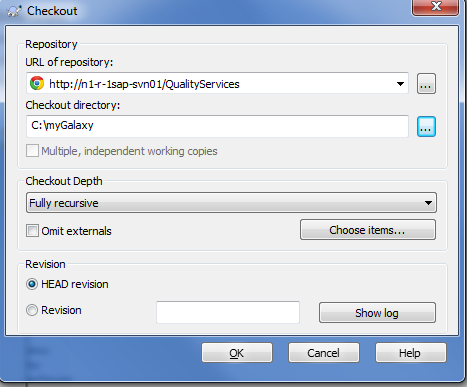
1. Checkout Test file from subversion
2. Change test data if needed
3. Add the test files into myGalaxy(Test Set)
4. Specific test case, url, browser, host etc
5. Execute
6. Review Result log

# Version Control – subversion

# Checkout Test Data File

* Install subversion client version on your desktop
* Right click on Windows Explorer, Select SVN Checkout…





# Folder Structure

* Vertical
* Applications
* Project
* Test case

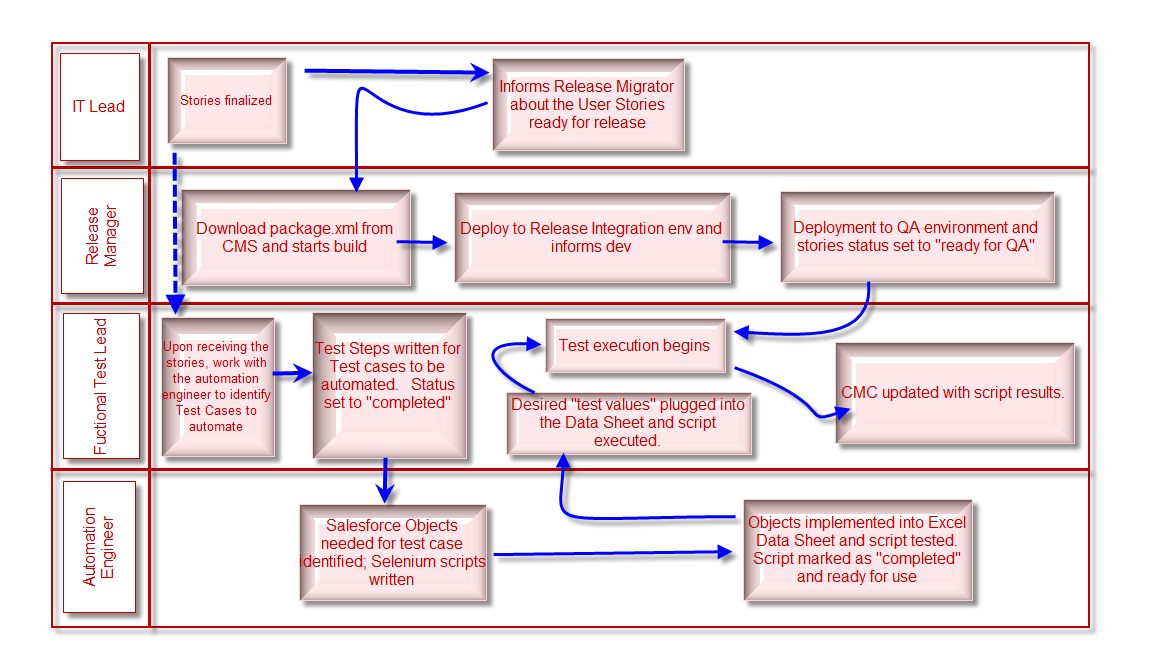
# Test Data Development Processes

* + When the Functional Test Lead (FT) identifies the test cases that needs to be automated and writes the test case steps, they will change the status for the test case to indicate that it is complete and ready to be worked on.
  + Based on the test steps, the automation developer (AD) will be able to decipher which pages and objects are needed for the test case to run. The automation engineer will then identify and implement those objects into the data file. They will plug in “dummy values” into the data file and ensure that the script works as desired.
  + Once the test data file has been turned over to the functional tester, they will then plug in the actual “test values” into the data file and run test case(s).

Workflow:

1. Functional Tester check test case as automatable. i.e. CMC (Sales Force).
2. Review by FT and AD to determine test case as automation candidate.
3. Write test case steps by FT.
4. Identify web elements by AD.
5. Shake down and execute test case.
6. Return test data file to FT and change data value applicable to the build by FT .
7. Add test data file in TestSet (myGalaxy) for execution.

# Automation Development Processes

* 1. Development Life Cycle

# Roles and Responsibilities

# Automation Developer (AD)

* Develop Java/Selenium scripts
* Maintain framework
* Maintain Java/Selenium scripts - via Subversion
* Pre-populate the test data file that the functional test leads will be utilizing for a given sprint
* Update test data files when necessary (functionality changes or bugs are found)
* Work with the functional test leads to determine the test cases requiring automation
* Work with the automation team to identify and develop best practices when choosing test cases to automate
* Advise Functional Test leads how to decipher whether a test case is automatable
* Train and support functional testers when necessary
* Keep update on automation tools and apply patches or tools upgrade to the framework.

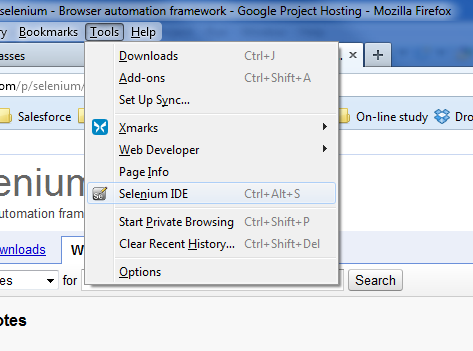
# Functional Tester (FT)

* Execute Test cases.
* Identify the test cases from a given requirement (Story- Sales force) that need to be automated or are candidates for regression
* Communicate to the automation team any functional changes to test cases (that were picked to be automated). This will ensure test cases automated are not obsolete
* Review execution log and also manually run failed test cases to determine if the failure is due to the script or the code
* Subject matter experts on testing Application
* Update and maintain test data files
* Maintain check-in/check-out process on Subversion.
* Update test results in Test management system (Quality Center , CMC for Salesforce)

# Selenium IDE

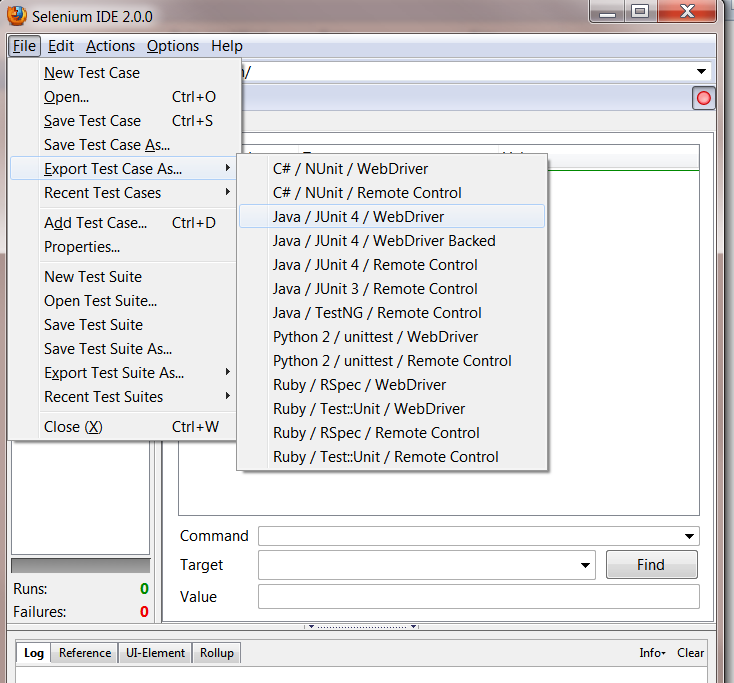
<http://docs.seleniumhq.org/download/>

**Selenium IDE** – A plug-in for Firefox browser, provides recording and playback feature. Download latest version 2.0.0. After installation you should see Selenium IDE option in Firefox.

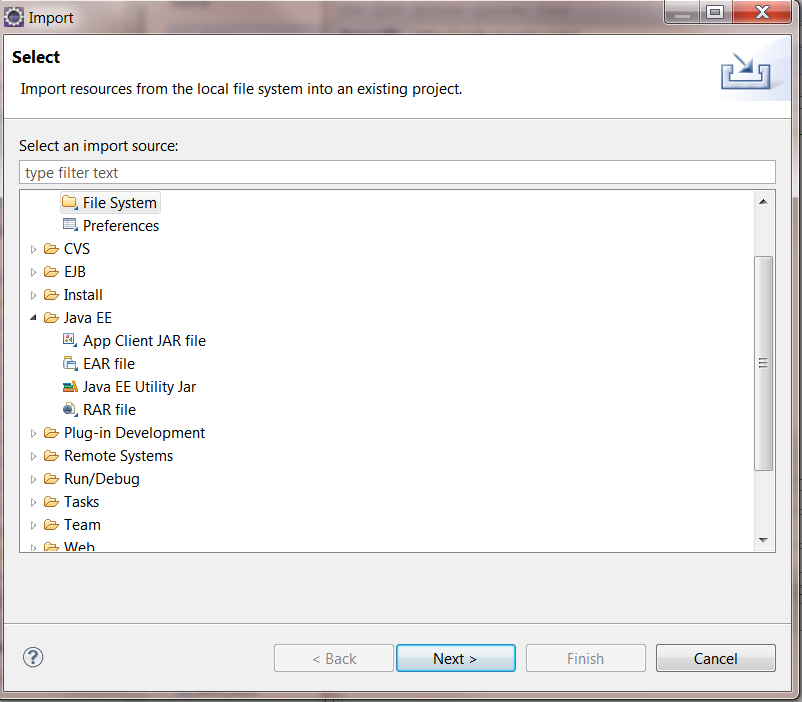


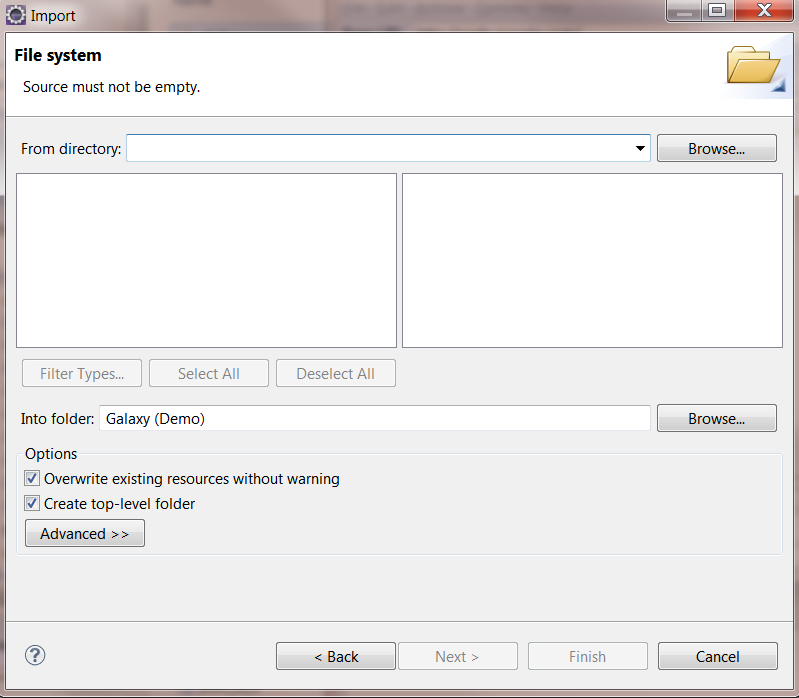
* **Run Selenium IDE Script via eclipse**

1. Export Selenium IDE Script as Java /JUnit 4/ Web Driver



1. Import to eclipse project -> Select File System

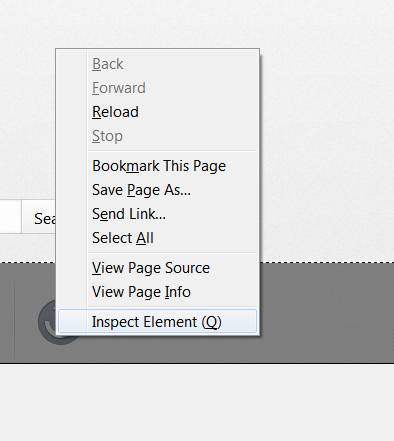




# Firebug

Firebug is add-on to Firefox that will help determine Web Element properties.

* Right Click on any web page/object -> Inspect Element (Q)



* html code display on the bottom of the page



