**Fall 2020**

**INFO 6255**

**Introduction to Selenium**

**Selenium** is a free (open source) automated testing suite for web applications across different browsers and platforms.

It is quite similar to HP UFT.

It focuses on automating web-based applications.

Testing done using Selenium tool is usually referred as Selenium Testing. Selenium is not just a single tool but a suite of software's, each catering to different testing needs of an organization.

**It has four components:**

* Selenium Integrated Development Environment (IDE)
* Selenium Remote Control (RC)
* Selenium Grid
* WebDriver

**Selenium IDE** is a complete [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) for Selenium tests. It is implemented as a [Firefox Add-On](https://en.wikipedia.org/wiki/Add-on_(Mozilla)) and as a [Chrome Extension](https://en.wikipedia.org/wiki/Chrome_Web_Store). It allows for recording, editing, and debugging of functional tests. It was previously known as Selenium Recorder. Scripts may be automatically recorded and edited manually providing [autocompletion](https://en.wikipedia.org/wiki/Autocomplete) support and the ability to move commands around quickly. Scripts are recorded in **Selenese**, a special test scripting language for Selenium. Selenese provides commands for performing actions in a browser (click a link, select an option), and for retrieving data from the resulting pages.

**Selenium RC** is the first automated web testing tool that allowed users to use a programming language they prefer. RC makes it possible to write automated tests for a web application in any programming language, which allows for better integration of Selenium in existing unit test frameworks. As of version 2.25.0, RC can support the following programming languages: Java, C#, PHP Python, Perl, Ruby.

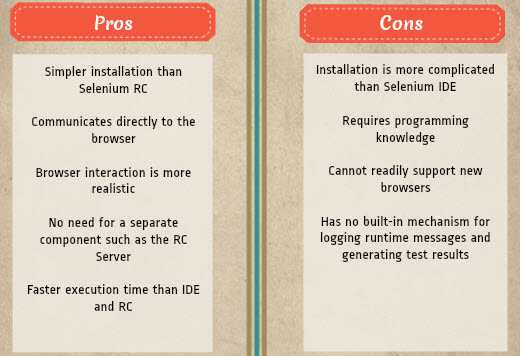
**Selenium Grid** is a tool used together with Selenium RC to run parallel tests across different machines and different browsers all at the same time. Parallel execution means running multiple tests at once. It is a server that allows tests to use web browser instances running on remote machines. With Selenium Grid, one server acts as the hub. It utilizes the hub-and-nodes concept. The hub acts as a central source of Selenium commands to each node connected to it.

**WebDriver** is a web automation framework that allows you to execute your tests against different browsers, not just Firefox, Chrome. WebDriver also enables you to use a programming language in creating your test scripts. WebDriver’s goal is to supply a well-designed object-oriented API that provides improved support for modern advanced web-app testing problems. Selenium-WebDriver makes direct calls to the browser using each browser’s native support for automation**.** How these direct calls are made, and the features they support depends on the browser you are using. If your browser and tests will all run on the same machine, and your tests only use the WebDriver API, then you do not need to run the Selenium-Server; WebDriver will run the browser directly. You can now use conditionaloperations like if-then-else or switch-case. You can also perform looping like do-while. Programming languages supported by WebDriver are: Java, .Net, PHP, Python, Perl and Ruby.

Selenium uses what is called **locators** to find and match the elements of your page that it needs to interact with. There are 8 locators strategies included in Selenium:

* Identifier
* Id
* Name
* Link
* DOM
* XPath
* CSS
* UI-element

**Pros and cons of selenium Webdriver:**



**Advantages of Selenium over QTP**

|  |  |
| --- | --- |
| **Selenium** | **QTP** |
| Open source, free to use, and free of charge. | Commercial. |
| Highly extensible | Limited add-ons |
| Can run tests across different browsers | Can only run tests in Firefox, Internet Explorer and Chrome |
| Supports various operating systems | Can only be used in Windows |
| Supports mobile devices | QTP Supports Mobile app test automation (iOS & Android) using HP solution called - HP Mobile Center |
| Can execute tests while the browser is minimized | Needs to have the application under test to be visible on the desktop |
| Can execute tests in parallel. | Can only execute in parallel but using Quality Center which is again a paid product. |

**CONCEPTS**

**What is Maven and why is it required?**

Maven is a project management tool which encompasses a project object model, a set of standards, a project life cycle, a dependency management system, and logic for executing plugin goals at defined phases in a life cycle. When you use Maven, you describe your project using a well-defined project object model, Maven can then apply cross-cutting logic from a set of shared (or custom) plugins. It mainly helps in below ways:

1. Managing and enforcing a complete project life-cycle i.e. in-short Code compilation>Unit testing>Build generation (jar and war files). So, here you can see a complete project management life cycle has been followed by Maven and if any of the one phase fails, Maven will not generate a build.

2. Easily managing external dependencies for the Java based project and keep the Java project light-weight (pom.xml). Easily managing external dependencies for the Java based project: Maven team has placed all the Java based libraries on cloud, which you can easily download and integrate with your project by adding dependencies under <dependencies> tag in pom.xml file. You can find all the dependencies at following website: [https://mvnrepository.com](https://mvnrepository.com/)

3) In maven, you don't have to manage jar files manually. Instead of jar files, we use jar's dependencies in one configuration file called pom.xml.

**Pom.xml**

Link: https://maven.apache.org/guides/introduction/introduction-to-the-pom.html

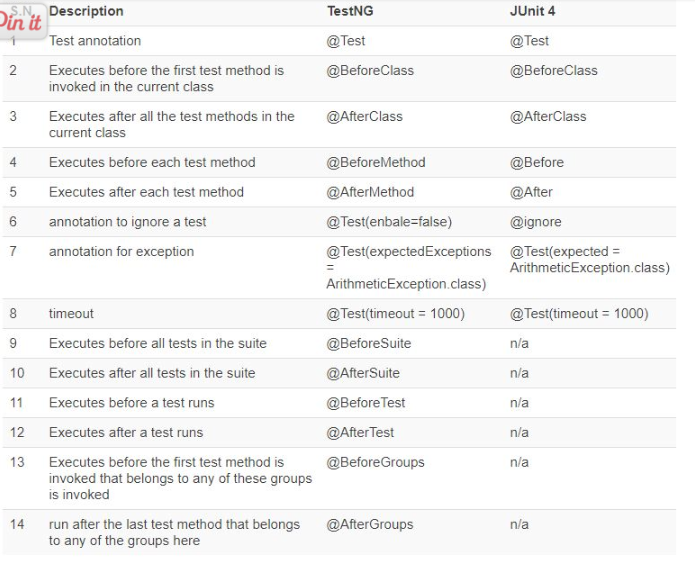
A Project Object Model or POM is the fundamental unit of work in Maven. It is an XML file that contains information about the project and configuration details used by Maven to build the project. It contains default values for most projects. Examples for this is the build directory, which is target; the source directory, which is src/main/java; the test source directory, which is src/test/java; and so on. When executing a task or goal, Maven looks for the POM in the current directory. It reads the POM, gets the needed configuration information, then executes the goal.

**Unit testing**

Unit Testing is a level of software testing where individual units of software are tested. A unit is the smallest testable part of software where usually few inputs are provided to test or verify a single functionality of a software system. Unit testing confirms that whether the project is developing in right direction or not.

**TestNG** is an automation testing framework in which NG stands for "Next Generation". TestNG is inspired from [JUnit](https://www.guru99.com/junit-tutorial.html)which uses the annotations (@). Using TestNG you can generate a proper report, and you can easily come to know how many test cases are passed, failed and skipped.

**Difference between JUnit and TestNG**



## **Sample Java Code**

import org.junit.Test;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class Sample {

@Test

public void hello() throws InterruptedException {

System.setProperty("webdriver.chrome.driver","C:\\Users\\Welcome\\Desktop\\SEM 2\\QA\\chromedriver.exe");

WebDriver driver = new ChromeDriver();

driver.get("http://mail.google.com/mail/");

driver.findElement(By.xpath("//input[@id='identifierId']")).sendKeys("username");

driver.findElements(By.className("RveJvd")).get(0).click();

Thread.sleep(2000);

driver.findElement(By.name("password")).sendKeys("password");

driver.findElements(By.className("RveJvd")).get(1).click();

//driver.quit();

}

}

## **POM.xml**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.mycompany</groupId>

<artifactId>Java\_Selenium</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.source>1.7</maven.compiler.source>

<maven.compiler.target>1.7</maven.compiler.target>

</properties>

<dependencies>

**<dependency>**

**<groupId>org.seleniumhq.selenium</groupId>**

**<artifactId>selenium-java</artifactId>**

**<version>2.45.0</version>**

**</dependency>**

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.10</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**How to download and install Selenium Webdriver:**

<https://www.guru99.com/installing-selenium-webdriver.html>

**selenium with Jenkins, maven and testng:**

<https://www.guru99.com/maven-jenkins-with-selenium-complete-tutorial.html>

**testng tutorial:** <https://www.guru99.com/all-about-testng-and-selenium.html>

**selenium with junit:** <https://www.softwaretestinghelp.com/selenium-junit-framework-selenium-tutorial-11/>