

# Selenium WebDriver

- QtpSudhakar

# Introduction

---

# About Trainer

- ❑ Sudhakar Kakunuri known as Qtp Sudhakar
- ❑ Worked as a Tester, Senior, Lead, Manager, Architect and a Trainer for Test Automation
- ❑ Blogs at QtpSudhakar.com on various Test Automation concepts
- ❑ Started career on WinRunner -> QTP/UFT -> Selenium
- ❑ Authored “Cracking the QTP Interview” book published by TATAMcGraw Hill
- ❑ Gives training on Selenium/WebDriver, QTP/UFT, LeanFT
- ❑ Agreed to write a book for UFTPro and Selenium for Shroff Publishers

## What will be covered in this Course....

- ❑ Java – To Write Tests Using WebDriver API
- ❑ Selenium WebDriver – To Automate Web Applications
- ❑ SIKULI – An Automation API useful in failure of WebDriver Locators
- ❑ Grid – For Remote Execution of Tests
- ❑ POM/POF Framework – Object Repository Based Framework
- ❑ TestNG – A Testing Framework to write Tests
- ❑ ExtentReports – To Generate Reports for Tests
- ❑ Maven – A Build Automation Tool
- ❑ GIT – Source Code Management Tool
- ❑ Cucumber – A tool based on BDD Framework

# Selenium Introduction

---

# Test Automation Life Cycle

## ↓ Decision to Automate

- Meetings and Discussions
- Tool Selection

## ↓ Planning

- How to Automate
- What to Automate
- When to Automate
- In Scope & Out Scope

## ↓ Design

- Test Automation Framework
- Environment Setup and Trainings

## ↓ Develop

- Develop Scripts for Automation
- Review Scripts

## ↓ Execute

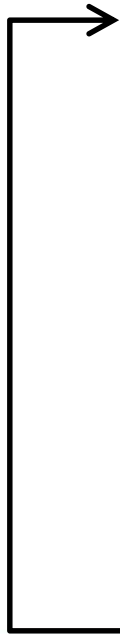
- Setup Test Execution Environment
- Execution of Scripts

## ↓ Report

- Analyse Results
- Report Defects

## ↓ Maintenance

- Enhance Scripts as per new CR's



# Different Test Automation Tools

Tool Name	Company
Unified Function Testing UFT Pro (LeanFT)	MicroFocus
Test Complete	SmartBear
Ranorex	Ranorex GmbH
Tosca Suite	Tricentis
SilkTest	MicroFocus
Rational Functional Tester	IBM
OATS	Oracle
CoadedUI	Microsoft
Selenium	Open Source
WatiR, WatiJ, WatiN	Open Source
Sahi	Open Source , Commercial

# What makes Selenium different from Other Tools?

- ☐ Selenium (WebDriver) is an open source Automation Solution
- ☐ Developed to Automate Web Applications Functional Testing
- ☐ You can test Web Applications on Multiple OS
  - ☐ Windows
  - ☐ Linux
  - ☐ OSX
- ☐ You can write programs in Multiple Languages
  - ☐ Java
  - ☐ Javascript
  - ☐ C#
  - ☐ Ruby
  - ☐ Python
- ☐ You can test Web Applications on Multiple Browsers
  - ☐ Firefox
  - ☐ IE
  - ☐ Chrome
  - ☐ Safari
  - ☐ Opera
  - ☐ ....



# Tools and Languages

Tool Name	Language
Unified Function Testing	VBScript
UFTPro (LeanFT)	Java, C#, JavaScript
Test Complete	JavaScript, Python, VBScript, C#
Ranorex	C#, VB.Net
Tosca Suite	Java, Delphi, VB
SilkTest	Java, C#
Rational Functional Tester	Java, VB.Net
OATS	Java, OpenScript
CoadedUI	C#, VB.Net
Selenium	Java, C#, Python, Perl, JavaScript
WatiR, WatiJ, WatiN	Ruby, Java, C#
Sahi	Java, SahiScript

## Selenium Version History

---

# Selenium Versions

- ❑ Selenium was originally developed by Jason Huggins in 2004 as an internal tool at ThoughtWorks.
- ❑ Huggins and Paul Hammant joined the team with other programmers and open sourced "Selenium Remote Control" (RC)
- ❑ In 2006, Shinya Kasatani donated Selenium-IDE a Firefox Addon to Selenium project.
- ❑ In 2007 Simon Stewart at ThoughtWorks developed a superior browser automation tool called WebDriver
- ❑ In 2008, Philippe Hanrigou made "Selenium Grid", which provides a hub allowing the running of multiple Selenium tests concurrently on any number of local or remote systems, thus minimizing test execution time.
- ❑ In 2009 at the Google Test Automation Conference, it was decided to merge the two projects, and call the new project Selenium WebDriver, or Selenium 2.0.
- ❑ It was released in 2011. The last version is 2.53.1
- ❑ In 2016 Selenium 3.0 released and unveiled a roadmap to make it a W3C specification.
- ❑ The latest version is 3.14 (Dec 2018)

## Selenium Versions - Continued

- ❑ Selenium RC has been officially deprecated in favor of Selenium WebDriver.
- ❑ Selenium 1.0 + WebDriver = Selenium 2.0
- ❑ WebDriver is designed in a simpler and more concise programming interface along with addressing some limitations in the Selenium-RC API.
- ❑ Selenium 2.0 supported Selenium core (Used for RC API Execution) deleted from Selenium 3.0
- ❑ They states that *“The needs of modern web testing have grown ever more complicated and Selenium Core is now less capable of meeting these needs than it was before.”*
- ❑ Selenium IDE will no longer work from Firefox 55 onwards as there are no people to maintain

# The First Program

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class FirstTest{

    public static void main(String[] args) {

        WebDriver driver = new FirefoxDriver();
        driver.get("http://facebook.com");

        System.out.println(driver.getTitle());

        driver.quit();
    }
}
```

## Program Explanation

- ❑ A class is an empty program by default
- ❑ You can add a main method while or after creating class
- ❑ A Method is a set of statements
- ❑ Main method is a driver for the class. When you click on “Run” whatever is there in main method will be executed.
- ❑ To automate an Application we need to import packages from selenium libraries. i.e. jars. You can import them out side of class.
- ❑ We can use classes, methods...etc. of selenium only by importing required packages from its library.
- ❑ We must use **import org.openqa.selenium.WebDriver;** to write any program of selenium
- ❑ **import org.openqa.selenium.firefox.FirefoxDriver;** will be used to work with firefox

## Program Explanation

- ❑ `public class FirstTest{ }` is defining a class
- ❑ `public static void main(String[] args) { }` is called method signature
- ❑ The statements between `{ }` are Method body.
- ❑ `WebDriver driver = new FirefoxDriver();` will open a new firefox window
- ❑ `FirefoxDriver` is a class name. “`new FirefoxDriver()`” will create an instance of firefox driver and returns to “`driver`” variable.
- ❑ We can use “`driver`” variable to perform operations in other statements.
- ❑ `driver.get("http://facebook.com");` will open facebook
- ❑ `driver.getTitle()` will get the title value of page
- ❑ `System.out.println("value");` will print some value in log
- ❑ `System.out.println(driver.getTitle());` will print the title in log

# Java Basics

---



# What level of java required to work with selenium?

- ❑ This is like “what level of knowledge required to do Manual Testing?”
- ❑ The answer is simple “The more knowledge you get The better results you see”
- ❑ A basic java knowledge is enough to learn and write automation programs
  - ❑ Variables, Datatypes, conditions, loops, Arrays, lists, packages...etc.
- ❑ Some more knowledge is required to write reusable automation programs
  - ❑ Classes, Objects, Methods...etc.
- ❑ Some more knowledge is required to develop Automation Framework
  - ❑ OOPS concepts like Inheritance, encapsulation, polymorphism...etc.
- ❑ Learning is a never end process.
- ❑ If you completed the first step successfully... you can easily complete all other steps

# Introduction







- ❑ Java is a high level language which contains English words which will be later translated to machine level language
- ❑ Programmers use any high level language to write programs
- ❑ A machine level language is what computer own language that is binary numbers all 1s and 0s.
- ❑ A compiler translates high level language in to machine level language
- ❑ Java has three part Java Development Kit, Java Run time Environment, Java Virtual Machine
- ❑ JDK provides all components that are required to develop, compile, debug and run a java program.
- ❑ JRE contains JVM and java libraries which will be used to run a java program. JVM creates machine specific instructions based on OS.
- ❑ If you install JDK all other components JRE, JVM will be installed

- ❑ What is a Program?
  - ❑ A program is a sequence of instructions and decisions that a computer perform to achieve a task.
- ❑ Any Program in Eclipse should be created in side of a project
- ❑ A Project contains Packages, Classes...etc.
- ❑ A Package is used to group the classes
- ❑ A Class is a program which contain methods, variables...etc.
- ❑ A Method is a reusable code block with a name assigned to it.
- ❑ A Variable is used to store value
- ❑ A Value is some data with type of integer, character, object, string and Boolean...etc.

# Variables and Datatypes

- ❑ Variables are used to store the value that can be used at later time
- ❑ Datatypes are two types
  - ❑ Primitive
  - ❑ Object Reference
- ❑ Primitive hold fundamental values like integer, float, Boolean, character that are predefined by the language
- ❑ Reference variable stores a reference to objects
- ❑ Any variable have type, name and value
  - ❑ Type is the type of data to be stored in that variable
  - ❑ Name is the name of variable
  - ❑ Value is what you have assigned to the variable

# Primitive Datatypes

						
<b>TYPE</b> >	<b>byte</b>	<b>short</b>	<b>int</b>	<b>long</b>	<b>float</b>	<b>double</b>
<b>BITS</b> >	<b>8 bits</b>	<b>16 bits</b>	<b>32 bits</b>	<b>64 bits</b>	<b>32 bits</b>	<b>64 bits</b>
<b>RANGE</b> >	<b>-128 to 127</b>	<b><math>-2^{15}</math> to <math>2^{15}-1</math></b>	<b><math>-2^{31}</math> to <math>2^{31}-1</math></b>	<b><math>-2^{63}</math> to <math>2^{63}-1</math></b>	<b><math>\pm 10^{38}</math></b>	<b><math>\pm 10^{308}</math></b>

	
<b>TYPE</b> >	<b>char</b>
<b>BITS</b> >	<b>16 bit Unicode Character</b>
<b>RANGE</b> >	<b>0 to 65535</b>



<b>boolean</b>
<b>1 bit of information</b>
<b>true / false</b>

# Variables and Datatypes

```
byte b = 100;
System.out.println("byte value is "+b);

short s = 10000;
System.out.println("short value is "+s);

int i = 100000;
System.out.println("int value is "+i);

long l = 1000000L;
System.out.println("long value is "+l);

float f = 2.1f;
System.out.println("float value is "+f);

double d = 2.1;
System.out.println("double value is "+d);

boolean isthere= true;
System.out.println("boolean value is "+ isthere);

char c = 'c';
System.out.println("char value is "+c);
```

# Type Casting

- ❑ We must specify the type of data when assigning a value to variable
- ❑ We can also assign data of one type to a variable of another type using casting
- ❑ `int x = 100;` | `short y=x;` will give an error. But you can cast x to short to assign value
- ❑ You can write `short y=(short) x;`
- ❑ This is called explicit type cast. You have to use it when casting bigger to small datatype
- ❑ `short x = 100;` | `int y=x;` will not give an error. The type cast will be done automatically because its small to bigger
- ❑ Byte -> short -> int -> long -> float -> double ... casting can be done automatically
- ❑ Double -> float -> long -> int -> short -> byte ... casting requires explicit mention of datatype.

## ❑ Wrapper Types

- ❑ For every primitive datatype there is a class created as a wrapper
- ❑ These classes are called as wrapper types
- ❑ These are mainly used for converting data and also in collection framework
- ❑ Names of these classes are almost equal to primitive type names with first letter as uppercase
- ❑ byte: Byte, short: Short, int: Integer, long: Long, float: Float, double: Double, char: Character, boolean: Boolean

## ❑ Reference Types

- ❑ A datatype which can be created using class reference.
- ❑ There are two main data types for storing values
  - String : For Storing group of characters
  - Object : For storing any type f data

❑ Object type the super type in Java and it can accept any type of data or reference



# Identifiers

- ❑ Identifier is a name and It can be a class name, variable name, method name...etc.
- ❑ A name can contain Alpha numeric or \$ or an underscore
- ❑ A name must start with Alpha character
- ❑ A java keyword cannot be used as identifier name
- ❑ All identifiers are case sensitive.
- ❑ Class/Interface Name should start with upper case letter
- ❑ Method/Variable/package Name should start with lower case letter
- ❑ Constant names should be in upper case letters
- ❑ If a name combined with two words, the second word always start with upper case letter

# Arrays

- ❑ Used to store multiple values in a single variable using an index number
- ❑ This is useful when storing data of similar type
- ❑ To store some user information we don't need to create more variables
- ❑ We can create a variable with [ ] and assign type of data by specify number of values to be stored
  - ❑ `String userDetails[] = new String[5];`
  - ❑ The above statement will accept 5 details of an user.
  - ❑ `userDetails[0] = "Selenium";`
  - ❑ `userDetails[1] = "hq";`
  - ❑ Like this we can store 5 values
  - ❑ Use `syso(userDetails[1]);` to print data
  - ❑ Array can be created in a single line like
    - `String userDetails[] = {"selenium","hq"};`

## Working with Elements

- ❑ Everything in a page is an Element.
- ❑ To work with any element in page we must first create the element reference in program.
- ❑ To click on a button we should create button reference in program and apply click operation on it.
- ❑ To create the reference of any object we have to use
  - ❑ `Driver.FindElement();`
- ❑ To Find an Element we can use some attributes that are given by developers while developing application
- ❑ An attribute is used to define the characteristics of an HTML element and is placed inside the element's opening tag.
- ❑ `<Input type="text" id="email" name="email"></input>`
- ❑ In above html code type, id, name are the attributes of input tag element.

## Example – FindElement(By)

```
WebDriver driver = new FirefoxDriver();  
driver.get("http://facebook.com");  
  
System.out.println(driver.getTitle());  
  
driver.findElement(By.id("email")).sendKeys("sudhakar");  
driver.findElement(By.id("pass")).sendKeys("password");  
driver.findElement(By.xpath("//Input[@value='Log In']")).click();  
driver.close();
```

- ❑ In selenium we can write programs to handle the elements using locators
- ❑ FindElement(By.LocatorName("LocatorValue")) will be used to get an object reference.
- ❑ SendKeys and Click are the methods to perform operations on Element

# Locators

- ❑ The attributes are given by the developer and selenium uses them to find elements. In selenium we call them as locators.
- ❑ There are 8 types of locators
  - ❑ By.Id – Get element with the specified @id attribute.
  - ❑ By.Name – Get first element with the specified @name attribute.
  - ❑ By.Linktext – Get link element which has same text
  - ❑ By.PartialLinktext – Get link element which contains partial text
  - ❑ By.TagName – Get Element using a Tag Name
  - ❑ By.Classname – Get Element using a class Name
  - ❑ By.CssSelector – Get the element using css selectors.
  - ❑ By.Xpath - Locate an element using an XPath expression.

# Configure Firefox

- ❑ We have to configure FireFox with Firebug and Firepath which will create easiness while writing selenium programs.
- ❑ In FindElement we have to give the locator values
- ❑ We can copy the locator values from Firefox -> Inspect Element
- ❑ Firebug will provide even more easy way to get locator values
- ❑ FirePath is useful to get/create/test X-Path of an element
- ❑ XPath is a path of html element that describes a way to locate elements.
- ❑ To add Firebug Open Firefox -> navigate to <https://addons.mozilla.org/en-US/firefox/addon/firebug/>
- ❑ Click on Add To Firefox -> Restart Firefox
- ❑ To Add Firepath navigate to -> <https://addons.mozilla.org/en-US/firefox/addon/firepath/>
- ❑ Click on Add To Firefox -> Restart Firefox

# HTML Basics

- ❑ To understand locating techniques you must have basic knowledge of html
- ❑ HTML is a markup language with set of html tags for describing web documents
- ❑ Each html tag describes different document content
- ❑ Html tags are normally in pairs (start tag, end tag) surrounded by angle brackets
- ❑ The end tag is written like the start tag, but with a slash before the tag name
- ❑ Web browsers will read the tags, understand and display them as content
- ❑ Every element in web page is a html tag
- ❑ `<html><head> <title> page title </title> </head> <body><h1>This is a heading</h1> </body></html>` will display “This is a heading” with big size letters
- ❑ Whatever is given in the body will be displayed in web page
- ❑ `<input>` tag will create user input elements like text, check, button, radio, file...etc.
- ❑ `<input type=“button”> </input>` will create a button
- ❑ `<input type=“button” value=“webdriver”> </input>` will create button with value webdriver
- ❑ Type, value are the attribute names and button, webdriver are the attribute values
- ❑ We can use these attribute names and values as locators to find element

# Sample HTML Document for Login

```
<table id="logindetailstable">
<tr><td> Username </td>
<td><input type="text" id="uname" class="logintbox" name="j_username"></td></tr>
<tr><td> Password </td>
<td><input type="password" id="pwd" class="logintbox" name="j_password"></td></tr>
<tr><td colspan=2 align="center"><input type="button" value="Login"></td></tr>
</table>
```

Username

Password

Login

```
<style>
.logintbox{
    border: 2px;
    border-style: dotted;
    background-color: AliceBlue;
}
</style>
```



## Locating Elements using X-Path

- ❑ Xpath is the path of element in page html source code
- ❑ In previous example the html path of the username text box is `html/body/table/tbody/tr[1]/td[2]/input`
- ❑ This is a absolute x path and it takes complete path of element and you can get this by right clicking on element -> Inspect in Firepath
- ❑ In real time we must always use relative xpath which doesn't depends on complete html path but based on its attributes
- ❑ The relative xpath for username text box is `//input[@id="uname"]` or `//*[@id='uname']`
- ❑ We can see absolute and relative xpaths in firepath
- ❑ To switch from absolute to relative xpath in firepath use below navigation
  - ❑ Right click on FirePath Tab in Firebug -> deselect Generate absolute xpath
  - ❑ Try to inspect element again using firepath

# X-Path Syntax

Expression	Description
/	Selects from the root element
//	Selects all matched elements
*	Matches any element
@	Selects attributes
//Tag[index]	Select first element based on index in parent
//Tag[last()]	Select last element in parent
//Tag[contains(@attribute, "value")]	Select the elements which contains the value
/html	Selects the element html Note: If the path starts with a slash ( / ) it always represents an absolute path to an element!
//Input	selects all input elements in html document
//*tr[1]	Selects first row
//Input[@id="email"]	Select username text box
//tr[1]	select first table row
//tr[last()]	select last table row
//input[contains(@id, "una")]	Select the element which has una in id attribute

# Locating Elements using CSS Selector - 1

Selector	Example	Example description
.class	.loginbox	Selects all elements with class="loginbox"
#id	#uname	Selects the element with id="uname"
*	*	Selects all elements
element	input	Selects all <input> elements
element,element	td,input	Selects all <td> elements and all <input> elements
element element	td input	Selects all <input> elements inside <td> elements
element>element	td > input	Selects all <input> elements where the parent is a <td> element
element+element	td + td	Selects all <td> elements that are placed immediately after <td> elements
element1~element 2	tr + tr	Selects every <tr> element that are preceded by a <tr> element

## Locating Elements using CSS Selector - 2

[attribute]	[id]	Selects all elements with a id attribute
[attribute=value]	[id=uname]	Selects all elements with id=uname
[attribute~=value]	[id~=uname]	Selects all elements with a id attribute containing the word "uname"
[attribute =value]	[id =pwd]	Selects all elements with a id attribute value starting with word "pwd"
[attribute^=value]	[id^=pw]	Selects all elements whose id attribute value begins with "pw"
[attribute\$=value]	[name\$=word]	Selects all elements whose id attribute value ends with "word"
[attribute*=value]	[name*=word]	Selects all elements whose id attribute value contains the substring "word"
:checked	input:checked	Selects every checked <input> element
:disabled	input:disabled	Selects every disabled <input> element
:enabled	input:enabled	Selects every enabled <input> element

## Locating Elements using CSS Selector - 3

:first-child	tr:first-child	Selects every <tr> element that is the first child of its parent
:last-child	input:last-child	Selects every <input> element that is the last child of its parent
:not(selector)	:not(input)	Selects every element that is not a <input> element
:nth-child(n)	tr:nth-child(1)	Selects every <tr> element that is the first child of its parent
:nth-last-child(n)	tr:nth-last-child(2)	Selects every <tr> element that is the second child of its parent, counting from the last child
:only-of-type	p:only-of-type	Selects every <p> element that is the only <p> element of its parent
:only-child	p:only-child	Selects every <p> element that is the only child of its parent

