

Rex Jones II, CSTE, TMap

PART 1
ABSOLUTE BEGINNER

Selenium WebDriver for Functional Automation Testing

Your
Beginners
Guide



Preface

Software QA/Testers are expected to test more development code in less time. The increased production from developers can cause testers to become bottlenecks. However, testing a web application utilizing an automation tool helps speed up the delivery of software products. A tool such as Selenium WebDriver reduces the time it takes to execute a test. In addition, test automation controls the execution of a test, verifies whether actual results return expected results and report results (Pass/Fail). The purpose of this book is to provide building blocks for automating any web application using Selenium WebDriver.

Target Audience

Absolute Beginner

Why You Should Learn Selenium WebDriver?

Selenium WebDriver is a powerful open source test automation tool. Open source refers to free software that is developed for the community. Many test automation tools such as Selenium IDE provide record and playback features. Record and playback allow users to record their actions and replay the actions any number of times. However, Selenium WebDriver does not include a record feature but allow an automation engineer to execute Test Scripts any number of times.

Selenium WebDriver Test Scripts are created and executed within an Integrated Development Environment (IDE). The IDE provide standards, assist with project management, and organize Test Scripts in a structure manner. Employing Selenium WebDriver is a valuable automation tool for assuring quality.

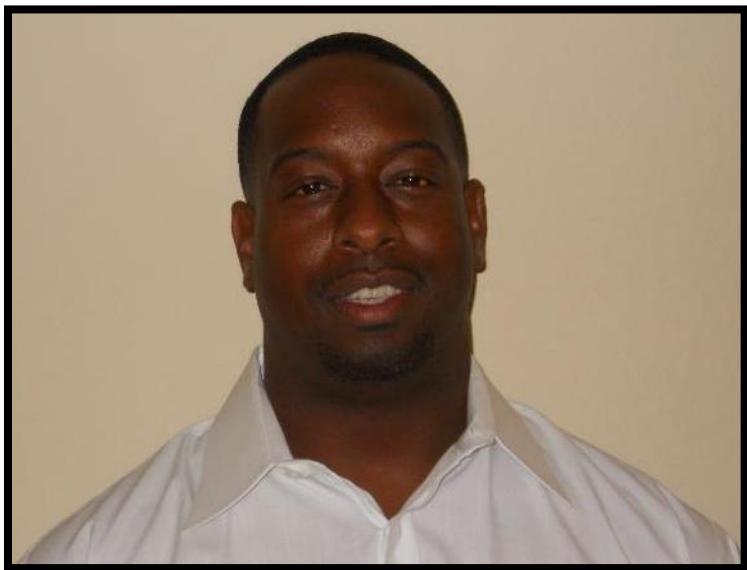
Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

About the Author



Rex Allen Jones II is a QA/Software Tester with a passion for sharing knowledge about testing software. He has been watching webinars, attending seminars, and testing applications since February 2005. Mr. Jones graduated from DeVry University in June 1999 with a Bachelor's of Science degree in Computer Information Systems (CIS).

Currently, Rex is a Sr. Consultant and former Board of Director for User Group: Dallas / Fort Worth Mercury User Group (DFWMUG) and member of User Group: Dallas / Fort Worth Quality Assurance Association (DFWQAA). In addition to his User Group memberships, he is a Certified Software Tester Engineer (CSTE) and has a Test Management Approach (TMap) certification.

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

Mr. Jones' advice for people interested in Functional Automation Testing is to learn the programming language. This advice led him to write 4 programming books “(Part 1 & Part 2) You Must Learn VBScript for QTP/UFT” and “(Part 1 & Part 2) Java 4 Selenium WebDriver”. VBScript is the programming language for Unified Functional Testing (UFT) formerly known as Quick Test Professional (QTP) and Java is one of the programming languages for Selenium WebDriver.

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Rex Jones' Contact Information

Email Address: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Books: <http://tinyurl.com/Rex-Allen-Jones-Books>

Twitter: @RexJonesII

Skype: rex.jones34

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

Free Webinars, Videos, and Live Training

Mr. Jones plans to have **free** step-by-step demonstration webinars, videos, and live trainings walking people through concepts of Selenium and QTP/UFT from A - Z. The material will teach/train individuals the fundamentals of the programming language, fundamentals of Selenium and QTP/UFT, and important concepts of Selenium and QTP/UFT. All of the webinars, videos, and live training will be directed toward beginners as well as mid-level automation engineers.

Sign Up to Receive

1. 3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>
2. 3 Tips To Master QTP/UFT Within 30 Days
<http://tinyurl.com/3-Tips-For-QTP-UFT>
3. Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Acknowledgements

I would like to express my gratitude to my wife Tiffany, children Olivia Rexe' and Rex III, family, friends, and the many people who provided encouragement. Writing this book took time and your support helped pushed this book forward.

Thank You,

A handwritten signature in black ink that reads "Rex Allen Jones II". The signature is fluid and cursive, with "Rex" and "Allen" stacked above "Jones" and "II".

Rex Allen Jones II

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

Copyright, Legal Notice, and Disclaimer

This publication is protected under the US Copyright Act of 1976. All rights are reserved including resale rights which applies to international, federal, state, and local laws. The purchaser is not allowed to share or sell this book to anyone.

Please note that much of this publication is based on personal experience and anecdotal evidence. The author has made every reasonable attempt to produce accurate content in this book. He assumes no responsibility for unknown errors or omissions. Therefore, the purchaser should use this information as he/she sees fit.

Any trademarks, service marks, product names or named features are assumed to be the property of their respective owners and used only for reference.

Copyright © 2016 Test 4 Success, LLC. All rights reserved worldwide.

Table of Contents

PREFACE.....	2
ABOUT THE AUTHOR	3
REX JONES' CONTACT INFORMATION	5
FREE WEBINARS, VIDEOS, AND LIVE TRAINING	6
ACKNOWLEDGEMENTS.....	7
COPYRIGHT, LEGAL NOTICE, AND DISCLAIMER	8
TABLE OF CONTENTS.....	9
 CHAPTER 1 INTRODUCTION TO SELENIUM WEBDRIVER.....	12
HISTORY	12
JAVA / OBJECT-ORIENTED PROGRAMMING	13
UNIT TEST FRAMEWORKS	14
FIRST SELENIUM TEST SCRIPT	14
WEBDRIVER AND WEBELEMENTS	16
INSTALLATIONS	16
 CHAPTER 2 JAVA / OBJECT-ORIENTED PROGRAMMING.....	52
VARIABLES AND DATA TYPES	53
OPERATORS	67
CONTROL STRUCTURES	78
CLASSES, OBJECTS, AND METHODS	101
ACCESS MODIFIERS	105

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

Table of Contents

INHERITANCE.....	105
ENCAPSULATION	108
POLYMORPHISM.....	108
PACKAGES	108
INTERFACES	111
CHAPTER 3 JUNIT VS TESTNG.....	115
WHAT ARE ANNOTATIONS	119
IMPORT ANNOTATIONS	120
CREATE ANNOTATIONS	121
EXECUTE ANNOTATIONS.....	130
CHAPTER 4 FIRST SELENIUM TEST SCRIPT	144
HOW TO ACCESS SELENIUM IDE.....	145
SELENIUM IDE USER INTERFACE.....	147
RECORD SELENIUM'S FIRST TEST SCRIPT	148
SAVE AND PLAYBACK SELENIUM'S FIRST TEST SCRIPT.....	154
EXPORT FROM SELENIUM IDE TO SELENIUM WEBDRIVER	157
LOCATE WEBELEMENTS VIA HTML.....	165
CHAPTER 5 WEBDRIVER AND WEBELEMENTS	173
WEBDRIVER PACKAGES AND CLASSES	174
WEBDRIVER OBJECT AND METHODS	174
FIND AND PERFORM ACTIONS ON WEBELEMENTS	181

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

CHAPTER 6 FIND WEBELEMENT BY ID	189
CHAPTER 7 FIND WEBELEMENT BY NAME	202
CHAPTER 8 FIND WEBELEMENT BY XPATH.....	207
CHAPTER 9 FIND WEBELEMENT BY CSS SELECTOR.....	224
CHAPTER 10 FIND WEBELEMENT BY LINK TEXT.....	237
CHAPTER 11 FIND WEBELEMENT BY PARTIAL LINK TEXT.....	242
CHAPTER 12 FIND WEBELEMENT BY TAG NAME.....	247
CHAPTER 13 FIND WEBELEMENT BY CLASS.....	252
CONCLUSION	261
JAVA	261
TESTNG	262
SELENIUM WEBDRIVER.....	263
RESOURCES.....	265
DOWNLOAD PDF VERSION	268
BOOKS BY REX JONES II	269
SIGN UP TO RECEIVE	271

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

Chapter 1

Introduction to Selenium WebDriver

*** PDF version updated on 12/06/16

History

According to [SeleniumHQ](#), the history of Selenium starts in 2004 at Thoughtworks with Jason Huggins building the core mode as JavaScriptTestRunner. He built the core mode because he did not want to manually step through the same test after every change. Therefore, Jason developed a JavaScript library that permitted him to run test repeatedly against multiple browsers such as Firefox, Google, and Internet Explorer.

Soon after, fellow coworker Paul Hammant saw a demo by Jason and started discussions about making Selenium open source. Open source is free software developed by the community and for the community. The JavaScript library became the core of Selenium that underlines Selenium Remote Control (RC).

Selenium RC is a powerful automation tool that allows an engineer to control the browser. The tool drastically improves productivity by reducing test time and cost. Although powerful, Selenium RC has drawbacks due to its JavaScript based automation engine. The drawback makes certain tasks impossible to carry out.

As a result of the drawbacks, an engineer named Simon Stewart started working on a project in 2006 called WebDriver. The goal of WebDriver was to address Selenium's drawbacks. After WebDriver solved the restrictions of Selenium, both developers decided to merge the two projects. The merge made Selenium WebDriver (also known as Selenium 2) a very robust test automation tool.

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Selenium WebDriver released in 2011 and became the successor to Selenium RC. Selenium WebDriver runs on multiple platforms and supported by multiple programming languages. The platforms are Windows, Macintosh, and Linux while the programming languages are C # pronounced as C Sharp, Java, Python, and Ruby.

Chapter 1 provides an overview of this book and introduce the following:

- ✓ [Java / Object-Oriented Programming](#)
- ✓ [Unit Test Frameworks](#)
- ✓ [First Selenium Test Script](#)
- ✓ [WebDriver and WebElements](#)

Java / Object–Oriented Programming

According to [TIOBE](#), Java is the most popular programming language within the programming community. Therefore, this book will cover Selenium WebDriver from Java's perspective. Java is a language that implements an object-oriented programming (OOP) model (see [Java / Object-Oriented Programming in Chapter 2](#)) that centers on objects.

An object contains data and logic. Variables hold data while methods carry out the logic. A variable is a memory location with a name. Methods instruct the program what action to perform and how to perform the action. In addition, a method provides access to data defined by a class. Classes characterize the structure of an object. Objects, methods, and classes are interconnected which serve as the basis for Object-Oriented Programming (OOP).

Moreover, Object-Oriented Programming (OOP) consists of three concepts, which are Inheritance, Encapsulation, and Polymorphism. Inheritance allows reusable code and objects to be extended. Encapsulation combines code and data for protection. Polymorphism allows multiple methods to utilize one method/interface.

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

Unit Test Frameworks

Unit testing is a development process where developers test the smallest part of an application. Like most processes, unit testing can be time-consuming and tedious. A unit test framework such as JUnit and Test Next Generation (TestNG) facilitates the process (see [JUnit vs TestNG in Chapter 3](#)). The frameworks execute an individual Test Script, execute a collection of Test Scripts (known as Test Suite), verify expected outcomes (Pass or Fail), and report the results.

Several unit test frameworks belong in the xUnit family. The xUnit family is a collection of unit test frameworks with a common architecture. Each framework has a structure for a specific programming language. For example, JUnit structured for Java and NUnit structured for C#. Both frameworks (JUnit and NUnit) are part of the xUnit family. The following is a list of xUnit test frameworks and their programming languages:

- AsUnit structured for ActionScript
- JUnit structured for Java
- NUnit structured for Microsoft.Net programming languages
- PHPUnit structured for Python

However, not all unit test frameworks are part of xUnit. TestNG is not included in the xUnit family. Although not included in the xUnit family, JUnit inspired TestNG. As a result, TestNG adopted some of JUnit's concepts then added more features for testing an application.

First Selenium Test Script

Many beginners with a desire to learn Selenium start with Selenium Integrated Development Environment (IDE). Selenium IDE does not require knowledge of programming but allows test creation. Therefore, Selenium IDE primarily used to create an automation engineer's first

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Test Script (see [First Selenium Test Script in Chapter 4](#)). A test created through its record and replay feature is valuable. The tool interactively records a user's actions and replays those same actions. In addition, the test is replayed any number of times with an ability to compare actual and expected results. Why use Selenium WebDriver if Selenium IDE create and play back Test Scripts? The following screenshot from [Selenium HQ](#) describes which Selenium is appropriate for testing:

Which part of Selenium is appropriate for me?

Selenium WebDriver	Selenium IDE
	
If you want to <ul style="list-style-type: none">• create robust, browser-based regression automation suites and tests• scale and distribute scripts across many environments	If you want to <ul style="list-style-type: none">• create quick bug reproduction scripts• create scripts to aid in automation-aided exploratory testing
Then you want to use Selenium WebDriver ; a collection of language specific bindings to drive a browser -- the way it is meant to be driven.	Then you want to use Selenium IDE ; a Firefox add-on that will do simple record-and-playback of interactions with the browser.

Figure 1.1 – Difference Between Selenium WebDriver and Selenium IDE

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

According to [Selenium HQ](#), the difference between Selenium WebDriver and Selenium IDE:

- Selenium WebDriver
 - create robust, browser-based regression automation suites and tests
 - scale and distribute scripts across many environments
- Selenium IDE
 - create quick bug reproduction scripts
 - create scripts to aid in automation-aided exploratory testing

Note: Selenium IDE executes Test Scripts only in Firefox while Selenium WebDriver executes Test Scripts in all major browsers. The record and replay features in Selenium IDE start the automation experience but Selenium WebDriver completes the automation experience.

WebDriver and WebElements

Most software applications are web based applications designed to run on browsers. WebDriver is an interface tool for testing web applications that contain WebElements (see [WebDriver and WebElements in Chapter 5](#)). WebElements (also known as elements) are buttons, text boxes, checkboxes, drop down menus, and hyperlinks. Identifying an element and performing an action on the element are keys to automating an application. It is important to know that WebDriver identify and perform actions on the elements.

Note: In the field of automation testing, objects or elements describe an entity “i.e., button, text box, etc.”

Installations

Eclipse requires several installations to execute automation Test Scripts. It is best to provide all of the installation steps in one section rather than listing the steps in different parts of the

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

book. Due to updates, website, and system changes, you may encounter different steps. The following is a list of installations to get started with Selenium WebDriver using Java:

- [Verify The System Type](#)
- [Install Browsers](#)
- [Install Selenium IDE](#)
- [Install Firebug and FirePath](#)
- [Install Java Development Kit \(JDK\)](#)
- [Install Eclipse IDE](#)
- [Install TestNG](#)
- [Configure WebDriver](#)
- [Download Browser Drivers](#)
- [Set Up A Firefox Profile](#)
- [Download Notepad ++](#)

Verify The System Type

A verification of the System Type determines the computer's version. Knowledge of the System Type helps when downloading/installing a particular product. All types of errors such as performance issues can occur soon after installing the wrong System Type. The following are steps to verify a computer's System Type:

1. Go to Start menu
2. Select System

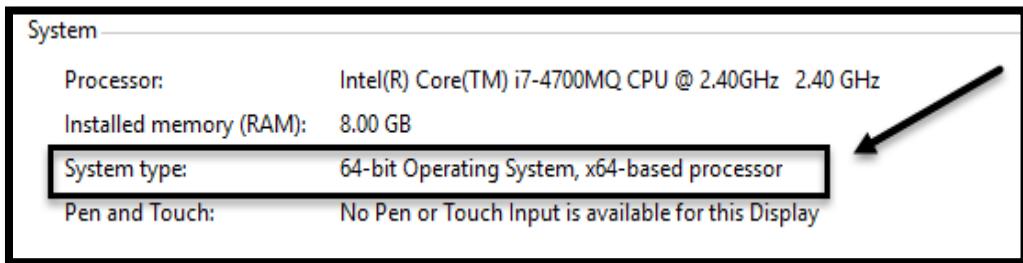
3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

3. System type = “i.e., 64-bit Operating System - Windows 64”



Install Browsers

A browser is a software application used to access the internet. It provides a way to interact with all of the information on the World Wide Web. There are many browser types. However, they contain similar functionalities. Each browser retrieves and presents information via hyperlinks. The most commonly used browsers are Firefox, Google Chrome, and Internet Explorer (IE). An installation of at least one browser is required in order to install the other products. The following are steps to install Firefox, Google Chrome, and Internet Explorer:

- [Steps To Install Firefox](#)
- [Steps To Install Google Chrome](#)
- [Steps To Install Internet Explorer](#)

Steps To Install Firefox:

1. Go to <https://www.mozilla.org/en-US/firefox/new/>
2. Click Free Download
3. Click Save then save to a location
4. Click Run
5. Click Yes
6. Click Next

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

7. Click Install
8. Click Finish

Steps To Install Google Chrome:

1. Go to <https://support.google.com/chrome/answer/95346?hl=en>
2. Click Download
3. Click Accept and Install
4. Click Run
5. Click Yes

Steps To Install Internet Explorer:

1. Go to <https://www.microsoft.com/en-us/download/details.aspx?id=39232>
2. Click Download
3. Click Run
4. Click Continue
5. Click Install

Note: The steps to install each browser are subject to change.

Install Selenium IDE

Selenium IDE is a Firefox extension. The tool records, replays, edits, and debugs a Test Script. Frequently, Selenium IDE is an introduction to automation due to its record and playback features. The tool permits code conversion to a specific language “i.e., Java” and unit test framework “i.e., TestNG”. This technique helps a beginner learn how to read the code before writing the code. The following are steps to install Selenium IDE:

1. Open Firefox browser
2. Go to <http://www.seleniumhq.org/download/>

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

Chapter 1

Introduction to Selenium WebDriver

(Part 1) Selenium WebDriver

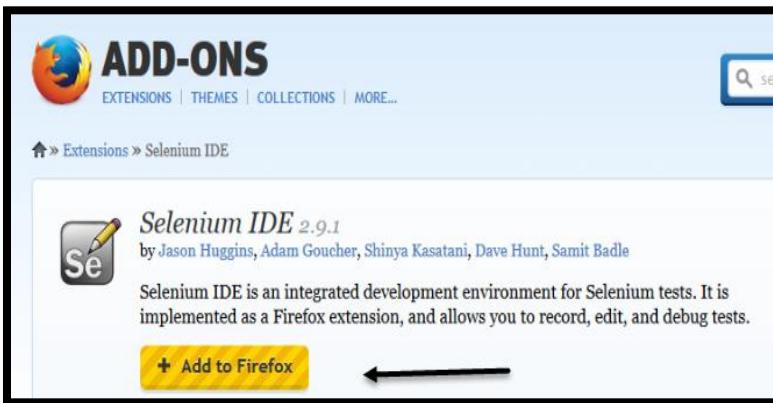
3. Select from addons.mozilla.org from Selenium IDE section

Selenium IDE

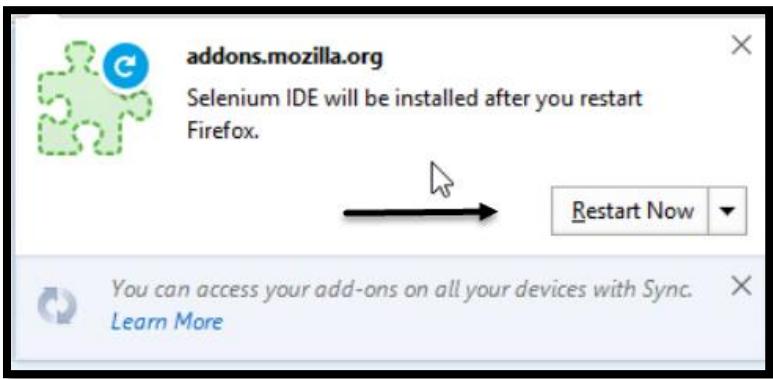
Selenium IDE is a Firefox plugin which records and plays back user interactions with the browser. Use this to either create simple scripts or assist in exploratory testing. It can also export Remote Control or WebDriver scripts, though they tend to be somewhat brittle and should be overhauled into some sort of Page Object-y structure for any kind of resiliency.

Download latest released version [from addons.mozilla.org](https://addons.mozilla.org) or view the [Release Notes](#) and then [install some plugins](#).

4. Click Add to Firefox



5. Click Restart Now



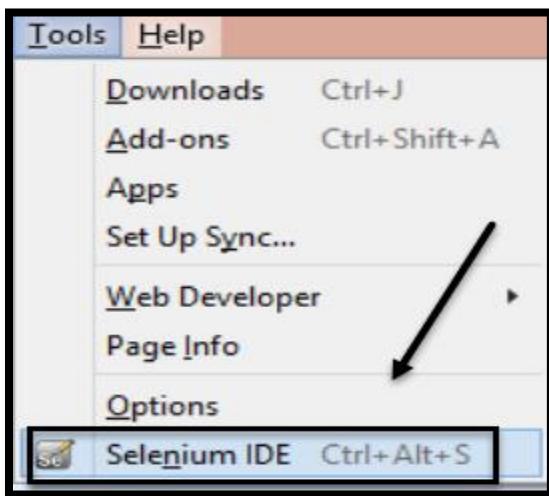
Skype: rex.jones34

Twitter: @RexJonesII

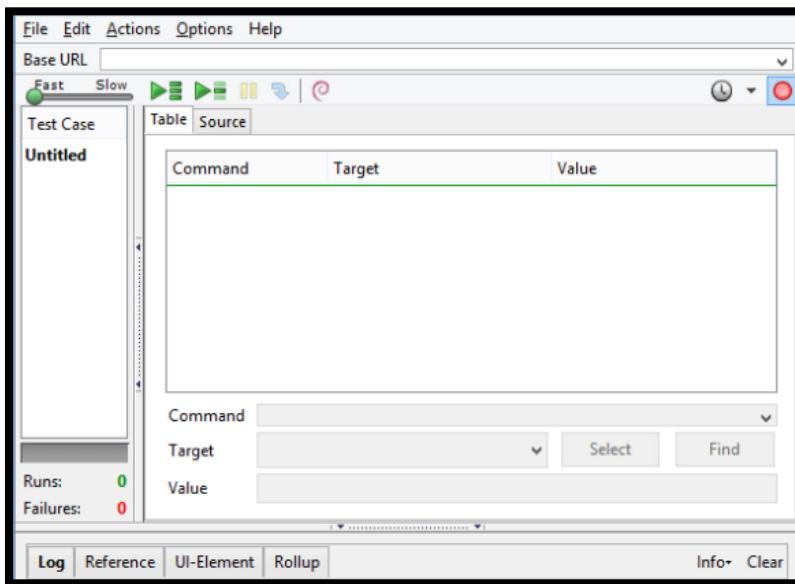
Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

6. Selenium IDE is installed into Firefox: Tools > Selenium IDE



7. Load Selenium IDE



3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

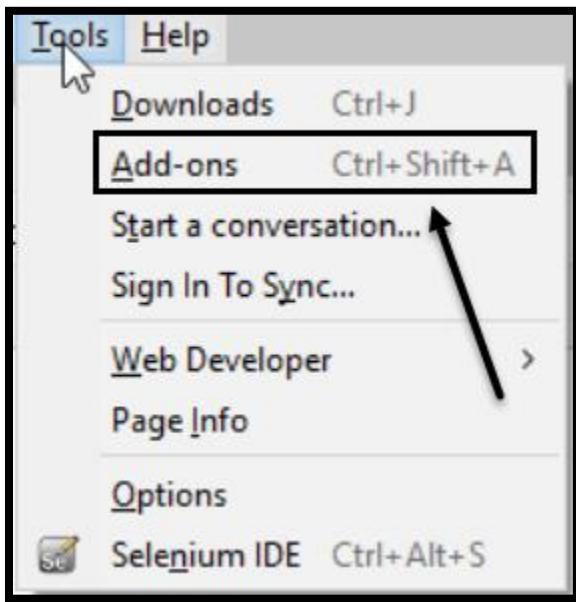
Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

Install Firebug and FirePath

Firebug is an open source Firefox extension that facilitates the process of recognizing WebElements. The Firebug tool allows an element inspection. Element inspection pinpoints a specific element within a web application. Selenium WebDriver provides eight locators to assist with finding an element. Two of the locators (CSS and XPath) must use FirePath to recognize the elements. FirePath is included with Firebug after installation. The following are steps to install Firebug and FirePath:

1. Open Firefox browser
2. Navigate to Tools > Add-ons



3. Select Extensions tab

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

4. Search for Firebug then Click Install

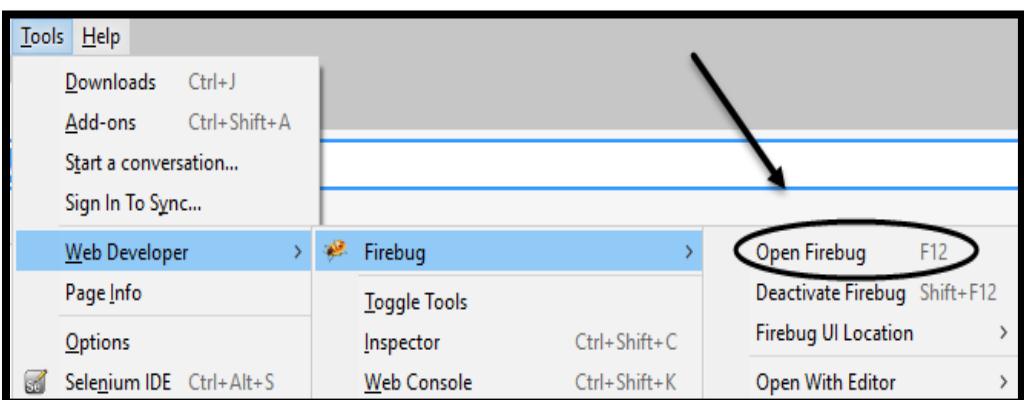


5. Search for FirePath then Click Install

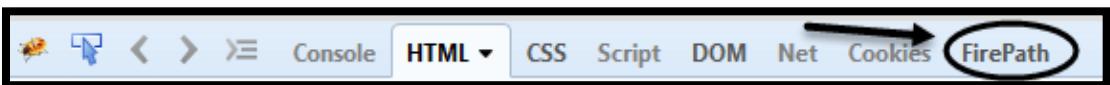


6. Click Restart Now

7. Open Firebug: Navigate to Tools > Web Developer > Firebug > Open Firebug



8. FirePath is loaded within Firebug



3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

Install Java Development Kit (JDK)

The Java Development Kit (JDK) is a software development environment used for writing code in Java. It includes many required components for creating and testing applications. Some of the components are Java Runtime Environment (JRE), Java Compiler, Java Interpreter, and Java Archiver (JAR).

- Java Runtime Environment (JRE) – provides the requirements to execute code in a web browser
- Java Compiler – primary program that reads class definitions then compiles it into bytecode class files
- Java Interpreter – primary program that executes bytecode for Java Virtual Machine
- Java Archiver (JAR) – files used to combine Java class files

The following are steps to install JDK 8:

1. Go to <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

- Click Accept License Agreement in the Java SE Development Kit 8u111 section

Note: Version 8u111 may not be the most recent version

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.78 MB	jdk-8u111-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.73 MB	jdk-8u111-linux-arm64-vfp-hflt.tar.gz
Linux x86	160.35 MB	jdk-8u111-linux-i586.rpm
Linux x86	175.04 MB	jdk-8u111-linux-i586.tar.gz
Linux x64	158.35 MB	jdk-8u111-linux-x64.rpm
Linux x64	173.04 MB	jdk-8u111-linux-x64.tar.gz
Mac OS X	227.39 MB	jdk-8u111-macosx-x64.dmg
Solaris SPARC 64-bit	131.92 MB	jdk-8u111-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	93.02 MB	jdk-8u111-solaris-sparcv9.tar.gz
Solaris x64	140.38 MB	jdk-8u111-solaris-x64.tar.Z
Solaris x64	96.82 MB	jdk-8u111-solaris-x64.tar.gz
Windows x86	189.22 MB	jdk-8u111-windows-i586.exe
Windows x64	194.64 MB	jdk-8u111-windows-x64.exe

- Click the Download link for the appropriate System Type “i.e., Windows x64”
- Go to the Download folder
- Open the downloaded executable file



- Click the Next button to Set Up Java SE Development Kit
- Click the Next button for Custom Set Up
- Click the Next button to Install to a specific location
“i.e., C:\Program Files\Java”

Note: Steps 9 – 20 are optional but beneficial. Eclipse automatically searches for the path “i.e., C:\Program Files\Java\jdk1.8.0_111\bin” that is placed in the Environment Variables

3 Tips To Master Selenium Within 30 Days

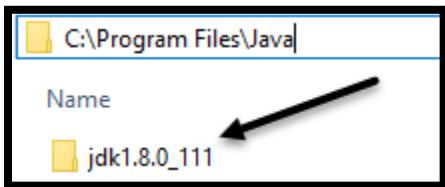
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

modal.

9. Go to the jdk 8 “i.e., jdk1.8.0_111” location and Open the jdk folder

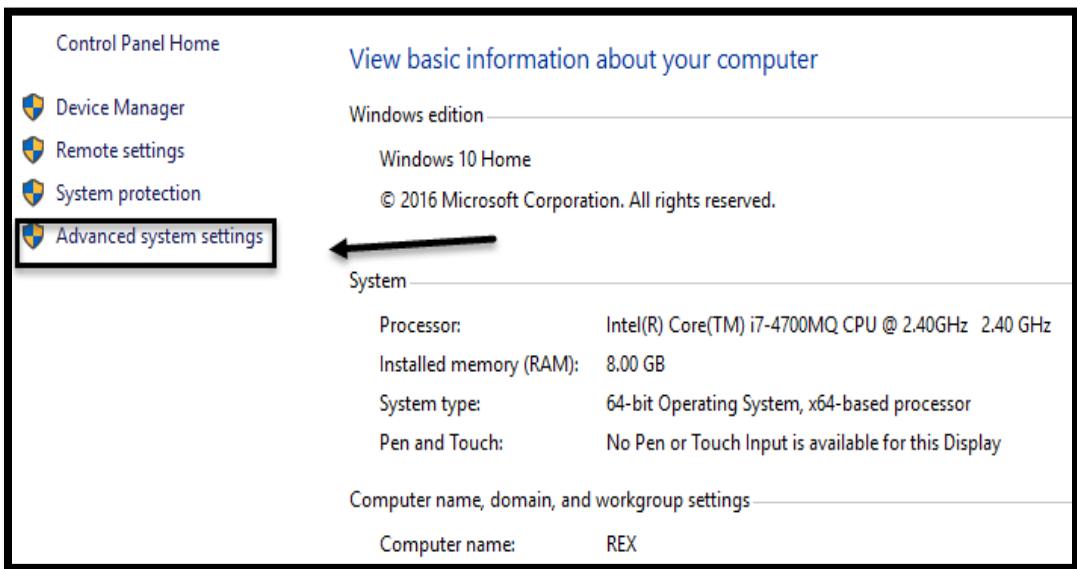


10. Open the bin folder

11. Copy the bin folder's location “i.e., C:\Program Files\Java\jdk1.8.0_111\bin”

Note: Shortcut to Copy = Shift + Right Click > Copy As Path

12. Access the Advanced system settings via System



13. Click the Advanced tab

14. Click Environment Variables

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

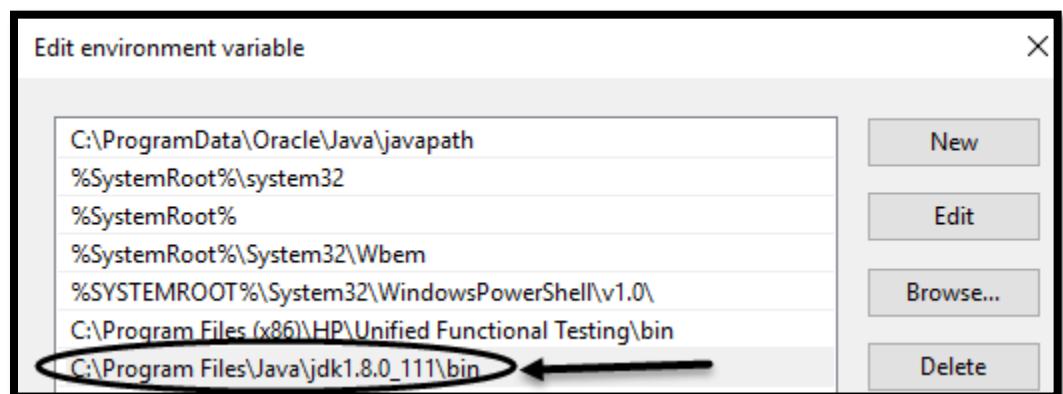
LinkedIn: <https://www.linkedin.com/in/rexjones34>

15. Go to Path within System variables section

System variables	
Variable	Value
NUMBER_OF_PROCESSORS	8
OS	Windows_NT
Path	C:\ProgramData\Oracle\Java\javapath;C:\WINDOWS\system32;C:\... ;.COM; .EXE; .BAT; .CMD; .VBS; .VBE; .JS; .JSE; .WSF; .WSH; .MSC
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 60 Stepping 3, GenuineIntel
PROCESSOR_LEVEL	6

16. Click Edit > New

17. Paste the bin folder's location "i.e., C:\Program Files\Java\jdk1.8.0_111\bin"



18. Click OK

19. Click OK

20. Click OK

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

Install Eclipse IDE

Eclipse is an open source IDE used for developing and testing applications. An IDE is comprehensive whereby it contains many features. The source code editor and debugger are some of the features. A source code editor allows code creation while a debugger examines the created code. Eclipse supports multiple programming languages but mainly used for Java. One of the benefits of Eclipse is the use of plugins. The plugins allow customizations and additional functionalities. The following are steps to install Eclipse IDE:

1. Go to <https://eclipse.org/downloads/>
2. Click the Download button “i.e., DOWNLOAD 64 BIT”



3. Click the Download button or Select another mirror
4. Go to Download folder
Note: Proceed to **Step 6c** if zip file is not available
5. Right click the Eclipse zip file and Extract All files
6. Open the eclipse.exe file to launch Eclipse IDE
 - a. Go the extracted folder

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

- b. Open eclipse folder
 - c. Right click eclipse.exe and Select Open
7. Select an Eclipse IDE “i.e., Eclipse IDE for Java EE Developers”



8. Click Install > Accept Now
9. Click the Remember Accepted Licenses Checkbox
10. Click the Accept button
11. Click the Launch

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

12. Load Eclipse IDE



Install TestNG

TestNG is a powerful testing framework structured for Java. The framework is designed to cover all types of testing such as unit, regression, functional, end-to-end, and integration.

Eclipse requires a TestNG installation to utilize TestNG features. The following are steps to install TestNG:

1. Open Eclipse
2. Load a Project “i.e., Hello World”

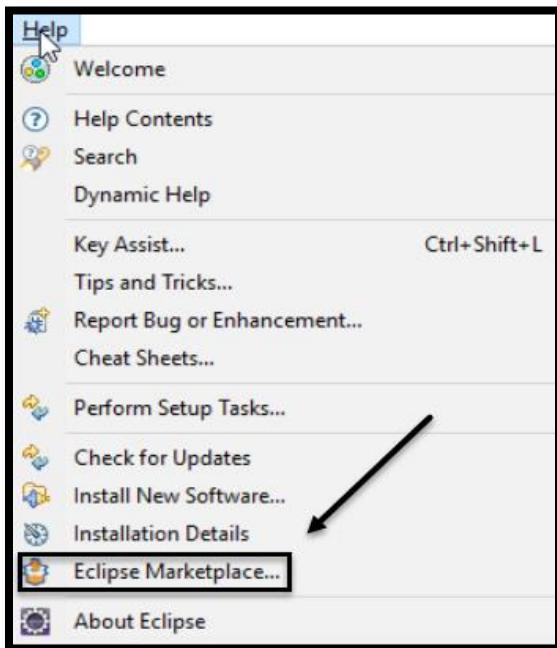
Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

3. Click Help > Eclipse Marketplace



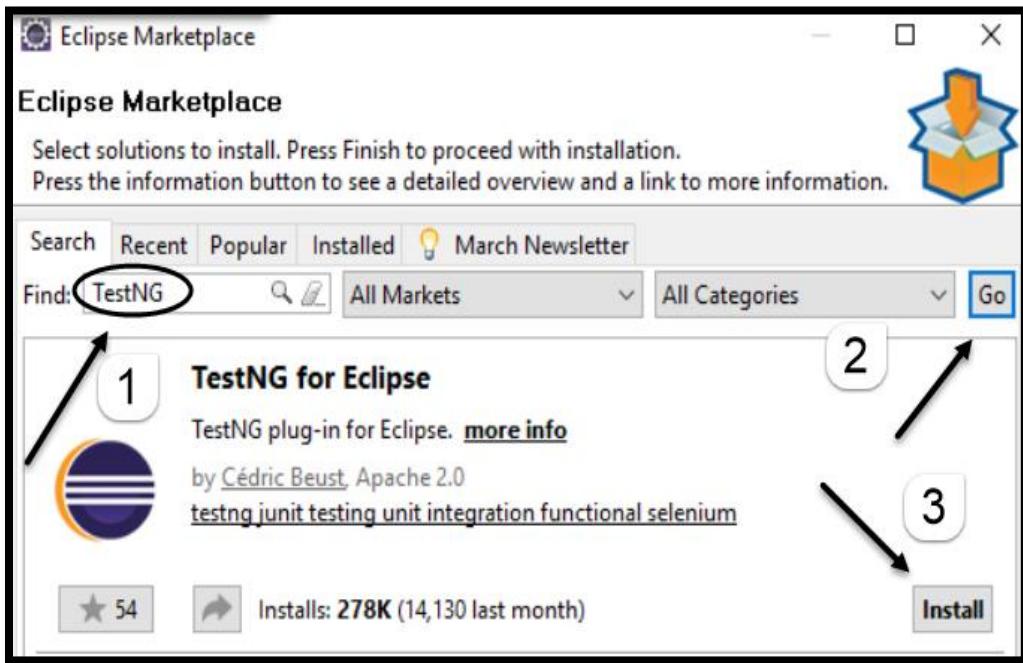
3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

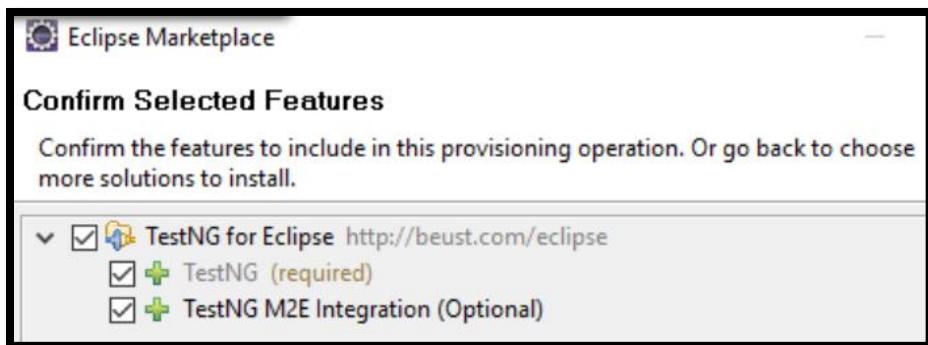
<http://tinyurl.com/Free-QTP-UFT-Selenium>

4. Search for TestNG, Click the Go button, then Click the Install button



5. Click the Confirm button to confirm the following features:

- a. TestNG (required)
- b. TestNG M2E Integration (Optional)



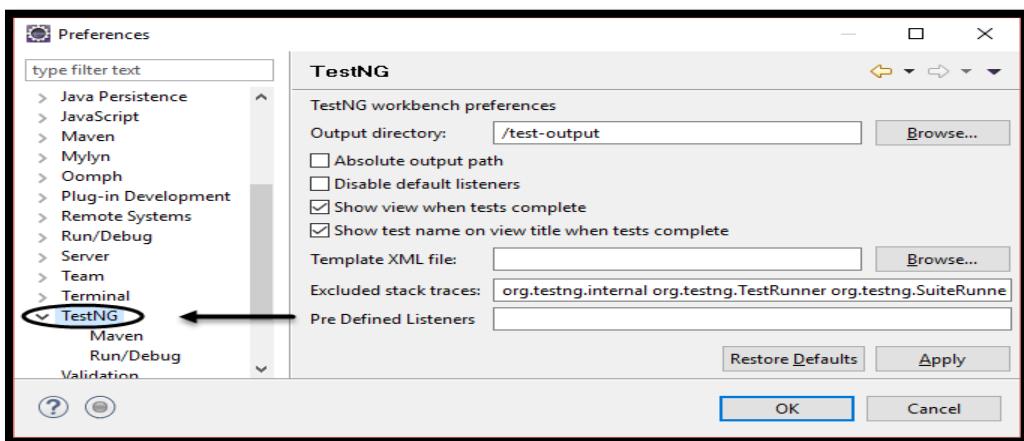
Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

6. Click the radio button “I accept the terms of the license agreement”
7. Click the Finish button
8. Click OK if a Security Warning appears in a modal
9. Click Yes to restart Eclipse
10. Verify TestNG has been added
 - a. Re-open Eclipse after restart
 - b. Click Window
 - c. Click Preferences



Configure WebDriver

As mentioned earlier, Selenium WebDriver supports multiple programming languages. The programming languages are C#, Java, Python, and Ruby. Download the client driver in order to execute Test Scripts for a particular language. In this case, WebDriver requires a client driver specific to Java. After downloading, configure the client driver in Eclipse by adding JAR files. A JAR file is a collection of multiple Java class files used to distribute Java libraries. [Selenium HQ](#) allows a download of the most recent client version for all supported languages. The following are steps to configure a Java client driver:

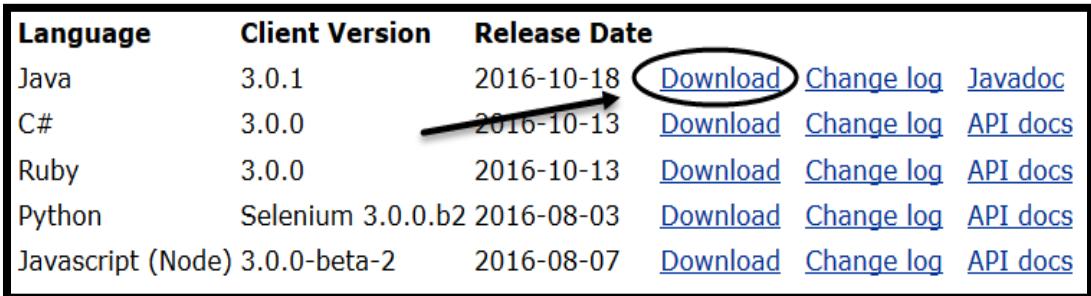
3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

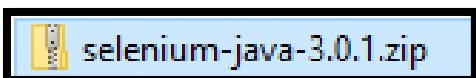
<http://tinyurl.com/Free-QTP-UFT-Selenium>

1. Go to <http://docs.seleniumhq.org/download/>
2. Download client driver for Java



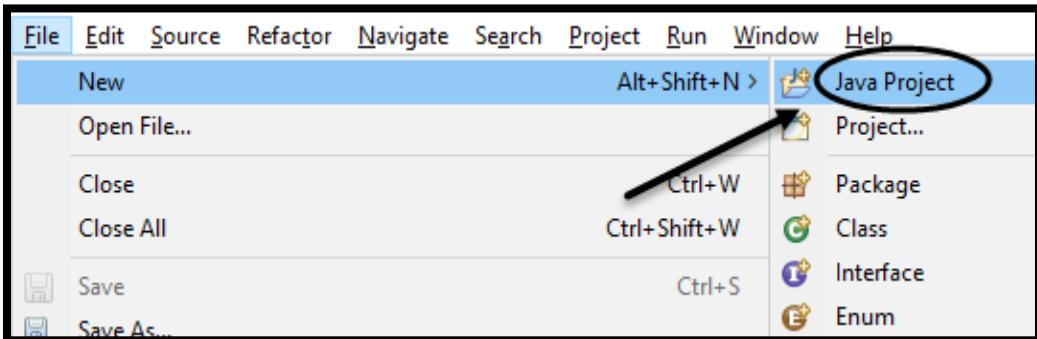
Language	Client Version	Release Date	Download	Change log	Javadoc
Java	3.0.1	2016-10-18	Download	Change log	Javadoc
C#	3.0.0	2016-10-13	Download	Change log	API docs
Ruby	3.0.0	2016-10-13	Download	Change log	API docs
Python	Selenium 3.0.0.b2	2016-08-03	Download	Change log	API docs
Javascript (Node)	3.0.0-beta-2	2016-08-07	Download	Change log	API docs

3. Go to Download folder
4. Right click the zip file “i.e., selenium-java-3.0.1.zip” and Extract All “jar files”



5. Open Eclipse IDE
6. Create New Project:

Navigate to File > New > Java Project > Project Name “i.e., Hello World” > Finish



7. Click File > Properties

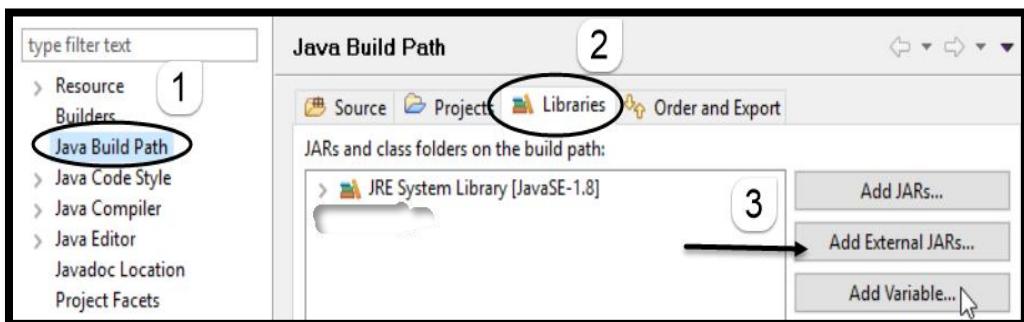
Skype: rex.jones34

Twitter: @RexJonesII

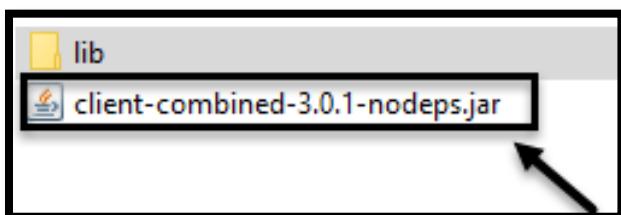
Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

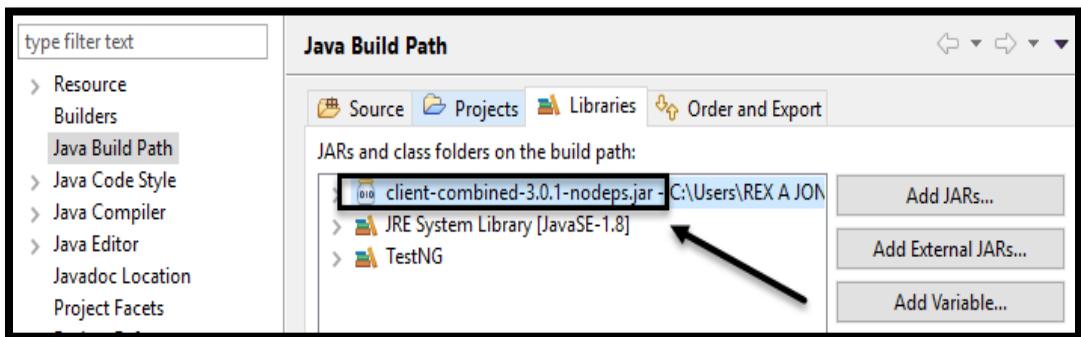
8. Select Java Build Path > Libraries > Add External JARs



9. Go to the Download folder and Select the extracted jar file "client-combined-3.0.1-nodeps.jar"



10. Click Open > jar file "client-combined-3.0.1-nodeps.jar"



11. Click Add External JARs

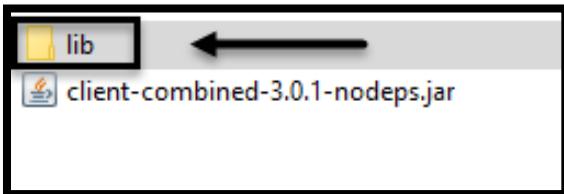
3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

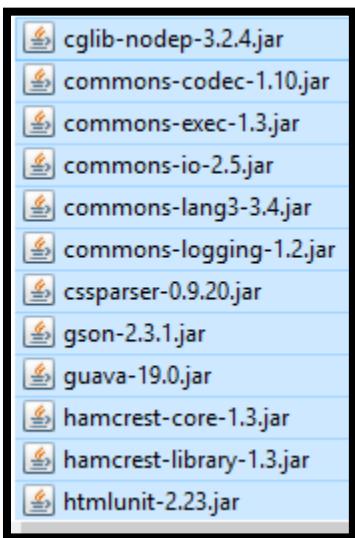
<http://tinyurl.com/Free-QTP-UFT-Selenium>

12. Select libs in the extracted folder

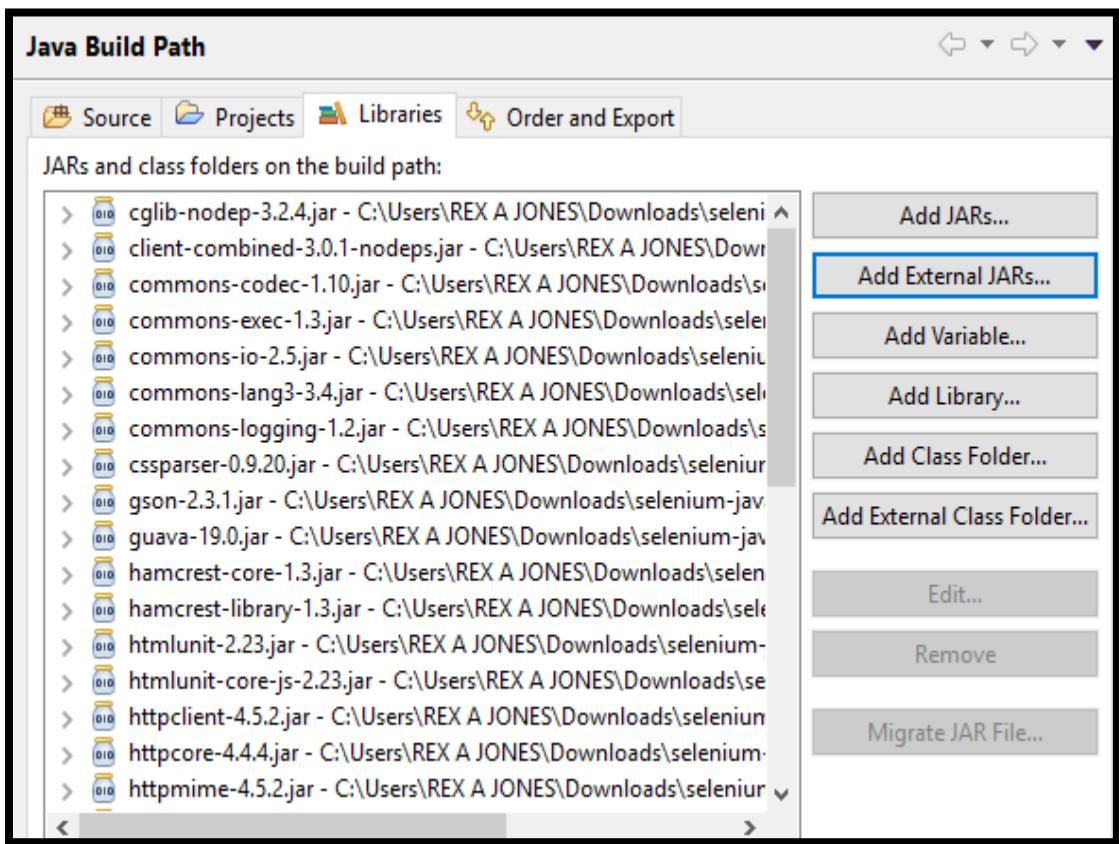


13. Select all of the jar files in libs folder

Note: Shortcut = Select 1 jar file > Ctrl + A



14. All of the jar files are added to the Java Build Path Library > Click OK



Download Browser Drivers

According to [Selenium HQ](#), “in order to create scripts that interact with the Selenium Server (Selenium RC, Selenium Remote WebDriver) or create local Selenium WebDriver scripts, you need to make use of language-specific client drivers”. The core languages are C#, Java, JavaScript, Python, and Ruby. Most of the drivers execute on Google Chrome, Internet Explorer, and Firefox. A profile is set up in Firefox while Google Chrome and Internet

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

Explorer require client drivers. The following provides steps to download drivers and set up a profile on the most used browsers:

- [Google Chrome](#)
- [Internet Explorer](#)
- [Firefox](#)

[Google Chrome](#)

Google Chrome requires a driver called ChromeDriver to run Selenium WebDriver Test Scripts. WebDriver uses ChromeDriver to control testing performed on Google Chrome. An error occurs in Eclipse if there is not a driver download. However, the error provides the following link to download the latest ChromeDriver:

<http://chromedriver.storage.googleapis.com/index.html>

The following are steps to download ChromeDriver from [Selenium HQ](#):

1. Go to <http://docs.seleniumhq.org/download/>
2. Click the link for Google Chrome Driver in Third Party Browser Drivers section

Third Party Browser Drivers NOT DEVELOPED by seleniumhq

Browser	
Mozilla GeckoDriver	
Google Chrome Driver	2.25
Opera	0.2.2
GhostDriver	(PhantomJS)
Microsoft Edge Driver	

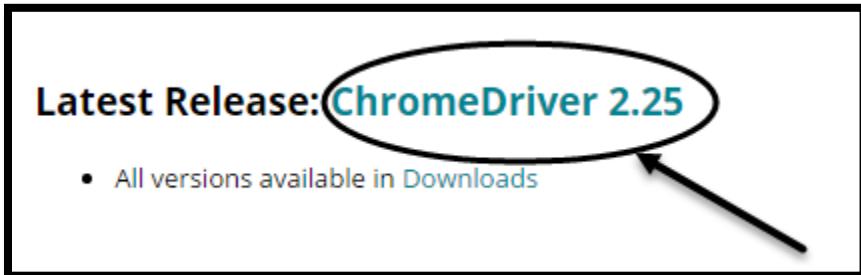
Skype: rex.jones34

Twitter: @RexJonesII

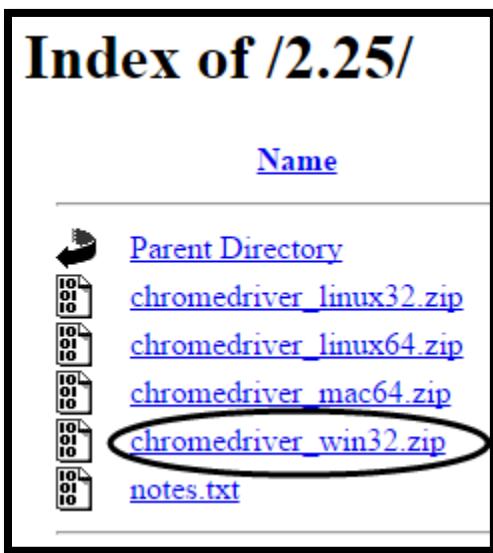
Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

3. Click the link for the most recent release “i.e., 2.25” via
<https://sites.google.com/a/chromium.org/chromedriver/>



4. Click the link for the most recent release again “i.e., 2.25” via
<https://sites.google.com/a/chromium.org/chromedriver/downloads>
5. Download the system’s zip file “i.e., chromedriver_win32.zip”



3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

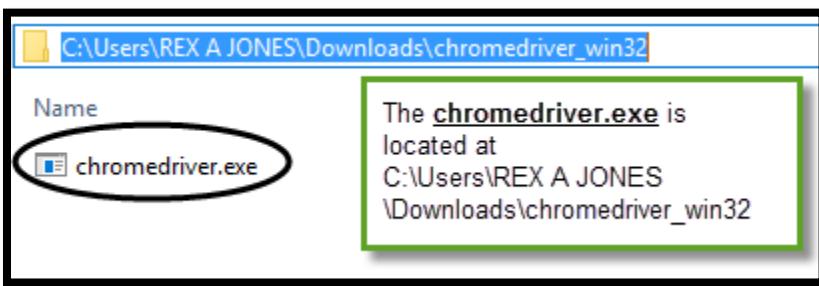
Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

Note: Zip file for Windows 64 is not available. Therefore, the zip file for Windows 32 “chromedriver_win32.zip” should be downloaded

6. Go to Download folder
7. Right click the zip file “i.e., chromedriver_win32.zip” and Extract All



8. Open the extracted folder “i.e., chromedriver_win32”
9. chromedriver.exe has been downloaded



Note: The chromedriver.exe is located at
C:\Users\REX A JONES\Downloads\chromedriver_win32.

There are two ways to ensure Test Scripts run successfully:

- Add path to Environment Variables – System Variables
- Add path to Selenium WebDriver Test Script. The following is an example of using the chromedriver.exe in a Test Script:

```
C:\\\\Users\\\\REX A JONES\\\\Downloads\\\\chromedriver_win32\\\\chromedriver.exe");
```

```
System.setProperty("webdriver.chrome.driver", C:\\\\Users\\\\REX A JONES\\\\Downloads\\\\chromedriver_win32\\\\chromedriver.exe");
```

Skype: rex.jones34

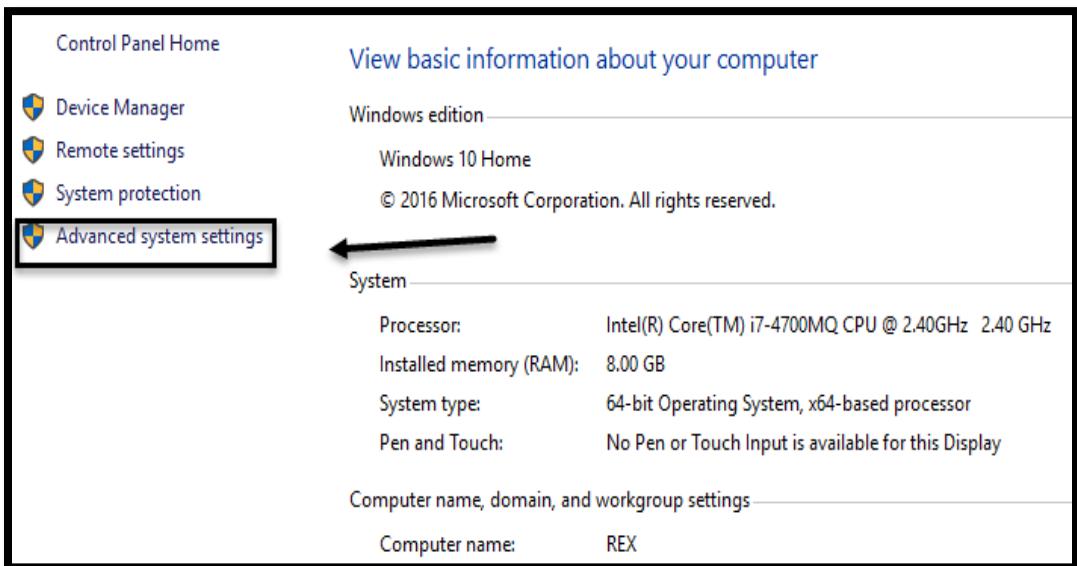
Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

The following illustrates how to add the executable file “i.e., chromedriver.exe” path to Environment Variables - System Variables:

10. Copy the path “i.e., C:\Users\REX A JONES\Downloads\chromedriver_win32”
11. Access the Advanced system settings via [System](#)



12. Click the Advanced tab
13. Click Environment Variables

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

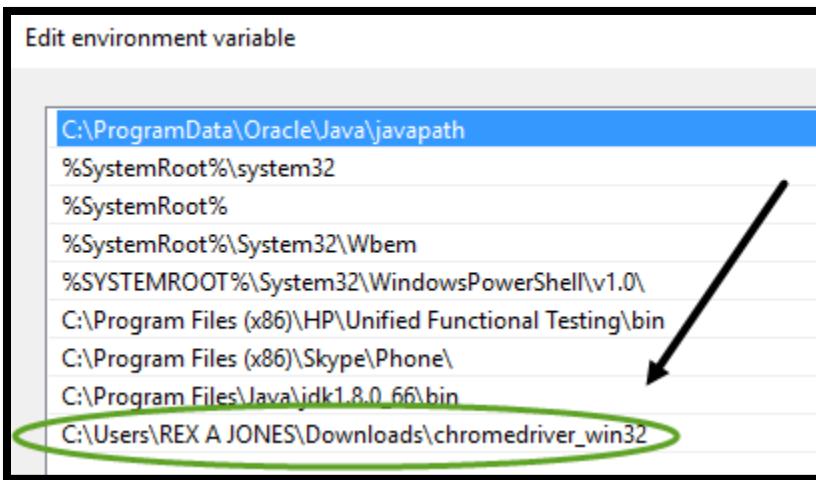
14. Go to Path within System variables section

System variables	
Variable	Value
NUMBER_OF_PROCESSORS	8
OS	Windows_NT
Path	C:\ProgramData\Oracle\Java\javapath;C:\WINDOWS\system32;C:\...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 60 Stepping 3, GenuineIntel
PROCESSOR_LEVEL	6

15. Click Edit

16. Paste the ChromeDriver's executable file location

"i.e., C:\Users\REX A JONES\Downloads\chromedriver_win32"



17. Click OK

18. Click Apply

19. Click OK

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Note: Most examples in this book include an extra code line via System.setProperty when executing Test Scripts in Google Chrome. However, the preferred way is to add the path to Environment Variables – System Variables.

Internet Explorer

Internet Explorer requires a driver called IEDriverServer to run Selenium WebDriver Test Scripts. WebDriver uses IEDriverServer to control testing performed on Internet Explorer. An error occurs in Eclipse if there is not a driver download. [Selenium HQ](#) contains a section called Internet Explorer Driver Server that allows an IEDriverServer download. Follow the same steps (6 - 19) from [Google Chrome's](#) section after downloading IEDriverServer:

The Internet Explorer Driver Server

This is required if you want to make use of the latest and greatest features of the WebDriver InternetExplorerDriver. Please make sure that this is available on your \$PATH (or %PATH% on Windows) in order for the IE Driver to work as expected.

Download version 2.53.1 for (recommended) [32 bit Windows IE](#) or [64 bit Windows IE](#)
[CHANGELOG](#)

Figure 1.2 – Internet Explorer Driver Server

Eclipse provides a link to download IEDriverServer if an error occurs. The following are steps to download IEDriverServer directly from that link:

1. Go to <http://selenium-release.storage.googleapis.com/index.html?path=2.53/>

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

2. Download the most recent Internet Explorer driver “i.e., 2.53”

Index of /2.53/

<u>Name</u>
Parent Directory
IEDriverServer_Win32_2.53.0.zip
IEDriverServer_Win32_2.53.1.zip
IEDriverServer_x64_2.53.0.zip
IEDriverServer_x64_2.53.1.zip
selenium-dotnet-2.53.0.zip
selenium-dotnet-strongnamed-2.53.0.zip
selenium-java-2.53.0.zip
selenium-server-2.53.0.zip
selenium-server-standalone-2.53.0.jar

3. Follow the same steps (6 - 19) from [Google Chrome's](#) section to execute Test Scripts for Internet Explorer browser.

Edit environment variable

```
C:\ProgramData\Oracle\Java\javapath
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
C:\Program Files (x86)\HP\Unified Functional Testing\bin
C:\Program Files (x86)\Skype\Phone\
C:\Program Files\Java\jdk1.8.0_66\bin
C:\Users\REX A JONES\Downloads\chromedriver_win32
C:\Users\REX A JONES\Downloads\IEDriverServer
```

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Note: The security and Firewall exception must be set up / added before running Test Scripts in Internet Explorer.

4. Go to Tools in Internet Explorer
5. Select Internet Options
6. Select Security
7. Confirm the Security Level for all zones (Internet, Local Intranet, Trusted Sites and Restricted Access) are the same
 - a. Uncheck or Check the checkbox for Enable Protected Mode
 - b. Click Apply
 - c. Click OK
 - d. Click OK
8. Add Firewall exception by double clicking the IEDriverServer.exe file in its path “i.e., C:\Users\REX A JONES\Downloads\IEDriverServer”
9. Click Allow Access

Firefox

Firefox does not require a separate driver server like [Google Chrome](#) and [Internet Explorer](#) if using Selenium 2.0. There is an in-built Firefox server supported by Selenium 2.0. However, Firefox requires a separate driver called GeckoDriver to run Test Scripts in Selenium 3.0.

WebDriver uses GeckoDriver to control testing performed on Firefox. An error occurs in Eclipse if there is not a driver download. The driver can be downloaded directly from [GitHub](#). In addition, users are directed to [GitHub](#) from [Selenium HQ](#) after clicking a link called Mozilla GeckoDriver. Follow the same steps (6 - 19) from [Google Chrome's](#) section after downloading GeckoDriver:

1. Go to <http://docs.seleniumhq.org/download/>
2. Click the link for Mozilla GeckoDriver in Third Party Browser Drivers section

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

Third Party Browser Drivers **NOT DEVELOPED** by seleniumhq

Browser		
Mozilla GeckoDriver		
Google Chrome Driver	2.25	change log
Opera	0.2.2	
GhostDriver	(PhantomJS)	
Microsoft Edge Driver		

Figure 1.3 – Mozilla GeckoDriver

3. Download the GeckoDriver “i.e., win64”

Downloads

geckodriver-v0.10.0-arm7hf.tar.gz	1.64 MB
geckodriver-v0.10.0-linux64.tar.gz	1.32 MB
geckodriver-v0.10.0-macos.tar.gz	1.19 MB
geckodriver-v0.10.0-win64.zip	2.28 MB
Source code (zip)	
Source code (tar.gz)	

Figure 1.4 – Download GeckoDriver From GitHub

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Set Up A Firefox Profile

A Firefox profile is a collection of user preferences, bookmarks, browser settings, extensions, passwords, and history. The Profile Manager permits a user to create, remove, and switch profiles. It is optional but beneficial to create a personal Firefox profile. One of the benefits involves troubleshooting a specific problem with Firefox. The profile is saved to a separate location from the other Firefox program files. The following are steps to set up a Firefox profile:

1. Exit Firefox browser (make sure all windows are closed) File > Exit
2. Go to Start menu then Select Run
3. Enter one of the following in the Run dialog
 - a. firefox.exe -p
 - b. firefox.exe -P
 - c. firefox.exe -ProfileManager

Note: Sometimes the Profile Manager does not load after entering the previous information (3a, 3b, 3c). If so, enter the following for a 32-bit Windows system or 64-bit Windows system:

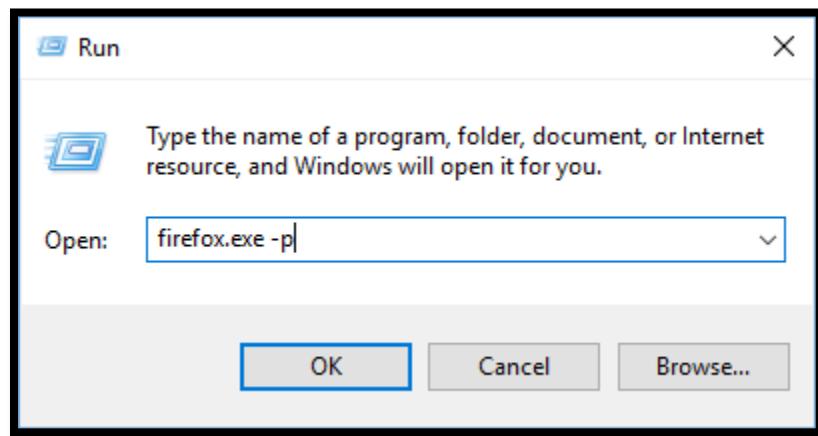
- d. 32 bit Windows = “C:\Program Files (x86)\Mozilla Firefox\firefox.exe” -P
- e. 64 bit Windows = “C:\Program Files\Mozilla Firefox\firefox.exe” -P

3 Tips To Master Selenium Within 30 Days

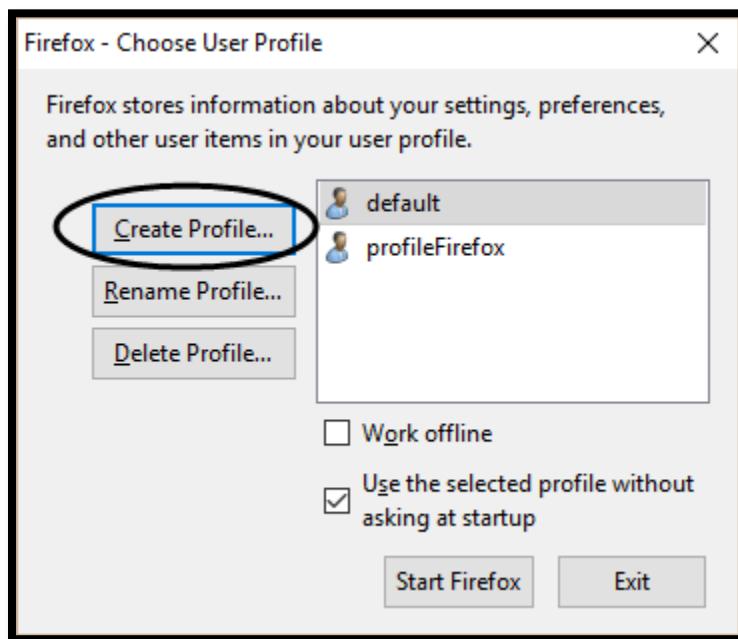
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>



4. Click Create Profile



5. Click Next on the Create Profile Wizard modal

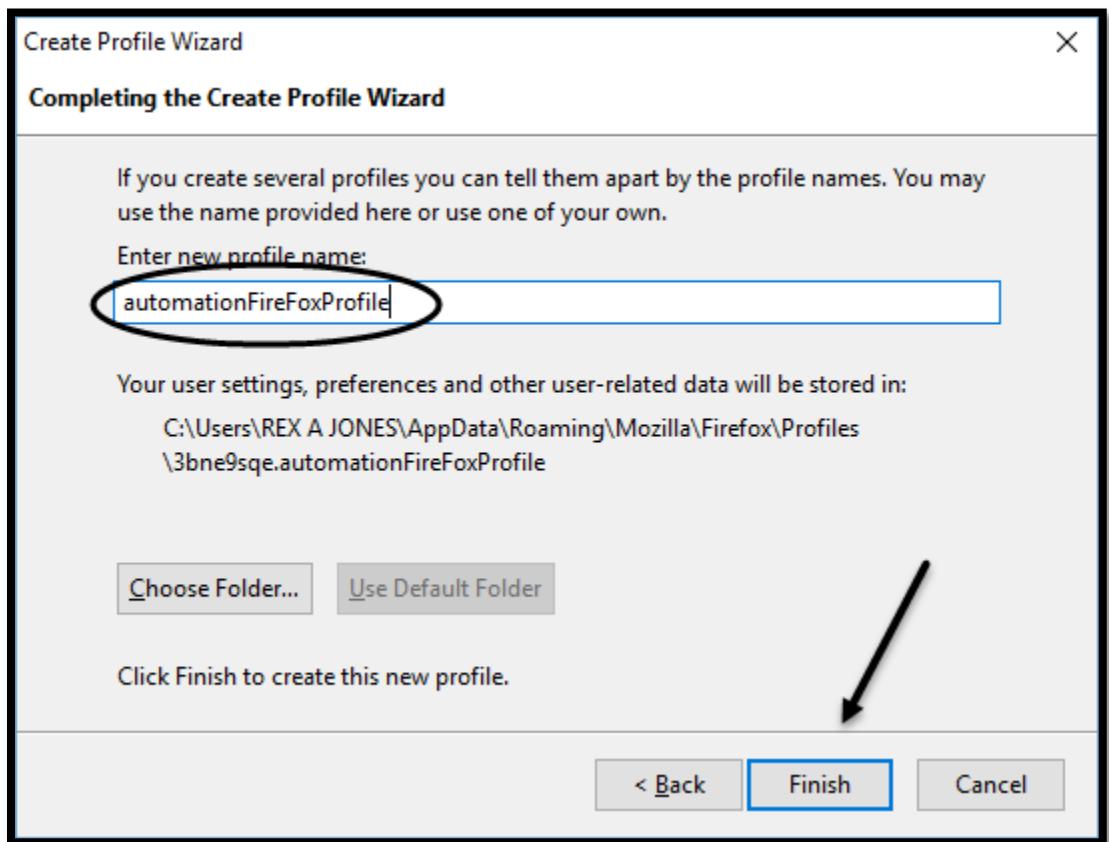
Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

6. Enter Profile Name “i.e., automationFireFoxProfile” > Click Finish



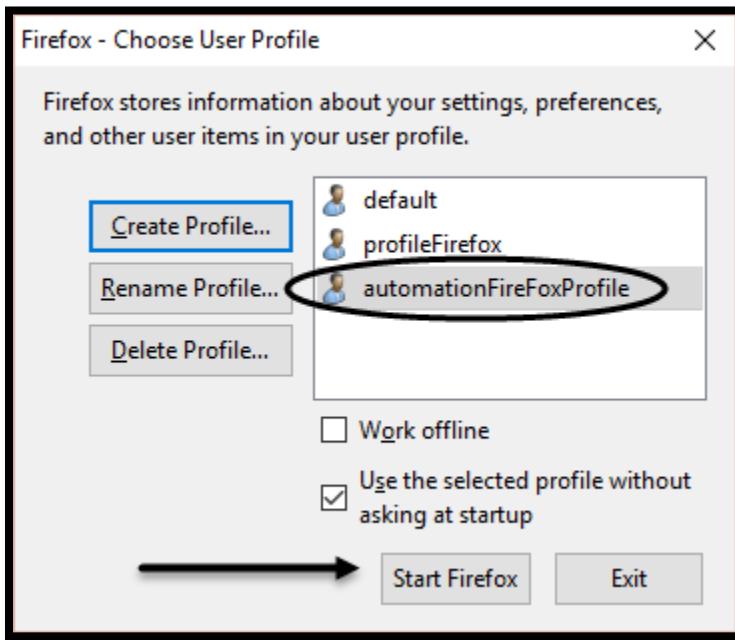
3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

7. Click Start Firefox



Note: Only the tabs will load in the Firefox window. The menu bar (File, Edit, View, etc.) will not be available. Switch profiles back to default in Profile Manager then click Start Firefox to view the menu bar.

Download Notepad ++

Notepad ++ is a free source editor that supports multiple languages. Java uses Notepad ++ and other source editors to view files that end with a .java extension. The following are steps to download Notepad ++:

1. Go to <https://notepad-plus-plus.org/download/v6.9.2.html>

Skype: rex.jones34

Twitter: @RexJonesII

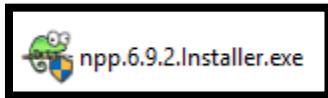
Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

2. Click the Download button



3. Go to Download folder
4. Right click npp.6.9.2.Installer.exe then Select Open



Chapter 1 provided an overview of the upcoming chapters. The subsequent chapters explore Java programming, Unit Test Frameworks (JUnit and TestNG), Selenium IDE, WebDriver, and how to find WebElements. Chapter 2 will explain Java, which is an object-oriented programming language.

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

Chapter 2

Java / Object-Oriented Programming

*** Some of the information in this chapter is a recap of books “[\(Part 1 & 2\) Java 4 Selenium WebDriver](#)”. Both books provide additional concepts regarding Java.

The programming languages offered in Selenium WebDriver are Java, C# (pronounced as C Sharp), Python, and Ruby. An understanding of the programming language sets the foundation for automation. That is the main reason for explaining Java at the beginning of this book. According to [TIOBE](#), Java is the most popular programming language within the programming community. It is an object-oriented programming (OOP) language developed by Sun Microsystems. OOP is an approach to programming that revolves around objects. In the real world, an object can be a person, place, or thing. Similarly, in programming, an object is anything that is visible or tangible.

The purpose of this chapter is provide information regarding [variables](#), [data types](#), [operators](#), [control structures](#), [classes](#), [objects](#), [methods](#), [access modifiers](#), [inheritance](#), [encapsulation](#), [polymorphism](#), [packages](#), and [interfaces](#).

A variable contains values that can change depending on information passed throughout the program. Inside a Java program, all variables must include a data type, which dictates the type of value. Operators assign and process values. An operator is a symbol that performs mathematical or logical manipulations on operands. Control structures are features within programming that includes operators. There are two types of control structures: branches and loops. The branch control structures are if branch and switch branch. The loop control structures are for loop, while loop, and do while loop.

A class is a template for creating objects whereby each class includes data and code that operates on the data. Objects are the foundation of Object – Oriented Programming (OOP).

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

The objective of methods is to provide access to data defined by the class. Access modifiers determine access to a class, methods, and variables. Inheritance allows reusable code, which facilitates maintenance. A package is a collection of related classes while an interface contains a collection of related methods.

Chapter 2 will cover the following regarding Core Java:

- ✓ [Variables and Data Types](#)
- ✓ [Operators](#)
- ✓ [Control Structures](#)
- ✓ [Classes, Objects, and Methods](#)
- ✓ [Access Modifiers](#)
- ✓ [Inheritance](#)
- ✓ [Encapsulation](#)
- ✓ [Polymorphism](#)
- ✓ [Packages](#)
- ✓ [Interfaces](#)

Variables and Data Types

A variable is a named container or memory location that holds a value. The value of the container or memory location can possibly change during program execution. Each variable has the ability to contain any kind of information, such as text or numbers. As a result, automation engineers are empowered to create flexible programs. Variables represent changeable data rather than hard-coded data. Hard-coded data is unchangeable data entered directly into a program.

All variables possess a name, data type, and value. A variable name uniquely identifies the variable. Data type refers to the type of variable, such as [int](#), [double](#), or [boolean](#) stored into a variable. Therefore, data type determines a variable's value. In Java, there are two kinds of

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

data types: primitive and reference. Primitive data type supports [eight basic data types](#) while reference data type supported by a class.

Data Types

The primitive data types give an account for the type of data that is stored in a variable. Each data type has a precise range and behavior. Consequently, a data type of [int](#) can store numerical data, but a type mismatch error will occur if [boolean](#) attempts to store numerical data. Furthermore, certain operations allow values according to the data type. As an example, a math calculation cannot be performed on a [boolean](#) data type because [boolean](#) cannot contain numbers. The following is a list of all eight primitive data types:

Type	Width in Bits (Bytes)	Description/Range
boolean		True or False values
byte	8-bit (1-byte)	-128 to 127
char	16-bit	Standard character set that can be a letter, control character, number, punctuation, or symbol representing all languages in the world
double	64-bit (8-byte)	-1.7976931348623157E+308 to 1.7976931348623157E+308
float	32-bit (4-byte)	-3.4028235E+38 to 3.4028235E+38
int	32-bit (4-byte)	-2,147,483,648 to 2,147,483,647
long	64-bit (8-byte)	-9,223,372,036,854,775,808 to 9,223,372,036,854,755,807
short	16-bit (2-byte)	-32,768 to 32,767

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Figure 2.1 – Eight Primitive Data Types

Note: See Variables and Data Types in Chapter 2 of “[Absolute Beginner \(Part 1\) Java 4 Selenium WebDriver](#)” for additional examples, figures, etc. The eBook and PDF formats are free.

Variable Names

The name of a variable is significant when identifying the variable in memory. All variable names range anywhere from one character to an unlimited number of characters.

Nevertheless, Java has rules to naming variables. One of the rules is to ensure each variable has a unique name. Unique names prevent errors from occurring, such as “Duplicate local variable”— meaning the same variable name has been entered more than one time. The following is a list of rules for naming a variable:

- Can contain case sensitive letters, numbers, dollar sign “\$,” and underscore “_”
- Can begin with a letter, dollar sign “\$,” or underscore “_”
- Cannot begin with a number
- Cannot contain a space or special character except dollar sign “\$,” and underscore “_”
- Cannot contain a reserve keyword. A list of Java’s keywords can be found on [Oracle’s website](#)

The following is a list of suggested conventions for naming a variable:

- Construct descriptive names that describe the variable’s purpose
- Compose names utilizing mixed case letters, unless the name is one word
 - If one word, then use all lowercase letters
 - If multiple words, then start the first word with a lowercase letter and each consecutive word with an uppercase letter (e.g., custFirstName)
- Create a name that begins with a letter and not a dollar sign "\$" or underscore “_”

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

- Choose [loop control variables](#) that begin with a single lowercase letter (e.g., i, x, y)

Variable Declaration and Initialization

Declaring a variable is stating clearly that a variable exists. All variables are associated with a [data type](#) in the event of declaring a variable. [Data types](#) guarantee the correct data assigned to a variable.

In general, variables sustain an initial value before the variables are used. The [Assignment Operator](#) separates two sides of an equation. There is a left and right side of every equation. The left side displays a [variable name](#) while the right side displays a value. The following is the syntax along with an example of declaring and initializing a variable:

Syntax

variableType variableName;

Syntax Details

Argument	Description
variableType	Data type of variable being declared
variableName	Name of variable being declared
;	Semi-colon completes the declaration statement

Figure 2.2 – Variable Declaration Syntax Details

The following is a variable declaration and initialization example:

```
import org.testng.annotations.Test;

public class DeclareInitializeVariables
{
```

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

```

@Test
public void testDeclareInitialize ()
{
    int x;
    int y;

    x = 10;
    y = 15;

    System.out.println("The values of x and y are: " + x + " and " + y);
}
}

```

```

1 import org.testng.annotations.Test;
2
3 public class DeclareInitializeVariables
4 {
5     @Test
6     public void testDeclareInitialize ()
7     {
8         int x;
9         int y;
10
11         x = 10;
12         y = 15;
13
14         System.out.println("The values of x and y are: " + x + " and " + y);
15     }
16 }

```

The diagram illustrates the flow of variable handling in the Java code. It shows two main components: 'Declare Variables' and 'Initialize Variables'. Arrows point from these labels to their respective code segments. 'Declare Variables' points to the declarations of 'int x;' and 'int y;'. 'Initialize Variables' points to the assignments 'x = 10;' and 'y = 15;'.

Figure 2.3 – Variable Declaration

Program Output:

The values of x and y are: 10 and 15

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

- Line 1 is a required import statement to utilize the annotation “@Test” at line 5. All annotations are imported if the word “Test” in line 1 is replaced with an asterisk (*): **import** org.testng.annotations.*; (see *JUnit vs TestNG in Chapter 3*)
- Lines 8 and 9 declare variables “x” and “y” with an int data type. Notice how each declaration ends with a semi-colon. The semi-colon completes the declaration statement.
- Lines 11 and 12 initialize the variables “x” and “y” with a value determined by the int data type. Variable “x” initialized to 10 and variable “y” initialized to 15.

Note: It is possible to declare multiple variables on the same line if the variable has the same data type. The following is a declaration example of multiple variables separated by a comma:

int x, y;

In like manner, it is possible to declare and initialize a variable or variables on the same line. The following is an example of declaring and initializing a variable(s) on the same line:

int x = 10;
int y = 15;

or

int x = 10, y = 15;

Variable Type, Scope, and Lifetime

Java allows a variable to be declared anywhere in a program. For that reason, it is valid to declare a variable within a class, within a method, or within a method as a parameter. The scope of a variable relates to its declaration inside a program. Lifetime is how long the

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

variable exists in the program. The following are four types of variables, their scope, and lifetime:

1. [Local Variables](#)
2. [Parameter Variables](#)
3. [Instance Variables](#)
4. [Class Variables](#)

Local Variables

Local variables are declared inside a [method](#). Individual [methods](#) can have the same [variable name](#) as another method. Local variables are only visible inside its [method](#). The following is a local variable example:

```
import org.testng.annotations.Test;

public class LocalVariable
{
    @Test
    public void testLocalVariable ()
    {
        int age = 34;

        System.out.println("How old is Joe Doe? " + age);
    }

    @Test
    public void testSameLocalVariableName ()
    {
        int age = 38;

        System.out.println("How old is Jane Doe? " + age);
    }
}
```

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

```
1 import org.testng.annotations.Test;  
2  
3 public class LocalVariable  
4 {  
5     @Test  
6     public void testLocalVariable ()  
7     {  
8         int age = 34; ←  
9  
10        System.out.println("How old is Joe Doe? " + age);  
11    }  
12  
13     @Test  
14     public void testSameLocalVariableName ()  
15     {  
16         int age = 38; ←  
17  
18         System.out.println("How old is Jane Doe? " + age);  
19     }  
20 }  
21
```

Figure 2.4 – Local Variable

Program Output:

How old is Joe Doe? 34

How old is Jane Doe? 38

- Lines 8 and 16 display a local variable “age” that is local to methods “testLocalVariable and testSameLocalVariableName.”
- An error will not occur because each variable is unique to its method. However, the same variable name cannot be declared multiple times within the same method. The

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

scope and lifetime of a local variable is limited to the block/curly braces in which it is declared.

Parameter Variables

Parameter variables are declared in the parenthesis “()” of a method. After a parameter variable is declared, it works like a local variable. Therefore, a local variable and parameter variable cannot have the same name. Eclipse displays “duplicate local variable” if a local variable and parameter variable contain the same name. The following is a parameter variable example:

```
import org.testng.annotations.Test;

public class ParameterVariable
{
    @Test
    public void testParameterVariable (int age)
    {
        System.out.println("Jane Doe is " + age + " years old");
    }

    @Test
    public void passAgeArgument ()
    {
        ParameterVariable objAge = new ParameterVariable ();
        objAge.testParameterVariable (43);
    }
}
```

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

```
1 import org.testng.annotations.Test;  
2  
3 public class ParameterVariable  
4 {  
5     @Test  
6     public void testParameterVariable (int age)  
7     {  
8         System.out.println("Jane Doe is " + age + " years old");  
9     }  
10  
11     @Test  
12     public void passAgeArgument ()  
13     {  
14         ParameterVariable objAge = new ParameterVariable ();  
15  
16         objAge.testParameterVariable (43);  
17     }  
18 }
```



Figure 2.5 – Parameter Variables

Program Output:

Jane Doe is 43 years old

- Line 16 pass an argument “43” into parameter variable “age” within method “testParameterVariable.”
- The scope of a parameter variable is a method’s header inside the parenthesis while the lifetime is a method’s body within the curly brackets.

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Instance Variables

Instance variables (also known as Global Variables) are declared inside a [class](#) but outside of a [method](#). They can be declared before or after it is initialized with visibility to all [methods](#) in a [class](#). The following is an instance variable example:

```
import org.testng.annotations.Test;

public class InstanceVariable
{
    int age = 18;

    @Test
    public void testInstanceVariable ()
    {
        System.out.println("Most people graduate from high school at " +
                           age + " years old");
    }

    @Test
    public void testInstanceVariableAgain ()
    {
        System.out.println("Some places state " + age + " years old" +
                           " is the legal age for adulthood");
    }
}
```

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

```

1 import org.testng.annotations.Test;
2
3 public class InstanceVariable
4 {
5     int age = 18; ←
6
7     @Test
8     public void testInstanceVariable ()
9     {
10         System.out.println("Most people graduate from high school at " +
11                         age + " years old");
12     }
13
14     @Test
15     public void testInstanceVariableAgain ()
16     {
17         System.out.println("Some places state " + age + " years old" +
18                         " is the legal age for adulthood");
19     }
20 }
```

Figure 2.6 – Instance Variable

Program Output:

Most people graduate from high school at 18 years old
 Some places state 18 years old is the legal age for adulthood

- Line 5 declares the instance variable “age” within the class “InstanceVariable,” but outside of methods “testInstanceVariable and testInstanceVariableAgain”.
- The instance variable is accessed in lines 11 and 17 because instance variables are visible to all methods. Therefore, the lifetime of the variable exists as long as there is a reference.

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Note: Some automation engineers' call instance variables global variables. Therefore, an instance variable can be accessed directly using the instance variable name or object reference.

Class Variables

Class Variables (known as Static Variables) are declared in a class but not in a method. This type of variable is declared using keyword static. The keyword static announces to the compiler that only one copy of a particular variable exists, but shared by all instances of an object. When a member is declared as static, the member can be accessed prior to creating an object. However to access a static member, it is best to precede the static member with a class name and dot operator. Accessing a static member using a class name separates class variables from instance variables. The following is a class variable example:

```
import org.testng.annotations.Test;

public class ClassStaticVariable
{
    static int age = 18;

    @Test
    public void testInstanceVariable ()
    {
        System.out.println("Most people graduate from high school at " +
                           ClassStaticVariable.age + " years old");
    }

    @Test
    public void testInstanceVariableAgain ()
    {
        System.out.println("Some places state " + ClassStaticVariable.age +
                           " years old is the legal age for adulthood");
    }
}
```

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

```
1 import org.testng.annotations.Test;  
2  
3 public class ClassStaticVariable  
4 {  
5     static int age = 18; ←  
6  
7     @Test  
8     public void testInstanceVariable ()  
9     {  
10         System.out.println("Most people graduate from high school at " +  
11             ClassStaticVariable.age + " years old");  
12     }  
13  
14     @Test  
15     public void testInstanceVariableAgain ()  
16     {  
17         System.out.println("Some places state " + ClassStaticVariable.age +  
18             " years old is the legal age for adulthood");  
19     }  
20 }
```

Figure 2.7 – Class Variable

Program Output:

Most people graduate from high school at 18 years old
Some places state 18 years old is the legal age for adulthood

- Line 5 declares the Class Variable “age” within the class “ClassStaticVariable” but outside of methods “testInstanceVariable and testInstanceVariableAgain”.
- The scope of a class variable is inside the block/curly braces of class and outside the block/curly braces of all methods. Therefore, the lifetime of the variable continues throughout execution of the program.

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Note: A class variable can be accessed via class name, object reference, or directly using the class variable name. However, it is recommended to use the class name to access the class variable.

Operators

Operators are symbols that perform mathematical or logical manipulations on one or more operands. An operand is anything that can be changed or manipulated. The most common type of operand is a variable. In Java, there are four types of operators: Arithmetic, Bitwise, Logical, and Relational. However, Arithmetic, Logical, and Relational operators are the most used operators in Selenium.

Arithmetic Operators

Arithmetic operators implement mathematical operations on numerical values. Therefore, the arithmetic operators apply to any data type involving numbers. The following is a list of arithmetic operators:

Symbol	Name	Description
+	Addition	Adds a value on both sides of the (+) operator Used for joining strings which is known as string concatenation
-	Subtraction	Subtracts right operand from left operand
*	Multiplication	Multiplies values on both sides of the (*) operand
/	Division	Divides left operand by right operand

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

%	Modulus	Divides left operand by right operand then returns the remainder
++	Increment	Increases the operand's value by one
--	Decrement	Decreases the operand's value by one

Figure 2.8 – Arithmetic Operators

Note: The Division Operator (/) truncates the remainder while the Modulus Operator (%) returns the remainder. For instance, 10/3 only returns three and truncates the remainder, which is one. On the other hand, 10%3 only returns the remainder of one. The following example demonstrates a Multiplication (*) Operator:

```
import org.testng.annotations.Test;

public class OperatorArithmetic
{
    @Test
    public void testMultiplication ()
    {
        int total;

        total = 4 * 5;
        System.out.println("What is 4 times 5? " + total);
    }
}
```

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

```

1 import org.testng.annotations.Test;
2
3 public class OperatorArithmetic
4 {
5     @Test
6     public void testMultiplication ()
7     {
8         int total;
9
10        total = 4 * 5;
11        System.out.println("What is 4 times 5? " + total);
12    }
13 }
```



Figure 2.9 – Operator and Operands

Program Output:

What is 4 times 5? 20

- The multiplication operator (*) is used at line 10
- 4 and 5 are the operands

Logical Operators

Logical Operators (known as Conditional Operators) return a boolean value based on one or more expressions. Therefore, the Logical Operator's data type must be boolean. The following is a list of logical operators:

Symbol	Name	Description

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

&&	Logical AND	Returns true if both operands are true Returns false if one operand or both operands are false
	Logical OR	Returns true if one operand or both operands are true Returns false if both operands are false
^	Logical Exclusive OR (XOR)	Returns true if only one operand is true Returns false if both operands are false and if both operands are true
!	Logical NOT	Returns the opposite value of the operand Returns true if the operand is false and return false if the operand is true

Figure 2.10 – Logical Operators

Note: The Bitwise Operators and Logical Operators perform some of the same functions. The following are examples:

```
import org.testng.annotations.Test;

public class OperatorLogicalBitwise
{
    @Test
    public void testLogicalBitwise ()
    {
        boolean x = 100 > 99, y = 99 > 100;

        // Logical AND '&&' operator
        System.out.println("What is the result of 100 > 99 && 99 > 100? " +
(x && y));

        // Bitwise AND '&' operator
    }
}
```

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

```
System.out.println("What is the result of 100 > 99 & 99 > 100? " +
(x & y));

// Logical OR '||' operator
System.out.println("What is the result of 100 > 99 || 99 > 100? " +
(x || y));

// Bitwise OR '|' operator
System.out.println("What is the result of 100 > 99 | 99 > 100? " +
(x | y));

// Logical XOR '^' operator
System.out.println("What is the result of 100 > 99 ^ 99 > 100? " +
(x ^ y));

// Logical NOT '!' operator
System.out.println("What is the result of Not 100 > 99? " + (!x));

// Logical NOT '!' operator (parenthesis is optional surrounding
this operator and operand)
System.out.println("What is the result of Not 99 > 100? " + !y);
    }
}
```

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

```

1 import org.testng.annotations.Test;
2
3 public class OperatorLogicalBitwise
4 {
5     @Test
6     public void testLogicalBitwise ()
7     {
8         boolean x = 100 > 99, y = 99 > 100;
9
10        // Logical AND '&&' operator
11        System.out.println("What is the result of 100 > 99 && 99 > 100? " + (x && y));
12
13        // Bitwise AND '&' operator
14        System.out.println("What is the result of 100 > 99 & 99 > 100? " + (x & y));
15
16        // Logical OR '||' operator
17        System.out.println("What is the result of 100 > 99 || 99 > 100? " + (x || y));
18
19        // Bitwise OR '|' operator
20        System.out.println("What is the result of 100 > 99 | 99 > 100? " + (x | y));
21
22        // Logical XOR '^' operator
23        System.out.println("What is the result of 100 > 99 ^ 99 > 100? " + (x ^ y));
24
25        // Logical NOT '!' operator
26        System.out.println("What is the result of Not 100 > 99? " + (!x));
27
28        // Logical NOT '!' operator (parenthesis is optional surrounding this operator and operand)
29        System.out.println("What is the result of Not 99 > 100? " + !y);
30    }
31 }

```

Figure 2.11 – Logical and Bitwise Operator Examples

Program Output:

What is the result of 100 > 99 && 99 > 100? false
 What is the result of 100 > 99 & 99 > 100? false
 What is the result of 100 > 99 || 99 > 100? true
 What is the result of 100 > 99 | 99 > 100? true
 What is the result of 100 > 99 ^ 99 > 100? true
 What is the result of Not 100 > 99? false
 What is the result of Not 99 > 100? True

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

In this example, line 8 declares and initializes the variables. Both variables “x and y” are assigned boolean expressions. Variable “x” assigned a true expression ($100 > 99$) while “y” assigned a false expression ($99 > 100$). A **&& (Logical AND)** operator in line 11 and **&** (Bitwise AND) operator in line 14 compares the operands “x and y” then returns a “false” value. False is returned because one operand “x” is true while the other operand “y” is false. A similar process is performed for all examples in [Figure 2.11](#) using different Logical and Bitwise Operators.

Relational Operators

Relational Operators return a boolean value after comparing operands. Normally, all of the Relational Operators apply to operands that are numbers. If the relationship between two operands is Yes, then true is returned. For example, if 34 equals 34, then the output returns true. The following is a list of Relational Operators:

Symbol	Name	Description
<code>==</code>	Equal To	Verifies if both operands are equal.
<code>!=</code>	Not Equal To	Verifies if both operands are not equal.
<code>></code>	Greater Than	Verifies if the left operand is greater than the right operand
<code>>=</code>	Greater Than or Equal To	Verifies if the left operand is greater than or equal to the right operand
<code><</code>	Less Than	Verifies if the left operand is less than the right operand

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

<=	Less Than or Equal To	Verifies if the left operand is less than or equal to the right operand
----	-----------------------	-------------------------------------------------------------------------

Figure 2.12 – Relational Operators

The following are examples of each relational operator:

```
import org.testng.annotations.Test;

public class OperatorRelational
{
    @Test
    public void testRelational ()
    {
        int x = 25, y = 50;

        // == Equal To operator
        System.out.println("Is 25 equal to 50? " + (x == y));

        // != Not Equal To operator
        System.out.println("Is 25 not equal to 50? " + (x != y));

        // > Greater Than operator
        System.out.println("Is 25 greater than 50? " + (x > y));

        // >= Greater Than or Equal To operator
        System.out.println("Is 25 greater than or equal to 50? " + (x >= y));

        // < Less Than operator
        System.out.println("Is 25 less than 50? " + (x < y));

        // <= Less Than or Equal To operator
        System.out.println("Is 25 less than or equal to 50? " + (x <= y));
    }
}
```

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

```

1 import org.testng.annotations.Test;
2
3 public class OperatorRelational
4 {
5     @Test
6     public void testRelational ()
7     {
8         int x = 25, y = 50;
9
10        // == Equal To operator
11        System.out.println("Is 25 equal to 50? " + (x == y));
12
13        // != Not Equal To operator
14        System.out.println("Is 25 not equal to 50? " + (x != y));
15
16        // > Greater Than operator
17        System.out.println("Is 25 greater than 50? " + (x > y));
18
19        // >= Greater Than or Equal To operator
20        System.out.println("Is 25 greater than or equal to 50? " + (x >= y));
21
22        // < Less Than operator
23        System.out.println("Is 25 less than 50? " + (x < y));
24
25        // <= Less Than or Equal To operator
26        System.out.println("Is 25 less than or equal to 50? " + (x <= y));
27    }
28 }

```

Figure 2.13 – Relational Operator Examples

Program Output:

Is 25 equal to 50? false
 Is 25 not equal to 50? true
 Is 25 greater than 50? false
 Is 25 greater than or equal to 50? false

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

```
Is 25 less than 50? true
Is 25 less than or equal to 50? True
```

In this example, line 8 declares and initializes the variables. Variable “x” assigned 25 while “y” assigned 50. An [== \(Equal To\)](#) operator determines if both variables ($x == y$) equals each other on line 11. The values 25 and 50 are not equal so the program returns false. A similar process is performed for all examples in [Figure 2.13](#) using a different Relational Operator according to [Figure 2.12](#).

Note: See Operators in Chapter 3 of “[Absolute Beginner \(Part 1\) Java 4 Selenium WebDriver](#)” for additional examples, figures, etc. The eBook and PDF formats are free.

Assignment Operator

An Assignment Operator ($=$) is positioned between a variable and value / expression. The purpose is to assign values to variables. Therefore, the value on the right side transferred to the variable name on the left side. The following is an assignment operator syntax:

Syntax

```
variableName = expression;
```

Syntax Details

Argument	Description
variableName	Name of variable that was declared
expression	Value that is assigned to the variable name
;	Semi-colon completes the initialization statement

Figure 2.14 – Assignment Operator Syntax Details

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

The following is an Assignment Operator example:

```
import org.testng.annotations.Test;

public class OperatorAssignment
{
    @Test
    public void testAssignmentOperator ()
    {
        int j;

        j = 34;
        System.out.println("The value " + j + " has been assigned to j");
    }
}
```

```
1 import org.testng.annotations.Test;
2
3 public class OperatorAssignment
4 {
5     @Test
6     public void testAssignmentOperator ()
7     {
8         int j;
9
10        j = 34; ←
11        System.out.println("The value " + j + " has been assigned to j");
12    }
13 }
```

Figure 2.15 – Assignment Operator Example

Program Output:

The value 34 has been assigned to j

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

- Line 10 assigns a value “34” to variable “j”
- The value is printed via line 11

Control Structures

Control structures are the process of using logic to force the program to skip statements while looping other statements. Forcing the program to skip statements is known as branching and looping specific statements is carried out via loops. The two types of branches are [if branch](#) and [switch branch](#). The three types of loops are [for loop](#), [while loop](#), and [do while loop](#).

If Branch

The if branch executes a statement when a condition is true. In other words, a specific statement executes if a condition is met. An if branch is a greatly utilized and considered an indispensable control structure. The following is the syntax for the if branch:

Syntax

```
if (condition)
{
    statement(s);
}
```

Syntax Details

Argument	Description
if	Keyword that starts the if branch
condition	Boolean expression which results in a true or false result
{	An opening curly bracket

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

statement(s)	Statement that will be executed if the condition is true
;	Semi-colon completes the true statement
}	A closing curly bracket

Figure 2.16 – If Branch Syntax Details

The following example displays a message if the customer brings three or more extra customers to a sporting event:

```
import org.testng.annotations.*;

public class IfBranch
{
    @Test
    public void testIfBranch ()
    {
        int extraCustomers = 4;

        if (extraCustomers >= 3)
        {
            System.out.println("Customer receives a discount");
        }
    }
}
```

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

```
1 import org.testng.annotations.*;
2
3 public class IfBranch
4 {
5     @Test
6     public void testIfBranch ()
7     {
8         int extraCustomers = 4;
9
10        if (extraCustomers >= 3) ←—————
11        {
12            System.out.println("Customer receives a discount");
13        }
14    }
15 }
```

Figure 2.17 – If Branch

Program Output:

Customer receives a discount

Line 8 assigns “4” to the variable “extraCustomers”. Line 10 displays keyword “if” followed by a parenthesis. Inside the parenthesis is a condition (`extraCustomers >= 3`) that returns true. True is returned because four is greater than three. The statement at line 12 (inside the curly brackets) executes after the true evaluation.

Note: The program would not execute the statement if the condition returned false. However, there are two variations of the if branch that can be executed when a condition is false:

1. If Else

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

2. If Else-If

If Else Branch

An optional else keyword extends the if branch just in case the condition returns false. Therefore, the statements following keyword “if” and the condition executes when a condition is true. Otherwise, the statement following keyword else, is executed when a condition is false. The following is the syntax for the if-else branch:

Syntax

```
if (condition)
{
    statement(s);
}
else
{
    statement(s);
}
```

The following example displays a message when the customer does not bring three or more extra customers to a sporting event:

```
import org.testng.annotations.*;

public class IfElseBranch
{
    @Test
    public void testIfElseBranch ()
    {
        int extraCustomers = 2;

        if (extraCustomers >= 3)
        {
            System.out.println("Customer receives a discount");
        }
    }
}
```

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

```
        }
    else
    {
        System.out.println("Customer does not receive a discount");
    }
}
```

```
1 import org.testng.annotations.*;
2
3 public class IfElseBranch
4 {
5     @Test
6     public void testIfElseBranch ()
7     {
8         int extraCustomers = 2;
9
10        if (extraCustomers >= 3)
11        {
12            System.out.println("Customer receives a discount");
13        }
14        else ←————
15        {
16            System.out.println("Customer does not receive a discount");
17        }
18    }
19 }
```

Figure 2.18 – If Else Branch

Program Output:

Customer does not receive a discount

Line 8 assigns “2” to the variable “extraCustomers”. Line 10 displays keyword “if” followed by a parenthesis. Inside the parenthesis is a condition (extraCustomers >= 3) that returns

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

false. False is returned because two is not greater than or equal to three. Therefore, the program bypasses the statement at line 12 and executes the statement at line 16.

If Else-if Branch

The first if keyword can optionally be followed by one or more if keywords. However, each subsequent if keyword must be preceded by a required else keyword. The else-if branch is only executed when the first if branch is false. All else-if branches are followed by a condition and one or more statements. The following is the syntax for the else if branch:

Syntax

```
if (condition)
{
    statement(s);
}
else if (condition)
{
    statement(s);
}
else if (condition)
{
    statement(s);
}
else
{
    statement(s);
}
```

The following example displays a message when the customer brings less than three extra customers to a sporting event:

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

```
import org.testng.annotations.Test;

public class ElseIfBranch
{
    @Test
    public void testElseIfBranch ()
    {
        int extraCustomers = 2;

        if (extraCustomers >= 3)
        {
            System.out.println("Customer receives a discount");
        }
        else if (extraCustomers <= 3)
        {
            System.out.println("No Discount: Customer count less than or
equal to 3");
        }
        else
        {
            System.out.println("Error: Not a valid customer count");
        }
    }
}
```

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

```
1 import org.testng.annotations.Test;
2
3 public class ElseIfBranch
4 {
5     @Test
6     public void testElseIfBranch ()
7     {
8         int extraCustomers = 2;
9
10        if (extraCustomers >= 3)
11        {
12            System.out.println("Customer receives a discount");
13        }
14        else if (extraCustomers <= 3) ←
15        {
16            System.out.println("No Discount: Customer count less than or equal to 3");
17        }
18        else
19        {
20            System.out.println("Error: Not a valid customer count");
21        }
22    }
23 }
```

Figure 2.19 – Else If Branch

Program Output:

No Discount: Customer count less than or equal to 3

Line 8 assigns “2” to the variable “extraCustomers”. Line 14 displays keywords “else” and “if” followed by a parenthesis. Inside the parenthesis is a condition (`extraCustomers <= 3`) that returns true. True is returned because two is less than or equal to three. The second condition (line 14) only executes after the first condition (line 10) returns false.

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

Note: Several else-if branches can be added to the if branch. On the other hand, line 20 executes if variable “extraCustomers” assigned an invalid value such as characters. The program output would have printed “Error: Not a valid customer count”.

Switch Branch

The switch branch evaluates a single variable then executes a statement according to the variable’s value. Primitive data types `byte`, `short`, `char`, and `int` are evaluated along with `String`. The switch and if branches have similar functionality. There are situations where either branch is suitable. However, the switch branch is most efficient when dealing with a specific number of values, such as days of the week. Otherwise, it is best to implement an if branch when handling an infinite number of values. The following is the syntax for the switch branch:

Syntax

```
switch (variableName)
{
    case constant1:
        statement(s);
        break;
    case constant2:
        statement(s);
        break;
    case constant3:
        statement(s);
        break;
    .
    .
    .
    default:
}
```

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

```
    statement;  
}
```

The following is a switch branch example:

```
import org.testng.annotations.Test;  
  
public class BranchSwitch  
{  
    @Test  
    public void testSwitchBranch ()  
    {  
        int day = 6;  
  
        switch (day)  
        {  
            case 1:  
                System.out.println("Sunday is the 1st day of the week");  
                break;  
            case 2:  
                System.out.println("Monday is the 2nd day of the week");  
                break;  
            case 3:  
                System.out.println("Tuesday is the 3rd day of the week");  
                break;  
            case 4:  
                System.out.println("Wednesday is the 4th day of the week");  
                break;  
            case 5:  
                System.out.println("Thursday is the 5th day of the week");  
                break;  
            case 6:  
                System.out.println("Friday is the 6th day of the week");  
                break;  
            case 7:  
                System.out.println("Saturday is the 7th day of the week");  
                break;  
            default:        }
```

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

```
        System.out.println("Not valid: There are only 7 days in a  
week");  
    }  
}  
}
```

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

```
1 import org.testng.annotations.Test;
2
3 public class BranchSwitch
4 {
5     @Test
6     public void testSwitchBranch ()
7     {
8         int day = 6;
9
10        switch (day) ←————
11        {
12            case 1:
13                System.out.println("Sunday is the 1st day of the week");
14                break;
15            case 2:
16                System.out.println("Monday is the 2nd day of the week");
17                break;
18            case 3:
19                System.out.println("Tuesday is the 3rd day of the week");
20                break;
21            case 4:
22                System.out.println("Wednesday is the 4th day of the week");
23                break;
24            case 5:
25                System.out.println("Thursday is the 5th day of the week");
26                break;
27            case 6:
28                System.out.println("Friday is the 6th day of the week");
29                break;
30            case 7:
31                System.out.println("Saturday is the 7th day of the week");
32                break;
33            default:
34                System.out.println("Not valid: There are only 7 days in a week");
35        }
36    }
37 }
```

Figure 2.20 – Switch Branch Example

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

Program Output:

Friday is the 6th day of the week

Line 8 assigns the variable “day” the value of “6.” Then the keyword “switch” starts the branch at line 10 by checking the variable’s value. Keyword “case” at line 27 matches the variable’s value “6”, and then executes the statement at line 28. The keyword “break” at line 32 is necessary to prevent case 7 (line 30) and default (line 33) from executing.

Note: All statements following a match will execute due to switch branches executing sequentially utilizing a top-down approach. Therefore, the keyword “break” must be used to jump out of the switch branch after a match is found.

For Loop

The for loop executes a block of code for a certain number of iterations. In other words, a statement executes as long as a condition is met. One of the benefits allows statements to execute without writing code repeatedly. The following is the for loop syntax:

Syntax

```
for (initialization; condition; iteration)
{
    statement(s);
}
```

Syntax Details

Argument	Description
for	Keyword that starts the for loop
initialization	Assignment that sets the loop control initial value

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

;	Semi-colon completes the initialization
condition	A boolean expression that determines if the loop will or will not repeat
;	Semi-colon completes the condition
iteration	Indicates how the loop control variable will change after each variation
{	An opening curly bracket
statement(s)	Statement(s) that will execute after the condition is met
;	Semi-colon completes the statement
}	A closing curly bracket

Figure 2.21 – For Loop Syntax Details

The initialization component declares a [data type](#) and assigns an initial value via loop control variable. Usually, the loop control variable is a single character variable name (e.g., i) that controls the entire loop. The condition is a boolean expression that specifies a maximum value for the loop control variable. All for loops continue executing while the condition is true. Execution continues on the statement immediately following the for loop when the condition becomes false. Most automation engineers use an [increment \(++\)](#) or [decrement \(--\)](#) operator as the iteration expression. The [increment](#) operator increases the [loop control variable](#) value by one, while the [decrement](#) operator decreases the value by one. An executable statement is placed between the optional curly brackets. Although, the curly brackets are optional, it is recommended to use the brackets to improve code readability. The following is a for loop example:

```
import org.testng.annotations.Test;
```

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

```

public class LoopFor
{
    @Test
    public void testForLoop ()
    {
        for (int i = 0; i < 5; i++ )
        {
            System.out.println("The loop control variable value is " + i);
        }
    }
}

```

```

1 import org.testng.annotations.Test;
2
3 public class LoopFor
4 {
5     @Test
6     public void testForLoop ()
7     {
8         for (int i = 0; i < 5; i++ ) ←
9         {
10            System.out.println("The loop control variable value is " + i);
11        }
12    }
13 }

```

Figure 2.22 – For Loop Example

Program Output:

The loop control variable value is 0
 The loop control variable value is 1
 The loop control variable value is 2

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

```
The loop control variable value is 3  
The loop control variable value is 4
```

Line 8 starts the for loop with keyword “for” followed by three arguments (initialization, condition, and iteration). Initialization (int i = 0;) assigns zero as the starting value. Condition (i < 5) sets less than five as the stopping point for the loop control variable. Increment (i++) increases the loop control variable by one. The statement prints the loop control variable via line 10.

It is important to use harmonious values in the for loop. The values lead to an infinite loop if the values are not created in agreement. An infinite loop is a loop that never stops. For example, the following for loop will repeat indefinitely because of the initial value, maximum value, and iteration expression:

```
for (int i = 3; i > 1; i++)
```

The initial value “3” starts at a greater value than the maximum value of “1,” while the iterator “++” increases after each loop. This example continues indefinitely since the condition always true. To correct this infinite loop, the initialization value “3” must decrease to less than the conditional value “1”; the conditional value “1” must increase to more than the initialization “3”; or the iterator must change from increasing “++” to decreasing “--” after each loop.

Note: Routinely, a condition using a greater than operator (>) implements a [decrement operator \(--\)](#), while a condition using a less than operator (<) implements an [increment operator \(++\)](#).

While Loop

The while loop repeats a statement while a condition is true. Conditions are boolean expressions that is checked prior to executing the statement. In addition, the [variable name](#) initialized before the loop and evaluated as part of the condition. When executing the

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

statement, the while loop continues until the condition becomes false. The following is the syntax of a while loop.

Syntax

```
while (condition)
{
    statement(s);
}
```

Syntax Details

Argument	Description
while	Keyword that starts the loop
condition	A boolean expression that determines if the loop will or will not repeat
{	An opening curly bracket
statement(s)	Statement(s) that will execute after the condition is met
;	Semi-colon that completes the statement
}	A closing curly bracket

Figure 2.23 – While Loop Syntax Details

The following is a while loop example:

```
import org.testng.annotations.Test;

public class LoopWhile
{
    @Test
```

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

```
public void testWhileLoop ()  
{  
    int i = 0;  
  
    while (i < 5)  
    {  
        System.out.println("The variables value is " + i);  
        i++;  
    }  
}
```

```
1 import org.testng.annotations.Test;  
2  
3 public class LoopWhile  
4 {  
5     @Test  
6     public void testWhileLoop ()  
7     {  
8         int i = 0;  
9  
10        while (i < 5) ←—————  
11        {  
12            System.out.println("The variables value is " + i);  
13            i++;  
14        }  
15    }  
16 }
```

Figure 2.24 – While Loop Example

Program Output:

The variables value is 0

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

```
The variables value is 1
The variables value is 2
The variables value is 3
The variables value is 4
```

Line 8 initializes the variable "i" to zero "0". The variable will be evaluated at line 10 as part of the condition ($i < 5$) after keyword "while". A value for the variable "i" prints repeatedly via line 12 while the condition is true. Notice the increment operator at line 13. It is important to know that the while loop never stops without an increment operator. Therefore, the loop would continue indefinitely, generating an infinite loop. In addition, the while loop becomes indefinite if the initialization and conditional variable values are not set in agreement.

Do While Loop

The do while loop evaluates a condition at the bottom of the loop. Therefore, the loop will execute the statement within the loop then evaluate the condition. As a result, the do while loop always executes a statement at least one iteration and continues as long as the condition is true. The following is the syntax for a do while loop:

Syntax

```
do
{
    statement(s);
}
while (condition);
```

Syntax Details

Argument	Description
do	Keyword that starts the loop

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

{	An opening curly bracket
statement(s)	Statement(s) that will execute at least once
;	Semi-colon that completes the statement
}	A closing curly bracket
while	Keyword that determines if the loop's condition will repeat
condition	A boolean expression that determines if the loop will or will not repeat
;	Semi-colon that completes the condition

Figure 2.25 – Do While Loop Syntax Details

The following is a do while loop example:

```
import org.testng.annotations.Test;

public class LoopDoWhile
{
    @Test
    public void testDoWhileLoop ()
    {
        int i = 0;

        do
        {
            System.out.println("The variables value is " + i);
            i++;
        }
        while (i < 5);
    }
}
```

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

```
1 import org.testng.annotations.Test;  
2  
3 public class LoopDoWhile  
4 {  
5     @Test  
6     public void testDoWhileLoop ()  
7     {  
8         int i = 0;  
9  
10        do ←  
11        {  
12            System.out.println("The variables value is " + i);  
13            i++;  
14        }  
15        while (i < 5); ←  
16    }  
17 }
```

Figure 2.26 – Do While Loop Example

Program Output:

```
The variables value is 0  
The variables value is 1  
The variables value is 2  
The variables value is 3  
The variables value is 4
```

Line 8 initializes the variable "i" to zero "0". The keyword "do" starts the do while loop followed by two statements surrounded by curly brackets. A value for the variable "i" prints repeatedly via line 12 while the condition ($i < 5$) is true. Coincidentally, the condition is evaluated after the statement at line 15. Like the while loop, an infinite loop would have

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

occurred if the increment operator (++) was not added at line 13. The initialization and conditional values also create an infinite loop if not set correctly. In this example, the statements repeated multiple iterations because the condition started with a true result. The following shows what happens when the condition starts with a false result:

```
import org.testng.annotations.Test;

public class LoopDoWhile
{
    @Test
    public void testDoWhileLoopFalse ()
    {
        int i = 0;

        do
        {
            System.out.println("The variables value is " + i);
            i++;
        }
        while (i > 5);
    }
}
```

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

```

1 import org.testng.annotations.Test;
2
3 public class LoopDoWhile
4 {
5     @Test
6     public void testDoWhileLoopFalse ()
7     {
8         int i = 0;
9
10        do
11        {
12            System.out.println("The variables value is " + i);
13            i++;
14        }
15        while (i > 5); ←
16    }
17 }
```

Figure 2.27 – Do While Loop Example (Start With False Condition)

Program Output:

The variables value is 0

Line 8 initializes the variable "i" to zero "0". Therefore, the condition ($i > 5$) at line 15 is false due to zero being less than five. The do while loop executed the statement because statements are executed prior to evaluating the condition.

Note: The loops ([for](#), [while](#), and [do while](#)) are similar in functionality. A rule of thumb when deciding which loop to implement is:

- Implement a [for loop](#) when executing a specific number of iterations

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

- Implement a [while loop](#) when the loop will repeat an uncertain number of iterations
- Implement a [do while loop](#) when a loop needs to be executed at least one iteration

Note: See Control Structures in Chapter 4 of “[Absolute Beginner \(Part 1\) Java 4 Selenium WebDriver](#)” for additional examples, figures, etc. The eBook and PDF formats are free.

Classes, Objects, and Methods

Object-Oriented Programming (OOP) is an approach to programming that centers around objects. As a result, identifying objects is one of the most essential principles in OOP.

Objects serve as the foundation for OOP while methods perform actions. Both objects and methods are located inside of a class. The following is an example of an English Course using a class, object, and method:

```
import org.testng.annotations.Test;

public class EnglishCourse
{
    int students;
    int weeks;
    int days;

    @Test
    public void testEnglishCourse ()
    {
        EnglishCourse ENG101 = new EnglishCourse ();
        int totalDays;

        ENG101.students = 10;
        ENG101.weeks = 4;
        ENG101.days = 3;

        totalDays = ENG101.weeks * ENG101.days;
        System.out.println("The English 101 course is a total of " +
totalDays + " days");
    }
}
```

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

```

    }
}

```

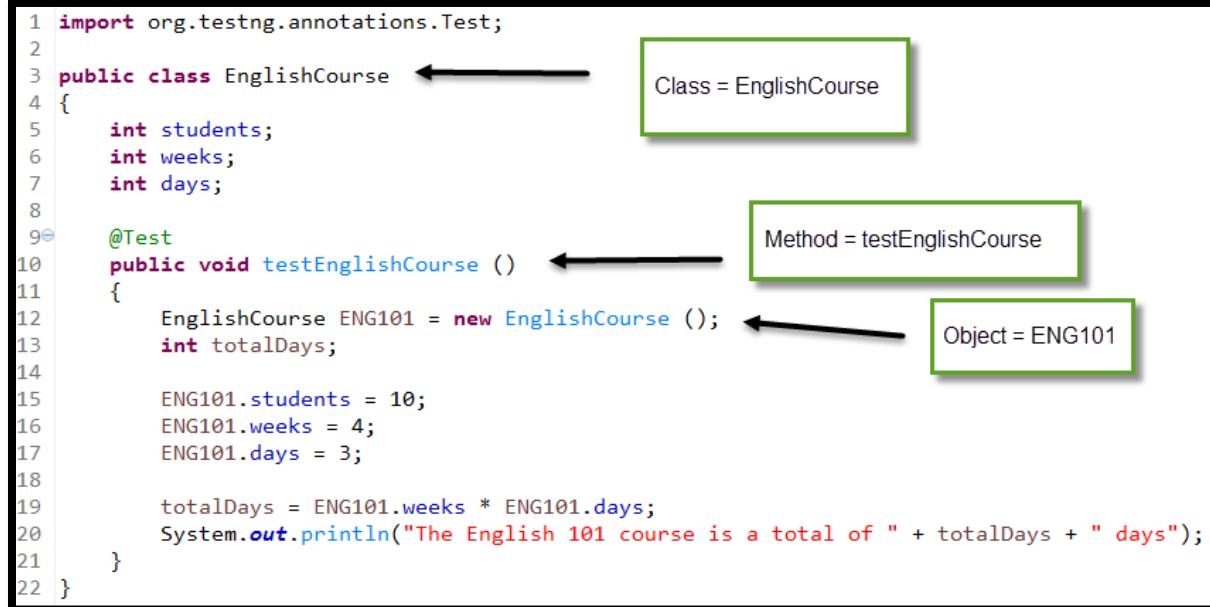


Figure 2.28 – Class, Object, and Method

Program Output:

The English 101 course is a total of 12 days

- Line 3 defines the class called “EnglishCourse”
- Line 10 defines the method called “testEnglishCourse”
- Line 12 creates a new object (`EnglishCourse ENG101 = new EnglishCourse ();`) by combining two steps. According to Java A Beginner’s Guide Sixth Edition (2014), the two steps combined can be rewritten like the following to show each step individually (page 106):

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

1. **EnglishCourse ENG101;** - The left side of the assignment declares a variable called ENG101 of class type EnglishCourse
2. **ENG101 = new EnglishCourse();** - The right side of the assignment creates a copy of the object and give ENG101 a reference to the object

ENG101 is a creation (known as instance) of EnglishCourse after creating the new object. Lines 15 – 17 assign values to the instance variables by accessing the variables using the dot (.) operator.

Class

A class is a blueprint for creating an object. Classes include data and code that operate on the data. Hence, a class is a template that defines the structure of an object. Therefore, the structure of a class must be precise. Classes formed with one logical entity makes the class complete. It is important to define classes with information that is logically connected. For example, a class that contains information about an English Course would not contain unrelated information about the school zone speed limit.

Note: A convention for naming classes is to use an UpperCamelCase where each word in the class name begins with a capital letter (i.e., EnglishCourse).

Objects

Objects serve as the cornerstone for OOP. All objects have two characteristics: state and behavior. State identifies the object and behavior represent actions of the object. For example, a dog has a state (name, breed, color) which identifies the dog and behavior (bark, jump, fetch) which represent the dog's actions. Objects are created using the keyword “new”.

Note: A convention for naming an object is similar to [naming a variable](#).

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

Methods

A method executes a job for the object. The state of an object is supported by variables while behavior is implemented through methods. A method is a block of code surrounded by curly brackets. The purpose is to manipulate and provide access to data defined by the class. In other words, a method's responsibility is to instruct the program what task/action to perform and how to perform the task/action.

A method performs actions on the data. It is best for all methods to carry out a single task. For example, a good method will only perform a single task of adding numbers but not adding numbers and saving data to a file. The program may become difficult to read and understand if a method carries out more than one task. The following is the syntax for a method:

Syntax

```
methodType methodName (parameter-list)
{
    // Method Body
}
```

Methods consist of a header and body. The method's header includes a method type and a method name. The convention for method name is similar to a variable name, which consist of a lowerCamelCase style. Each word begins with a capital letter except the first word. Succeeding the method name is a required pair of parenthesis that sets apart variables from methods.

A good technique to employ for naming a method is verb-noun combinations, such as “getOrder or addNumbers”. Method body executes code to carry out a task. The parameter list contains variables that receive arguments passed to the method. If the method has no parameters then the parameter list must remain empty. The method type (known as return type) determines the data type returned by the method. A programmer is forced to use keyword “void” for the method type if no values are returned.

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Note: See Classes, Objects, and Methods in Chapter 2 of “[\(Part 2\) Java 4 Selenium WebDriver](#)” for additional examples, figures, etc.

Access Modifiers

Access modifiers are helpful features of object-oriented programming. They are helpful because of the access limitation it places on every class and class members (variables and methods). One of the access modifiers “public” has been used throughout this chapter. In Java, there are four kinds of access modifiers:

1. public – indicates a member can be accessed by all classes
2. protected – indicates a member can be accessed by all classes and subclasses within its own package (see subclass and package in [\(Part 2\) Java 4 Selenium WebDriver](#))
3. no modifier – indicates a member can be accessed by all classes within its own package
4. private – indicates a member can be accessed within its own [class](#)

	public	protected	no modifier	private
Class	Yes	Yes	Yes	Yes
Package	Yes	Yes	Yes	No
Subclass	Yes	Yes	No	No
World	Yes	No	No	No

Figure 2.29 – Access Modifier Levels

Inheritance

Inheritance is a hierarchical concept that allows code and objects to be reused. Each class allows other classes to inherit its code. As a result, the relationship between the classes are

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

superclass and subclass. Superclass is the parent class and subclass is the child class. The classes maintain common data while holding unique characteristics. In order for the subclass to inherit the superclass, the keyword “extends” must be used in the class declaration. Keyword “extends” means the subclass will add to the superclass. The following is the syntax for a subclass inheriting a superclass:

Syntax

```
class SubClassName extends SuperClassName
{
    //Class Body
}
```

The following is an example for a subclass inheriting a superclass:

```

1  class School
2  {
3      int numTeachers;
4      int numStudents;
5
6      void showNumberOfPeople ()
7      {
8          System.out.println("There are " + numTeachers + " teachers and " + numStudents + " students");
9      }
10 }
11 class ElementarySchool extends School
12 {
13     String principalName;
14
15     int totalTeacherStudents ()
16     {
17         return numTeachers + numStudents;
18     }
19
20     void displayPrincipal ()
21     {
22         System.out.println("The principal name is " + principalName);
23     }
24 }
25 class SchoolDistrict
26 {
27     public static void main(String[] args)
28     {
29         ElementarySchool BishopHeights = new ElementarySchool ();
30         ElementarySchool AltaMesa = new ElementarySchool ();
31
32         BishopHeights.numTeachers = 10;
33         BishopHeights.numStudents = 130;
34         BishopHeights.principalName = "Joe Doe";
35
36         System.out.println("My elementary school is Bishop Heights \n");
37         BishopHeights.displayPrincipal ();
38         BishopHeights.showNumberOfPeople ();
39         System.out.println("Therefore the total of teachers and students is " + BishopHeights.totalTeacherStudents ());
40     }
41 }
```

The class "ElementarySchool" is a subclass while "School" is a superclass.

Figure 2.30 – Subclass Extends Superclass Example

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Program Output:

My elementary school is Bishop Heights

The principal name is Joe Doe

There are 10 teachers and 130 students

Therefore the total of teachers and students is 140

- Line 1 displays superclass “School”
- Line 11 uses the keyword “extends” to create subclass “ElementarySchool” which inherits superclass “School”
- Line 17 utilize variables “numTeachers and numStudents” of the superclass
- Lines 29 and 30 display two objects “BishopHeights and AltaMesa”
- Lines 32 and 33 assign values “10 and 130” to variables “numTeachers and numStudents” which are inherited from superclass “School”
- Line 34 assign a value “Joe Doe” to variable “principalName” which is a member of subclass “Elementary School”

The subclass “ElementarySchool” is a unique type of superclass “School”. More subclasses such as HighSchool, University, etc. can inherit superclass “School”. The “ElementarySchool” class inherits all of the “School” class members then add a variable “principalName” and two methods “totalTeacherStudents () and displayPrincipal ()”. Notice how line 30 creates an object that was not utilized. This shows that more Elementary School objects can be created to access members of the superclass “School”.

Note: Objects from the superclass do not have knowledge of the subclass. Therefore, the superclass objects cannot access the subclass.

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

Encapsulation

The concept of encapsulation consists of binding code and data. A class encapsulates code and data giving structure to the class members. As a result of this structure, the code has access to manipulate data while preventing interference from outside sources.

Declaring a variable as private is one of the ways to prevent interference. Private is an access modifier that only allow a class member (variable or method) to be accessed within its own class. However, the accessor methods provide a bridge to private variables. Although there is a bridge, each private variable remains safe from misuse such as entering a negative value for only positive values.

Polymorphism

Polymorphism is an Object-Oriented Programming (OOP) concept that allows one interface to be utilized by multiple methods. In the case of superclass and subclass, a superclass provides a structure whereby a subclass can define its own method operation. The methods are inherited automatically when the subclass extends the superclass.

Packages

A package is a collection of related classes. The package is comparable to a folder and classes are similar to files within the folder. Each class within the package is accessed by the package name. It is important to know that Java requires a package import if a class wants to use members from a different class. A Java package is imported by using the keyword “import”. After importing a package, the members of the package are used directly without any additional syntax. The following is the syntax for importing a package:

Syntax

```
import packagename.ClassName;
```

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

The following screenshot shows a package “examplepackage” and class “PackageTest” that will be imported into a different class “PackageOne”.

```
1 package examplepackage;
2 import org.testng.annotations.Test;
3
4 class PackageTest
5 {
6     @Test
7     void printTest ()
8     {
9         System.out.println("This is a test");
10    }
11 }
```

Class "**PackageTest**" will be imported into a different class

Method "**printTest**" will print the Program Output

Figure 2.31 – Class PackageTest

The following screenshot shows an import of class “PackageTest” into class “PackageOne” from the same package “examplepackage”:

```

1 package examplepackage;
2 import examplepackage.PackageTest;
3 import org.testng.annotations.Test;
4
5 public class PackageOne
6 {
7     @Test
8     void printFromADifferentClass()
9     {
10         PackageTest objPackage = new PackageTest ();
11         objPackage.printTest();
12     }
13 }
14
15
16
17
18
19
20

```

The code illustrates Java's package system and object-oriented programming concepts. It defines a class `PackageOne` in the `examplepackage` package. The class contains a test method `printFromADifferentClass`. Inside this method, an object `objPackage` is created from the `PackageTest` class, also located in the same package. The code uses annotations `@Test` from the `org.testng.annotations` package.

Annotations:

- Line 2:** `import examplepackage.PackageTest;` - An arrow points to this line with the annotation: "Class 'PackageTest' was imported from the same package 'examplepackage'".
- Line 7:** `@Test` - An arrow points to this line with the annotation: "Object Reference 'objPackage' is used to call method 'printTest'" which prints the Program Output".

Figure 2.32 – Package Import

Program Output:

This is a test

- Line 1 contains the package statement “`package examplepackage;`”. The package statement reveals the package name which is `examplepackage`
- Line 2 contains the import statement “`import examplepackage.PackageTest;`”. The package name is “`examplepackage`” and class name is “`PackageTest`” which is located in a different file
- Line 3 imports the `@Test` annotation on line 7
- Line 5 begins the class declaration “`public class PackageOne`”
- Line 7 contains the `@Test` annotation
- Line 10 creates an object “`objPackage`” using the imported class “`PackageTest`”

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

- Line 11 calls the method “`printTest();`” which is located in a different class “`PackageTest`”. The method prints the Program Output “This is a test”

Note: All of the classes from a different package or same package “i.e., `examplepackage`” are imported using the package name and an asterisk: `import packagename.*;`

The following screenshot is the Package Directory for package “`examplepackage`” and classes “`PackageTest` and `PackageOne`”. It resembles a folder “`examplepackage`” that contains files “`PackageTest` and `PackageOne`”:

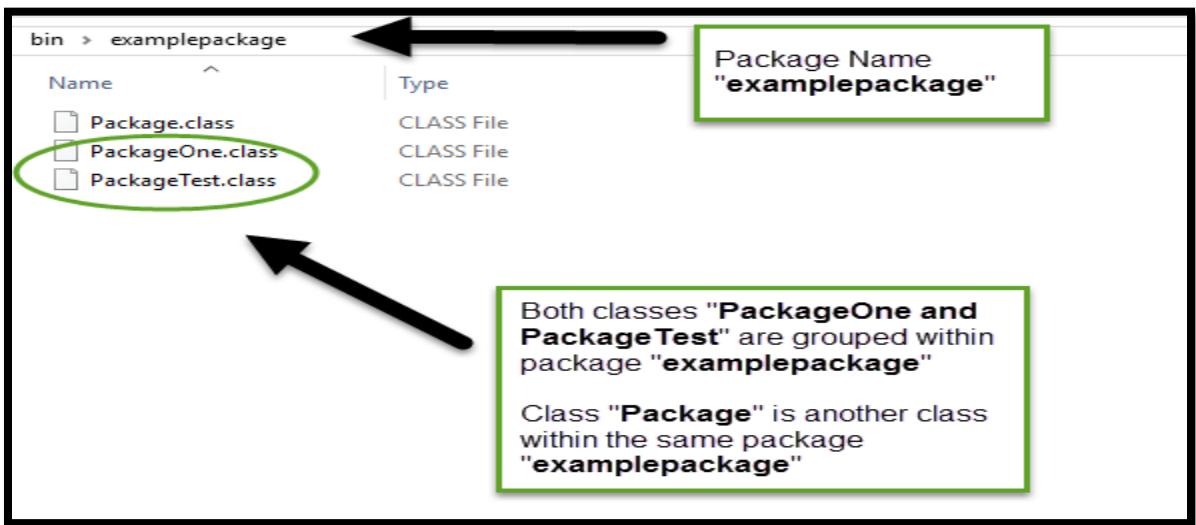


Figure 2.33 – Package Directory Screenshot

Interfaces

An interface is a collection of related methods. Generally, most interfaces do not include the body of a method. Therefore an interface method reveals what action to perform but not how

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

to perform the action. This concept allows a class to implement the interface method and decide how to perform the action. Each class has the ability to implement a different action for the same interface method. Interfaces optionally extend one or more interfaces similar to a subclass extending a superclass. The following is the syntax for an interface:

Syntax

```
accessmodifier interface interfaceName extends interfaceName1, interfaceName2,  
interfaceNameN  
{  
    variableType variableName1;  
    variableType variableName2;  
    variableType variableNameN;  
  
    methodType methodName1 (parameter-list);  
    methodType methodName2 (parameter-list);  
    methodType methodNameN (parameter-list);  
}
```

Syntax Details

Argument	Description
accessmodifier	Can be public or no access modifier. A public declaration indicates the interface can be used by any code. A no access modifier is the default which indicates the interface is only available to members of its package
interface	A keyword used to declare an interface
interfaceName	The name of an interface
extends	An optional keyword that extends one or more interfaces

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

interfaceName1, 2, N	One or more interfaces separated by a comma that will be extended
variableType variableName1, 2, N;	One or more interface variables
methodType methodName1 (parameter-list);	One or more interface methods

Figure 2.34 – Interface Syntax Details

The following is an interface example:

```

1 package InterfaceTesting;
2
3 public interface InterfaceTest
4 {
5     public void methodOne();
6     public void methodTwo();
7 }
```

Figure 2.35 – Interface Example

- Line 3 declares an interface by using the keyword “interface” with a “public” access modifier. Therefore, any class within any package can implement the code. The interface name is InterfaceTest.
- Lines 5 and 6 are void methods “methodOne and methodTwo”. Notice the semi-colon at the end of each method. The semi-colon is placed at the end of the methods because there is no body/implementation.

Note: The following concepts were not discussed in this chapter but are explained in “[\(Part 2\) Java 4 Selenium WebDriver](#)”:

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

- Arrays
- Strings
- Errors
- Exceptions
- Debugging
- File Input/Output

Chapter 2 provided an overview of Core Java (Variables, Data Types, Operators, Control Structures, Classes, Objects, Methods, Access Modifiers, Inheritance, Encapsulation, Polymorphism, Packages, and Interfaces). Chapter 3 will discuss JUnit and TestNG, which are Unit Test Frameworks. JUnit inspired TestNG then TestNG added more features for testing applications.

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Chapter 3

JUnit vs TestNG

JUnit and Test Next Generation (known as TestNG) are unit test frameworks structured for Java. Both frameworks support testing in Eclipse. JUnit is part of a unit testing framework family called xUnit. The xUnit family is a collection of unit test frameworks with a common architecture.

TestNG is not part of the xUnit family but influenced by JUnit. Consequently, TestNG added new more powerful test features, which includes unit, integration, end-to-end (E2E), acceptance, and functional testing.

In spite of TestNG containing more features, JUnit is more popular than TestNG. A lot of mainstream IDE's are automatically furnished with JUnit. On the other hand, TestNG must be installed as a plugin to the IDE's. The following illustrates one of the ways to install TestNG into Eclipse IDE:

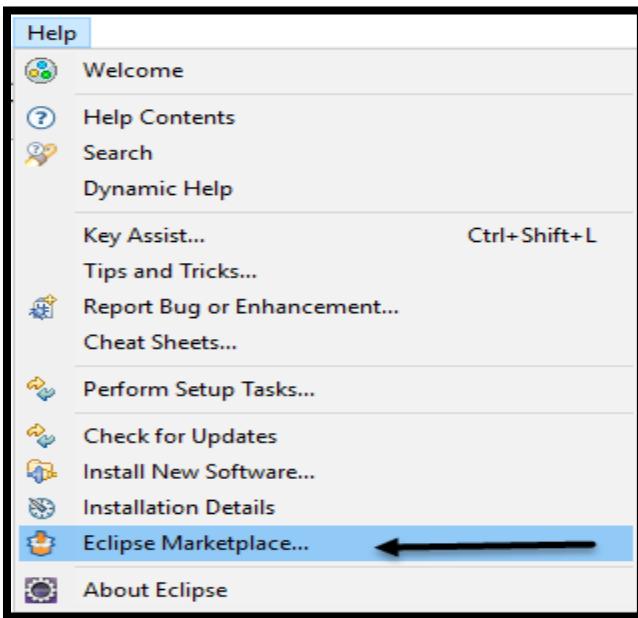
Install TestNG via Marketplace:

1. Open Eclipse
2. Click Help

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

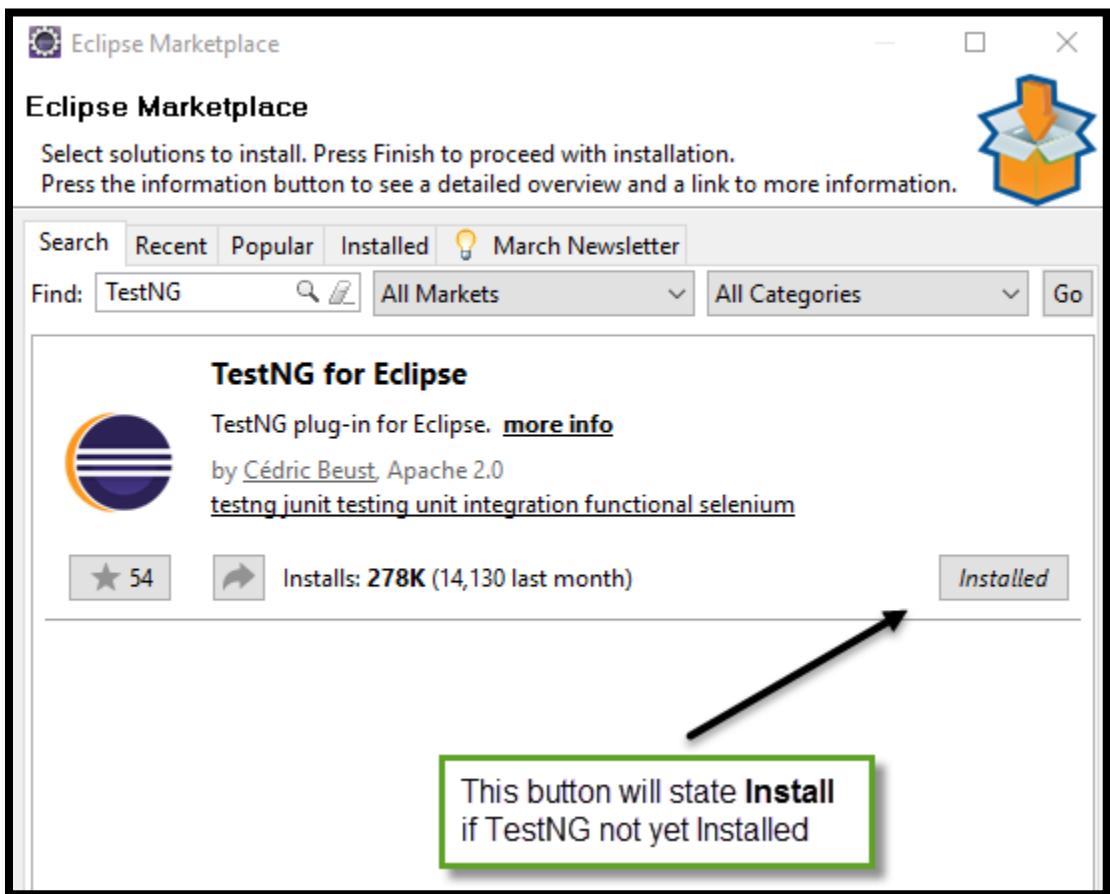
Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

3. Click Eclipse Marketplace...



Skype: rex.jones34
Twitter: @RexJonesII
Email: Rex.Jones@Test4Success.org
LinkedIn: <https://www.linkedin.com/in/rexjones34>

4. Enter TestNG in the Find text field



5. Click Go
6. Click Install

The following are core functions of TestNG and JUnit testing frameworks:

- Create Test Scripts via Selenium WebDriver
- Generate Test Reports

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

- Generate Logs
- Read / Write Data in Excel

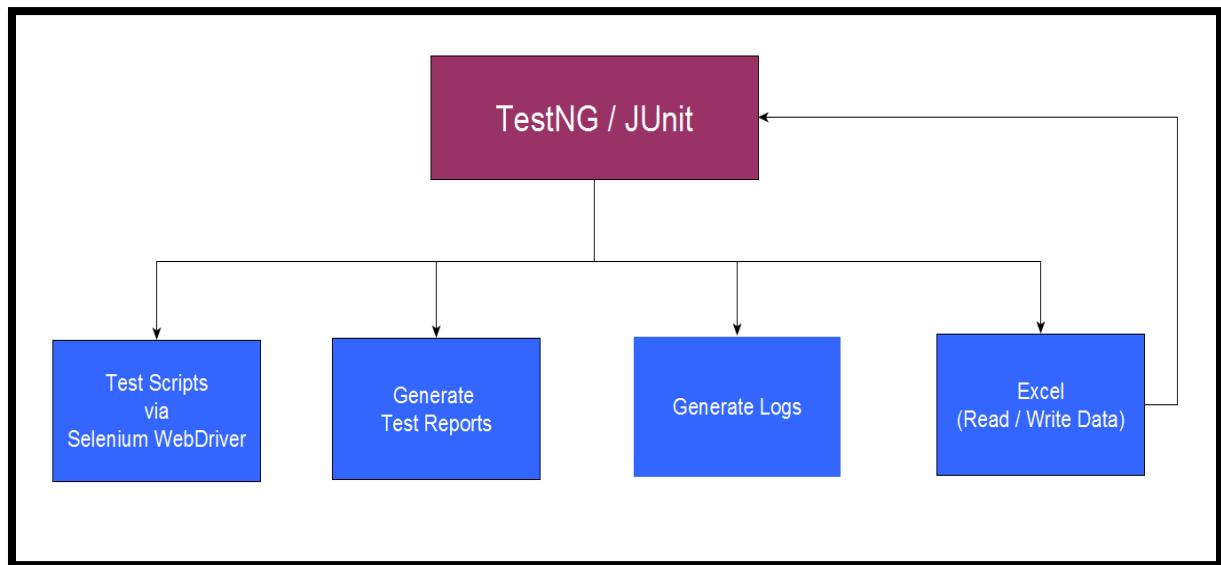


Figure 3.1 – TestNG and JUnit Core Test Functions

Chapter 3 will focus on the following TestNG features although both frameworks perform similar functions:

- ✓ [What Are Annotations](#)
- ✓ [Import Annotations](#)
- ✓ [Create Annotations](#)
- ✓ [Execute Annotations](#)

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

What Are Annotations

Annotations provide metadata to the compiler. Metadata is data that describes data. All annotations start with an at “@” symbol followed by the annotation name. The goal of annotations is to specify the purpose of a [class](#), [method](#), or test. Annotations have the ability to receive parameters like methods. The following is a list of annotations and their descriptions according to [testng.org](#).

- @BeforeSuite – The annotated method will run before all tests in this suite have run.
- @AfterSuite – The annotated method will run after all tests in this suite have run.
- @BeforeTest – The annotated method will run before any test method belonging to the classes inside the <test> tag runs.
- @AfterTest – The annotated method will run after all the test methods belonging to the classes inside the <test> tag runs.
- @BeforeGroups – The list of groups that this configuration method will run before. This method is guaranteed to run before the first test method that belongs to any of these groups is called/invoked.
- @AfterGroups – The list of groups that this configuration method will run after. This method is guaranteed to run after the last test method that belongs to any of these groups is called/invoked.
- @BeforeClass – The annotated method will run before the first test method in the current class is called/invoked.
- @AfterClass – The annotated method will run after all the test methods in the current class runs.
- @BeforeMethod – The annotated method will run before each test method.
- @AfterMethod – The annotated method will run after each test method.
- @DataProvider – Marks a method as providing data for a test method. The annotated method must return an Object[][] where each Object[] can be assigned the parameter

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

list of the test method. The @Test method that wants to receive data from this DataProvider needs to use a dataProvider name equals to the name of this annotation.

- @Factory – Marks a method as a factory that returns objects that will be used by TestNG as Test classes. The method must return Object[].
- @Listeners - Defines listeners on a test class.
- @Parameters - Describes how to pass parameters to a @Test method.
- @Test - Marks a class or a method as part of the test.

Note: Annotations are case-sensitive so the compiler will generate an error if they are not written with the correct syntax.

Import Annotations

Importing a package is a feature that allow a class to access members of another class (see *Chapter 5 in (Part 2) Java 4 Selenium WebDriver*). In the case of annotations, there are packages that contain annotation classes. The annotation classes must be imported for another class to use a specific annotation. An error occurs if the correct class is not imported. The following is an example of several TestNG annotation imports:

```
import org.testng.annotations.Test;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.DataProvider;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeSuite;
import org.testng.annotations.AfterSuite;
```

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

```
3@ import org.testng.annotations.Test;
4 import org.testng.annotations.BeforeMethod;
5 import org.testng.annotations.AfterMethod;
6 import org.testng.annotations.DataProvider;
7 import org.testng.annotations.BeforeClass;
8 import org.testng.annotations.AfterClass;
9 import org.testng.annotations.BeforeTest;
10 import org.testng.annotations.AfterTest;
11 import org.testng.annotations.BeforeSuite;
12 import org.testng.annotations.AfterSuite;
```

Figure 3.2 – TestNG Annotation Import

Note: An automation engineer can use an asterisk (*) or enter the shortcut (Ctrl + Shift + O) to import all classes. The following is an example of using an asterisk to import all required annotation classes.

```
import org.testng.annotations.*;
```

Create Annotations

Annotations are created to identify test methods. As a result, an annotation is placed one line prior to the test method. Annotations have a syntax that must be followed but test methods can be any name. The annotations are selected via New TestNG class modal or entered by the automation engineer.

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

How To Add Annotations via New TestNG Class

In Eclipse, a New TestNG class modal is accessed multiple ways. The following are steps to access the TestNG class modal via project:

1. Right click the project name “e.g., Chapter_3”
2. Select TestNG
3. Select Create TestNG class
4. Select / Enter Source folder, Package name, and Class name
5. Check one or more the annotations
6. Click Finish

Chapter 3 JUnit vs TestNG

(Part 1) Selenium WebDriver

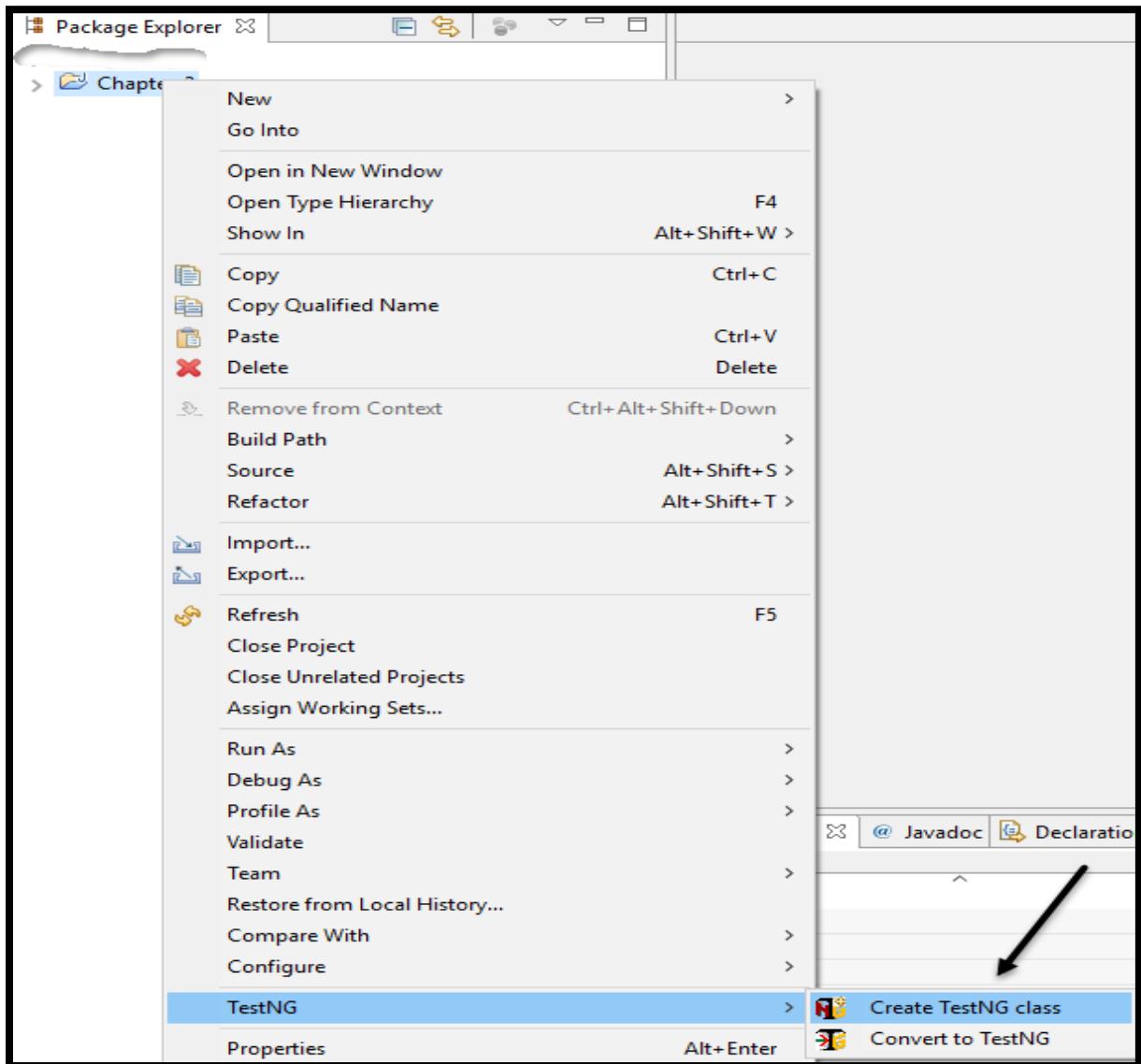


Figure 3.3 – Steps To Access New TestNG Class Modal

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

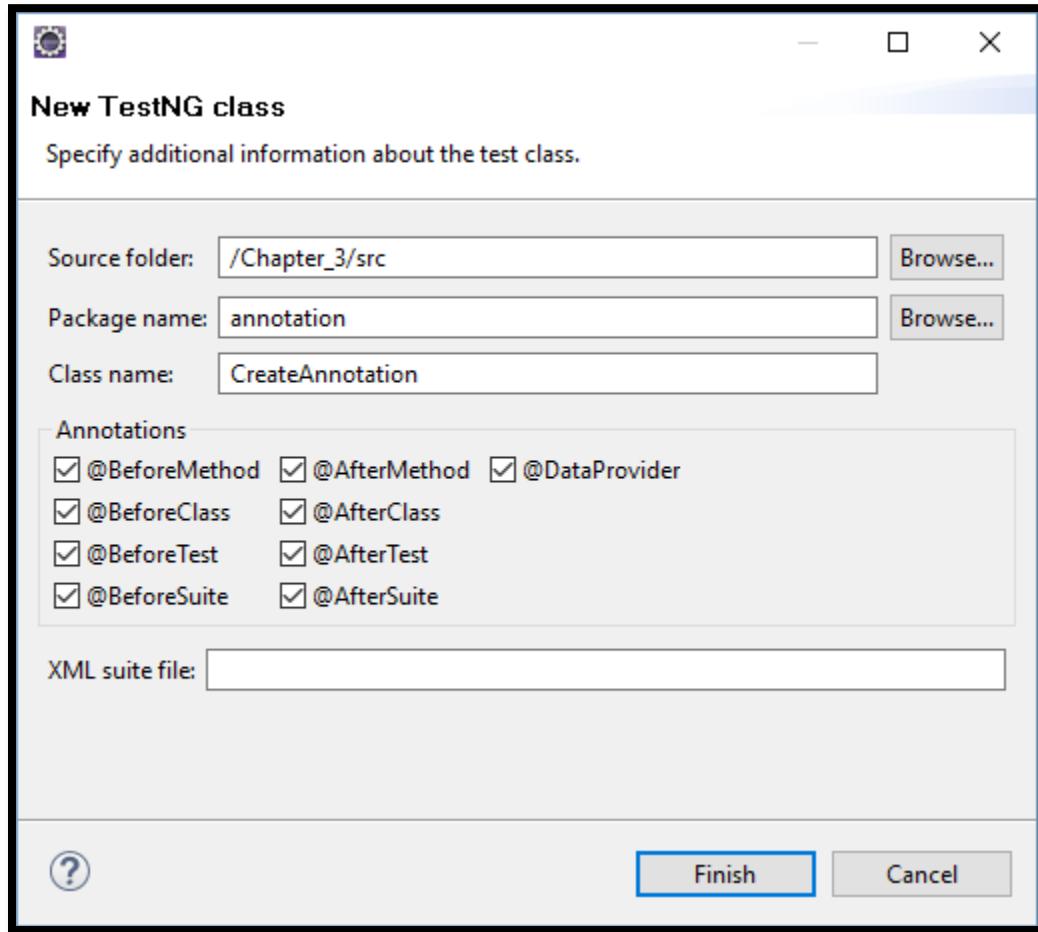


Figure 3.4 – New TestNG Class Modal

Skype: rex.jones34
Twitter: @RexJonesII
Email: Rex.Jones@Test4Success.org
LinkedIn: <https://www.linkedin.com/in/rexjones34>

Note: The annotations and imports are automatically created in the program after clicking the Finish button. The following is an example of annotations after checking the annotations from New TestNG Class modal:

```
package annotation;

import org.testng.annotations.Test;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.DataProvider;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeSuite;
import org.testng.annotations.AfterSuite;

public class CreateAnnotationsAutomatically
{
    @Test(dataProvider = "dp")
    public void f(Integer n, String s) {
    }

    @BeforeMethod
    public void beforeMethod() {
    }

    @AfterMethod
    public void afterMethod() {
    }

    @DataProvider
    public Object[][] dp() {
        return new Object[][] {
            new Object[] { 1, "a" },
            new Object[] { 2, "b" },
        };
    }
}
```

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

```
@BeforeClass
public void beforeClass() {
}

@BeforeClass
public void afterClass() {
}

@BeforeTest
public void beforeTest() {
}

@AfterTest
public void afterTest() {
}

@BeforeSuite
public void beforeSuite() {
}

@AfterSuite
public void afterSuite() {
}

}
```

Skype: rex.jones34
Twitter: @RexJonesII
Email: Rex.Jones@Test4Success.org
LinkedIn: <https://www.linkedin.com/in/rexjones34>

Chapter 3

JUnit vs TestNG

(Part 1) Selenium WebDriver

```

1 package annotation;
2
3+ import org.testng.annotations.Test; ↗
4
5 public class CreateAnnotationsAutomatically
6 {
7     @TestdataProvider = "dp"
8     public void f(Integer n, String s) {
9     }
10
11     @BeforeMethod
12     public void beforeMethod() {
13     }
14
15     @AfterMethod
16     public void afterMethod() {
17     }
18
19     @DataProvider
20     public Object[][] dp() {
21         return new Object[][] {
22             new Object[] { 1, "a" },
23             new Object[] { 2, "b" },
24         };
25     }
26
27     @BeforeClass
28     public void beforeClass() {
29     }
30
31     @AfterClass
32     public void afterClass() {
33     }
34
35     @BeforeTest
36     public void beforeTest() {
37     }
38
39     @AfterTest
40     public void afterTest() {
41     }
42
43     @BeforeSuite
44     public void beforeSuite() {
45     }
46
47     @AfterSuite
48     public void afterSuite() {
49     }
50
51     @BeforeMethod
52     public void beforeMethod() {
53     }
54
55     @AfterMethod
56     public void afterMethod() {
57     }
58
59 }

```

All of the imports except for one has been minimized.
Notice line number shows 3 then 13

Figure 3.5 – Automatically Created Annotations From New TestNG Class Modal

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

Note: Lines 3 – 12 have imports. However, lines 4 – 12 are minimized so all of the imports are not visible. The remaining lines similar to [Figure 3.2 – TestNG Annotation Import](#) are shown by clicking the plus icon (+) next to line 3.

How Automation Engineers Write Annotations

In the previous example, all of the annotations were checked in the New TestNG Class modal. Generally, all annotations are not checked or written by an automation engineer. Only the annotations necessary for testing is utilized. An automation engineer can write any of the annotations from the list of annotations. The following is a Test annotation “@Test” example:

```
package annotation;

public class WriteAnnotations
{
    @Test
    public void testFirstScript ()
    {
        System.out.println("This is the first test script");
    }
}
```

```
1 package annotation;
2
3 public class WriteAnnotations
4 {
5     @Test
6     public void testFirstScript ()
7     {
8         System.out.println("This is the first test script");
9     }
10 }
```

Figure 3.6 - @Test Annotation

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Notice, the error at line 5 for the Test annotation “@Test”. A TestNG library and test annotation import must be added to resolve the error. Eclipse displays several solutions by hovering over the error. The following shows how to add an import.

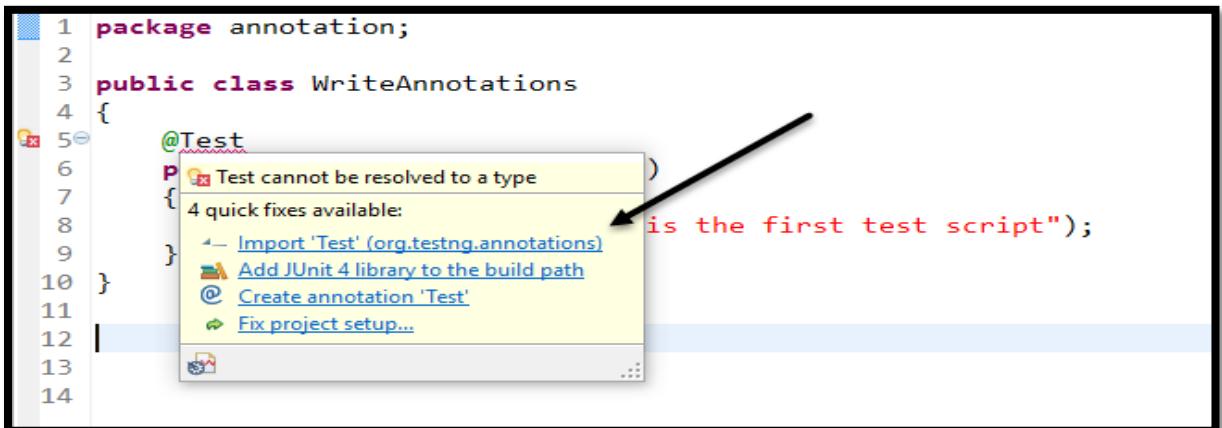


Figure 3.7 – Import Test Annotation

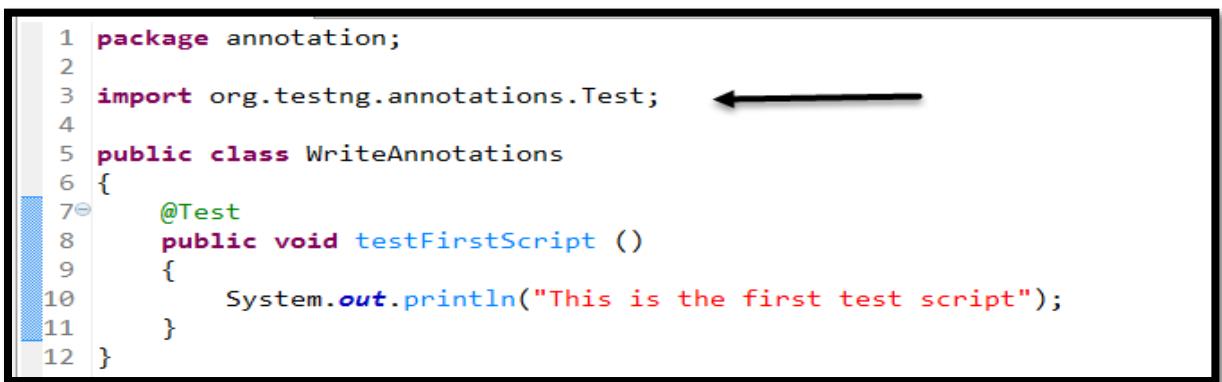


Figure 3.8 – Test Annotation Imported

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

- The option “Import ‘Test’ (org.testng.annotations)” was selected to import the Test annotation
- Line 3 automatically populates the program with an import to resolve the error

Note: In this example, TestNG was added as a library to the Project Explorer before importing the Test annotation. The following is a screenshot of the TestNG library:



Figure 3.9 – TestNG Library Added To The Project Explorer

Execute Annotations

Annotations are executed in a predefined order. The annotations can be located anywhere in the program and still execute according to the predefined order. TestNG contains an in-built mechanism for executing annotations. According to Next Generation Java™ Testing TestNG and Advanced Concepts, every time a method is annotated with one of these annotations, it will run at the following time (page 20):

- @BeforeSuite / @AfterSuite – before a suite starts / after all the test methods in a certain suite have been run
- @BeforeTest / @AfterTest – before a test starts / after all the test methods in a certain test have been run

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

- @BeforeClass / @AfterClass – before a test class starts / after all the test methods in a certain class have been run
- @BeforeMethod / @AfterMethod – before a test method is run / after a test method has been run

The following is an example illustrating an annotation execution order:

```
package annotation;

import org.testng.annotations.*;

public class AnnotationExecutionOrder
{
    @Test // This is a test method
    public void testMethod ()
    {
        System.out.println("@Test - This is a test method");
    }

    @BeforeMethod // Executes before each test
    public void beforeMethod()
    {
        System.out.println("@BeforeMethod - Executes before each test method");
    }

    @AfterMethod
    public void afterMethod()
    {
        System.out.println("@AfterMethod - Executes after each test method");
    }

    @BeforeClass
    public void beforeClass()
    {
        System.out.println("@BeforeClass - Executes first in the class");
    }
}
```

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

```
@AfterClass
public void afterClass()
{
    System.out.println("@AfterClass - Executes last in the class");
}

@BeforeTest
public void beforeTest()
{
    System.out.println("@BeforeTest - Executes before all test methods");
}

@AfterTest
public void afterTest()
{
    System.out.println("@AfterTest - Executes after all test methods");
}

@BeforeSuite
public void beforeSuite()
{
    System.out.println("@BeforeSuite - Executes first in the suite");
}

@AfterSuite
public void afterSuite()
{
    System.out.println("@AfterSuite - Executes last in the suite");
}
```

Skype: rex.jones34
Twitter: @RexJonesII
Email: Rex.Jones@Test4Success.org
LinkedIn: <https://www.linkedin.com/in/rexjones34>

```
7@  @Test // This is a test method
8  public void testMethod ()
9  {
10     System.out.println("@Test - This is a test method");
11 }
12
13@  @BeforeMethod // Executes before each test
14  public void beforeMethod()
15  {
16     System.out.println("@BeforeMethod - Executes before each test method");
17 }
18
19@  @AfterMethod
20  public void afterMethod()
21  {
22     System.out.println("@AfterMethod - Executes after each test method");
23 }
24
25@  @BeforeClass
26  public void beforeClass()
27  {
28     System.out.println("@BeforeClass - Executes first in the class");
29 }
30
31@  @AfterClass
32  public void afterClass()
33  {
34     System.out.println("@AfterClass - Executes last in the class");
35 }
36
37@  @BeforeTest
38  public void beforeTest()
39  {
40     System.out.println("@BeforeTest - Executes before all test methods");
41 }
42
43@  @AfterTest
44  public void afterTest()
45  {
46     System.out.println("@AfterTest - Executes after all test methods");
47 }
48
49@  @BeforeSuite
50  public void beforeSuite()
51  {
52     System.out.println("@BeforeSuite - Executes first in the suite");
53 }
54
55@  @AfterSuite
56  public void afterSuite()
57  {
58     System.out.println("@AfterSuite - Executes last in the suite");
59 }
```

Figure 3.10 – TestNG Annotations And A Single Test Annotation

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

Program Output:

@BeforeSuite - Executes first in the suite
 @BeforeTest - Executes before all test methods
 @BeforeClass - Executes first in the class
 @BeforeMethod - Executes before each test method
 @Test - This is a test method
 @AfterMethod - Executes after each test method
 @AfterClass - Executes last in the class
 @AfterTest - Executes after all test methods
 @AfterSuite - Executes last in the suite

- Lines 7, 13, 19, 25, 31, 37, 43, 49, and 55 contain TestNG annotations
- The program displays the annotations in the following order: @Test, @BeforeMethod, @AfterMethod, @BeforeClass, @AfterClass, @BeforeTest, @AfterTest, @BeforeSuite, @AfterSuite
- The annotations execute in the following predefined order: @BeforeSuite, @BeforeTest, @BeforeClass, @BeforeMethod, @Test, @AfterMethod, @AfterClass, @AfterTest, @AfterSuite

The test annotations “@Test” do not execute in a predefined order if multiple Test annotations are created in a program. The following is an example execution of multiple Test annotations in a single program.

```

package annotation;

import org.testng.annotations.*;

public class ExecuteTestAnnotations
{
  @Test
  public void testInternetExplorer ()
  {
  }
}
  
```

Skype: rex.jones34
 Twitter: @RexJonesII
 Email: Rex.Jones@Test4Success.org
 LinkedIn: <https://www.linkedin.com/in/rexjones34>

```
        System.out.println("Test Script 3 - Cross Browser Testing in
Internet Explorer");
    }

    @Test
    public void testFirefox ()
    {
        System.out.println("Test Script 1 - Cross Browser Testing in
Firefox");
    }

    @Test
    public void testGoogle ()
    {
        System.out.println("Test Script 2 - Cross Browser Testing in
Google");
    }
}
```

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

```
1 package annotation;
2
3 import org.testng.annotations.*;
4
5 public class ExecuteTestAnnotations
6 {
7     @Test
8     public void testInternetExplorer ()
9     {
10         System.out.println("Test Script 3 - Cross Browser Testing in Internet Explorer");
11     }
12
13     @Test
14     public void testFirefox ()
15     {
16         System.out.println("Test Script 1 - Cross Browser Testing in Firefox");
17     }
18
19     @Test
20     public void testGoogle ()
21     {
22         System.out.println("Test Script 2 - Cross Browser Testing in Google");
23     }
24 }
```

Figure 3.11 – Test Annotations

Program Output:

Test Script 1 - Cross Browser Testing in Firefox

Test Script 2 - Cross Browser Testing in Google

Test Script 3 - Cross Browser Testing in Internet Explorer

- Lines 7, 13, and 19 display multiple Test annotations “@Test” which identify test methods at lines 8, 14, and 20
- The program displays the test methods in the following order: testInternetExplorer, testFirefox, and testGoogle
- The test methods execute in the following order: testFirefox, testGoogle, then testInternetExplorer

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Note: The test methods were executed in alphabetical order according to method name. However, it is best to pass a priority parameter to each test annotation. The following is an example using a priority parameter to specify an execution order:

```
package annotation;

import org.testng.annotations.*;

public class OrderTestAnnotations
{
    @Test (priority = 2)
    public void testInternetExplorer ()
    {
        System.out.println("Test Script 3 - Cross Browser Testing in
Internet Explorer");
    }

    @Test (priority = 3)
    public void testFirefox ()
    {
        System.out.println("Test Script 1 - Cross Browser Testing in
Firefox");
    }

    @Test (priority = 1)
    public void testGoogle ()
    {
        System.out.println("Test Script 2 - Cross Browser Testing in
Google");
    }
}
```

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

Figure 3.12 – Test Annotation With Priority Parameter

Program Output:

Test Script 2 - Cross Browser Testing in Google
Test Script 3 - Cross Browser Testing in Internet Explorer
Test Script 1 - Cross Browser Testing in Firefox

- Lines 7, 13, and 19 include a priority parameter within the Test annotations. The priority parameter sets the precedence for each test method on lines 8, 14, and 20.

- Execution order for the test methods are testGoogle (line 20), testInternetExplorer (line 8), then testFirefox (line 14). As a result, the Program Output displays Test Script 2, Test Script 3, and Test Script 1

The following is an example displaying the execution order for multiple TestNG annotations:

```
package annotation;

import org.testng.annotations.*;

public class ExecuteMostAnnotations
{
    @BeforeMethod // Execute before each test (i.e., @Test)
    public void openBrowser()
    {
        System.out.println("Open Browser");
    }

    @AfterMethod // Execute after each test (i.e., @Test)
    public void closeBrowser()
    {
        System.out.println("Close Browser \n");
    }

    @BeforeTest // Before all test scripts are executed
    public void connectDataBase()
    {
        System.out.println("Connect to a Database \n");
    }

    @AfterTest // After all test scripts are executed
    public void disconnectDataBase()
    {
        System.out.println("Disconnect from the Database");
    }

    @Test (priority = 1)
    public void testFirefox ()
}
```

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

```
{  
    System.out.println("Test Script 1 - Cross Browser Testing in  
Firefox");  
}  
  
@Test (priority = 2)  
public void testGoogle ()  
{  
    System.out.println("Test Script 2 - Cross Browser Testing in  
Google");  
}  
  
@Test (priority = 3)  
public void testInternetExplorer ()  
{  
    System.out.println("Test Script 3 - Cross Browser Testing in  
Internet Explorer");  
}  
}
```

Chapter 3

JUnit vs TestNG

(Part 1) Selenium WebDriver

```

1 package annotation;
2
3 import org.testng.annotations.*;
4
5 public class ExecuteMostAnnotations
6 {
7     @BeforeMethod // Execute before each test (i.e., @Test)
8     public void openBrowser()
9     {
10         System.out.println("Open Browser");
11     }
12
13     @AfterMethod // Execute after each test (i.e., @Test)
14     public void closeBrowser()
15     {
16         System.out.println("Close Browser \n");
17     }
18
19     @BeforeTest // Before all test scripts are executed
20     public void connectDataBase()
21     {
22         System.out.println("Connect to a Database \n");
23     }
24
25     @AfterTest // After all test scripts are executed
26     public void disconnectDataBase()
27     {
28         System.out.println("Disconnect from the Database");
29     }
30
31     @Test (priority = 1)
32     public void testFirefox ()
33     {
34         System.out.println("Test Script 1 - Cross Browser Testing in Firefox");
35     }
36
37     @Test (priority = 2)
38     public void testGoogle ()
39     {
40         System.out.println("Test Script 2 - Cross Browser Testing in Google");
41     }
42
43     @Test (priority = 3)
44     public void testInternetExplorer ()
45     {
46         System.out.println("Test Script 3 - Cross Browser Testing in Internet Explorer");
47     }
48 }
```

Figure 3.13 –TestNG Annotations And Multiple Test Annotations

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

Program Output:

Connect to a Database

Open Browser

Test Script 1 - Cross Browser Testing in Firefox

Close Browser

Open Browser

Test Script 2 - Cross Browser Testing in Google

Close Browser

Open Browser

Test Script 3 - Cross Browser Testing in Internet Explorer

Close Browser

Disconnect from the Database

- Lines 7, 13, 19, 25, 31, 37, and 43 contain different TestNG annotations. Lines 31, 37, and 43 are the test annotations
- The annotations execute in the following order:
 - **Line 19**
@BeforeTest
 - **Line 7, Line 31, Line 13**
@BeforeMethod, @Test – testFirefox, @AfterMethod
 - **Line 7, Line 37, Line 13**
@BeforeMethod, @Test – testGoogle, @AfterMethod
 - **Line 7, Line 43, Line 13**
@BeforeMethod, @Test – testInternetExplorer, @AfterMethod
 - **Line 25**
@AfterTest

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Note: Some additional powerful features of TestNG include the following:

- Automatically identify and run JUnit test
- Generate reports in HTML and XML
- Trace code that handles exceptions
- Group test methods and specify groups that contain other groups

Chapter 3 discussed JUnit and TestNG frameworks. JUnit is part of the xUnit family, which is a collection of unit test frameworks with a common architecture. TestNG and JUnit are similar due to JUnit influencing TestNG. Chapter 4 introduces Selenium IDE, which is a good automation tool for creating Selenium's first Test Script.

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

Chapter 4

First Selenium Test Script

Most books and videos create the first Selenium test script via Selenium IDE. A Selenium test script is a set of instructions executed on the Application Under Test (AUT). Selenium IDE is a record and replay automation tool integrated within the Firefox browser. The tool generates Selenium test scripts by extracting locators from the web application. Locators help find and match WebElements. WebElements are buttons, radio buttons, text boxes, checkboxes, drop down menus, and hyperlinks (also known as links) located on a web application.

The record and replay feature is a good introduction for testers starting to learn automation testing. However, Selenium IDE has limitations such as cross browser testing. Therefore, it cannot test across multiple browsers (i.e., Google Chrome, Internet Explorer, etc.) because it is a Firefox extension.

Although Selenium IDE is limited, the record and replay feature is the main reason for discussing Selenium's first test script. The record feature captures a user's interaction with the AUT then provides an ability to playback the interactions. For example, Selenium IDE can record then playback the following interactions (also known as steps):

1. Enter User Name
2. Enter Password
3. Click Login

Chapter 4 will cover the following regarding Selenium's First Test Script:

- ✓ [How To Access Selenium IDE](#)
- ✓ [Selenium IDE User Interface](#)

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

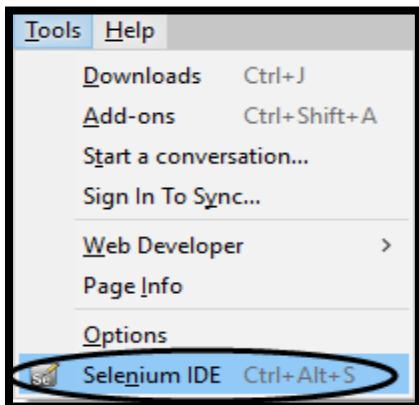
- ✓ [Record Selenium's First Test Script](#)
- ✓ [Save and Playback Selenium's First Test Script](#)
- ✓ [Export From Selenium IDE To Selenium WebDriver](#)
- ✓ [Locate WebElements via HTML](#)

How To Access Selenium IDE

Selenium IDE must be installed as a Firefox extension before it can be accessed. Steps for installing Selenium IDE are located in [Chapter 1 – Install Selenium IDE](#) section. The following are steps to access Selenium IDE:

Steps To Access Selenium IDE: (Shortcut = Ctrl + Alt + S)

1. Open the Firefox browser
2. Click Tools
3. Click Selenium IDE



Result: The Selenium IDE automation tool opens.

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

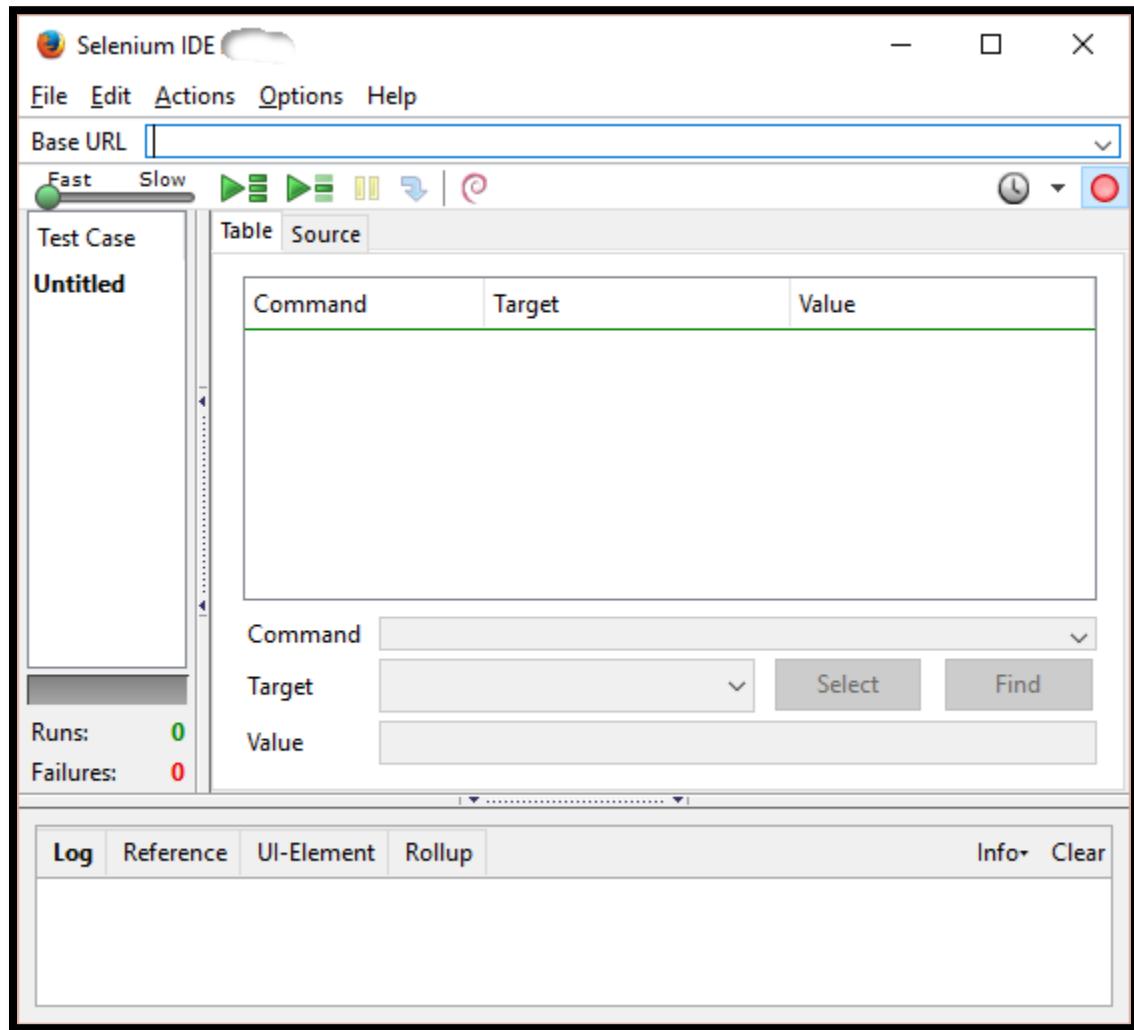


Figure 4.1 – Selenium IDE Screenshot

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Selenium IDE User Interface

The Selenium IDE testing tool comes equipped with a Menu Bar and Tool Bar. Both bars form the User Interface (UI), which makes Selenium IDE more than a record and replay tool. The Menu Bar consist of a File menu, Edit menu, Actions menu, Options menus, and Help menu.



Figure 4.2 – Selenium IDE Menu Bar Screenshot

- **File menu** - allows a Test Script and Test Suite to be added, opened, saved, and exported
- **Edit menu** - allows Test Script commands to be copied, pasted, and deleted
- **Actions menu** - allows a Test Script to be recorded, played, and debugged
- **Options menu** - allows a change in the Selenium IDE settings
- **Help menu** - contains information about Selenium's UI/Elements, website, and blog

The following is a screenshot of Selenium IDE Tool Bar:



Figure 4.3 – Selenium IDE Tool Bar Screenshot

- **Speed Control** – controls how slow/fast the test script runs
- **Play Entire Test Suite** – plays the entire Test Suite when multiple Test Scripts are loaded

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

- **Play Current Test Script** – plays the current loaded Test Script
- **Pause/Resume Test Script** – allows the loaded Test Script to pause and resume
- **Step Through The Test Script** – allows a Test Script to get debugged one command at a time
- **Apply Rollup Rules** – allows one or more Selenium commands to be grouped into one command
- **Turn Test Scheduler On/Off** – allows a Test Suite to be scheduled
- **Record Button** – records the user's interactions with the web browser

Note: There is a difference between a Test Script and Test Case although used interchangeably. Selenium IDE mislabels a Test Script by calling it a Test Case. In reality, a Test Script refers to steps from an automation perspective while Test Case refers to steps from a manual testing perspective.

The following is a list of Selenium IDE features that is not visible on the Menu Bar or Tool Bar:

- Add comments to a test script
- Store information in variables
- Work with multiple windows such as pop up windows
- Work with AJAX applications
- Parameterize the test scripts
- Prioritize the locators

Record Selenium's First Test Script

Most first time automation engineers utilize the record feature in Selenium IDE. After recording the interactions, a set of Test Scripts is hardcoded into Selenium IDE. Commands, targets, and values form the Test Scripts. Command shows the actions performed by the user.

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Target shows the element that received an action from the user. Value is the data entered by the user.

A web application called “[Welcome to the Internet](#)” powered by [Elemental Selenium](#) will be used to record Selenium’s first Test Script. Dave Haeffner is the writer of [Elemental Selenium](#) which provides a free, once a week email on how to use Selenium like a Pro. The following is a set of manual Test Case steps simulating a user logging into a web application.

Test Case Steps:

1. Open Firefox
2. [Access Selenium IDE](#)
3. Click the red button if recording is not turned on by default
4. Enter URL for the website “<http://the-internet.herokuapp.com/>”
5. Click the Form Authentication link
6. Enter Username “tomsmith”
7. Enter Password “SuperSecretPassword!”
8. Click the Login button

The following screenshot shows the recorded Test Scripts in Selenium IDE:

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

Command	Target	Value
open	/	
clickAndWait	link=Form Authentication	
type	id=username	tomsmith
type	id=password	SuperSecretPassword!
clickAndWait	css=button.radius	

Figure 4.4 – Selenium IDE Recorded Test Script

All of the manual Test Case steps (4 - 8) recorded as a single Test Script. The following explains Base URL along with columns Command, Target, and Value:

- **Base URL:** web application loaded by the user
- **Command:** represents the action performed by the user
 - **open** – opens the Base URL
 - **clickAndWait** – clicks on a hyperlink, button, checkbox, or radio button
 - **type** – sets the value of a text field (also known as input field)
- **Target:** represents the element that received an action from the user
 - **link=Form Authentication** – element Form Authentication received an action from the user
 - **id=username** – element Username received an action from the user
 - **id=password** – element Password received an action from the user
 - **css=button.radius** - element Login received an action from the user

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Note: The attributes (link and id) and attribute value (radius) are derived from HTML and will be explained in [Locate WebElements via HTML](#).

- **Value:** represents the data entered by the user
 - **tomsmith** – data entered in the Username text field
 - **SuperSecretPassword!** – data entered in the Password text field

The following screenshots illustrate loading the website, clicking the Form Authentication link, entering the Username, entering the Password, and clicking the Login button:

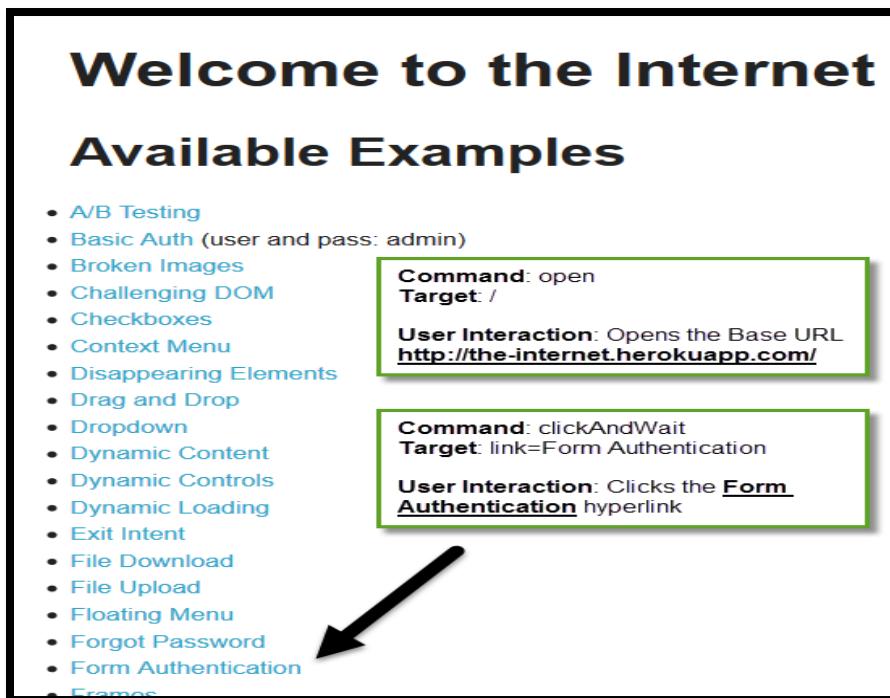


Figure 4.5 – Clicks The Form Authentication Link

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

Login Page

This is where you can log into the secure area. Enter **tomsmith** for the username and **SuperSecretPassword!** for the password. If the information is wrong you should see error messages.

Username
tomsmith

Password
|

Login

Command: type
Target: id=username
Value: tomsmith

User Interaction: Enter tomsmith in the Username text field

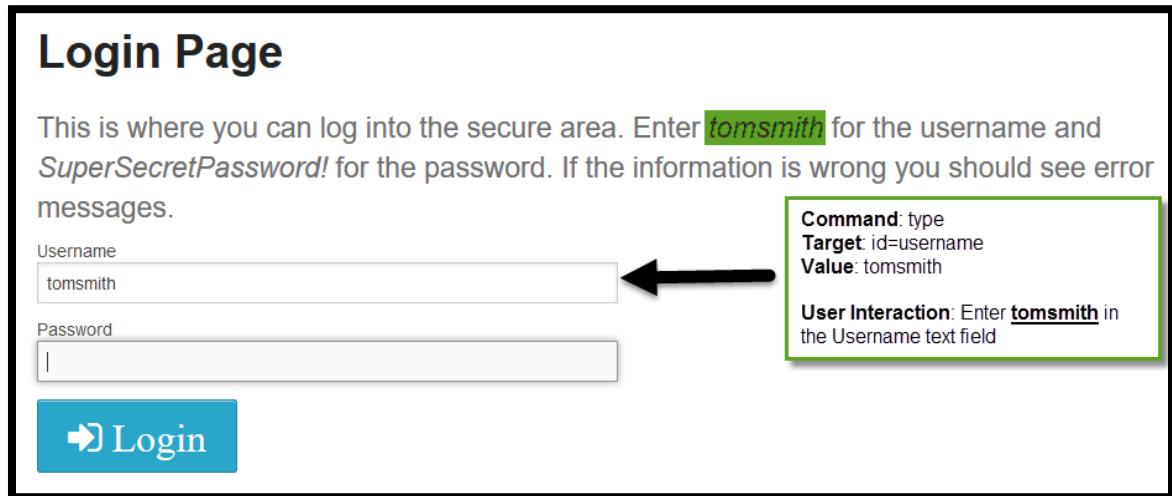


Figure 4.6 – Enter Username In The Text Field

Login Page

This is where you can log into the secure area. Enter **tomsmith** for the username and **SuperSecretPassword!** for the password. If the information is wrong you should see error messages.

Username
tomsmith

Password

Login

Command: type
Target: id=password
Value: SuperSecretPassword!

User Interaction: Enter SuperSecretPassword! in the Password text field

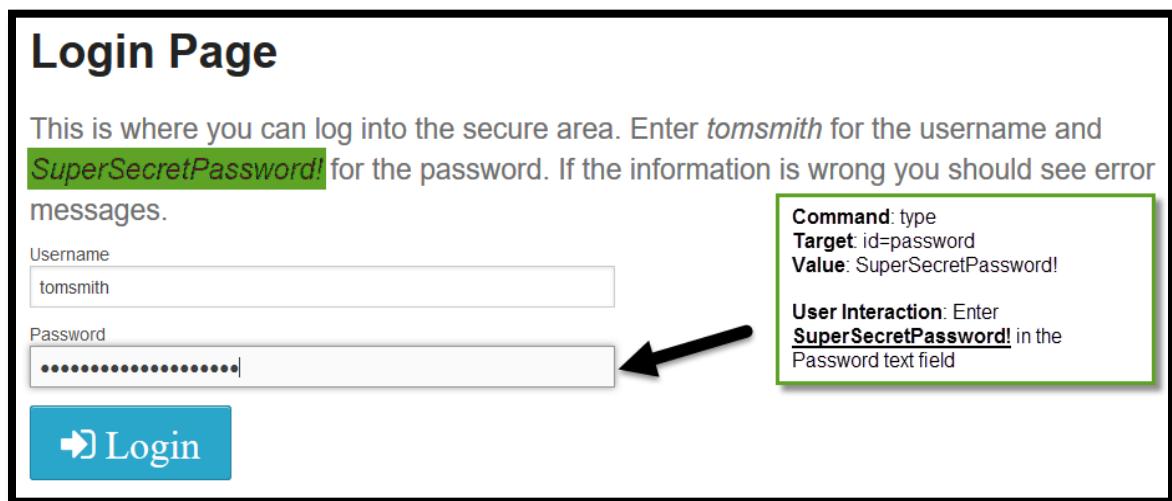


Figure 4.7 – Enter Password In The Text Field

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

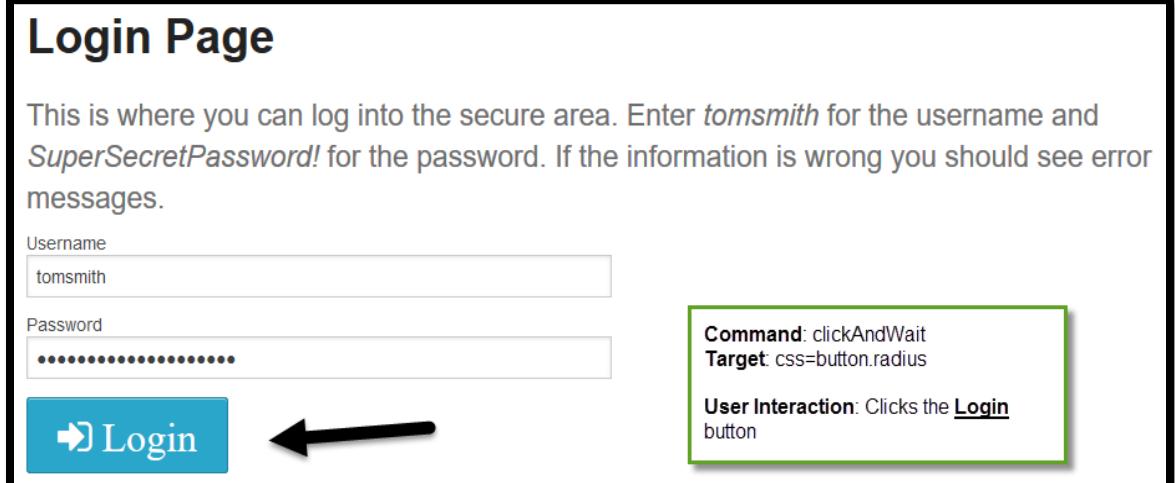


Figure 4.8 – Clicks The Login Button

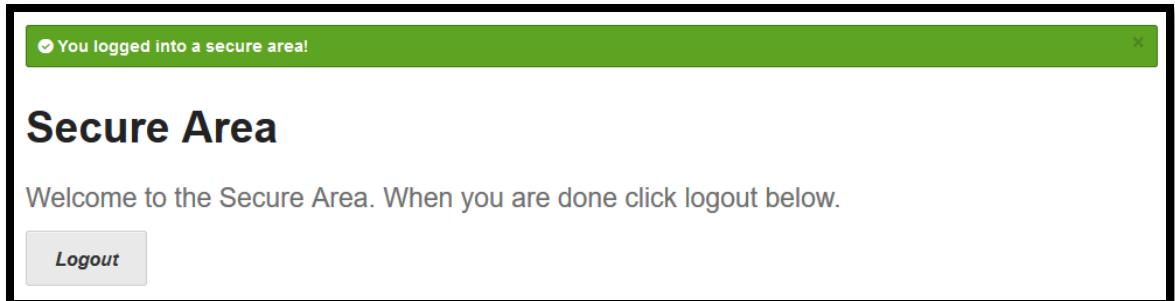


Figure 4.9 – Successful Login

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

Save and Playback Selenium's First Test Script

All Test Scripts in Selenium IDE can be saved and played back. It is best to save the Test Script before replaying the Test Script. Replaying a Test Script allows the user to see every recorded action.

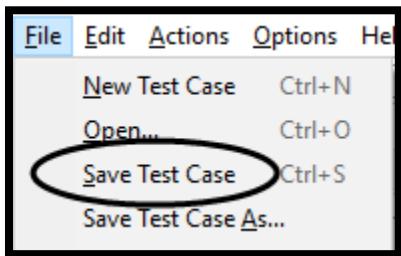
As mentioned in “[Record Selenium's First Test Script](#)”, the Test Scripts are formed by commands, targets, and values. However, their terminology is different when recording the Test Script and replaying the Test Script. Recording the Test Script represents actions performed by the user. Replayng the Test Script are instructions to Selenium IDE. The following is terminology for command, target, and value while replaying a Test Script:

- **Command:** instructs Selenium IDE what action to perform
- **Target:** instructs Selenium IDE what element to locate
- **Value:** instructs Selenium IDE what data to enter

The following are steps to save and playback an automation Test Script:

Steps To Save The Test Script:

1. Click File from the Menu Bar
2. Click Save Test Case



3. Name the Test Script “i.e., Enter Username – Password (Form Authentication)”

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Steps To Playback The Test Script

1. Click the icon for Play Entire Test Suite or Play Current Test Case



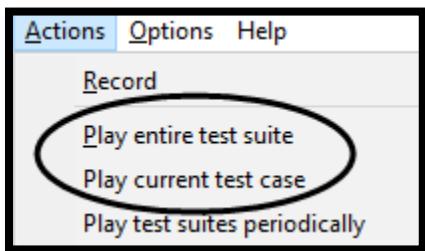
or



or

Click Actions from the Menu Bar

2. Select Play Entire Test Suite or Play Current Test Case



Note: Playing the entire Test Suite is suitable for replaying one Test Script or multiple Test Scripts.

Selenium IDE stores information (Commands, Targets, Values) about WebElements in Test Scripts while recording. Then uses the information to identify each WebElement for replaying the Test Scripts. For example, when the user enters a value in the Username text field, enters a value in the Password text field, and clicks the Login button. Selenium IDE stores the Commands, Targets, and Values then uses the information to replay the Test Script. Therefore, Selenium IDE will enter the same value in the text fields and click the Login button.

Replaying a Test Script offers useful information in the bottom pane of Selenium IDE. The bottom pane includes four tabs (Log, Reference, UI-Element, and Rollup). Two of the tabs (UI-Element and Rollup) are advanced features that are not relevant for this chapter. However, more information about both tabs is covered in the UI-Element Documentation

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

located in Selenium's Help menu. Log reveals all steps and whether the Test Script Passed or Failed. Reference allows an automation engineer to investigate a specific step in the Test Script by displaying information regarding the command. The following are Log and Reference screenshots covering [Figure 4.4.](#)

The screenshot shows the Selenium IDE interface with the 'Log' tab selected. The log window displays the following sequence of events:

```
[info] Playing test case Enter Username - Password (Form Authentication)
[info] Executing: |open| / ||
[info] Executing: |clickAndWait| link=Form Authentication ||
[info] Executing: |type| id=username | tomsmith |
[info] Executing: |type| id=password | SuperSecretPassword! |
[info] Executing: |clickAndWait| css=button.radius ||
[info] Test case passed
```

A large black arrow points from the left towards the final line of the log, which is 'Test case passed'.

Figure 4.10 – Log Tab

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Chapter 4

First Selenium Test Script

(Part 1) Selenium WebDriver

The screenshot shows the Selenium IDE interface with the 'Reference' tab selected. At the top, there is a table of recorded commands:

Command	Target	Value
open	/	
clickAndWait	link=Form Authentication	
type	id=username	tomsmith
type	id=password	SuperSecretPassword!
clickAndWait	css=button.radius	

Below this, a specific command is expanded:

Command	type
Target	id=password
Value	SuperSecretPassword!

At the bottom of the expanded view, there is a detailed description of the 'type(locator, value)' command:

type(locator, value)

Arguments:

- locator - an element locator
- value - the value to type

Sets the value of an input field, as though you typed it in.

Can also be used to set the value of combo boxes, check boxes, etc. In these cases, value should be the value of the option selected, not the visible text.

Figure 4.11 – Reference Tab

Export From Selenium IDE To Selenium WebDriver

After recording a Test Script, the Test Script can be exported from Selenium IDE to Selenium WebDriver in different formats. Selenium IDE provides bindings for the available programming languages in Selenium WebDriver. The available languages are C# (pronounced C Sharp), Java, Python, and Ruby.

In addition, the testing framework such as JUnit and TestNG are exported along with the programming language. This section exports code for Java / TestNG / WebDriver since Java, TestNG, and WebDriver are the book's focal point. The following are steps for exporting code from Selenium IDE to Selenium WebDriver:

Steps For Exporting Code From Selenium IDE:

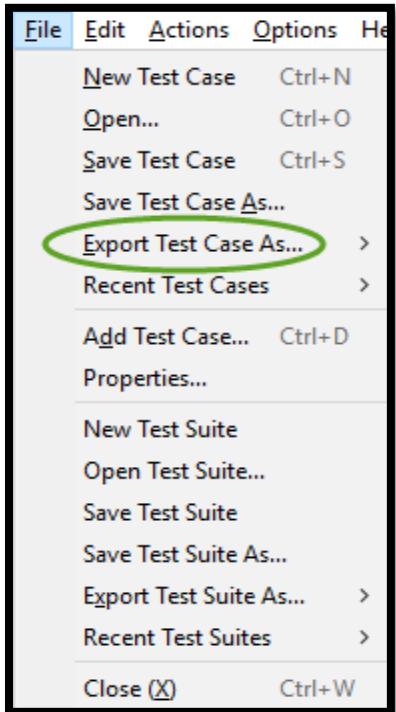
3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

1. Click File from the Menu Bar
2. Select Export Test Case As



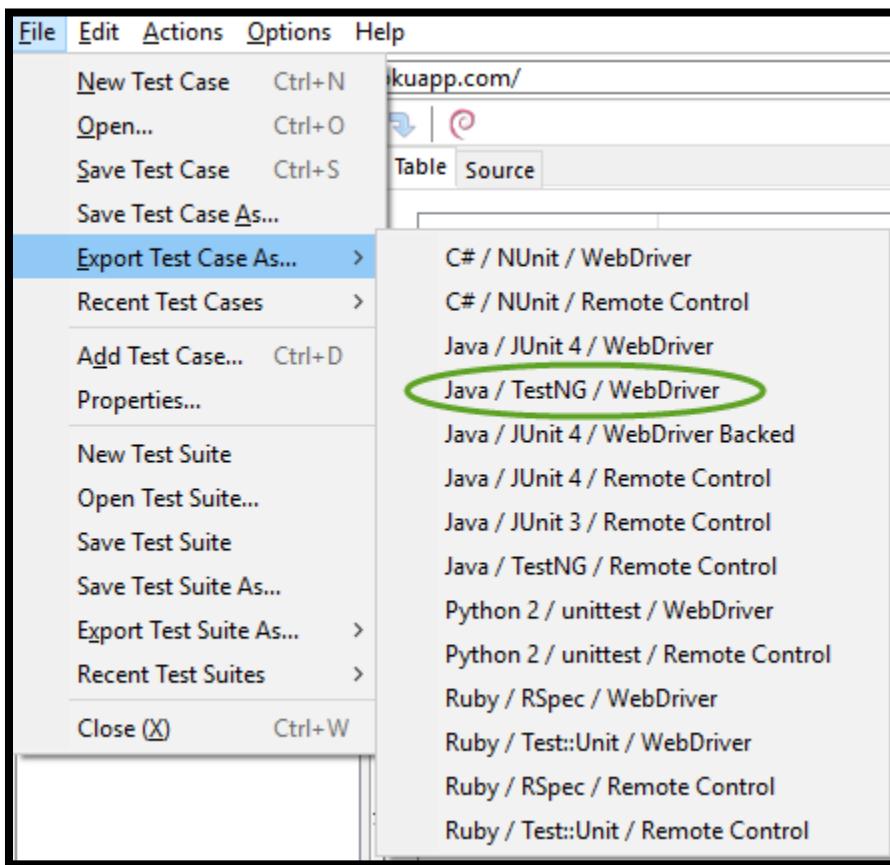
Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

3. Select Java / TestNG / WebDriver



4. Save the file “i.e., **First Selenium Test Script**”. The file will be saved with a .java extension “i.e., **First Selenium Test Script.java**”
5. Open the file with a text editor such as Notepad ++. [Chapter 1 – Download Notepad ++](#) provides information for downloading Notepad ++:

Result: The Test Script is available in Java / TestNG / WebDriver format.

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

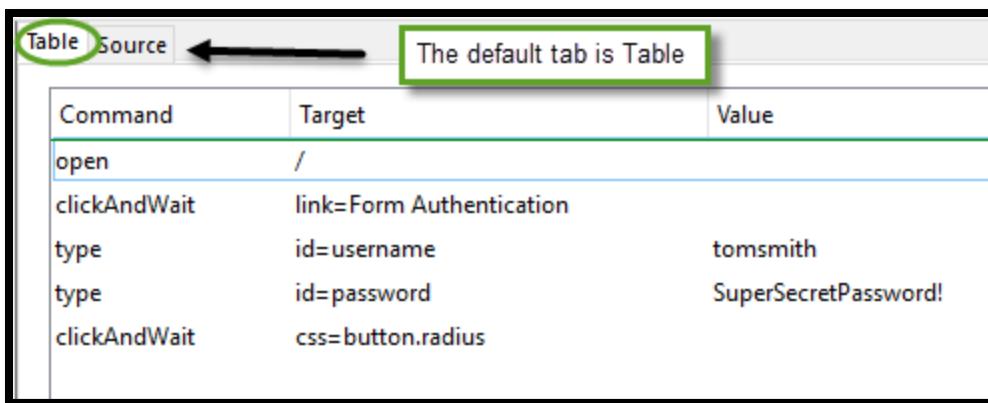
Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

Selenium IDE has a Source feature similar to the Export feature that reveals the code in any language and test framework. Therefore, the Java / TestNG / WebDriver format can be seen without Notepad ++. However, the difference between the Source feature and Notepad ++ is the code's color. Notepad ++ displays some of the code with colors while the Source feature only displays black and white colors. The following are steps for utilizing the Source feature in Selenium IDE:

Steps For Viewing Code Within Selenium IDE:

1. Select the Source tab (The Table tab is selected by default)



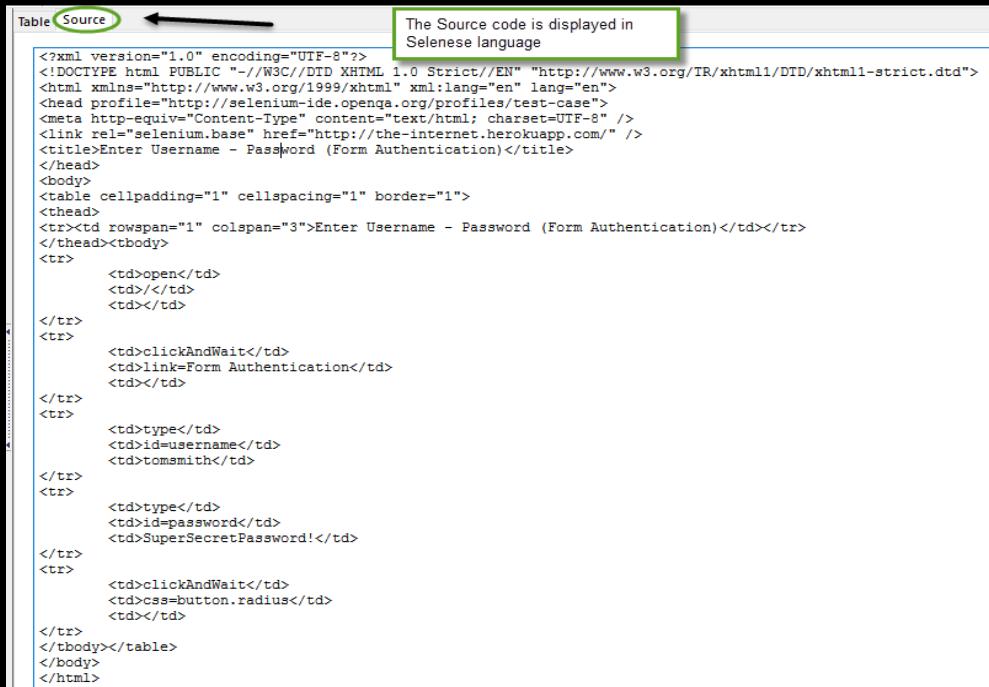
Command	Target	Value
open	/	
clickAndWait	link=Form Authentication	
type	id=username	tomsmith
type	id=password	SuperSecretPassword!
clickAndWait	css=button.radius	

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>



The Source code is displayed in Selenese language

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head profile="http://selenium-ide.openqa.org/profiles/test-case">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="selenium.base" href="http://the-internet.herokuapp.com/" />
<title>Enter Username - Password (Form Authentication)</title>
</head>
<body>
<table cellpadding="1" cellspacing="1" border="1">
<thead>
<tr><td rowspan="3" colspan="3">Enter Username - Password (Form Authentication)</td></tr>
</thead><tbody>
<tr>
<td>open</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clickAndWait</td>
<td>link=Form Authentication</td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>id=username</td>
<td>tomsmith</td>
</tr>
<tr>
<td>type</td>
<td>id=password</td>
<td>SuperSecretPassword!</td>
</tr>
<tr>
<td>clickAndWait</td>
<td>css=button.radius</td>
<td></td>
</tr>
</tbody></table>
</body>
</html>

```

2. Click Options from the Menu Bar
3. Check the Enable experimental features checkbox
4. Click OK
5. Click Options from the Menu Bar
6. Click Format

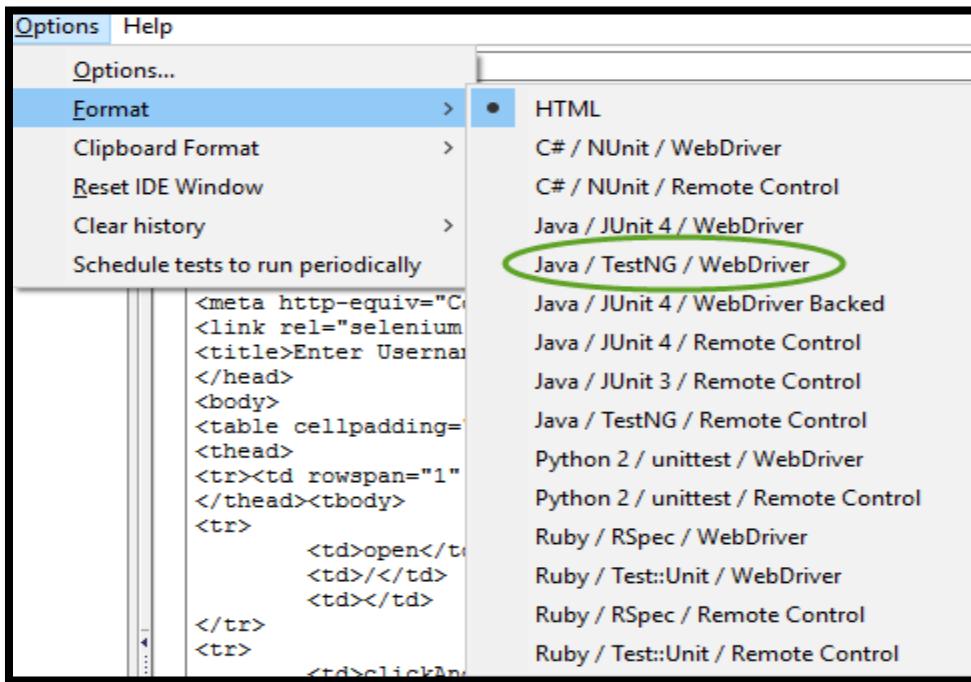
3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

7. Select Java / TestNG / WebDriver



8. Click OK on the JavaScript Application modal

Result: The Test Script is available in Java / TestNG / WebDriver format.

After viewing the code, it can be copied from the Source tab or text editor “i.e., Notepad ++”, pasted into Eclipse, then executed like all Selenium WebDriver Test Scripts. The following is a portion of the Test Script from Eclipse.

```
public class FirstSeleniumTestScript {
    private WebDriver driver;
    private String baseUrl;
    private boolean acceptNextAlert = true;
    private StringBuffer verificationErrors = new StringBuffer();
```

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

```
@BeforeClass(alwaysRun = true)
public void setUp() throws Exception {
    driver = new FirefoxDriver();
    baseUrl = "http://the-internet.herokuapp.com/";
    driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
}

@Test
public void testFirstSeleniumTestScript() throws Exception {
    driver.get(baseUrl + "/");
    driver.findElement(By.linkText("Form Authentication")).click();
    driver.findElement(By.id("username")).clear();
    driver.findElement(By.id("username")).sendKeys("tomsmith");
    driver.findElement(By.id("password")).clear();
    driver.findElement(By.id("password")).sendKeys("SuperSecretPassword!");
    driver.findElement(By.cssSelector("button.radius")).click();
}
```

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

```

10 public class FirstSeleniumTestScript {
11     private WebDriver driver;
12     private String baseUrl;
13     private boolean acceptNextAlert = true;
14     private StringBuffer verificationErrors = new StringBuffer();
15
16     @BeforeClass(alwaysRun = true)
17     public void setUp() throws Exception {
18         driver = new FirefoxDriver();
19         baseUrl = "http://the-internet.herokuapp.com/";
20         driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
21     }
22
23     @Test
24     public void testFirstSeleniumTestScript() throws Exception {
25         driver.get(baseUrl + "/");
26         driver.findElement(By.linkText("Form Authentication")).click();
27         driver.findElement(By.id("username")).clear();
28         driver.findElement(By.id("username")).sendKeys("tomsmith");
29         driver.findElement(By.id("password")).clear();
30         driver.findElement(By.id("password")).sendKeys("SuperSecretPassword!");
31         driver.findElement(By.cssSelector("button.radius")).click();
32     }

```

Class Name = FirstSeleniumTestScript

The @Test annotation performs all of the Test Script steps

Figure 4.12 – First Selenium Test Script

- Line 10 displays the Class Name “FirstSeleniumTestScript” which is the same name as the saved file “First Selenium Test Script” from Selenium IDE
- Line 23 “@Test” identifies the method as a test method and performs all of the Test Script steps (lines 25 - 31)
- Line 25 gets the baseUrl, which is <http://the-internet.herokuapp.com/>. A new page loads “Welcome to the Internet” in the browser window
- Line 26 clicks the Form Authentication hyperlink
- Line 28 types the Username “tomsmith” into the text field
- Line 30 types the Password “SuperSecretPassword!” into the text field
- Line 31 clicks the Login button

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.orgLinkedIn: <https://www.linkedin.com/in/rexjones34>

Note: The [WebDriver Methods](#) “i.e., `findElement`”, [Locator Types](#) “i.e., `id`”, and [methods that perform an action](#) “i.e., `click`” are explained in [Chapter 5 – WebDriver and WebElements](#).

Locate WebElements via HTML

All web applications and browsers contain WebElements “i.e., buttons, links, etc.” The WebElements are derived from HTML (Hyper Text Markup Language). HTML is the standard markup language for creating web applications. Furthermore, each browser reads an HTML document to display the browser pages.

A great deal of HTML markup tags includes an opening (also known as starting) tag, a closing (also known as ending) tag, and at least one attribute. In regards to web applications and browsers, the markup tags determine the characteristics of a WebElement. Attributes provide additional information about the WebElement. It is important to know that attributes are defined in the opening tag.

Both tags have the same name surrounded by angle brackets (`<>`). However, the difference between an opening and closing tag is the forward slash (`/`). Closing tags must include a forward slash before the tag name. Unlike the tags, an attribute contains a name/value pair. The following is the syntax for an opening tag, closing tag, and attribute name/value pair.

Syntax

`<Opening AttributeName="AttributeValue">`

Content placed in between the tags

`</Closing>`

Note: Some elements do not contain a closing tag.

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

Recall three of the Targets (Username, Password, Login) from [Figure 4.4 - Selenium IDE Recorded Test Script](#) displayed the following information:

- id=username
 - id is the AttributeName
 - username is theAttributeValue
- id=password
 - id is the AttributeName
 - password is theAttributeValue
- css=button.radius
 - css is the Locator Type (see [Chapter 9 – Find WebElement by CSS Selector](#))
 - button is the opening/closing tag name
 - radius is theAttributeValue

The following screenshot provides an illustration of TagNames, AttributeNames, and AttributeValues using HTML's source code for Username, Password, and Login:

Skype: rex.jones34
Twitter: @RexJonesII
Email: Rex.Jones@Test4Success.org
LinkedIn: <https://www.linkedin.com/in/rexjones34>

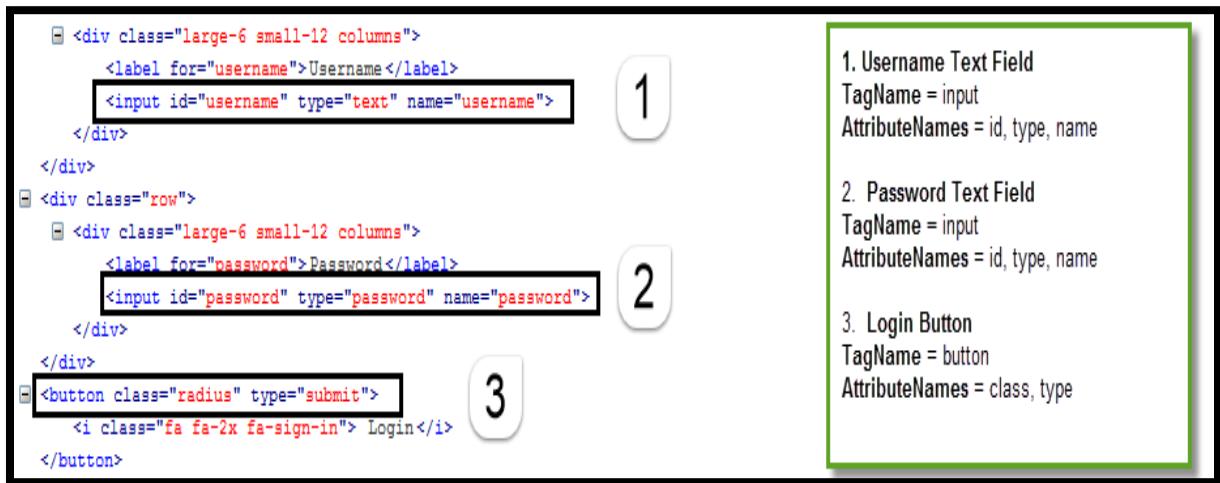


Figure 4.13 – HTML For The-Internet.Herokuapp.com/Login

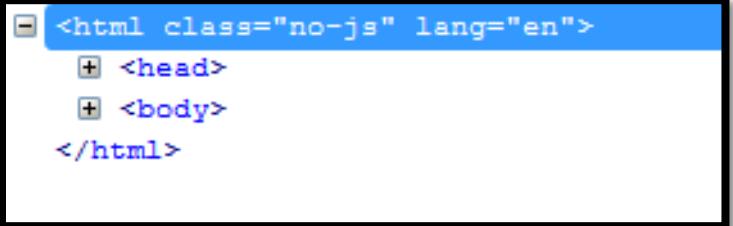
Understanding The HTML Document Object Model (DOM)

The Document Object Model (DOM) represents an HTML document as a tree. The tree consists of parent-child relationship nodes. A parent can have one child node or many children nodes. Everything is considered a node in the HTML DOM. All of the HTML elements/tags are element nodes and all attributes are attribute nodes. Thus, the HTML document is a document node.

The HTML document becomes a document object when loaded into a browser. As a result, the top level is the owner of all nodes that contains children nodes. The parent node is labeled HTML while the two children nodes are labeled head and body. The following screenshot shows the parent node “HTML” along with its two children nodes “head and body”.

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

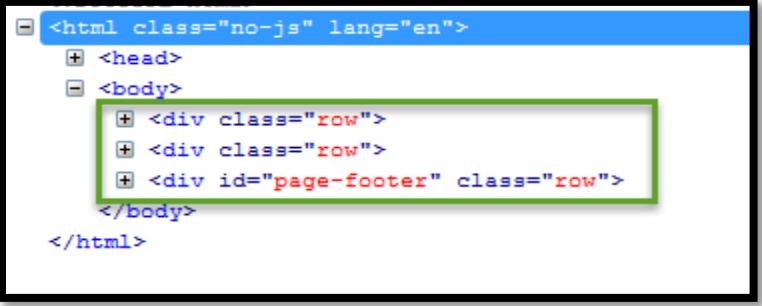
Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>



```
<html class="no-js" lang="en">
  + <head>
  + <body>
</html>
```

Figure 4.14 – HTML Parent Node And Two Children Nodes

As mentioned earlier, everything is considered a node in HTML DOM. Therefore, the html opening/closing tag and two attribute name/value pairs are nodes. The minus icon (-) next to html tag discloses the child nodes “head and body”. An automation engineer can click the plus icon (+) to view all children nodes within a specific parent node. In our example website “[Welcome to the Internet](#)”, the body node contains three div tags. The second div tag holds the Username text field, Password text field, and Login button. The following screenshots show each div tag and how the WebElements appear within the second div tag:



```
<html class="no-js" lang="en">
  + <head>
  - <body>
    + <div class="row">
    + <div class="row">
    + <div id="page-footer" class="row">
      </body>
    </html>
```

Figure 4.15 – Multiple Div Tags Within Body Node

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Chapter 4

First Selenium Test Script

(Part 1) Selenium WebDriver

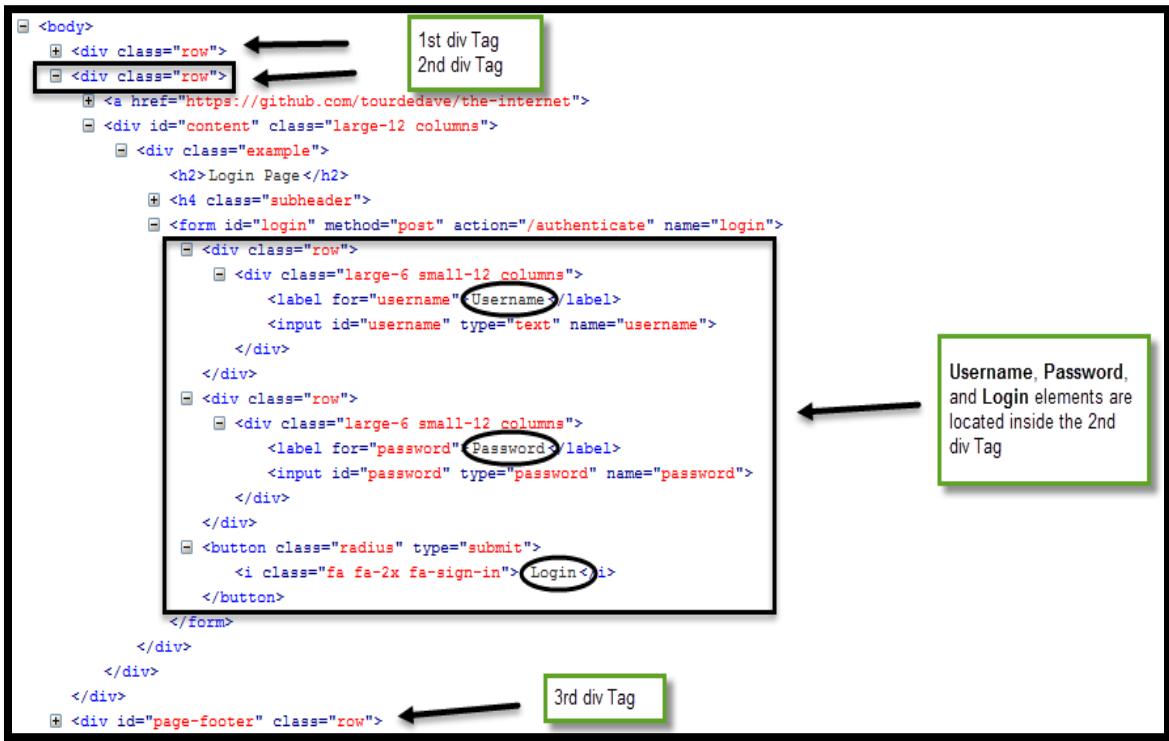


Figure 4.16 – Username, Password, and Login WebElements Inside 2nd Div Tag

Note: The WebElements “Username, Password, and Login” are located within the 2nd div tag. However, there are several nodes between the 2nd div tag and each WebElement.

How To Locate A WebElement

A WebElement is located utilizing the tag name of an opening/closing tag, attribute name, and/or attribute value. They are located depending on how the extractors find and match the elements. Each element is extracted from HTML’s source code into Selenium IDE. Firebug is one of many tools used to read HTML’s source code. Steps for installing Firebug are

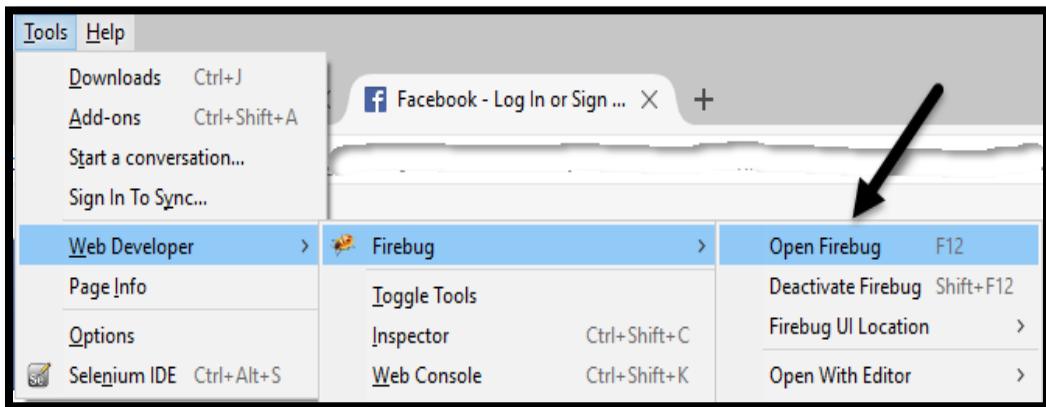
3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

located in [Chapter 1 – Install Firebug and FirePath](#). The following are steps for locating a WebElement.

Steps for locating a WebElement:

1. Open Firefox browser
2. Load a web application “i.e., [Welcome to the Internet](#)”
3. Click shortcut “F12” or Tools > Web Developer > Firebug > Open Firebug



4. Navigate to the HTML tab within Firebug



Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

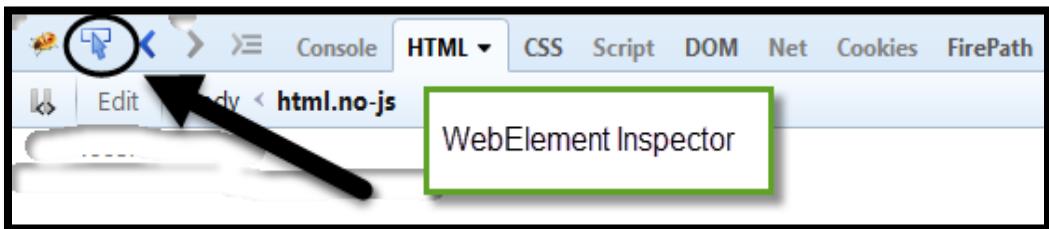
LinkedIn: <https://www.linkedin.com/in/rexjones34>

Chapter 4

First Selenium Test Script

(Part 1) Selenium WebDriver

5. Click the WebElement Inspector



6. Select the WebElement "i.e., Username"

Result: The WebElement is highlighted in the web application and HTML's source code.

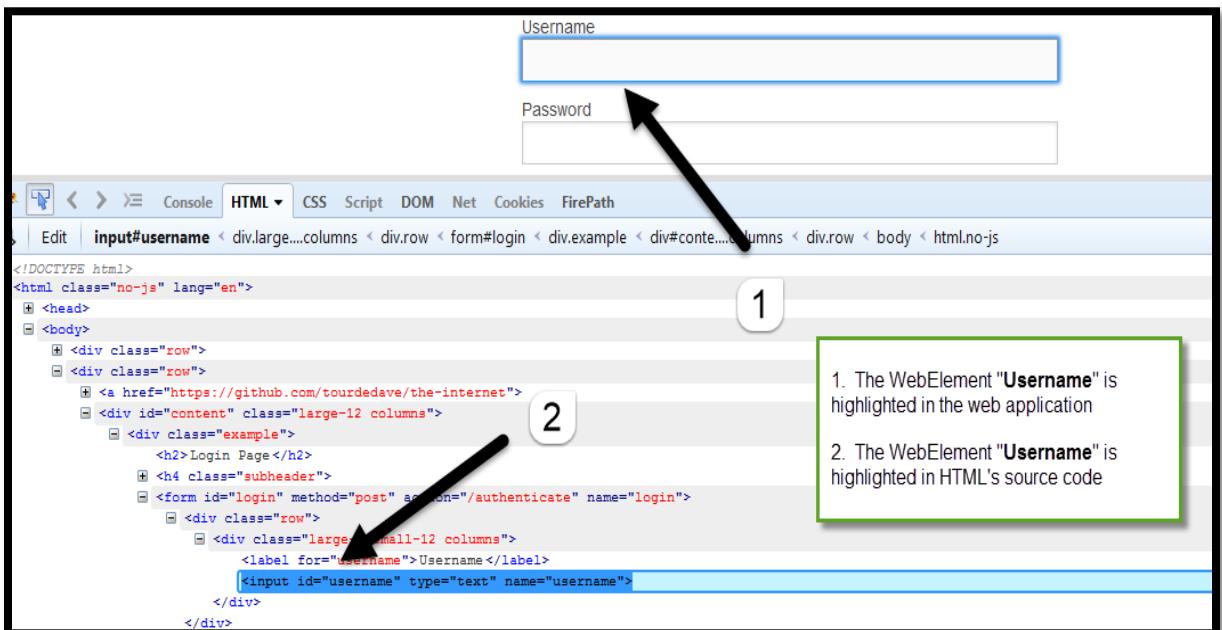


Figure 4.17 – WebElement In The Web Application And HTML Source Code

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

Chapter 4 explained Selenium IDE and HTML. Selenium IDE primarily used for recording and replaying Test Scripts. HTML is the standard markup language used for creating web applications. Web applications contain WebElements identified by tags and attributes within HTML. The Test Scripts from Selenium IDE can be exported to Selenium WebDriver. Chapter 5 will describe WebDriver, how to find WebElements via WebDriver, and how to perform actions on WebElements via WebDriver.

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Chapter 5

WebDriver and WebElements

The starting point for testing all web applications begins with a browser. Selenium WebDriver provides support for major browsers such as Firefox, Google Chrome, Internet Explorer, Safari, and Opera. Calls are directed to the browser by using the browser's native support for automation. Common calls such as performing actions on WebElements are executed through a WebDriver. WebDriver is an interface tool used for testing web applications.

As mentioned in Chapter 4, WebElements are buttons, text boxes, checkboxes, drop down menus, and hyperlinks. The building blocks of Selenium WebDriver is locating a WebElement and performing an action on the WebElement. Therefore, if an automation engineer identifies the element then an appropriate action can be determined for the element.

Chapter 5 will discuss the following regarding WebDriver and WebElements:

- ✓ [WebDriver Packages and Classes](#)
- ✓ [WebDriver Object and Methods](#)
- ✓ [Find and Perform Actions On WebElements](#)

Note: A configuration for WebDriver is required before importing its packages. The driver object and methods are put to use after a WebDriver package import. [Chapter 1 – Configure WebDriver](#) provides details for configuring WebDriver.

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

WebDriver Packages and Classes

The WebDriver interface contains several classes. Each class utilized through an imported package. In order to use WebDriver, the appropriate package must be imported to execute specific commands. As an example, package “org.openqa.selenium.chrome” must be imported to use class “ChromeDriver” to execute a command such as load the Google Chrome browser.

As mentioned in Chapter 3 (see *Import Annotations*), importing a package is a feature that allows a class to access members of another class. Per the example, a separate class accesses the “ChromeDriver” class in order to load Google Chrome. The following is an example of importing a package for a Firefox driver:

```
import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.firefox.FirefoxDriver;
```

WebDriver Object and Methods

Executions of the WebDriver commands are carried out through the driver object. The driver object is the main WebDriver component, which assist with locating and performing actions on WebElements. Like objects in Java programming, the driver object must be instantiated “created” before it is used. The following is an example of instantiating a Firefox driver object:

```
WebDriver driver = new FirefoxDriver();
```

After creating a driver object, the WebDriver methods are available to perform actions on WebElements. An automation engineer can type the word “driver” followed by the dot operator (.) to view the methods. The following is a screenshot of the WebDriver methods:

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

The screenshot shows a Java code editor with the following code:

```
1 import org.openqa.selenium.*;
2 import org.openqa.selenium.firefox.*;
3 import org.testng.annotations.*;
4
5 public class SeleniumExamples
6 {
7     @BeforeTest
8     void createWebDriver()
9     {
10         WebDriver driver = new FirefoxDriver();
11
12         driver.|
```

A tooltip is displayed over the `driver.` part of the code, listing various WebDriver methods:

- close() : void - WebDriver
- equals(Object obj) : boolean - Object
- findElement(By arg0) : WebElement - WebDriver
- findElements(By arg0) : List<WebElement> - WebDriver
- get(String arg0) : void - WebDriver
- getClass() : Class<?> - Object
- getCurrentUrl() : String - WebDriver
- getPageSource() : String - WebDriver
- getTitle() : String - WebDriver
- getWindowHandle() : String - WebDriver
- getWindowHandles() : Set<String> - WebDriver
- hashCode() : int - Object
- manage() : Options - WebDriver
- navigate() : Navigation - WebDriver
- notify() : void - Object
- notifyAll() : void - Object
- quit() : void - WebDriver
- switchTo() : TargetLocator - WebDriver
- toString() : String - Object
- wait() : void - Object
- wait(long timeout) : void - Object
- wait(long timeout, int nanos) : void - Object

The IDE interface includes tabs for Markers, Properties, and Se, and a status bar at the bottom.

Figure 5.1 – WebDriver Methods

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

The WebDriver methods either return a value or return no value. A method “i.e., **close() : void**” with a return value of “void” indicates a value will not be returned. On the other hand, a method “i.e., **findElement(By arg0) : WebElement**” with a return value of “WebElement” indicates a WebElement will be returned.

Notice a description of the highlighted WebDriver method “**close() : void**” is not available. Eclipse has a feature that allows documentation to be imported from [SeleniumHQ](#). After importing the documentation, a description of the WebDriver methods will be available. The following are steps to add documentation from [SeleniumHQ](#):

Steps to import documentation from Selenium HQ:

1. Navigate to <http://www.seleniumhq.org/download/>
2. Right click [Javadoc](#)



A screenshot of a table on the SeleniumHQ download page. The table lists client versions and release dates for various programming languages. The Java row is highlighted with a green arrow pointing to the Javadoc link. The table has a black border.

Language	Client Version	Release Date	Download	Change log	Javadoc
Java	2.53.0	2016-03-15	Download	Change log	Javadoc
C#	2.53.0	2016-03-16	Download	Change log	API docs
Ruby	2.53.0	2016-03-15	Download	Change log	API docs
Python	2.53.0	2016-03-15	Download	Change log	API docs
Javascript (Node)	2.53.1	2016-03-15	Download	Change log	API docs

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

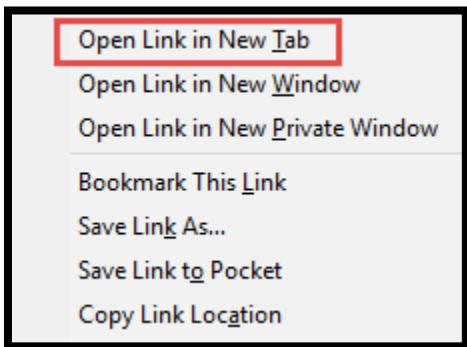
LinkedIn: <https://www.linkedin.com/in/rexjones34>

Chapter 5

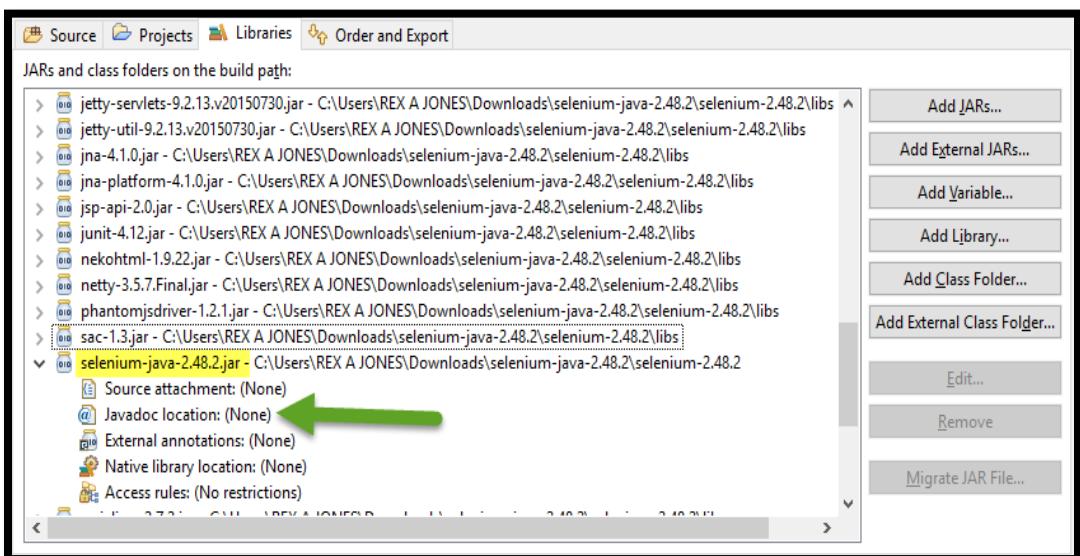
WebDriver and WebElements

(Part 1) Selenium WebDriver

3. Select Open Link in New Tab



4. Open Eclipse IDE
5. Right click on the project “i.e., Chapter_5”
6. Select Build Path => Configure Build Path
7. Search for Selenium jar files “i.e., selenium-java-2.48.2.jar”
8. Select Javadoc location: None

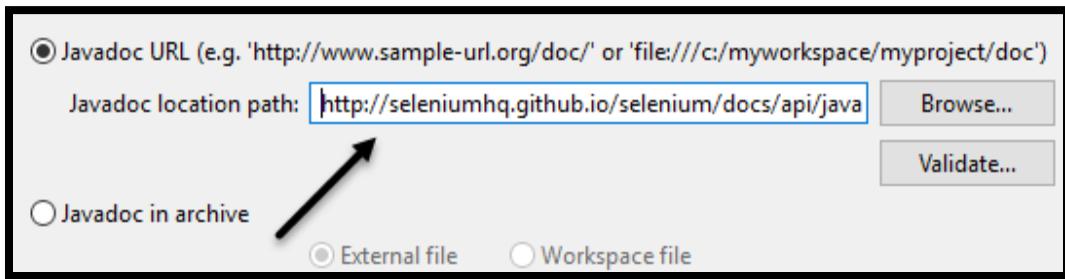


3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

9. Click the Edit button
10. Paste the Javadoc URL into Javadoc location path

<http://seleniumhq.github.io/selenium/docs/api/java/index.html>



11. Remove index.html from the end of the location path
<http://seleniumhq.github.io/selenium/docs/api/java/>
12. Click OK => Apply
13. Javadoc is successfully imported into Eclipse

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

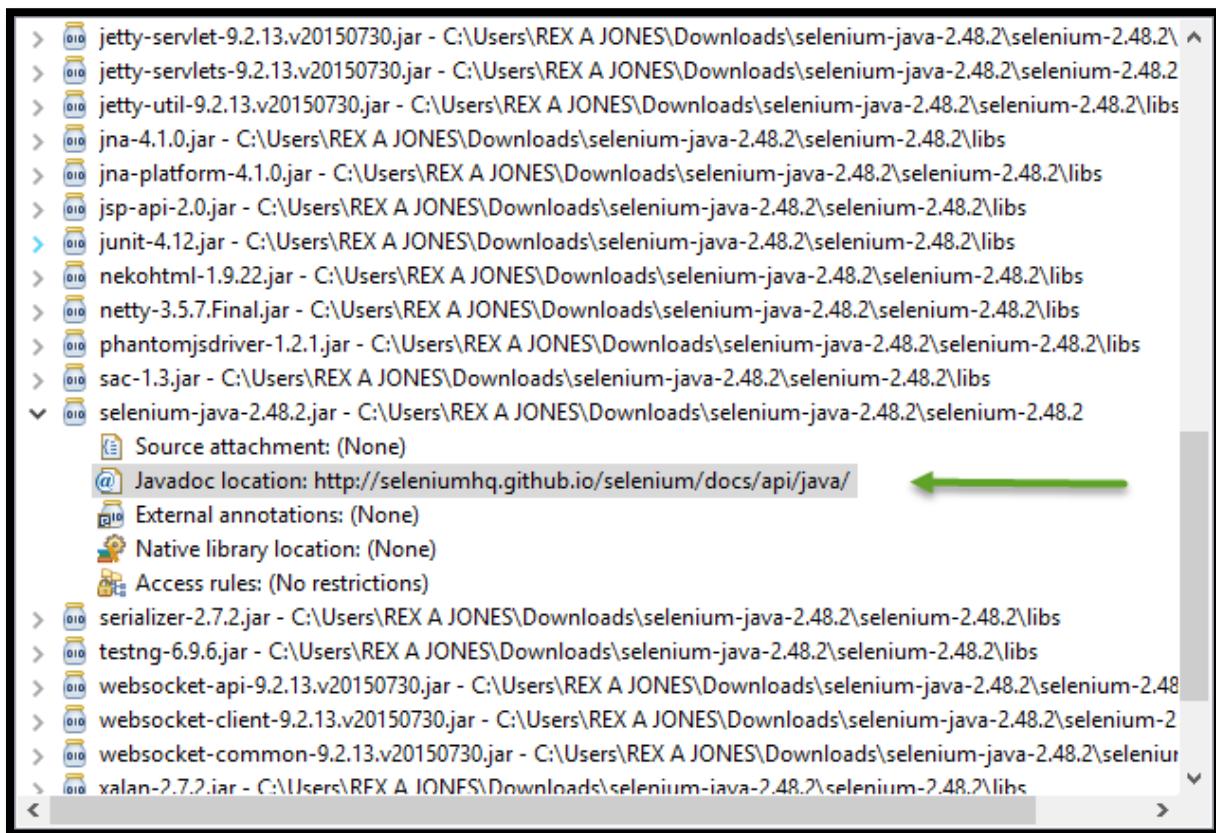


Figure 5.2 – Import Javadoc Into Eclipse

A description of the WebDriver methods is available after importing the Javadoc. Descriptions are valuable if an automation engineer lacks knowledge about a method. The following is a screenshot of the previous example but with a WebDriver description:

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

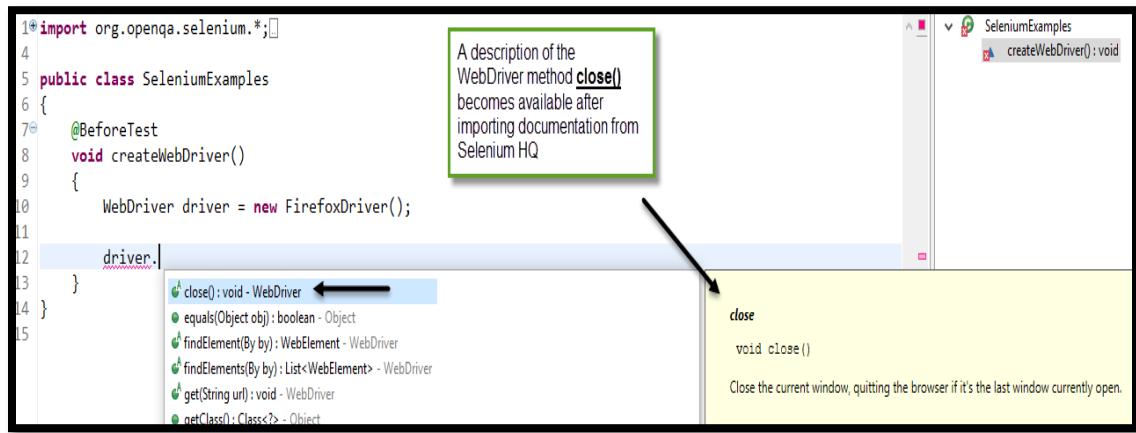


Figure 5.3 – WebDriver Methods With A Description

The following describes each WebDriver method:

WebDriver Method	Description
<code>close()</code>	Closes the current active window if there are multiple windows. The browser quits if only one window is active
<code>findElement()</code>	Find the first WebElement based on the locator type
<code>findElements()</code>	Find all elements within the current page based on the locator type
<code>get()</code>	Loads a new web page in the current browser window
<code>getCurrentUrl()</code>	Gets a string defining the current web page URL
<code>getPageSource()</code>	Gets the complete page source of the loaded web page.

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

getTitle()	Gets the current page title
getWindowHandle()	Handles a browser after switching a specific window
getWindowHandles()	Handles all browser windows and permits the user to switch control between the parent window and child window
manage()	Receives the option interface
navigate()	Navigates to a specific URL
quit()	Stops/Quits the driver instance and close all open browser windows
switchTo()	Switch from one browser window to another browser window

Figure 5.4 – WebDriver Methods and Descriptions

Find And Perform Actions On WebElements

WebDriver methods [findElement\(\)](#) and [findElements\(\)](#) are used to find WebElements. The difference between both methods are returned number of WebElements. WebDriver method [findElement\(\)](#) returns one element while [findElements\(\)](#) return multiple elements. If no elements are found then [findElement\(\)](#) throws an exception “NoSuchElementException” but [findElements\(\)](#) returns an empty list.

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

It is important to know a driver object must be created before searching for a WebElement. After [creating a driver object](#), the driver object connects to one of the WebDriver methods via dot operator. The WebDriver method is followed by an open parenthesis, a By object, dot operator, and a locator type. The following is the syntax for [findElement\(\)](#) and [findElements\(\)](#):

Syntax for findElement()

```
driver.findElement(By.locatorType("value"));
```

// or

```
WebElement element = driver.findElement(By.id("value"));
```

Syntax for findElements()

```
driver.findElements(By.locatorType("value"));
```

// or

```
List<WebElement> elements = element.findElements(By.id("value"));
```

Note: Package “java.util.List” must be imported to write the last syntax which starts with “List <WebElement> ...”.

The object “By” is used to help locate WebElements. However, the object must be used in conjunction with a locator type. A locator type is the main component for finding and matching WebElements. There are eight locator types to find WebElements within an application. The following is a screenshot of all locator types in Eclipse after writing

```
driver.findElement(By.
```

// or

```
driver.findElements(By.
```

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

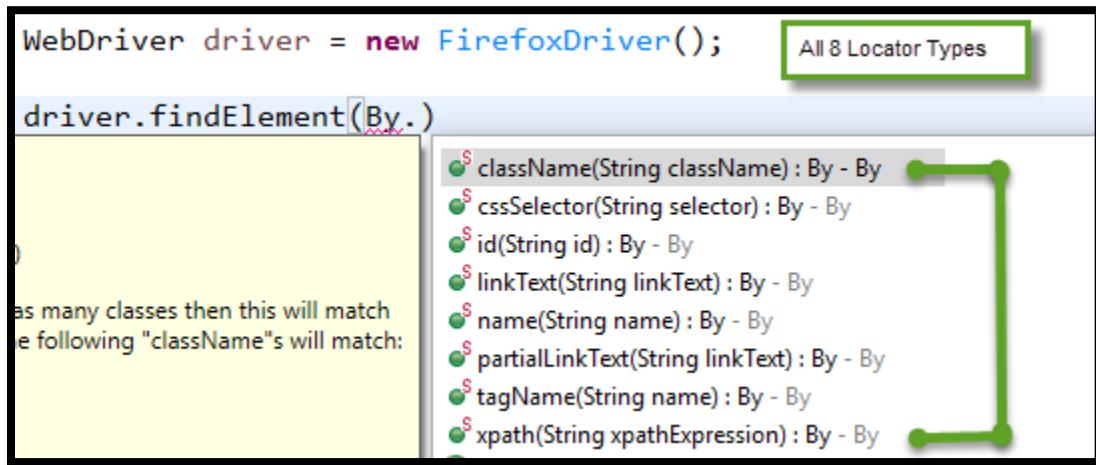


Figure 5.5 – Screenshot of All Locator Types

The following is a description of each locator type in alphabetical order:

Locator Type	Description
<code>By.className</code>	Find WebElements by the value of its Class attribute
<code>By.cssSelector</code>	Find WebElements by the CSS Selector's engine
<code>By.id</code>	Find WebElements by the value of its ID attribute
<code>By.linkText</code>	Find hyperlink WebElements by its complete text
<code>By.name</code>	Find WebElements by the value of its Name attribute
<code>By.partialLinkText</code>	Find hyperlink WebElements by partial text contained within the complete text

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

<i>By.tagName</i>	Find WebElements by its tag name
<i>By.xpath</i>	Find WebElements by its XPath

Figure 5.6 – Eight Locator Types

The following chapters describe each `findElement` by locator type:

- [Chapter 6 – Find WebElement By ID](#)
- [Chapter 7 – Find WebElement By Name](#)
- [Chapter 8 – Find WebElement By XPath](#)
- [Chapter 9 – Find WebElement By CSS Selector](#)
- [Chapter 10 – Find WebElement By Link Text](#)
- [Chapter 11 – Find WebElement By Partial Link Text](#)
- [Chapter 12 – Find WebElement By Tag Name](#)
- [Chapter 13 – Find WebElement By Class](#)

An element can be located using a locator type if the HTML property (e.g., ID, Name, etc.) is present. Each locator type provides interaction between Selenium WebDriver and the web application. However, there is a precedence regarding which locator type is used to find a specific WebElement. The following is prioritization of the locator types:

1. By.*id*
2. By.*name*

An automation engineer should search for a WebElement starting with an ID attribute. If the ID attribute is not available then the next attribute is Name. CSS Selector and XPath locator types are the next priority locators behind ID and Name. Some automation engineers use CSS Selector as the third priority while others use XPath. As you will see in [Chapter 8 - Find](#)

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

[WebElements by XPath](#) and [Chapter 9 - Find WebElements by CSS Selector](#), the XPath and CSS Selector locators are very similar.

After finding a WebElement, an action such as enter value is performed on the WebElement. The following is a screenshot of available actions / methods in Eclipse after writing

```
driver.findElement(By.id("value")).
```

// or

```
WebElement element = driver.findElement(By.id("value"));  
element.
```

Note: Both statements end with a dot operator (.).

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

```
WebDriver driver = new FirefoxDriver();

WebElement element = driver.findElement(By.id("value"));
element.

    A clear() : void - WebElement
    A click() : void - WebElement
    ● equals(Object obj) : boolean - Object
    A findElement(By by) : WebElement - WebElement
    A findElements(By by) : List<WebElement> - WebElement
    A getAttribute(String name) : String - WebElement
    ● getClass() : Class<?> - Object
    A getCssValue(String propertyName) : String - WebElement
    A getLocation() : Point - WebElement
    A getScreenshotAs(OutputType<X> target) : X - TakesScreenshot
    A getSize() : Dimension - WebElement
    ● getTagName() : String - WebElement
    A getText() : String - WebElement
    ● hashCode() : int - Object
    A isDisplayed() : boolean - WebElement
    A isEnabled() : boolean - WebElement
    A isSelected() : boolean - WebElement
    ● notify() : void - Object
    ● notifyAll() : void - Object
    A sendKeys(CharSequence... keysToSend) : void - WebElement
    A submit() : void - WebElement
    ● toString() : String - Object
    ● wait() : void - Object
    ● wait(long timeout) : void - Object
    ● wait(long timeout, int nanos) : void - Object
```

Figure 5.7 – Available Actions After Finding A WebElement

The following describes each method that performs an action:

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Action/Method	Description
clear()	Clear or erase a value in a text field
click()	Clicks a link, button, checkbox, or radio button
findElement()	Find the first WebElement based on the locator type
findElements()	Find all elements within the current page based on the locator type
getAttribute()	Get the value of a specified WebElement attribute
getCssValue()	Returns the value of CSS properties
getLocation()	Returns the x and y coordinates with the top left corner of an element as the point
getSize()	Returns the width and height of a rendered element
getTagName()	Get the tag name of a specified WebElement
getText()	Get the visible innerText attribute of a WebElement
isDisplayed()	Verifies if a WebElement is present on a page
isEnabled()	Verifies if a WebElement is enabled on a page
isSelected()	Verifies if a WebElement is selected on a page
sendKeys()	Insert or type text in a text field
submit()	Operates like the click () method if the WebElement is a form or within a form

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

Figure 5.8 - Methods That Perform Actions On WebElements

Chapter 5 discussed WebDriver and WebElements. WebDriver is a tool used for testing web applications. All web applications contain WebElements. WebElements are buttons, text boxes, checkboxes, drop down menus, and hyperlinks. The key to testing a web application is finding the WebElement then performing an action on the WebElement. Chapters 6 through 13 explain how to find a WebElement by each locator type and provide examples for performing an action:

- [Chapter 6 – Find WebElement By ID](#)
- [Chapter 7 – Find WebElement By Name](#)
- [Chapter 8 – Find WebElement By XPath](#)
- [Chapter 9 – Find WebElement By CSS Selector](#)
- [Chapter 10 – Find WebElement By Link Text](#)
- [Chapter 11 – Find WebElement By Partial Link Text](#)
- [Chapter 12 – Find WebElement By Tag Name](#)
- [Chapter 13 – Find WebElement By Class](#)

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Chapter 6

Find WebElement By ID

Finding an element by ID is the first priority of all locator types. According to [W3C](#), id attributes are unique on a web page. Therefore, the id attribute is extremely safe when locating an element within HTML. It is possible the developer will not change the ID when modifying a web page. The following is a screenshot of [LinkedIn](#) and its HTML markup tags for a User's First Name:

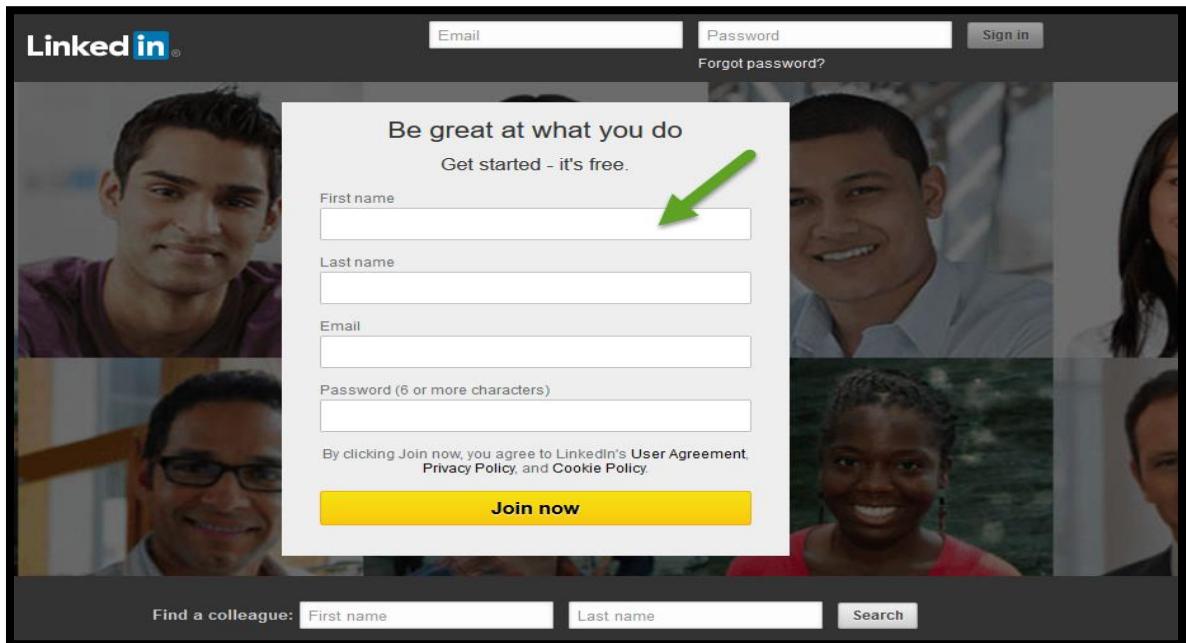


Figure 6.1 – LinkedIn's Home Screen (User's First Name)

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

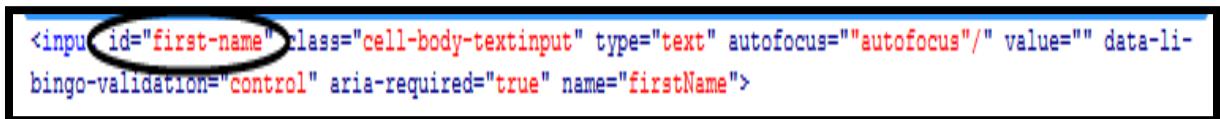


Figure 6.2 – HTML For LinkedIn's User First Name

The following is an example of finding [LinkedIn's](#) User First Name using the HTML id attribute:

```
driver.findElement(By.id("first-name"));
```

// or

```
WebElement userFirstName = driver.findElement(By.id("first-name"));
```

Enter Text In The User's First Name Text Field

The following is code for entering text in [LinkedIn's](#) User First Name text field via id attribute:

```
package LinkedInHomePage;
import org.openqa.selenium.*;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.testng.annotations.*;

public class JoinLinkedIn1
{
    WebDriver driver;

    @BeforeTest
    public void setUp() throws Exception
    {
        driver = new FirefoxDriver();

        // Go to LinkedIn's Home Page
        driver.get("https://www.linkedin.com/");
    }
}
```

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

```
}

{@AfterTest
public void tearDown() throws Exception
{
    driver.quit();
}

@Test
void joinLinkedIn_1()
{
    // Find first name via ID locator type
    // Enter first name via send keys
    driver.findElement(By.id("first-name")).sendKeys("Test First Name");
}

}
```

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

```

1 package LinkedInHomePage;
2 import org.openqa.selenium.*;
3
4 public class JoinLinkedIn1
5 {
6     WebDriver driver;
7
8     @BeforeTest
9     public void setUp() throws Exception
10    {
11         driver = new FirefoxDriver();
12
13         // Go to LinkedIn's Home Page
14         driver.get("https://www.linkedin.com/"); ← 1
15
16     }
17
18     @AfterTest
19     public void tearDown() throws Exception
20     {
21         driver.quit(); ← 4
22     }
23
24
25     @Test
26     void joinLinkedIn_1()
27     {
28         // Find first name via ID locator type
29         // Enter first name via send keys
30         driver.findElement(By.id("first-name")).sendKeys("Test First Name"); ← 3
31     }
32 }
```

1. The `get()` method loads LinkedIn's Home page

2. The First Name WebElement is found using `findElement()` and locator type `By.id`

3. Text is entered using `sendKeys()`

4. The `quit()` method closes the browser

Figure 6.3 – Perform Actions On The User’s First Name Text Field (1)

- Line 8 “WebDriver `driver`” is the interface for driving the browser. Currently, the object reference variable “`driver`” points to nothing but will point to a `FirefoxDriver` object in a subsequent line “`driver = new FirefoxDriver()`”.
- Line 13 “`driver = new FirefoxDriver()`” is an implementation of the WebDriver interface. The object reference variable “`driver`” is pointing to `new FirefoxDriver()` which means testing is controlled on the Firefox browser.

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.orgLinkedIn: <https://www.linkedin.com/in/rexjones34>

- Line 16 “`driver.get("https://www.linkedin.com/")`” loads a new LinkedIn Home page in the current browser window
- Line 22 “`driver.quit()`” quits the driver instance and closes the open browser window
- Line 30 “`driver.findElement(By.id("first-name")).sendKeys("Test First Name")`”:
 - `driver` – WebDriver object reference variable that assist with finding a WebElement
 - `findElement` – a WebDriver method that finds the text field WebElement “First name” on LinkedIn’s Home page
 - `(By.id("first-name"))` – By and id are parameters of the findElement WebDriver method. By is an object which locate elements and id is the locator type. The id locator type accepts a string parameter “first-name” which is the value of the HTML id attribute
 - `sendKeys("Test First Name")` – types the text “Test First Name” in the First Name text field

Note: An exception is an error that occurs at runtime. Lines 11 and 20 define an exception via `throws` Exception. It is used to explicitly identify an exception that the `setUp()` and `tearDown()` methods might come across at runtime. See Errors, Exceptions, and Debugging in Chapter 7 of “[\(Part 2\) Java 4 Selenium WebDriver](#)”.

The following is a screenshot of [LinkedIn's](#) Home page after entering text in the User’s First Name text field:

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

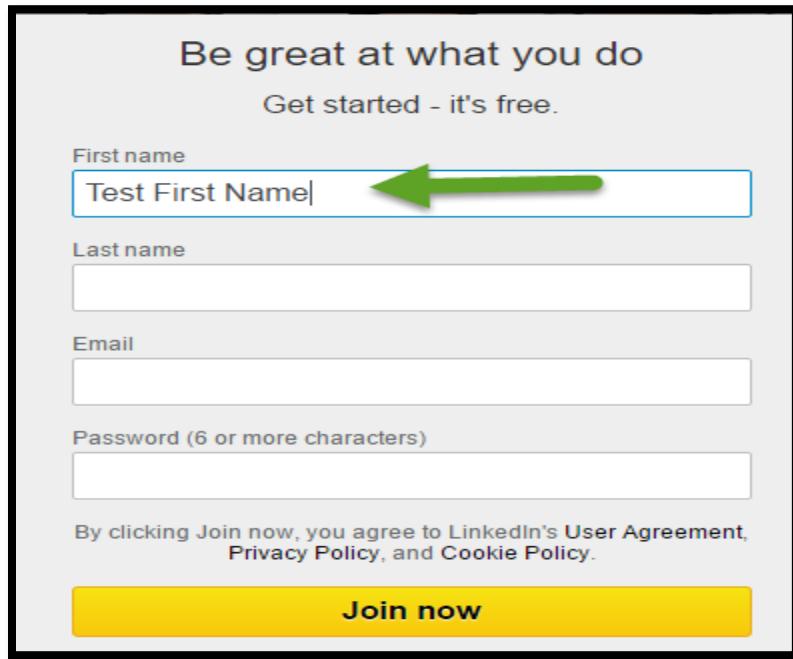


Figure 6.4 – Enter Text In The User’s First Name Text Field

In some cases, an extra tab opens with a problem while running Test Scripts in Firefox. Although there is a problem, the automation Test Scripts still run successfully. The following is a screenshot of two tabs opening up while running Test Scripts in Firefox:

Skype: rex.jones34
Twitter: @RexJonesII
Email: Rex.Jones@Test4Success.org
LinkedIn: <https://www.linkedin.com/in/rexjones34>

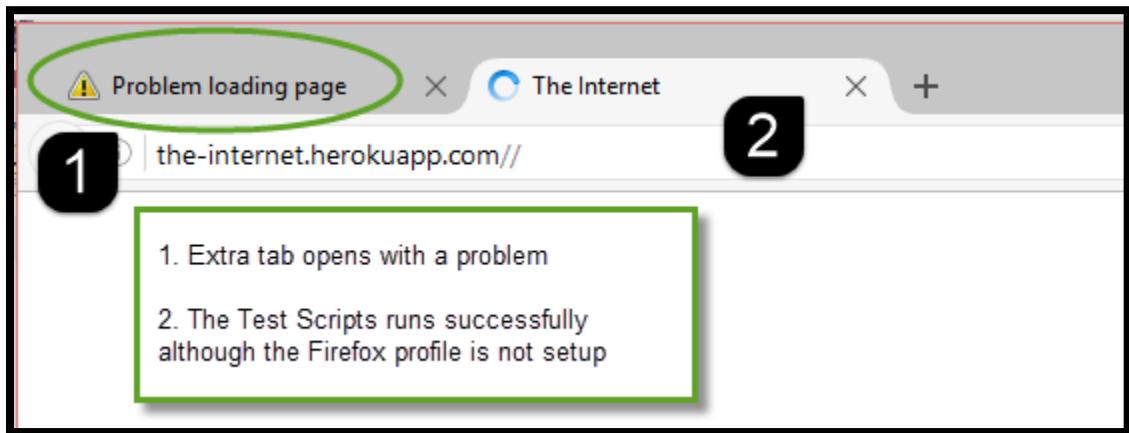


Figure 6.5 – Problem Loading A Tab In Firefox

The problem with Firefox loading an extra tab does not happen every time. However, prevent the problem by creating a Firefox profile then access the profile from Selenium WebDriver. Enter the following code lines in Selenium WebDriver to access the profile:

```
ProfilesIni profile = new ProfilesIni();
FirefoxProfile autoProfile = profile.getProfile("automationFireFoxProfile");
WebDriver driver = new FirefoxDriver(autoProfile);
```

The new profile “i.e., **automationFireFoxProfile**” reflects where the file is stored. Steps for creating a Firefox profile are located in [Chapter 1 – Set Up A Firefox Profile](#) section. The following screenshot is an example of how to access / get the Firefox profile (lines 10, 11, and 18) using Selenium WebDriver:

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

```
8 public class JoinLinkedIn
9 {
10     ProfilesIni profile = new ProfilesIni();
11     FirefoxProfile autoProfile = profile.getProfile("automationFireFoxProfile");
12
13     WebDriver driver;
14
15     @BeforeTest
16     public void setUp() throws Exception
17     {
18         driver = new FirefoxDriver(autoProfile);
19
20         // Go to LinkedIn's Home Page
21         driver.get("https://www.linkedin.com/");
22     }
}
```

Figure 6.6 – Screenshot of Code To Prevent A Tab Loading Problem

The following is a screenshot of a successful run after accessing the Firefox profile:

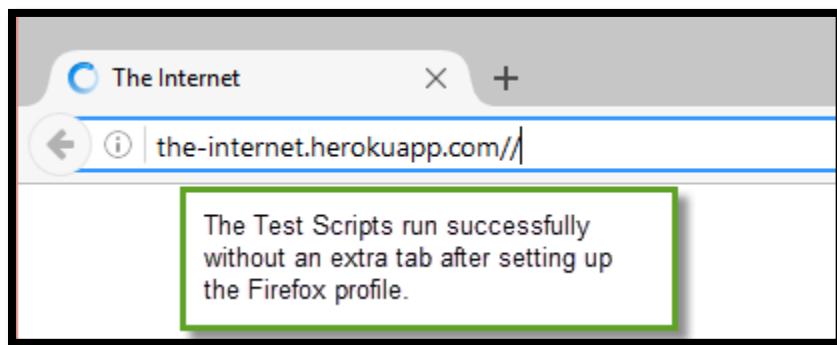


Figure 6.7 – Successful Run After Adding Code Lines In Selenium WebDriver

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Enter and Clear Text In The User's First Name Text Field

The following is code for entering and clearing text in [LinkedIn's](#) User First Name text field via id attribute:

```
package LinkedInHomePage;
import org.openqa.selenium.*;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.*;

public class JoinLinkedIn2
{
    WebDriver driver;

    @BeforeTest
    public void setUp() throws Exception
    {
        driver = new ChromeDriver();

        // Go to LinkedIn's Home Page
        driver.get("https://www.linkedin.com/");
    }

    @AfterTest
    public void tearDown() throws Exception
    {
        driver.quit();
    }

    @Test
    void enter_clearFirstName()
    {
        // Find first name via ID locator type
        WebElement userFirstName = driver.findElement(By.id("first-name"));

        // Enter first name via send keys
        userFirstName.sendKeys("Test First Name");

        // Clear first name via clear
    }
}
```

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

Chapter 6

Find WebElement By ID

(Part 1) Selenium WebDriver

```

        userFirstName.clear();
    }
}
}

```

```

1 package LinkedInHomePage;
2 import org.openqa.selenium.*;
3
4 public class JoinLinkedIn2
5 {
6     WebDriver driver;
7
8     @BeforeTest
9     public void setUp() throws Exception
10    {
11         driver = new ChromeDriver();
12
13         // Go to LinkedIn's Home Page
14         driver.get("https://www.linkedin.com/");
15     }
16
17     @AfterTest
18     public void tearDown() throws Exception
19    {
20         driver.quit();
21     }
22
23     @Test
24     void enter_clearFirstName()
25    {
26
27         // Find first name via ID locator type
28         WebElement userFirstName = driver.findElement(By.id("first-name"));
29
30         // Enter first name via send keys
31         userFirstName.sendKeys("Test First Name");
32
33         // Clear first name via clear
34         userFirstName.clear();
35     }
36 }
37

```

1. The `get()` method loads LinkedIn's Home page
2. A `WebElement` object "userFirstName" is created then the first name WebElement is found using `findElement()` and locator type `By.id`
3. Text is entered using `sendKeys()`
4. Text is cleared using `clear()`
5. The `quit()` method closes the browser

Figure 6.8 – Perform Actions On The User's First Name Text Field (2)

- Line 8 “WebDriver `driver`” is the interface for driving the browser. Currently, the object reference variable “`driver`” points to nothing but will point to a Chrome Driver object in a subsequent line “`driver = new ChromeDriver()`”.

Note: The `System.setProperty()` method was not used in this example since the

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

ChromeDriver path was placed in Environment Variables – System Variables. However, the subsequent examples will contain an extra code line for `System.setProperty()` method:

```
System.setProperty("webdriver.chrome.driver", C:\\Users\\REX A JONES\\Downloads\\chromedriver_win32\\chromedriver.exe");
```

Figure 6.9 – `System.setProperty()` Prevents An Exception For Google Chrome

- The following are parameters for `System.setProperty`:
 - **key** = `webdriver.chrome.driver`
 - **value** = `C:\\Users\\REX A JONES\\Downloads\\chromedriver_win32\\chromedriver.exe`
- Line 13 “`driver = new ChromeDriver()`” is an implementation of the WebDriver interface. The object reference variable “`driver`” is pointing to `new ChromeDriver()` which means testing is controlled on the Chrome browser.
- Line 16 “`driver.get("https://www.linkedin.com/")`” loads a new LinkedIn Home page in the current browser window
- Line 22 “`driver.quit()`” quits the driver instance and closes the open browser window
- Line 29 “`WebElement userFirstName = driver.findElement(By.id("first-name"));`”:
 - `WebElement userFirstName` – creates an object reference variable called `userFirstName` that refers or points to a `WebElement`
 - `driver` – WebDriver object reference variable that assist with finding the `WebElement`

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

- `findElement` – a WebDriver method that finds the text field WebElement “First name” on LinkedIn’s Home page
 - `(By.id("first-name"))` – By and id are parameters of the `findElement` WebDriver method. By is an object which locate elements and id is the locator type. The id locator type accepts a string parameter “first-name” which is the value of the HTML id attribute
-
- Line 32 “`userFirstName.sendKeys("Test First Name")`” types the text “Test First Name” in the First Name text field via object reference variable “`userFirstName`”
 - Line 35 “`userFirstName.clear()`” clears the text “Test First Name” from the First Name text field via object reference variable “`userFirstName`”

Note: This example was an illustration for `clear()` method. However, it is useful to create a WebElement object like `userFirstName` if multiple actions (i.e., enter text and clear text) are performed on the same WebElement (i.e., First Name text field).

The following is a screenshot of [LinkedIn's](#) Home page after clearing the User’s First Name:

Skype: rex.jones34
Twitter: @RexJonesII
Email: Rex.Jones@Test4Success.org
LinkedIn: <https://www.linkedin.com/in/rexjones34>

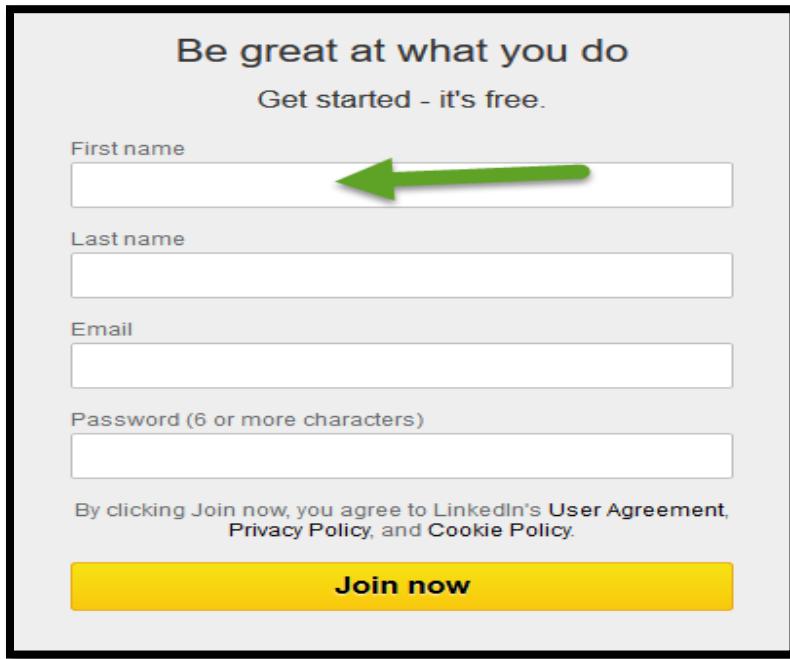


Figure 6.10 – Enter And Clear Text In The User’s First Name Text Field

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

Chapter 7

Find WebElement By Name

There are times when a web page may not contain an id attribute. If so, then an automation engineer should search for the value of a name attribute. Finding an element by name is the second priority behind the ID locator type. Usually, the name attribute is located in WebElements such as text fields and buttons. However, unlike the id attribute, which contains a unique value, the name attribute can contain the same value within HTML's source code. The following is a screenshot of [LinkedIn](#) and its HTML markup tags for a Colleague's First and Last Name:

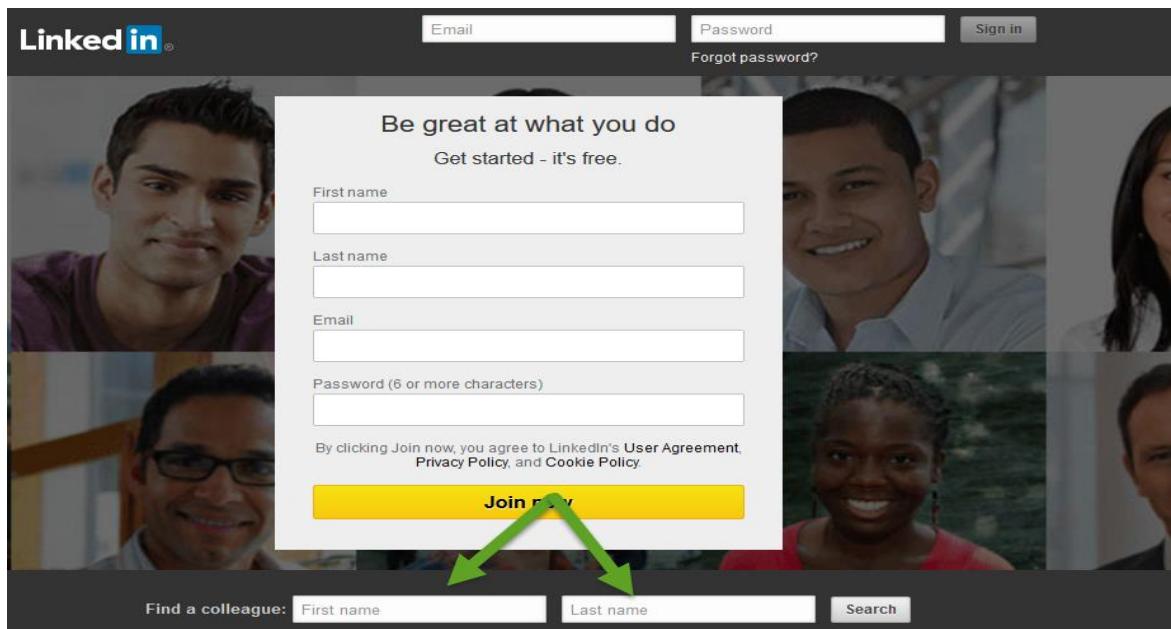


Figure 7.1 – LinkedIn's Home Screen (Colleague's First and Last Name)

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

```
<h3>Find a colleague:</h3>
<input type="text" placeholder="First name" name="first">
<input type="text" placeholder="Last name" name="last">
<input type="submit" value="Search" name="search">
```

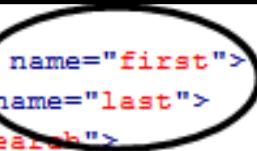


Figure 7.2 – HTML For LinkedIn's Colleague First and Last Name

Note: The HTML markup tags do not contain an id attribute for Colleague's First or Last Name:

Enter Text In The Colleague's First and Last Name Text Fields

The following is code for entering text in [LinkedIn's](#) Colleague First and Last Name text fields via name attribute:

```
package LinkedInHomePage;
import org.openqa.selenium.*;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.*;

public class FindColleague
{
    WebDriver driver;

    @BeforeTest
    public void setUp() throws Exception
    {
        System.setProperty("webdriver.chrome.driver", "C:\\\\Users\\\\REX A
JONES\\\\Downloads\\\\chromedriver_win32\\\\chromedriver.exe");
        driver = new ChromeDriver();

        driver.get("https://www.linkedin.com/");
    }

    @AfterTest
```

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

```

public void tearDown() throws Exception
{
    driver.quit();
}

@Test
public void enterColleagueName()
{
    driver.findElement(By.name("first")).sendKeys("Rex");
    driver.findElement(By.name("last")).sendKeys("Jones");
}
}

```

```

1 package LinkedInHomePage;
2 import org.openqa.selenium.*;
3
4
5 public class FindColleague
6 {
7     WebDriver driver;
8
9
10 @BeforeTest
11 public void setUp() throws Exception
12 {
13     System.setProperty("webdriver.chrome.driver", "C:\\\\Users\\\\REX A JONES\\\\Downloads\\\\chromedriver_win32\\\\chromedriver.exe");
14     driver = new ChromeDriver();
15
16     driver.get("https://www.linkedin.com/"); ← 1
17 }
18
19 @AfterTest
20 public void tearDown() throws Exception
21 {
22     driver.quit(); ← 5
23 }
24
25 @Test
26 public void enterColleagueName()
27 {
28     driver.findElement(By.name("first")).sendKeys("Rex");
29     driver.findElement(By.name("last")).sendKeys("Jones"); ← 2
30 } ← 3
31 }

```

1. The [get\(\)](#) method loads LinkedIn's Home page

2. The First Name WebElement is found using [findElement\(\)](#) and locator type [By.name](#)

3. The Last Name WebElement is found using [findElement\(\)](#) and locator type [By.name](#)

4. Text is entered using [sendKeys\(\)](#) for First and Last Name

5. The [quit\(\)](#) method closes the browser window

Figure 7.3 – Perform Actions On The Colleague's First Name Text Field

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.orgLinkedIn: <https://www.linkedin.com/in/rexjones34>

- Line 8 “WebDriver **driver**” is the interface for driving the browser. Currently, the object reference variable “driver” points to nothing but will point to a Chrome Driver object in a subsequent line “**driver = new ChromeDriver()**”.
- Line 13 tells Selenium where the executable file for Chrome driver is located via **System.setProperty**. The executable file operates like a bridge between Selenium and the browser. All browsers except Firefox require an executable file. Steps for downloading the executable file are located in [Chapter 1 – Download Browser Drivers](#) section. The following are parameters for **System.setProperty**:
 - key** = webdriver.chrome.driver
 - value** = C:\\Users\\REX A JONES\\Downloads\\chromedriver_win32\\chromedriver.exe

Note: Value is the path location of the executable file

- Line 14 “**driver = new ChromeDriver()**” is an implementation of the WebDriver interface. The object reference variable “driver” is pointing to new ChromeDriver() which means testing is controlled on the Chrome browser.
- Line 16 “**driver.get("https://www.linkedin.com/")**” loads a new LinkedIn Home page in the current browser window
- Line 22 “**driver.quit()**” quits the driver instance and closes the open browser window
- Line 28 “**driver.findElement(By.name("first")).sendKeys("Rex")**”:
 - driver** – WebDriver object reference variable that assist with finding a WebElement
 - findElement** – a WebDriver method that finds the text field WebElement “First name” on LinkedIn’s Home page

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

- (By.name("first")) – By and *name* are parameters of the findElement WebDriver method. By is an object which locate elements and name is the locator type. The name locator type accepts a string parameter “first” which is the value of the HTML name attribute
- sendKeys("Rex") – types the text “Rex” in the Colleague’s First Name text field

- Line 29 “`driver.findElement(By.name("last")).sendKeys("Jones")`”:
 - `driver` – WebDriver object reference variable that assist with finding a WebElement
 - `findElement` – a WebDriver method that finds the text field WebElement “First name” on LinkedIn’s Home page
 - (By.name("last")) – By.name are parameters of the findElement WebDriver method. By is an object which locate elements and name is the locator type. The name locator type accepts a string parameter “last” that is the value of the HTML name attribute
 - `sendKeys("Jones")` – types the text “Jones” in the Colleague’s Last Name text field

The following is a screenshot of [LinkedIn's](#) Home page after entering text in the Colleague’s First and Last Name field:

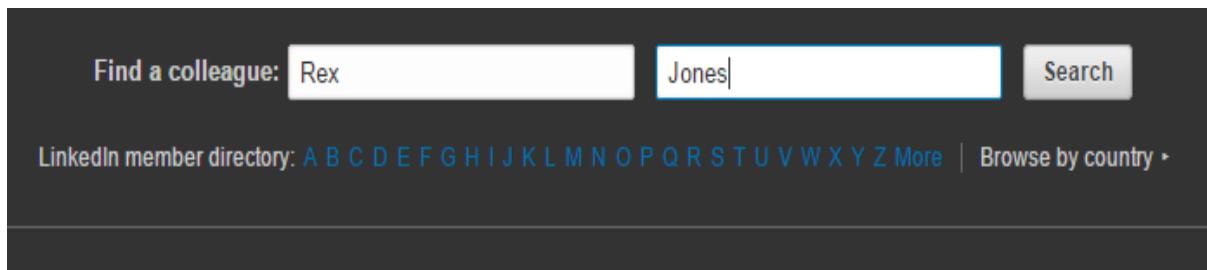


Figure 7.4 – Enter Text In The Colleague’s First and Last Name Text Field

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Chapter 8

Find WebElement By XPath

XML Path Language “XPath” is a powerful method for finding WebElements. The XPath locator type refers to a specific node in the DOM tree. It is important to know that most WebElements can be located using the XPath locator type. The XPath locator accepts two types of parameter strings to find a WebElement.

1. [Absolute XPath](#) – generates a complete XPath starting from the <html> tag or single forward slash (/)
2. [Relative XPath](#) – generates a shortened version of the XPath starting with double forward slashes (//)

An automation engineer can manually create an XPath using the web page’s HTML or a tool that generates the XPath. FirePath is a tool that generates an absolute and relative XPath.

Steps for installing FirePath are located in [Chapter 1 – Install Firebug and FirePath](#) section.

The following is a screenshot of FirePath within Firebug:

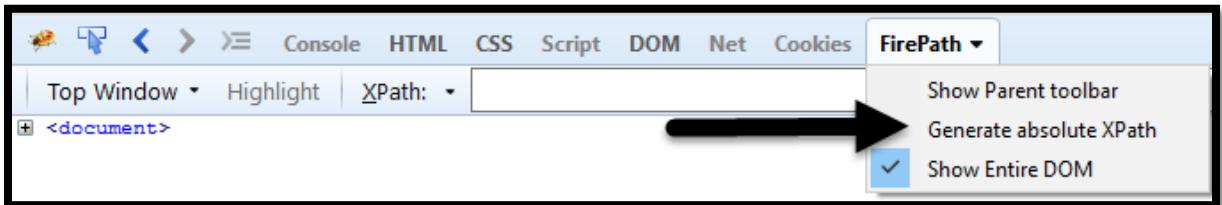


Figure 8.1 – FirePath / XPath Screenshot

The option labeled “Generate absolute XPath” determines whether FirePath generates an absolute or relative XPath. Below are screenshots of [Facebook's](#) Sign Up page, its markup

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

tags for radio buttons “Female and Male”, followed by steps to generate an absolute and relative XPath:

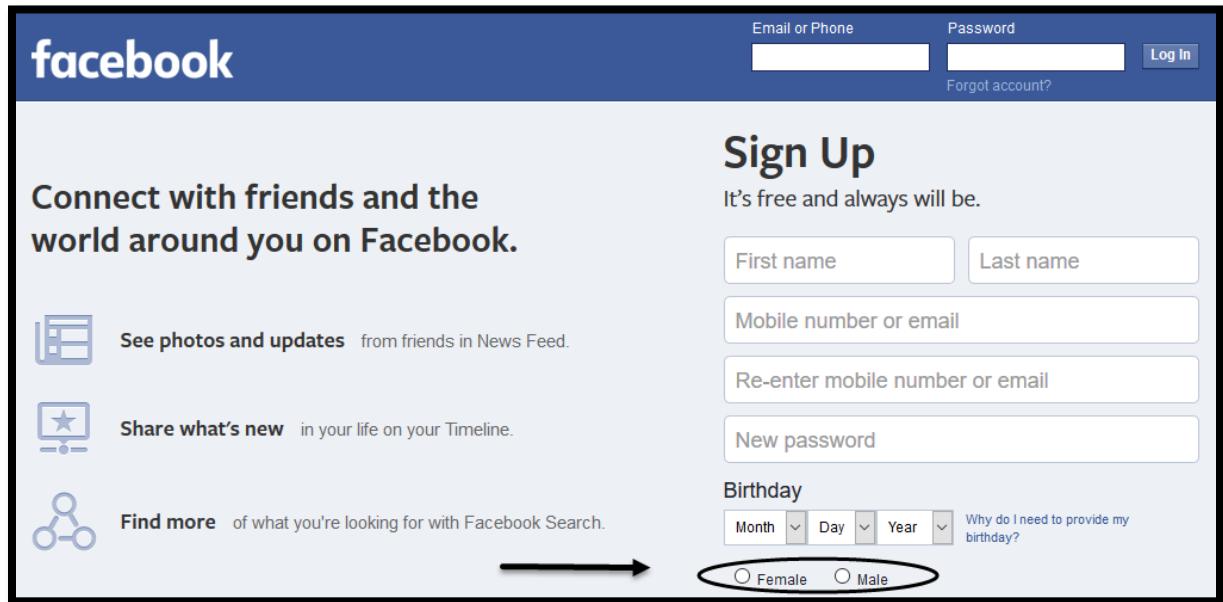


Figure 8.2 – Facebook’s Sign Up Page / Female and Male Radio Buttons

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

```

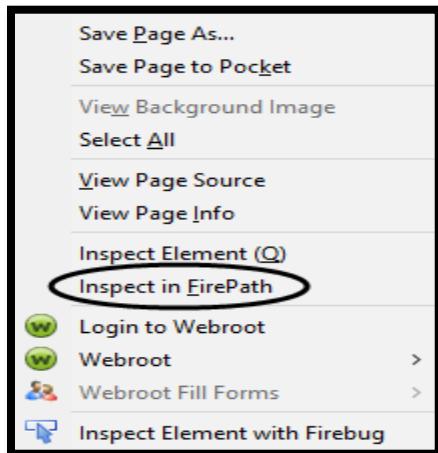
<span id="u_0_h" class="_5k_3" data-name="gender_wrapper" data-type="radio">
  <span class="_5k_2 _5dba">
    <input id="u_0_e" type="radio" value="1" name="sex"/>
    <label class=_58mt for="u_0_e">Female</label>
  </span>
  <span class="_5k_2 _5dba">
    <input id="u_0_f" type="radio" value="2" name="sex"/>
    <label class=_58mt for="u_0_f">Male</label>
  </span>
</span>

```

Figure 8.3 – HTML For Female and Male Radio Buttons On Facebook’s Sign Up Page (1)

Steps For Generating An Absolute XPath:

1. Open Firefox browser
2. Load a web application “i.e., [Facebook](#)”
3. Right click the WebElement “i.e., radio button”
4. Select Inspect in FirePath



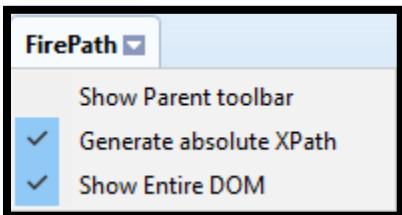
3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

5. Click the drop down for FirePath
6. Check the option “Generate absolute XPath”



Result: The XPath text box displays a string parameter for absolute XPath.

The screenshot shows the FirePath interface with the 'FirePath' tab selected. The 'XPath' field contains the absolute XPath: `html/body/div[1]/div[2]/div[1]/div/div/div/div[2]/div[2]/div/div/div[1]/form/div[1]/div[6]/span/span[1]/input`. The DOM tree below shows two spans: 'span [1]' and 'span [2]'. The first span contains a radio button for 'Female' and a label. The second span contains a radio button for 'Male' and a label. A callout box on the right explains the XPath: 1. Absolute XPath starts with html, and 2. span [1] identifies the Female radio button.

Figure 8.4 – Absolute XPath For The Female Radio Button

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

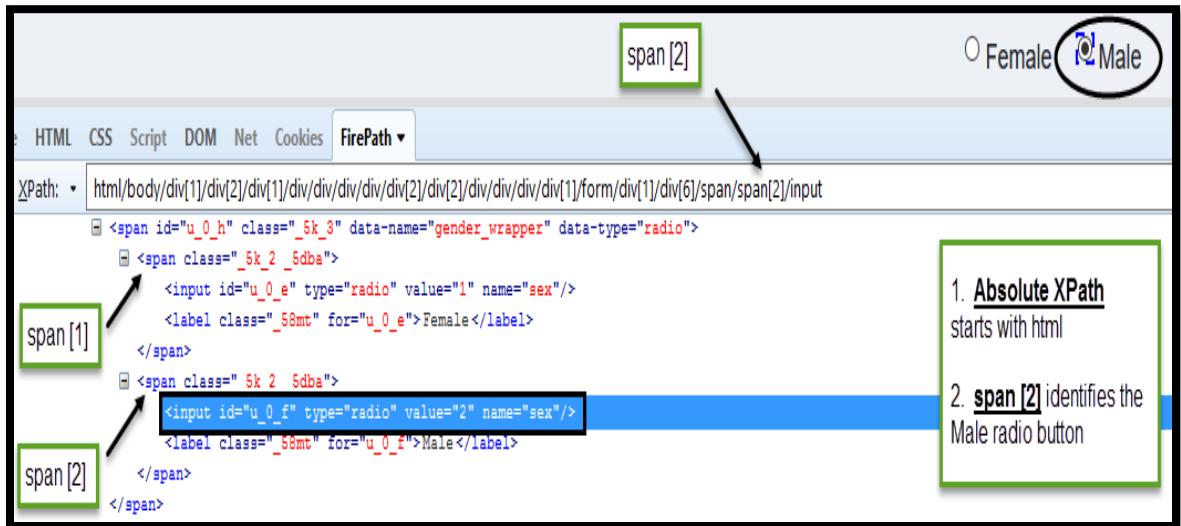


Figure 8.5 – Absolute XPath For The Male Radio Button

Steps For Generating A Relative XPath:

1. Open Firefox browser
2. Load a web application “i.e., [Facebook](#)”
3. Right click the WebElement “i.e., radio button”

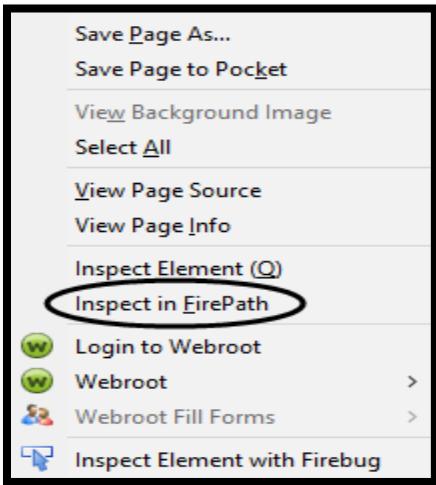
3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

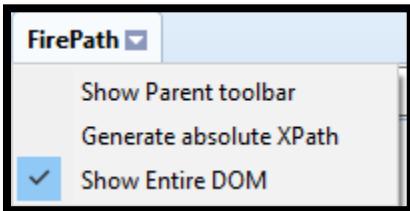
Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

4. Select Inspect in FirePath



5. Uncheck the option “Generate absolute XPath”



Result: The XPath text box displays a string parameter for relative XPath.

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>



Figure 8.6 – Relative XPath For The Female Radio Button

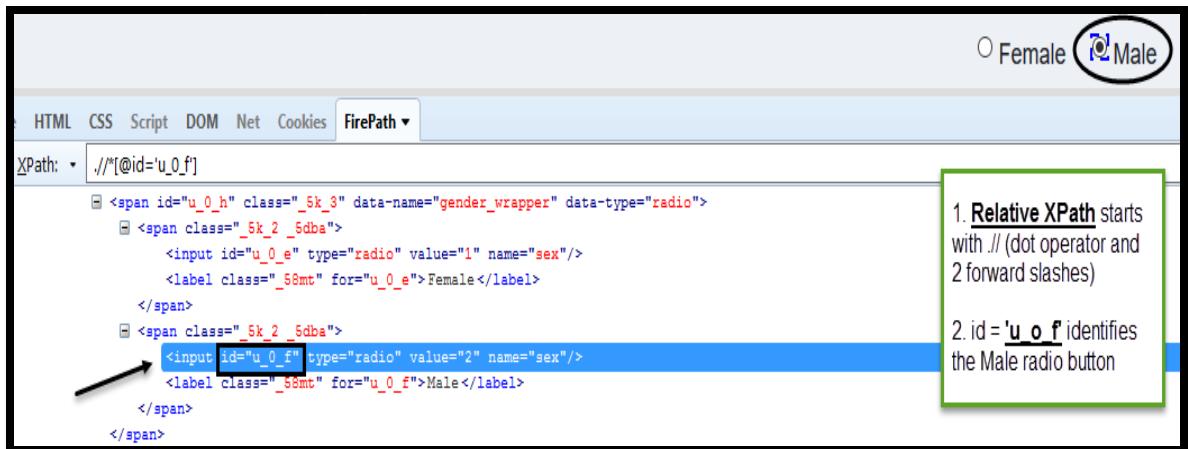


Figure 8.7 – Relative XPath For The Male Radio Button

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

Note: FirePath can also be accessed via Firefox > F12 or Tools > Firebug > Open Firebug then navigate to the FirePath tab.

The relative XPath finds the nearest id attribute for the chosen WebElement. In this case, the value for Female id attribute is u_0_e and Male id attribute is u_0_f. However, the id attribute value of “u_0_h” would have been chosen as the relative XPath if the radio buttons “Female & Male” did not contain an id attribute.

It is recommended to use the relative XPath rather than the absolute XPath. Both XPaths are valuable but the absolute XPath is more vulnerable to an HTML code change. The chosen WebElement “i.e., radio button” becomes invalid if a developer adds or removes a tag prior to the WebElement. For example, the Test Script will not be functional if a developer modifies the HTML code “i.e., div tag” prior to span[1] in [Figure 8.4](#).

Select A Radio Button via Relative XPath

The following is code for selecting the Female radio button on [Facebook's](#) Sign Up Page via XPath locator type:

```
package FacebookSignUpPage;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.*;

public class SelectRadioButton
{
    WebDriver driver;

    @BeforeTest
    public void setUp () throws Exception
    {
        System.setProperty("webdriver.chrome.driver", "C:\\Users\\REX A
JONES\\Downloads\\chromedriver_win32\\chromedriver.exe");
        driver = new ChromeDriver ();
    }
}
```

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

```
        driver.get("https://www.facebook.com/");
    }

    @AfterTest
    public void tearDown () throws Exception
    {
        driver.quit();
    }

    @Test
    public void selectGender ()
    {
        driver.findElement(By.xpath("//*[@id='u_0_e']")).click();
    }
}
```

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

Chapter 8

Find WebElement By XPath

(Part 1) Selenium WebDriver

```

1 package FacebookSignUpPage;
2 import org.openqa.selenium.By;
3
4 public class SelectRadioButton
5 {
6     WebDriver driver;
7
8     @BeforeTest
9     public void setUp () throws Exception
10    {
11        System.setProperty("webdriver.chrome.driver", "C:\\Users\\REX A JONES\\Downloads\\chromedriver_win32\\chromedriver.exe");
12        driver = new ChromeDriver ();
13
14        driver.get("https://www.facebook.com/");
15
16    }
17
18    @AfterTest
19    public void tearDown () throws Exception
20    {
21        driver.quit();
22    }
23
24
25    @Test
26    public void selectGender ()
27    {
28        driver.findElement(By.xpath("//*[@id='u_0_e']")).click();
29    }
30
31 }
```

1. The `get()` method loads Facebook's Sign Up page
2. The Radio Button WebElement is found using `findElement()` and locator type `By.xpath`
3. The radio button is clicked using `click()`
4. The `quit()` method closes the

Figure 8.8 – Perform Actions On The Radio Button

- Line 9 “WebDriver `driver`” is the interface for driving the browser. Currently, the object reference variable “`driver`” points to nothing but will point to a Chrome Driver object in a subsequent line “`driver = new ChromeDriver()`”.
- Line 14 tells Selenium where the executable file for Chrome driver is located via `System.setProperty`. The executable file operates like a bridge between Selenium and the browser. All browsers except Firefox require an executable file. Steps for downloading the executable file are located in [Chapter 1 – Download Browser Drivers](#) section. The following are parameters for `System.setProperty`:
 - key** = `webdriver.chrome.driver`
 - value** = `C:\\Users\\REX A JONES\\Downloads\\chromedriver_win32\\chromedriver.exe`

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.orgLinkedIn: <https://www.linkedin.com/in/rexjones34>

Note: Value is the path location of the executable file

- Line 15 “`driver = new ChromeDriver()`” is an implementation of the WebDriver interface. The object reference variable “`driver`” is pointing to `new ChromeDriver()` which means testing is controlled on the Chrome browser.
- Line 17 “`driver.get("https://www.facebook.com/")`” loads a new Facebook page in the current browser window
- Line 23 “`driver.quit()`” quits the driver instance and closes the open browser window
- Line 29 “`driver.findElement(By.xpath("//*[@id='u_0_e']")).click()`”:
 - `driver` – WebDriver object reference variable that assist with finding a WebElement
 - `findElement` – a WebDriver method that finds the radio button WebElement “Female” on Facebook’s Sign Up page
 - (`By.xpath("//*[@id='u_0_e']")`) – `By` and `xpath` are parameters of the `findElement` WebDriver method. `By` is an object which locate elements and `xpath` is the locator type. The XPath locator type accepts a string parameter “`//*[@id='u_0_e']`” which is the relative XPath for the Female radio button.
 - `click()` – clicks the Female radio button

Note: The dot (.) operator which is the first character in FirePath’s relative XPath must be removed before executing the test script.

- FirePath’s relative XPath contains the dot operator “`//*[@id='u_0_e']`”
- Test Script removes the dot operator (`By.xpath("//*[@id='u_0_e']")`)

The following is a screenshot of [Facebook’s](#) Sign Up page after selecting the Female radio button:

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

The screenshot shows a 'Sign Up' form with the following fields:

- First name
- Last name
- Mobile number or email
- Re-enter mobile number or email
- New password

Birthday input fields (Month, Day, Year) with a link to 'Why do I need to provide my birthday?'.
Gender selection radio buttons: Female (selected, circled) and Male.
A note below states: "By clicking Sign Up, you agree to our [Terms](#) and that you have read our [Data Policy](#), including our [Cookie Use](#)."
A large green 'Sign Up' button at the bottom.

Figure 8.9 – Select The Female Radio Button

Relative XPath Alternatives To FirePath

The following HTML screenshot of [Facebook's](#) Female and Male radio buttons will be used to explain alternatives for executing relative XPath:

Skype: rex.jones34
Twitter: @RexJonesII
Email: Rex.Jones@Test4Success.org
LinkedIn: <https://www.linkedin.com/in/rexjones34>

```

<span id="u_0_h" class="_5k_3" data-name="gender_wrapper" data-type="radio">
  <span class="_5k_2 _5dba">
    <input id="u_0_e" type="radio" value="1" name="sex"/>
    <label class=_58mt for="u_0_e">Female</label>
  </span>
  <span class="_5k_2 _5dba">
    <input id="u_0_f" type="radio" value="2" name="sex"/>
    <label class=_58mt for="u_0_f">Male</label>
  </span>
</span>

```

Figure 8.10 – HTML For Female and Male Radio Buttons On Facebook's Sign Up Page (2)

The relative xpath's locator (By.xpath("//*[@id='u_0_e']")) was used to click the Female radio button. An asterisk (*) in the relative XPath represents any tag within the HTML code. Therefore, an automation engineer can choose to be more specific and replace the asterisk with a tag name. The following is the syntax for finding a WebElement by relative XPath:

Syntax

//TagName[@AttributeName='AttributeValue']

The following code shows how input is a suitable tag name for clicking the radio button.

(By.xpath("//input[@id='u_0_e']"))

According to the syntax, the attribute “id” can be replaced with any attribute. An attribute such as name or type that identifies the WebElement is suitable. However, there are attributes/values that are duplicated within HTML’s source code. If duplicate attributes/values exist for a WebElement then an automation engineer can use multiple attributes. Multiple attributes are beneficial if it is difficult find a unique attribute/value. The following are examples for utilizing a single attribute and multiple attributes:

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

- (By.xpath("//*[@for='u_0_e']"))
- (By.xpath("//label[@for='u_0_e']"))
- (By.xpath("//*[@type='radio' and @value='1']"))
- (By.xpath("//input[@type='radio' and @name='sex' and @value='1']"))

Working With Dynamic WebElements Using XPath

There are situations when a web application may contain dynamic WebElements. A dynamic WebElement is when the value on an application changes at runtime. For example, the time will change every second/minute on an application during runtime.

In addition, a dynamic WebElement manifests when the value of an attribute changes upon reloading the web application. The Home page of [Yahoo](#) is a perfect example of a dynamic WebElement. All of the list box WebElements change when the page reloads. A user has to search for information, select an option from the list of options, and then return to [Yahoo's](#) Home page. The following screenshots show how the same set of options contains a different value for ID attribute:

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

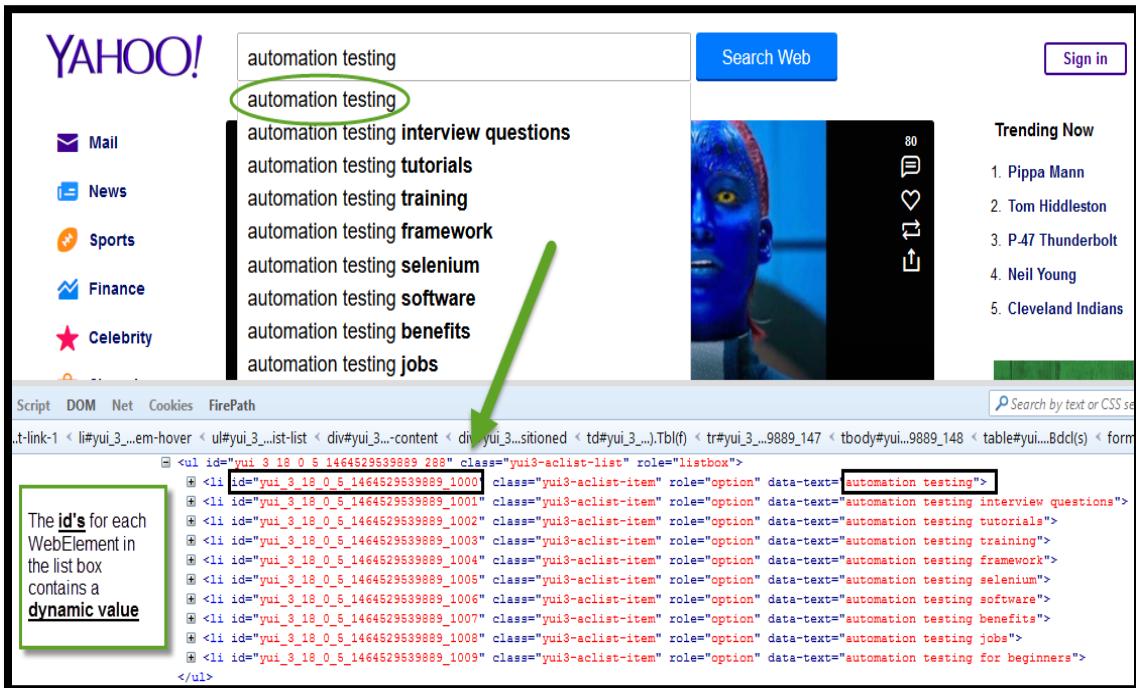


Figure 8.11 – HTML For Yahoo’s Home Page Listbox (1)

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

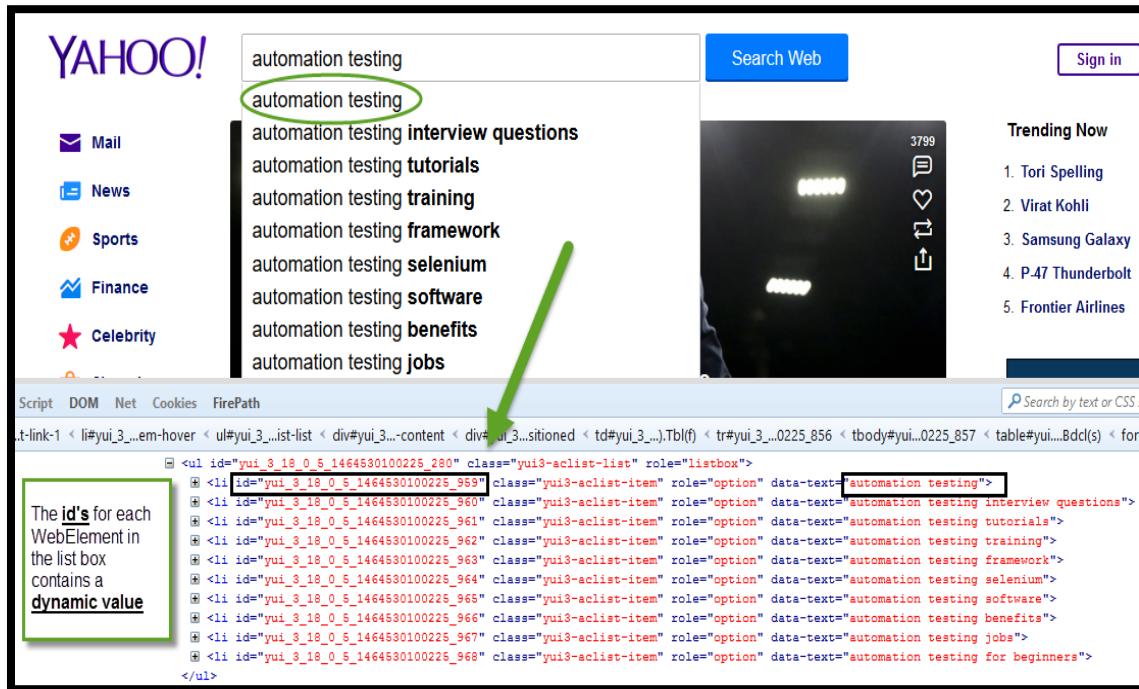


Figure 8.12 – HTML For Yahoo’s Home Page Listbox (2)

The screenshots display a dynamic value for ID attribute. In particular, the ID changes for list box option “automation testing” from “yui_3_18_0_5_1464529539889_1000” to “yui_3_18_0_5_1464530100225_959”.

An exception occurs if the locator attempts to find a dynamic element. The Test Script will not find the element because the value changed. Nevertheless, the Relative XPath can find dynamic WebElements utilizing a partial pattern match. A partial pattern match locates a WebElement based on part of the value. The following is the syntax for using a partial pattern match via relative XPath:

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Syntax

[PartialPatternMatch(@AttributeName,’AttributeValue’)]

Assume the value for an ID attribute changes from “Test123” to “Test456”, “123Test” to “456Test”, and “123Test456” to “789Test123”. The constant value in each example is Test. As a result, an automation engineer can manually write a partial pattern match to find the dynamic WebElement. The following are three partial pattern match types that assist with dynamic values: starts-with, ends-with, contains

1. **starts-with** – finds a WebElement if the value starts with a constant string value “Test123” and “Test456”

`driver.findElement(By.xpath("//[starts-with(@id,'Test')]"))`

2. **ends-with** – finds a WebElement if the value ends with a constant string value “123Test” and “456Test”

`driver.findElement(By.xpath("//[ends-with(@id,'Test')]"))`

3. **contains** – finds a WebElement if the value contains a constant string value “123Test456” and “789Test123”

`driver.findElement(By.xpath("//[contains(@id,'Test')]"))`

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

Chapter 9

Find WebElement By CSS Selector

Cascade Style Sheet “CSS” Selector is a robust method for finding WebElements. This locator type is very similar to the XPath locator type. An automation engineer can manually create a CSS Selector from the web page’s HTML or use FirePath. Steps for installing FirePath are located in [Chapter 1 – Install Firebug and FirePath](#) section. The following is a screenshot “How To Select CSS” from FirePath:

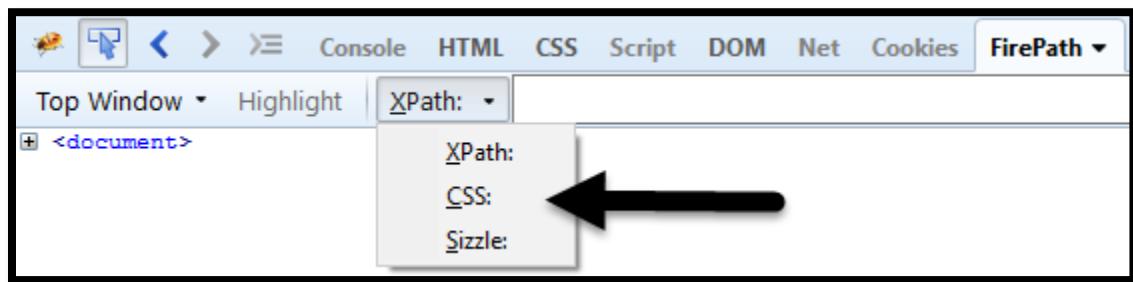


Figure 9.1 – FirePath / CSS Selector Screenshot

The CSS feature within FirePath provides an ability to retrieve a string parameter for CSS Selector locator. Below are screenshots of [Facebook's](#) Sign Up page, its markup tags for radio buttons “Female and Male”, followed by steps to generate a CSS Selector string parameter:

Skype: rex.jones34
Twitter: @RexJonesII
Email: Rex.Jones@Test4Success.org
LinkedIn: <https://www.linkedin.com/in/rexjones34>

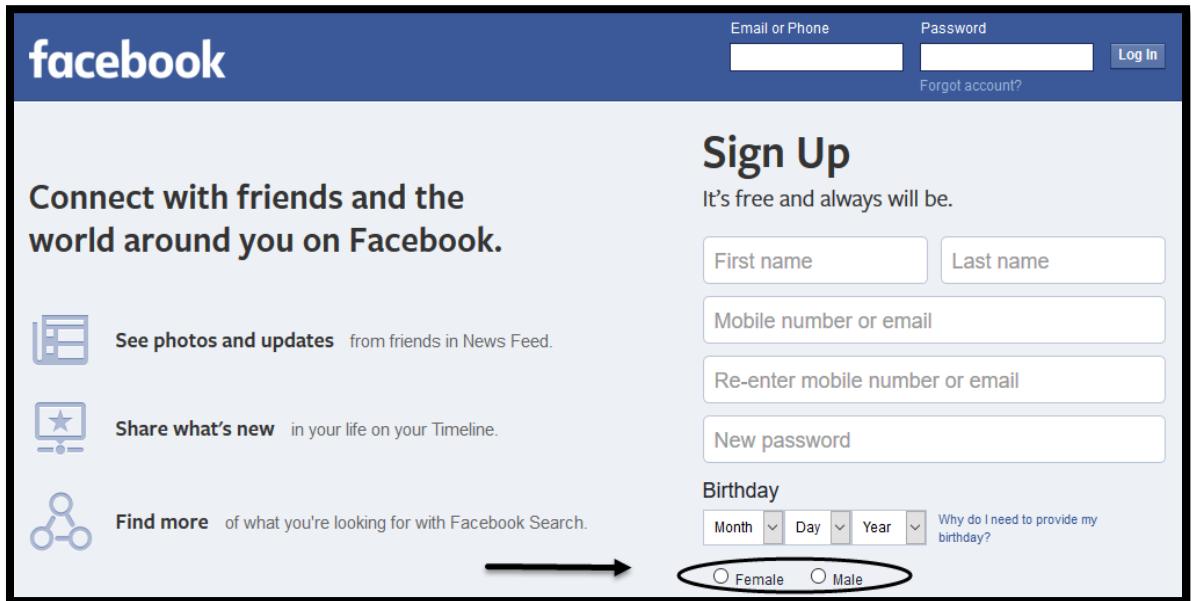


Figure 9.2 – Facebook’s Sign Up Page / Female and Male Radio Buttons

```




<label class=_58mt for="u_0_e">Female</label>



<label class=_58mt for="u_0_f">Male</label>



```

Figure 9.3 – HTML For Female and Male Radio Buttons On Facebook’s Sign Up Page

3 Tips To Master Selenium Within 30 Days

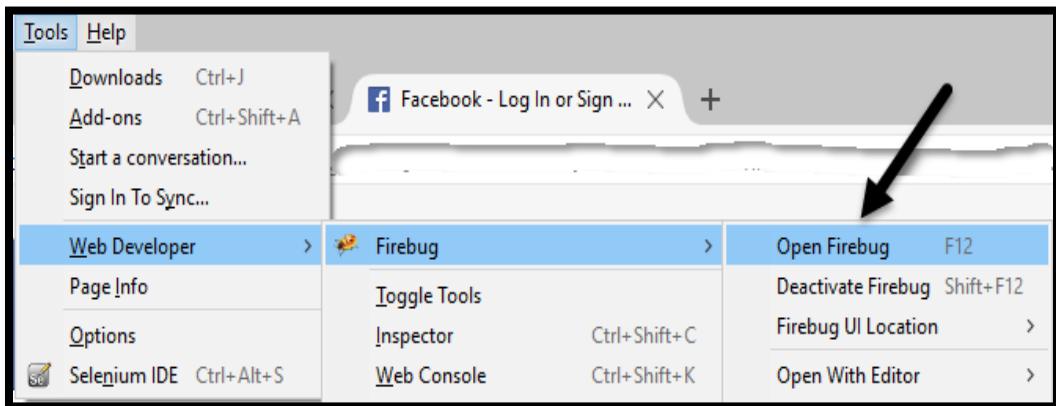
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

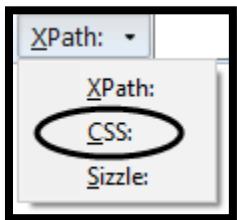
<http://tinyurl.com/Free-QTP-UFT-Selenium>

Steps for generating a CSS Selector:

1. Open Firefox browser
2. Load a web application “i.e., [Facebook](#)”
3. Click shortcut “F12” or Tools > Web Developer > Firebug > Open Firebug



4. Navigate to the FirePath tab within Firebug
5. Click the drop down for XPath
6. Select CSS:



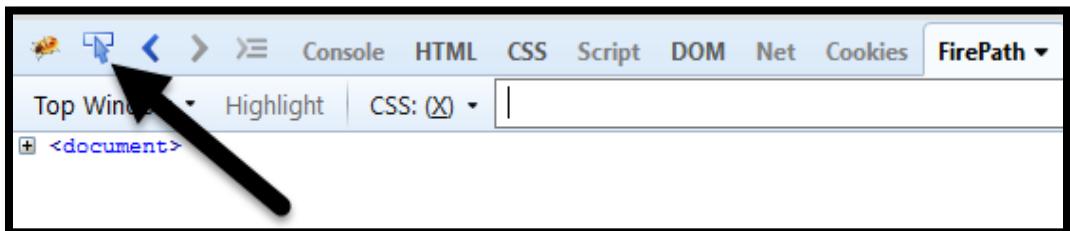
Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

7. Click the WebElement Inspector



8. Select the WebElement "i.e., radio button"

Result: The CSS text box displays a string parameter for CSS Selector.

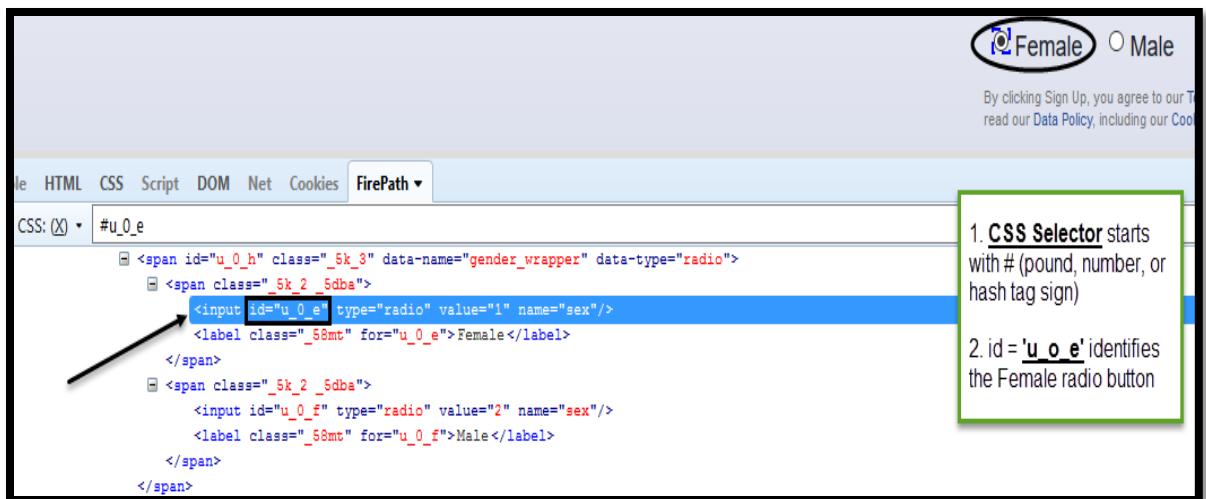


Figure 9.4 – CSS Selector For The Female Radio Button

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

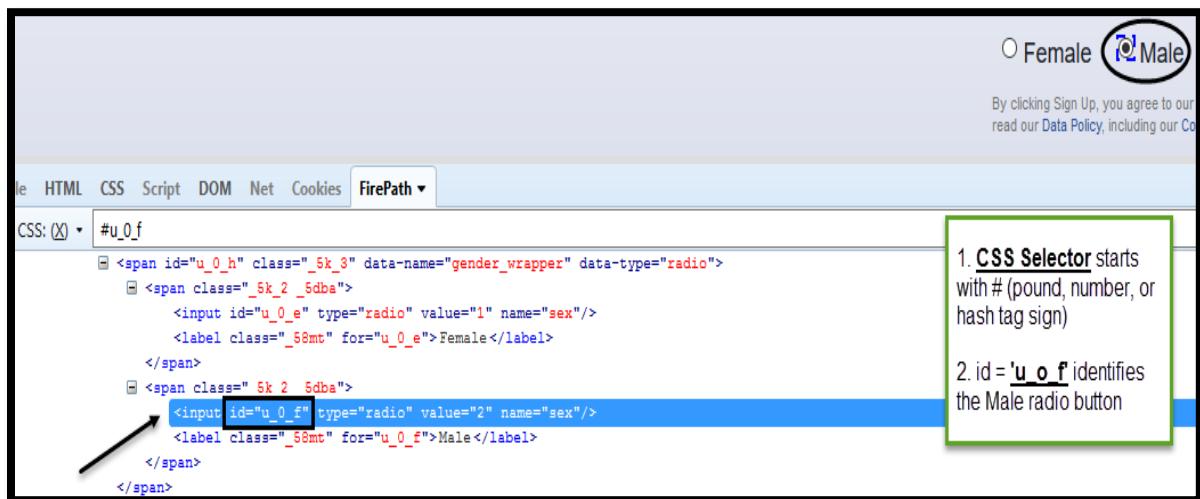


Figure 9.5 – CSS Selector For The Male Radio Button

Select A Radio Button via CSS Selector

The following is code for selecting the Male radio button on [Facebook's](#) Sign Up Page via CSS Selector locator type:

```
package FacebookSignUpPage;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.*;

public class SelectRadioButton2
{
    WebDriver driver;

    @BeforeTest
    public void setUp () throws Exception
    {
```

Skype: rex.jones34
 Twitter: @RexJonesII
 Email: Rex.Jones@Test4Success.org
 LinkedIn: <https://www.linkedin.com/in/rexjones34>

```
System.setProperty("webdriver.chrome.driver", "C:\\Users\\REX A  
JONES\\Downloads\\chromedriver_win32\\chromedriver.exe");  
driver = new ChromeDriver();  
  
driver.get("https://www.facebook.com/");  
}  
  
@AfterTest  
public void tearDown () throws Exception  
{  
    driver.quit();  
}  
  
@Test  
public void selectGender ()  
{  
    driver.findElement(By.cssSelector("#u_0_f")).click();  
}  
}
```

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

Chapter 9

Find WebElement By CSS Selector

(Part 1) Selenium WebDriver

```

1 package FacebookSignUpPage;
2 import org.openqa.selenium.By;
3
4 public class SelectRadioButton2
5 {
6     WebDriver driver;
7
8     @BeforeTest
9     public void setUp () throws Exception
10    {
11        System.setProperty("webdriver.chrome.driver", "C:\\Users\\REX A JONES\\Downloads\\chromedriver_win32\\chromedriver.exe");
12        driver = new ChromeDriver ();
13
14        driver.get("https://www.facebook.com/");
15    }
16
17    @AfterTest
18    public void tearDown () throws Exception
19    {
20        driver.quit();
21    }
22
23    @Test
24    public void selectGender ()
25    {
26        driver.findElement(By.cssSelector("#u_0_f")).click();
27    }
28
29
30
31 }
```

1. The `get()` method loads Facebook's Sign Up page
2. The Radio Button WebElement is found using `findElement()` and locator type `By.cssSelector`
3. The radio button is clicked using `click()`
4. The `quit()` method closes the

Figure 9.6 – Perform Actions On The Radio Button

- Line 9 “WebDriver `driver`” is the interface for driving the browser. Currently, the reference “`driver`” points to nothing but will point to a Chrome Driver object in a subsequent line “`driver = new ChromeDriver()`”.
- Line 14 tells Selenium where the executable file for Chrome driver is located via `System.setProperty`. The executable file operates like a bridge between Selenium and the browser. All browsers except Firefox require an executable file. Steps for downloading the executable file are located in [Chapter 1 – Download Browser Drivers](#) section. The following are parameters for `System.setProperty`:
 - key** = `webdriver.chrome.driver`
 - value** = `C:\\Users\\REX A JONES\\Downloads\\chromedriver_win32\\chromedriver.exe`

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.orgLinkedIn: <https://www.linkedin.com/in/rexjones34>

Note: Value is the path location of the executable file

- Line 15 “`driver = new ChromeDriver()`” is an implementation of the WebDriver interface. The reference “`driver`” is pointing to `new ChromeDriver()` which means testing is controlled on the Chrome browser.
- Line 17 “`driver.get("https://www.facebook.com/")`” loads a new Facebook page in the current browser window
- Line 23 “`driver.quit()`” quits the driver instance and closes the open browser window
- Line 29 “`driver.findElement(By.cssSelector("#u_0_f")).click();`”:
 - `driver` – WebDriver object which assist with finding a WebElement
 - `findElement` – a WebDriver method that finds the WebElement “Male” radio button on Facebook’s Sign Up page
 - `(By.cssSelector("#u_0_f"))` – `By` and `cssSelector` are parameters of the `findElement` WebDriver method. `By` is an object which locate elements and `cssSelector` is the locator type. The CSS Selector locator type accepts a string parameter “`#u_0_f`” which is the CSS Selector for the Male radio button.
 - `click()` – clicks the Male radio button

The following is a screenshot of [Facebook’s](#) Sign Up page after selecting the Male radio button:

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

The screenshot shows a 'Sign Up' form with the following fields:

- First name
- Last name
- Mobile number or email
- Re-enter mobile number or email
- New password

Birthday

Month ▾ Day ▾ Year ▾ Why do I need to provide my birthday?

Female Male

By clicking Sign Up, you agree to our [Terms](#) and that you have read our [Data Policy](#), including our [Cookie Use](#).

Sign Up

A red oval highlights the 'Male' radio button.

Figure 9.7 – Select The Male Radio Button

CSS Selector Alternatives To FirePath

The number “#” (also known as pound or hashtag) symbol starts a CSS Selector representing an ID attribute. In the previous example, CSS Selector (By.cssSelector("#u_0_f")) means the ID attribute’s value is “u_0_f” for Male radio button. However, the dot (.) operator is a symbol that represents the Class attribute. The following is the syntax for identifying all elements using class if the class name is Automation:

Syntax

(By.cssSelector(".Automation"))

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

The following is the syntax for identifying an element using class if the tag type is Test and class name is Automation:

Syntax

(By.cssSelector("Test.Automation"))

The first syntax only used a dot operator with the class name (Automation). Therefore, all elements are identified if the class name is Automation. The second syntax used the tag name (Test), dot operator (.), and class name (Automation). A specific element will be identified since the Test Script contains a tag name and class name. CSS Selector has a syntax similar to relative XPath that identify WebElements. The difference between the CSS Selector and relative XPath is the two forward slashes and the at symbol “@” before attribute name:

CSS Selector Syntax

TagName[AttributeName='AttributeValue']

Relative XPath

//TagName[@AttributeName='AttributeValue']

The following is a screenshot of [WordPress](#) and its HTML markup tags for Remember Me checkbox:

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

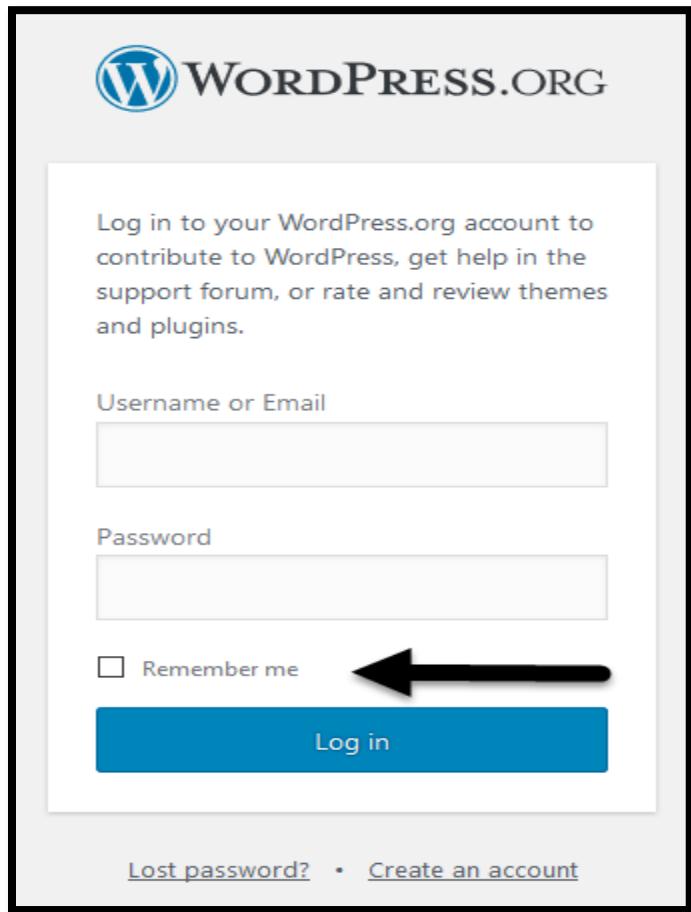


Figure 9.8 – WordPress Remember Me Checkbox

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

```

<p class="login-remember">
  <label>
    <input id="rememberme" type="checkbox" value="forever" name="rememberme">
    Remember me
  </label>
</p>

```

Figure 9.9 – HTML For The Remember Me Checkbox On WordPress Log In Page

The following table compares CSS Selector and relative XPath using the ID attribute for Remember Me checkbox:

CSS Selector	XPath
[id='rememberme']	//[@id='rememberme']
input[id='rememberme']	//input[@id='rememberme']

Figure 9.10 – Similarities Between CSS Selector and Relative XPath

Working With Dynamic WebElements Using CSS Selector

As mentioned earlier, a dynamic WebElement manifests when the value on an application changes or the value of an HTML attribute changes. The WebElement can be found although the value changes at runtime or after a page reloads. CSS Selector has three pattern matching symbols (^, \$, *) that assist with finding dynamic WebElements. The following is the syntax for finding dynamic WebElements via CSS Selector:

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

Syntax

TagName[AttributeName Symbol='AttributeValue']

Assume the value for an ID attribute changes from “Hello123” to “Hello456”, “123Hello” to “456Hello”, and “123Hello456” to “789Hello123”. The constant value in each example is Hello. As a result, an automation engineer can insert a symbol (^, \$, *) to find the dynamic WebElement. The following are three CSS Selector pattern matching symbols that find dynamic values:

1. ^ (caret) – finds a WebElement if the value starts with a constant string value “Hello123” and “Hello456”

```
driver.findElement(By.cssSelector("*[id ^= 'Hello']"))
```

2. \$ (dollar sign) – finds a WebElement if the value ends with a constant string value “123Hello” and “456Hello”

```
driver.findElement(By.cssSelector("*[id $= 'Hello']"))
```

3. * (asterisk) – finds a WebElement if the value contains a constant string value “123Hello456” and “789Hello123”

```
driver.findElement(By.cssSelector("*[id *= 'Hello']"))
```

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Chapter 10

Find WebElement By Link Text

Finding an element by Link Text is carried out via the hyperlink's text name. Using a hyperlink's text name is the best way to click a hyperlink. The entire hyperlink's text name is used as the string parameter for the Link Text locator type. It is important to know that the hyperlink's text name is placed within HTML's anchor 'a' tags and after the href attribute/value. The following is a screenshot of [LinkedIn](#) and its HTML markup tags for the Forgot Password hyperlink:

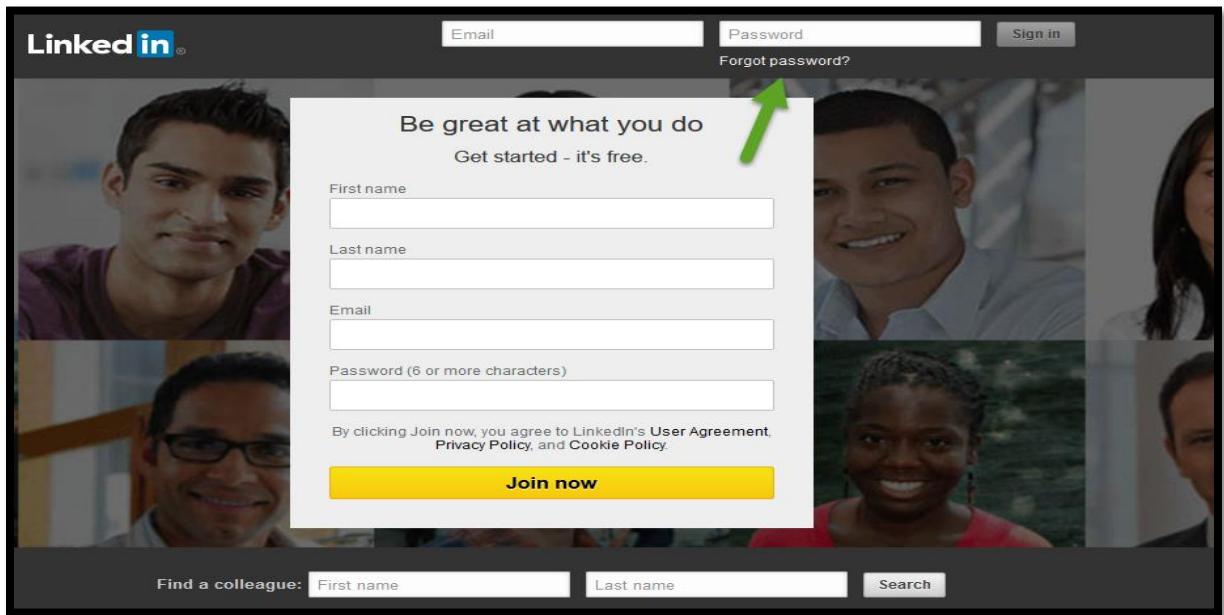


Figure 10.1 – LinkedIn's Home Screen (Forgot Password?)

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

```
<a class="link-forgot-password" href="https://www.linkedin.com/uas/request-password-
reset?trk=uno-reg-guest-home-forgot-password">Forgot password?</a>
```

Figure 10.2 – HTML For The Forgot Password Hyperlink On LinkedIn's Home Page

Click The Hyperlink For Forgot Password (Link Text)

The following is code for clicking the Forgot Password hyperlink on [LinkedIn's](#) Home Page via linkText locator type:

```
package LinkedInHomePage;
import org.openqa.selenium.*;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.*;

public class ForgotPassword
{
    WebDriver driver;

    @BeforeTest
    public void setUp() throws Exception
    {
        System.setProperty("webdriver.chrome.driver", "C:\\\\Users\\\\REX A
JONES\\\\Downloads\\\\chromedriver_win32\\\\chromedriver.exe");
        driver = new ChromeDriver();

        // Go to LinkedIn's Home Page
        driver.get("https://www.linkedin.com/");
    }

    @AfterTest
    public void tearDown() throws Exception
    {
        driver.quit();
    }
}
```

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

```

1 package LinkedInHomePage;
2 import org.openqa.selenium.*;
3
4 public class ForgotPassword
5 {
6     WebDriver driver;
7
8     @BeforeTest
9     public void setUp() throws Exception
10    {
11        System.setProperty("webdriver.chrome.driver", "C:\\Users\\REX A JONES\\Downloads\\chromedriver_win32\\chromedriver.exe");
12        driver = new ChromeDriver();
13
14        driver.get("https://www.linkedin.com/"); ← 1
15    }
16
17    @AfterTest
18    public void tearDown() throws Exception
19    {
20        driver.quit(); ← 4
21    }
22
23    @Test
24    public void clickForgotPassword()
25    {
26        driver.findElement(By.linkText("Forgot password?")).click(); ← 2
27    }
28
29    }
30
31

```

1. The `get()` method loads LinkedIn's Home page

2. The Link Text WebElement is found using `findElement()` and locator type `By.linkText`

3. The hyperlink is clicked using `click()`

4. The `quit()` method closes the browser

Figure 10.3 – Perform Actions On The Forgot Password Hyperlink

- Line 8 “WebDriver `driver`” is the interface for driving the browser. Currently, the object reference variable “`driver`” points to nothing but will point to a Chrome Driver object in a subsequent line “`driver = new ChromeDriver()`”.

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

- Line 13 tells Selenium where the executable file for Chrome driver is located via `System.setProperty`. The executable file operates like a bridge between Selenium and the browser. All browsers except Firefox require an executable file. Steps for downloading the executable file are located in [Chapter 1 – Download Browser Drivers](#) section. The following are parameters for `System.setProperty`:
 - **key** = `webdriver.chrome.driver`
 - **value** = `C:\\\\Users\\\\REX A JONES\\\\Downloads\\\\chromedriver_win32\\\\chromedriver.exe`

Note: Value is the path location of the executable file

- Line 14 “`driver = new ChromeDriver()`” is an implementation of the WebDriver interface. The object reference variable “`driver`” is pointing to `new ChromeDriver()` which means testing is controlled on the Chrome browser.
- Line 16 “`driver.get("https://www.linkedin.com/")`” loads a new LinkedIn Home page in the current browser window
- Line 22 “`driver.quit()`” quits the driver instance and closes the open browser window
- Line 28 “`driver.findElement(By.linkText("Forgot password?")).click()`”:
 - `driver` – WebDriver object reference variable that assist with finding a WebElement
 - `findElement` – a WebDriver method that finds the hyperlink WebElement “Forgot Password” on LinkedIn’s Home page
 - `(By.linkText("Forgot password"))` – `By` and `linkText` are parameters of the `findElement` WebDriver method. `By` is an object which locate elements and `linkText` is the locator type. The Link Text locator type accepts a string parameter “Forgot password” which is the text name of the hyperlink
 - `click()` – clicks the Forgot Password hyperlink

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Note: The `submit()` method can also be used to click the Forgot Password hyperlink.

The following is a screenshot of [LinkedIn's](#) Home page after clicking the Forgot Password hyperlink:

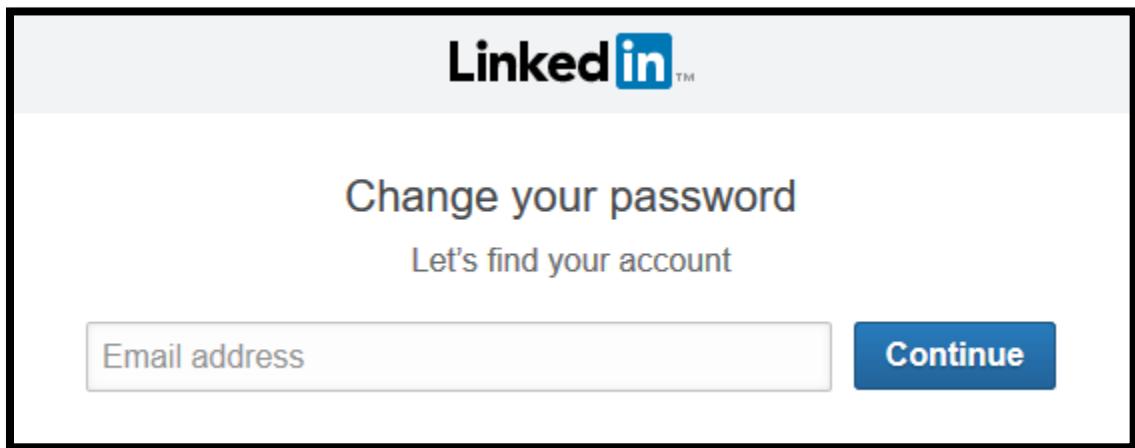


Figure 10.4 – Change Your Password

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

Chapter 11

Find WebElement By Partial Link Text

Finding an element by Partial Link Text is carried out similar to Link Text. Both techniques only work for clicking hyperlinks. However, the difference between Partial Link Text and Link Text is hyperlink's text name. Part of the hyperlink's text name is suitable for the Partial Link Text locator type. For example, an automation engineer can enter "Password" and the "Forgot Password" link is clicked. The following is a screenshot of [LinkedIn](#) and its HTML markup tags for the Forgot Password hyperlink:

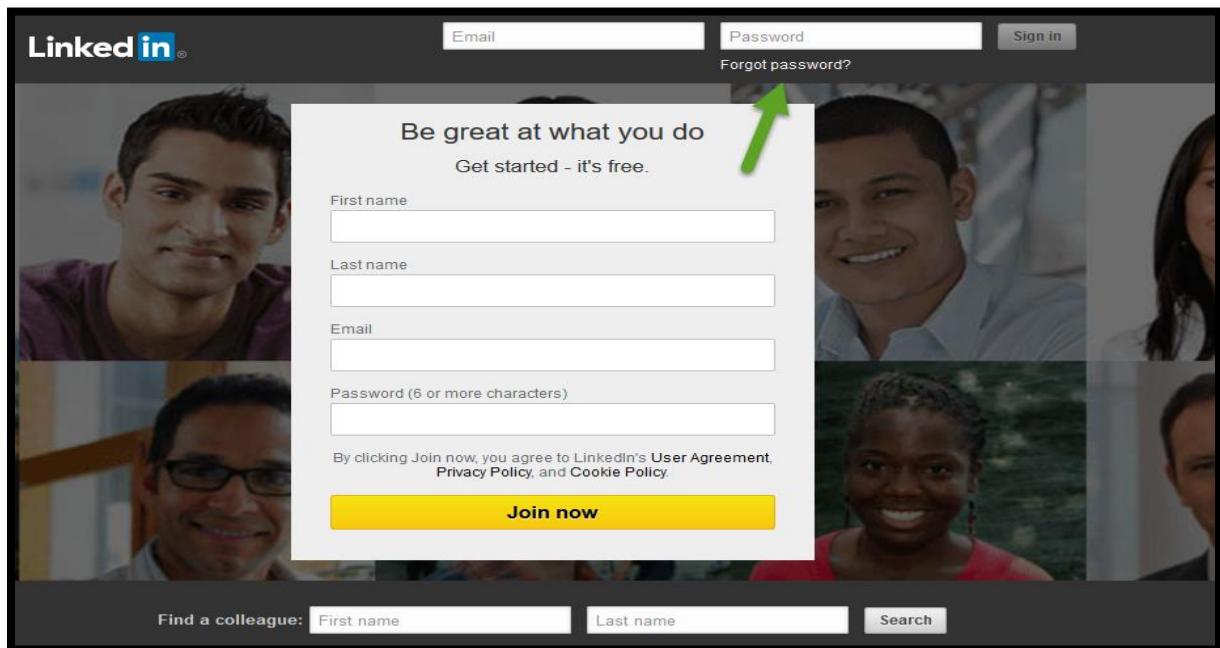


Figure 11.1 – LinkedIn's Home Screen (Forgot Password?)

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

```
<a class="link-forgot-password" href="https://www.linkedin.com/uas/request-password-
reset?trk=uno-reg-guest-home-forgot-password">Forgot password?</a>
```

Figure 11.2 – HTML For The Forgot Password Hyperlink On LinkedIn's Home Page (2)

Click The Hyperlink For Forgot Password (Partial Link Text)

The following is code for clicking the Forgot Password hyperlink on [LinkedIn's](#) Home Page via partialLinkText locator type:

```
package LinkedInHomePage;
import org.openqa.selenium.*;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.*;

public class ForgotPassword2
{
    WebDriver driver;

    @BeforeTest
    public void setUp() throws Exception
    {
        System.setProperty("webdriver.chrome.driver", "C:\\\\Users\\\\REX A
JONES\\\\Downloads\\\\chromedriver_win32\\\\chromedriver.exe");
        driver = new ChromeDriver();

        driver.get("https://www.linkedin.com/");
    }

    @AfterTest
    public void tearDown() throws Exception
    {
        driver.quit();
    }
}
```

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

```

1 @Test
2 public void clickForgotPassword()
3 {
4     driver.findElement(By.partialLinkText("Forgot")).click();
5 }
6

```

```

1 package LinkedInHomePage;
2 import org.openqa.selenium.*;
3
4 public class ForgotPassword2
5 {
6     WebDriver driver;
7
8     @BeforeTest
9     public void setUp() throws Exception
10    {
11        System.setProperty("webdriver.chrome.driver", "C:\\Users\\REX A JONES\\Downloads\\chromedriver_win32\\chromedriver.exe");
12        driver = new ChromeDriver();
13
14        driver.get("https://www.linkedin.com/");
15    }
16
17    @AfterTest
18    public void tearDown() throws Exception
19    {
20        driver.quit();
21    }
22
23    @Test
24    public void clickForgotPassword()
25    {
26        driver.findElement(By.partialLinkText("Forgot")).click();
27    }
28
29 }
30
31

```

1. The `get()` method loads LinkedIn's Home page

2. The Link Text WebElement is found using `findElement()` and locator type `By.partialLinkText`

3. The hyperlink is clicked using `click()`

4. The `quit()` method closes the browser

Figure 11.3 – Perform Actions On The Forgot Password Hyperlink

- Line 8 “WebDriver `driver`” is the interface for driving the browser. Currently, the object reference variable “`driver`” points to nothing but will point to a Chrome Driver object in a subsequent line “`driver = new ChromeDriver()`”.
- Line 13 tells Selenium where the executable file for Chrome driver is located via `System.setProperty`. The executable file operates like a bridge between Selenium and the browser. All browsers except Firefox require an executable file. Steps for

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.orgLinkedIn: <https://www.linkedin.com/in/rexjones34>

downloading the executable file are located in [Chapter 1 – Download Browser Drivers](#) section. The following are parameters for `System.setProperty`:

- **key** = webdriver.chrome.driver
- **value** = C:\\Users\\REX A JONES\\Downloads\\chromedriver_win32\\chromedriver.exe

Note: Value is the path location of the executable file

- Line 14 “`driver = new ChromeDriver()`” is an implementation of the WebDriver interface. The object reference variable “`driver`” is pointing to `new ChromeDriver()` which means testing is controlled on the Chrome browser.
- Line 16 “`driver.get("https://www.linkedin.com/")`” loads a new LinkedIn Home page in the current browser window
- Line 22 “`driver.quit()`” quits the driver instance and closes the open browser window
- Line 28 “`driver.findElement(By.partialLinkText("Forgot")).click()`”:
 - `driver` – WebDriver object reference variable that assist with finding a WebElement
 - `findElement` – a WebDriver method that finds the hyperlink WebElement “Forgot Password” on LinkedIn’s Home page
 - `(By.partialLinkText("Forgot"))` – By and `partialLinkText` are parameters of the `findElement` WebDriver method. By is an object which locate elements and `partialLinkText` is the locator type. The Partial Link Text locator type accepts a string parameter “Forgot” which is part of the hyperlink’s text name..
 - `click()` – clicks the Forgot Password hyperlink

Note: The `submit()` method can also be used to click the Forgot Password hyperlink.

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

The following is a screenshot of [LinkedIn's](#) Home page after clicking the Forgot Password hyperlink:

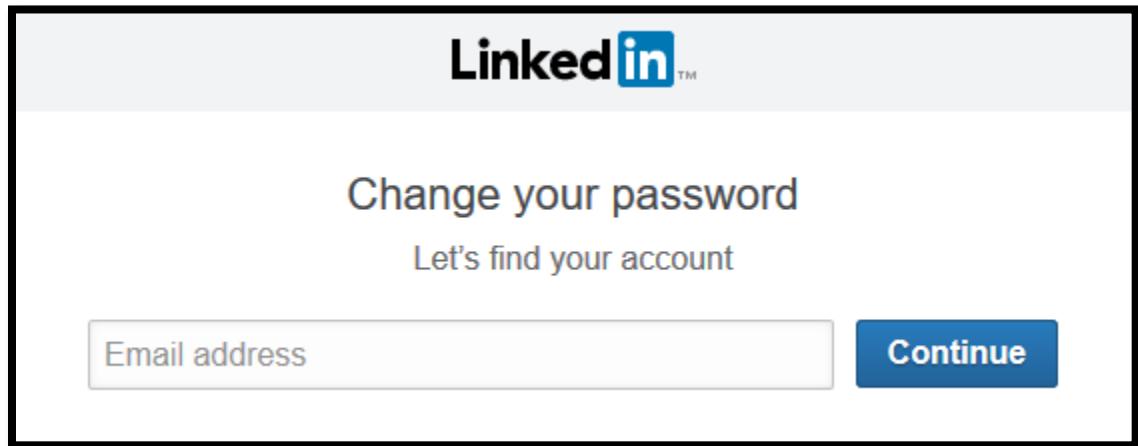


Figure 11.4 – Change Your Password

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Chapter 12

Find WebElement By Tag Name

In HTML, multiple elements share the same Tag Name. Therefore finding an element by Tag Name is limited. It is best to use another locator type in conjunction with the Tag Name locator type to identify an element. The following is a screenshot of [LinkedIn](#) and its HTML markup tags for the Join Now button:

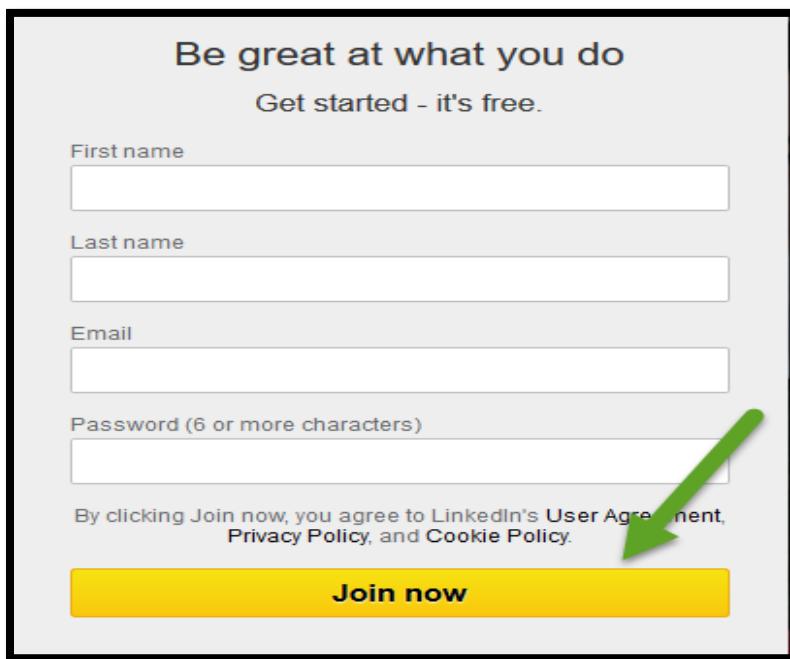


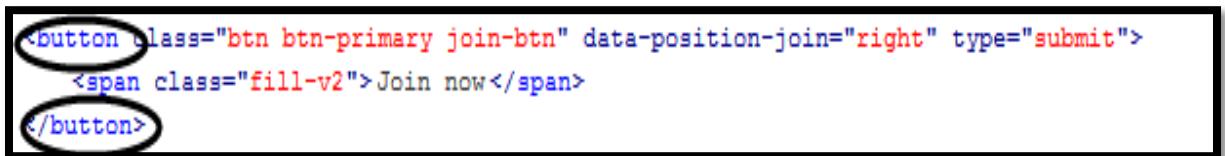
Figure 12.1 – LinkedIn’s Join Now Button

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>



```
<button class="btn btn-primary join-btn" data-position-join="right" type="submit">
    <span class="fill-v2">Join now</span>
</button>
```

Figure 12.2 – HTML For The Join Now Button On LinkedIn's Home Page

Click The Join Now Button

The following is code for clicking the Join Now button on [LinkedIn's](#) Home Page via tagname locator type:

```
package LinkedInHomePage;
import org.openqa.selenium.*;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.*;

public class JoinNow
{
    WebDriver driver;

    @BeforeTest
    public void setUp() throws Exception
    {
        System.setProperty("webdriver.chrome.driver", "C:\\\\Users\\\\REX A
JONES\\\\Downloads\\\\chromedriver_win32\\\\chromedriver.exe");
        driver = new ChromeDriver();

        driver.get("https://www.linkedin.com/");
    }

    @AfterTest
    public void tearDown() throws Exception
    {
        driver.quit();
    }
}
```

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

```

1 @Test
2 public void clickJoinNowButton ()
3 {
4     // This button can be clicked using submit() or click()
5     driver.findElement(By.tagName("button")).submit();
6 }
7 }
```

```

1 package LinkedInHomePage;
2 import org.openqa.selenium.*;
3
4 public class JoinNow
5 {
6     WebDriver driver;
7
8     @BeforeTest
9     public void setUp() throws Exception
10    {
11        System.setProperty("webdriver.chrome.driver", "C:\\Users\\REX A JONES\\Downloads\\chromedriver_win32\\chromedriver.exe");
12        driver = new ChromeDriver();
13
14        driver.get("https://www.linkedin.com/");
15    }
16
17    @AfterTest
18    public void tearDown() throws Exception
19    {
20        driver.quit();
21    }
22
23    @Test
24    public void clickJoinNowButton ()
25    {
26        // This button can be clicked using submit() or click()
27        driver.findElement(By.tagName("button")).submit();
28    }
29 }
30 }
```

1. The `get()` method loads LinkedIn's Home page

2. The Button WebElement is found using `findElement()` and locator type `By.tagName`

3. The button is clicked using `submit()`

4. The `quit()` method closes the browser

Figure 12.3 – Perform Actions On The Join Now Button

- Line 8 “WebDriver `driver`” is the interface for driving the browser. Currently, the object reference variable “`driver`” points to nothing but will point to a Chrome Driver

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

object in a subsequent line “`driver = new ChromeDriver()`”.

- Line 13 tells Selenium where the executable file for Chrome driver is located via `System.setProperty`. The executable file operates like a bridge between Selenium and the browser. All browsers except Firefox require an executable file. Steps for downloading the executable file are located in [Chapter 1 – Download Browser Drivers](#) section. The following are parameters for `System.setProperty`:
 - **key** = `webdriver.chrome.driver`
 - **value** = `C:\\\\Users\\\\REX A JONES\\\\Downloads\\\\chromedriver_win32\\\\chromedriver.exe`

Note: Value is the path location of the executable file

- Line 14 “`driver = new ChromeDriver()`” is an implementation of the WebDriver interface. The object reference variable “`driver`” is pointing to `new ChromeDriver()` which means testing is controlled on the Chrome browser.
- Line 16 “`driver.get("https://www.linkedin.com/")`” loads a new LinkedIn Home page in the current browser window
- Line 22 “`driver.quit()`” quits the driver instance and closes the open browser window
- Line 29 “`driver.findElement(By.tagName("button")).submit()`”:
 - `driver` – WebDriver object reference variable that assist with finding a WebElement
 - `findElement` – a WebDriver method that finds the button WebElement “Join Now” on LinkedIn’s Home page
 - `(By.tagName("button"))` – `By` and `tagName` are parameters of the `findElement` WebDriver method. `By` is an object which locate elements and `tagName` is the locator type. The Tag Name locator type accepts a string parameter “button” which is the tag name for the Join Now button.
 - `submit()` – clicks the Join Now button

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Note: The `click()` method can also be used to click the Join Now button. There is only one button on LinkedIn's Home page. Therefore, the Tag Name locator type is used by itself because multiple elements do not share the same Tag Name "button".

The following is a screenshot of [LinkedIn's](#) Home page after clicking the Join Now button without entering required fields:

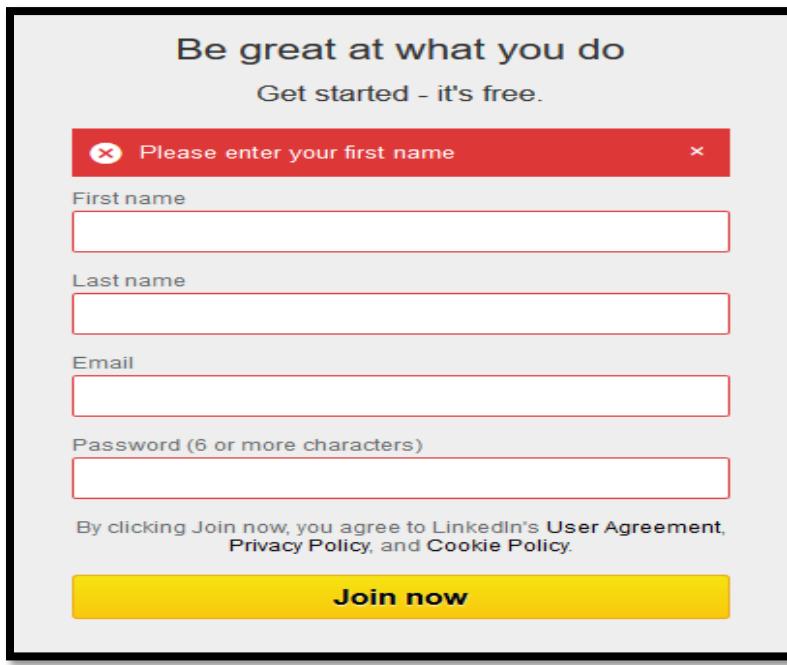


Figure 12.4 – Click The Join Now Button Without Entering Required Fields

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

Chapter 13

Find WebElement By Class

Finding a WebElement by Class is unique from most locator types. A specific WebElement may contain multiple class attributes. On the other hand, there may be multiple WebElements within the same class attribute. If the latter scenario occurs then the first WebElement will be returned if the WebElement is not identified with another locator type. The following is a screenshot of [LinkedIn](#) and its HTML markup tags for the Email text box, Password text box, and Sign In button:

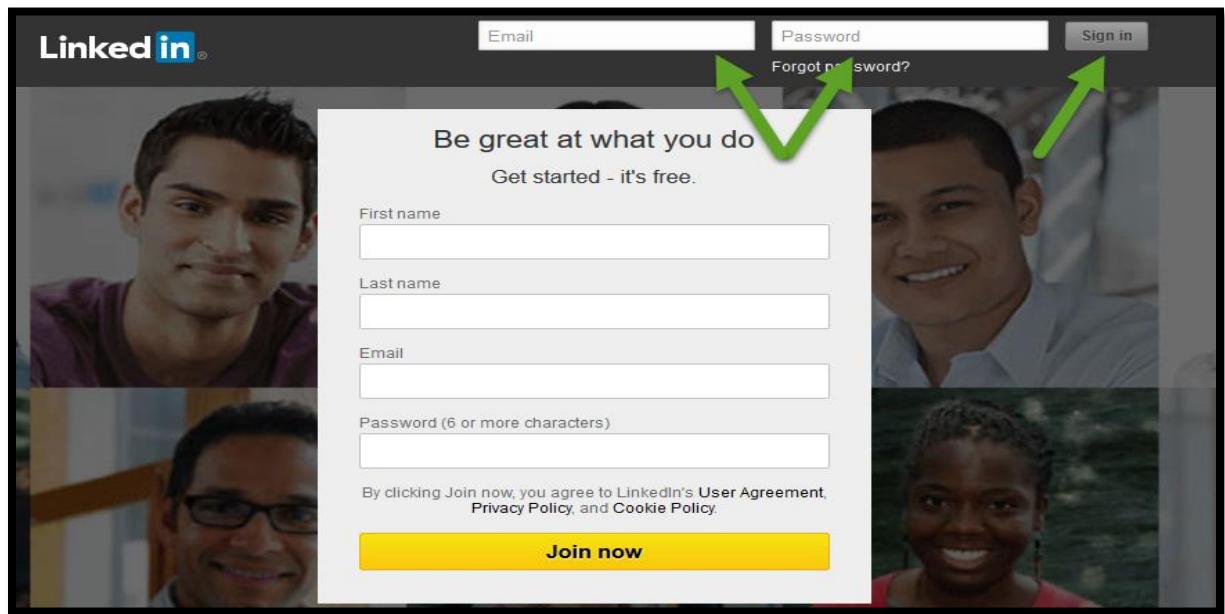


Figure 13.1 – LinkedIn’s Email Text Field, Password Text Field, And Sign In Button

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

```
<form class="login-form" data-autologin="true" method="POST" action="https://www.linkedin.com/uas/login-submit">
    <label for="login-email">Email</label>
    <input id="login-email" type="text" autofocus="autofocus" placeholder="Email" name="session_key">
    <label for="login-password">Password</label>
    <input id="login-password" type="password" placeholder="Password" aria-required="true" name="session_password">
    <input type="hidden" value="false" name="isJsEnabled">
    <input id="loginCsrfParam-login" type="hidden" value="ffaa5956-19fc-4026-86b2-e367e2f2d8ae" name="loginCsrfParam">
    <input id="sourceAlias-login" type="hidden" value="0_7r5yezRXCiA_H0CRD8sf6Dh0jTKUNps5xGTqeX8EEoi" name="sourceAlias">
    <input type="submit" value="Sign in" name="submit">
```

Figure 13.2 – HTML For The Email Text Field, Password Text Field, And Sign In Button

Enter Email, Password, and Click The Sign In Button

The following is code for entering an email, password, and clicking the Sign In button on [LinkedIn's](#) Home Page via classname locator type:

```
package LinkedInHomePage;
import org.openqa.selenium.*;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.*;

public class CaptureErrorMessage
{
    WebDriver driver;

    @BeforeTest
    public void setUp() throws Exception
    {
        System.setProperty("webdriver.chrome.driver", "C:\\\\Users\\\\REX A JONES\\\\Downloads\\\\chromedriver_win32\\\\chromedriver.exe");
        driver = new ChromeDriver();

        driver.get("https://www.linkedin.com/");
    }
}
```

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

```

@AfterTest
public void tearDown() throws Exception
{
    driver.quit();
}

@Test
public void captureSignInErrorMessage()
{

    WebElement parentSignIn = driver.findElement(By.className("login-
form"));

    WebElement childEmail = parentSignIn.findElement(By.id("login-
email"));
    WebElement childPassword = parentSignIn.findElement(By.id("login-
password"));
    WebElement childSignInButton =
parentSignIn.findElement(By.name("submit"));

    // Enter Email and Password then Click the Sign In button
    childEmail.sendKeys("Rex.Jones@Test4Success.org");
    childPassword.sendKeys("34Tester");
    childSignInButton.click();

    // Capture the error messages
    String firstSignInErrorMessage =
driver.findElement(By.className("alert")).getText();
    System.out.println(firstSignInErrorMessage);
}
}

```

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Chapter 13

Find WebElement By Class

(Part 1) Selenium WebDriver

```

1 package LinkedInHomePage;
2 import org.openqa.selenium.*;
3
4 public class CaptureErrorMessage
5 {
6     WebDriver driver;
7
8     @BeforeTest
9     public void setUp() throws Exception
10    {
11         System.setProperty("webdriver.chrome.driver", "C:\\Users\\REX A JONES\\Downloads\\chromedriver_win32\\chromedriver.exe");
12         driver = new ChromeDriver();
13
14         driver.get("https://www.linkedin.com/"); ← 1
15
16     }
17
18     @AfterTest
19     public void tearDown() throws Exception
20     {
21         driver.quit(); ← 5
22     }
23
24     @Test
25     public void captureSignInErrorMessage()
26     {
27
28         WebElement parentSignIn = driver.findElement(By.className("login-form"));
29
30         WebElement childEmail = parentSignIn.findElement(By.id("login-email"));
31         WebElement childPassword = parentSignIn.findElement(By.id("login-password"));
32         WebElement childSignInButton = parentSignIn.findElement(By.name("submit"));
33
34         // Enter Email and Password then Click the Sign In button
35         childEmail.sendKeys("Rex.Jones@Test4Success.org");
36         childPassword.sendKeys("34Tester");
37         childSignInButton.click(); ← 3
38
39         // Capture the error messages
40         String firstSignInErrorMessage = driver.findElement(By.className("alert")).getText();
41         System.out.println(firstSignInErrorMessage); ← 4
42
43     }
44 }

```

1. The `get()` method loads LinkedIn's Home page

2. The text fields (Email & Password) and Sign In button are child WebElements found using `findElement()` but different locator types (`By.id` & `By.name`). The parent WebElement was located using locator type `By.className`

3. Text entered for Email and Password using `sendKeys()` but the Sign In button is clicked using `click()`

4. Capture the error message using `getText()` then prints the message

5. The `quit()` method closes the browser

Figure 13.3 – Perform Actions On Email, Password, And Sign In

Program Output:

There were one or more errors in your submission. Please correct the marked fields below.

- Line 8 “WebDriver `driver`” is the interface for driving the browser. Currently, the object reference variable “`driver`” points to nothing but will point to a Chrome Driver

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

object in a subsequent line “`driver = new ChromeDriver()`”.

- Line 13 tells Selenium where the executable file for Chrome driver is located via `System.setProperty`. The executable file operates like a bridge between Selenium and the browser. All browsers except Firefox require an executable file. Steps for downloading the executable file are located in [Chapter 1 – Download Browser Drivers](#) section. The following are parameters for `System.setProperty`:
 - **key** = `webdriver.chrome.driver`
 - **value** = `C:\\\\Users\\\\REX A JONES\\\\Downloads\\\\chromedriver_win32\\\\chromedriver.exe`

Note: Value is the path location of the executable file

- Line 14 “`driver = new ChromeDriver()`” is an implementation of the WebDriver interface. The object reference variable “`driver`” is pointing to `new ChromeDriver()` which means testing is controlled on the Chrome browser.
- Line 16 “`driver.get("https://www.linkedin.com/")`” loads a new LinkedIn Home page in the current browser window
- Line 22 “`driver.quit()`” quits the driver instance and closes the open browser window
- Line 29
`“WebElement parentSignIn = driver.findElement(By.className("login-form"))”;`
 - WebElement `parentSignIn` - creates an object reference variable called `parentSignIn` that refers or points to a WebElement. Subsequent lines for Email, Password, and Sign In will point to the same form WebElement via `parentSignIn`.
 - `driver` – WebDriver object reference variable that assist with finding the WebElement

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

- `findElement` – a WebDriver method that finds the form WebElement on LinkedIn’s Home page. The form includes an email text box, password text box, Sign In button, and a Forgot Password hyperlink
 - `(By.className("login-form"))` – By and *className* are parameters of the `findElement` WebDriver method. By is an object which locate elements and *className* is the locator type. The Class Name locator type accepts a string parameter “login-form” which is the value of the HTML class attribute.
- Line 31
`“WebElement childEmail = parentSignIn.findElement(By.id("login-email"))”:`
 - WebElement `childEmail` = creates an object reference variable called `childEmail` that refers or points to a WebElement
 - `parentSignIn` – object reference variable that assist with finding the WebElement by pointing to the same form WebElement from line 29
 - `findElement` – a WebDriver method that finds the text box WebElement “Email” on LinkedIn’s Home page.
 - `(By.id("login-email"))` – By and *id* are parameters of the `findElement` WebDriver method. By is an object which locate elements and *id* is the locator type. The ID locator type accepts a string parameter “login-email” which is the value of the HTML *id* attribute.
 - Line 32
`“WebElement childPassword = parentSignIn.findElement(By.id("login-password"))”:`
 - WebElement `childPassword` = creates an object reference variable called `childPassword` that refers or points to a WebElement
 - `parentSignIn` – object reference variable that assist with finding the WebElement by pointing to the same form WebElement from line 29
 - `findElement` – a WebDriver method that finds the text box WebElement “Password” on LinkedIn’s Home page.

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

- (By.id("login-password")) – By and *id* are parameters of the findElement WebDriver method. By is an object which locate elements and id is the locator type. The ID locator type accepts a string parameter “login-password” which is the value of the HTML id attribute.
- Line 33
 “WebElement childSignInButton = parentSignIn.findElement(By.name("submit"));”
 - WebElement childSignInButton = creates an object reference variable called childSignInButton that refers or points to a WebElement
 - parentSignIn – object reference variable that assist with finding the WebElement by pointing to the same form WebElement from line 29
 - **findElement** – a WebDriver method that finds the button WebElement “Sign In” on LinkedIn’s Home page.
 - (By.name("submit")) – By and *name* are parameters of the findElement WebDriver method. By is an object which locate elements and name is the locator type. The Name locator type accepts a string parameter “submit” which is the value of the HTML name attribute.
- Line 36 “childEmail.sendKeys("Rex.Jones@Test4Success.org)” types the text “Rex.Jones@Test4Success.org” in the Email text field via object reference childEmail
- Line 37 “childPassword.sendKeys("34Tester)” types the text “34Tester” in the Password text field via object reference childPassword
- Line 38 “childSignInButton.click()” clicks the Sign In button via object reference childSignInButton

Note: The **submit()** method can also be used to click the Sign In button.

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

- Line 41


```
"String firstSignInErrorMessage = driver.findElement(By.className("alert")).getText()";
```

 - `String firstSignInErrorMessage` = creates an object reference variable called `firstSignInErrorMessage` that refers or points to a string
 - `driver` – WebDriver object reference variable that assist with finding a WebElement
 - `findElement` – a WebDriver method that finds the WebElement string “error message” on LinkedIn’s Sign In page
 - `(By.className("alert"))` – By and `className` are parameters of the `findElement` WebDriver method. By is an object which locate elements and `className` is the locator type. The Class Name locator type accepts a string parameter “alert” which is the value of the HTML class attribute
 - `getText()` – gets the WebElement string
- Line 42 “`System.out.println(firstSignInErrorMessage)`” prints the WebElement string “error message”. View the error message (Class’s [Program Output](#) and/or [Figure 13.4 – Sign In Page After Entering Invalid Credentials](#))

Note: In this example scenario, Email, Password, and Sign In contained unique attribute/values within HTML’s source code. However, the Class locator type is beneficial if there are multiple WebElements with the same attribute/value. The WebElement with a duplicate attribute/value should be located first by finding the value for its class attribute followed by attributes ID, Name, etc. This explains why object reference “parentSignIn” was created for the form followed by Email “childEmail”, Password “childPassword”, and Sign In “childSignInButton”.

Lines 29 – 38 could have been created without references. The following code also enters an email, password, and clicks the Sign In button:

```
driver.findElement(By.id("login-email")).sendKeys("Rex.Jones@Test4Success.org");
driver.findElement(By.id("login-password")).sendKeys("34Tester");
```

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

```
driver.findElement(By.name("submit")).click();
```

The following is a screenshot of [LinkedIn's](#) Sign In page after entering invalid credentials:

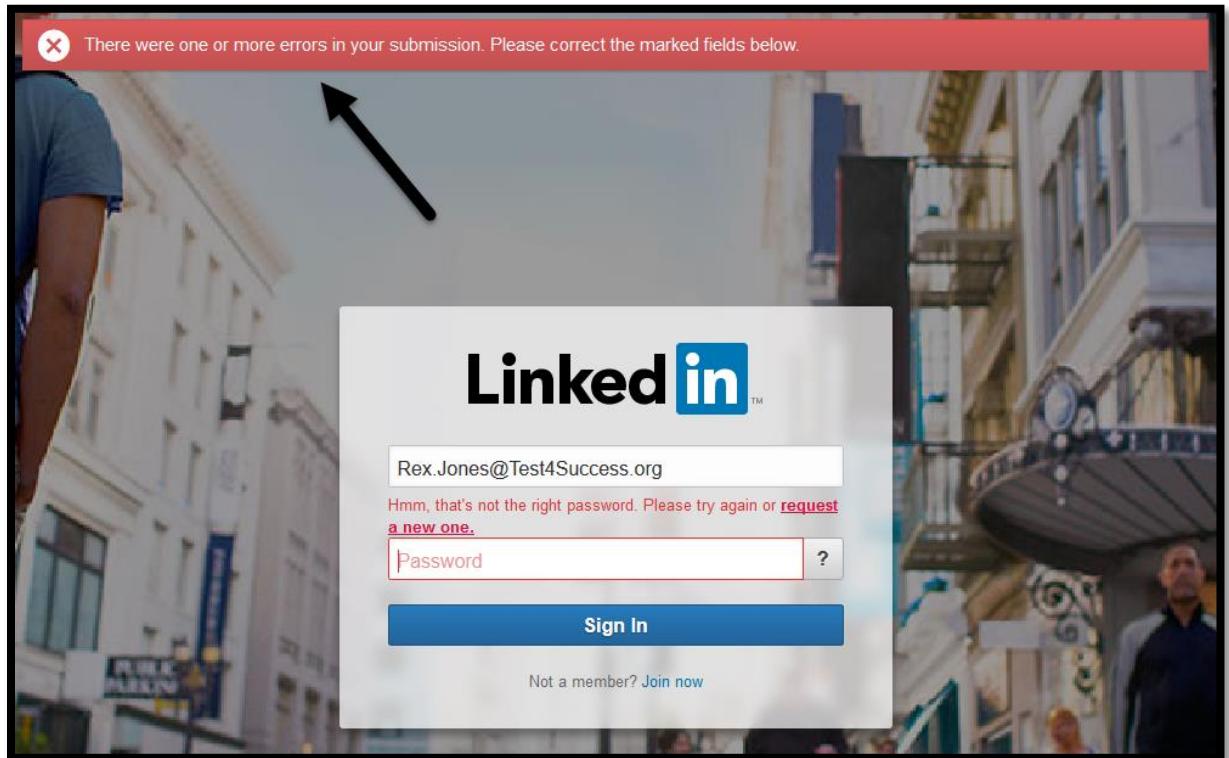


Figure 13.4 – Sign In Page After Entering Invalid Credentials

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Conclusion

Selenium WebDriver is a very popular open source automation tool in the Software QA/Testing industry. The goal of “*(Part 1) Selenium WebDriver for Functional Automation Testing*” was to provide a foundation for beginners. An automation engineer is effective when learning how to program, how to apply a unit test framework, and understand two automation principles. The two automation principles encompass finding a WebElement then deciding what action to perform on the WebElement. Automation engineers build on top of that foundation with knowledge of classes, methods, and annotations within Java, TestNG, and Selenium WebDriver. The following are take-away topics from this book:

Java

Java is one of the programming languages supported by Selenium WebDriver. An automation engineer can transition to a different language (C#, Python, Ruby) after understanding the fundamentals of programming. One of the fundamentals is to learn the syntax for that desired programming language. The following are several concepts regarding Core Java:

- Variables: A location that holds data
- Data Types: Refers to the type of variable
- Operators: A symbol that performs mathematical or logical operations
- Control Structures: Refers to the process of using logic to force the program to skip or loop statements
- Class: A template for objects which contains data and code that operates on the data

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

- **Objects:** Objects are the foundation to object-oriented programming (OOP). It consists of two characteristics: state and behavior. State identifies the object and behavior represent actions of the object
- **Methods:** A method is a block of code that perform a specific task / action
- **Access Modifiers:** Determines access to a class and all of the class members (variables and methods)
- **Inheritance:** Inheritance is a hierarchical concept which allows reusable code and objects to be extended
- **Encapsulation:** Encapsulation is a concept where a class encapsulates code and data for protection
- **Polymorphism:** Polymorphism is concept that allows multiple methods to utilize one interface
- **Packages:** A package is a collection of related classes
- **Interfaces:** An interface is a collection of related methods

TestNG

TestNG is a unit test framework influenced by JUnit. JUnit comes equipped with Eclipse but TestNG must be installed as a plugin. Both frameworks contain annotations that are required for testing an application. However, TestNG added more features that facilitate unit, integration, end-to-end (E2E), acceptance, and functional testing. The following are some of the annotations within TestNG:

- **@BeforeSuite / @AfterSuite** – before a suite starts / after all the test methods in a certain suite have run

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

- @BeforeTest / @AfterTest – before a test starts / after all the test methods in a certain test have run
- @BeforeClass / @AfterClass – before a test class starts / after all the test methods in a certain class have run
- @BeforeMethod / @AfterMethod – before a test method runs / after a test method has run
- @Test - Marks a class or a method as part of the test

Selenium WebDriver

Selenium WebDriver is an automation tool used for testing web applications. The tool generates a call to a browser utilizing each browser's native support for automation. It is important to know that all web applications and browsers contain WebElements “i.e., buttons, links, etc.” The WebElements are derived from HTML, which is an acronym for Hyper Text Markup Language.

HTML is the standard language for creating web applications. As a result, the key to automating an application is to first find the WebElement via HTML. After locating the WebElement, an automation engineer decides the appropriate action to perform on the WebElement. Finding the WebElement and performing an action on the WebElement are building blocks to test automation. A WebElement is found using the following eight locator types:

1. ID - Find WebElements by the value of its ID attribute
2. Name - Find WebElements by the value of its Name attribute

3 Tips To Master Selenium Within 30 Days
<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings
<http://tinyurl.com/Free-QTP-UFT-Selenium>

3. [XPath](#) - Find WebElements by its XPath
4. [CSS Selector](#) - Find WebElements by the CSS Selector's engine
5. [Link Text](#) - Find hyperlink WebElements by its complete text
6. [Partial Link Text](#) - Find hyperlink WebElements by partial text contained within the complete text
7. [Tag Name](#) - Find WebElements by its tag name
8. [Class Name](#) - Find WebElements by the value of its Class attribute

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Resources

1. Selenium HQ

<http://www.seleniumhq.org/>

<http://www.seleniumhq.org/about/history.jsp>

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

<https://seleniumhq.github.io/selenium/docs/api/java/index.html?org/openqa/selenium/WebDriver.html>

2. TIOBE

http://www.tiobe.com/tiobe_index

3. GitHub

<https://github.com/mozilla/geckodriver/releases>

4. ORACLE Java Documentation

The Java™ Tutorials

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html

5. Java A Beginner's Guide Sixth Edition

Create, Compile, and Run Java Programs Today

Herbert Schildt

6. TestNG

<http://testng.org/doc/documentation-main.html#annotations>

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

Resources

7. Next Generation Java™ Testing

TestNG and Advanced Concepts

Cédric Beust | Hani Suleiman

8. W3C

<http://www.w3.org/TR/WCAG20-TECHS/H93.html>

9. LinkedIn

<https://www.linkedin.com/>

10. Facebook

<https://www.facebook.com/>

11. Yahoo

<https://www.yahoo.com/>

12. WordPress

<https://login.wordpress.org/>

13. Welcome to the Internet

<http://the-internet.herokuapp.com/>

The following resources are links to the installations in Chapter 1. Some links duplicate the previous 12 resources:

14. Install Selenium IDE via Selenium HQ

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

15. Install Java Development Kit (JDK) via Oracle

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads->

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

[2133151.html](#)

16. Install Eclipse IDE via Eclipse

<https://eclipse.org/downloads/>

17. Install Java Client Driver via Selenium HQ

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

18. Install ChromeDriver via Selenium HQ or Google Sites

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

<https://sites.google.com/a/chromium.org/chromedriver/downloads>

19. Install IEDriverServer via Selenium HQ or Selenium Server

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

<http://selenium-release.storage.googleapis.com/index.html?path=2.53/>

20. Install Notepad ++

<https://notepad-plus-plus.org/download/v6.9.2.html>

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

Download PDF Version

The PDF Version of this book is available to you at the following link:

<http://tinyurl.com/Part-1-Selenium-WebDriver>

If the book was helpful, can you leave a favorable review?

<http://tinyurl.com/Review-Pt-1-Selenium-WebDriver>

Thanks in advance,



Rex Allen Jones II

Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Books by Rex Jones II

www.tinyurl.com/Rex-Allen-Jones-books

1. **Free Book** Absolute Beginner

(Part 1) You Must Learn VBScript for QTP/UFT

Don't Ignore The Language For Functional Automation Testing

2. (Part 2) You Must Learn VBScript for QTP/UFT

Don't Ignore The Language For Functional Automation Testing

3. **Free Book** Absolute Beginner

(Part 1) Java 4 Selenium WebDriver

Come Learn How To Program For Automation Testing

4. (Part 2) Java 4 Selenium WebDriver

Come Learn How To Program For Automation Testing

5. **Free Book** Absolute Beginner

(Part 1) Selenium WebDriver for Functional Automation Testing

Your Beginners Guide

Coming Soon

6. (Part 2) Selenium WebDriver for Functional Automation Testing

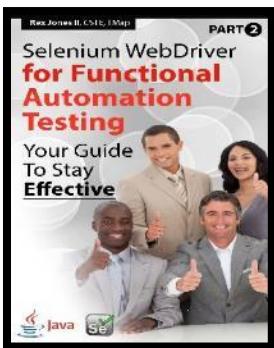
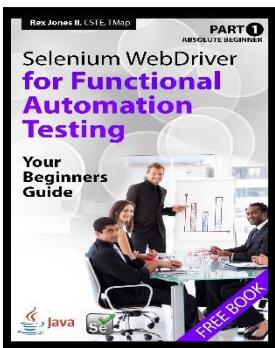
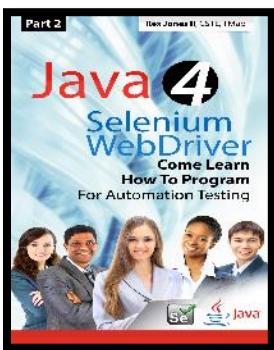
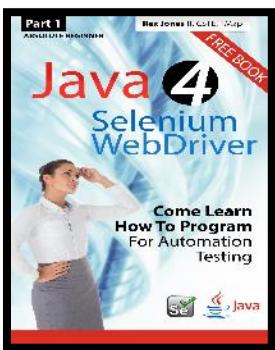
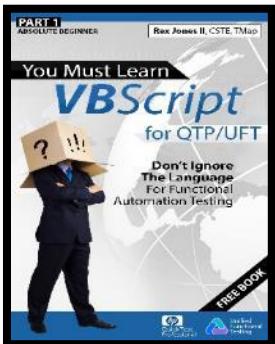
Your Guide To Stay Effective

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>



Skype: rex.jones34

Twitter: @RexJonesII

Email: Rex.Jones@Test4Success.org

LinkedIn: <https://www.linkedin.com/in/rexjones34>

Sign Up To Receive

1. 3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

2. 3 Tips To Master QTP/UFT Within 30 Days

<http://tinyurl.com/3-Tips-For-QTP-UFT>

3. Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>

3 Tips To Master Selenium Within 30 Days

<http://tinyurl.com/3-Tips-For-Selenium>

Free Webinars, Videos, and Live Trainings

<http://tinyurl.com/Free-QTP-UFT-Selenium>