Selenium Overview, Introduction and setting up project

QAUBER

1159 Sonora Court Suite 205, Sunnyvale CA, 94086 USA
Tel. +1 (408) 220 9912 Ext 1 www.qauber.com

What is Selenium WebDriver?

WebDriver is a tool for automating web application testing, and in particular to verify that they work as expected. It aims to provide a friendly API that's easy to explore and understand, easier to use than the Selenium-RC (1.0) API, which will help to make your tests easier to read and maintain.

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.

Introduction as tool for web application test automation

WebDriver is a web automation framework that allows you to execute your tests against different browsers, not just Firefox (unlike Selenium IDE).

WebDriver also enables you to use a programming language in creating your test scripts (not possible in Selenium IDE).

You can now use conditional operations like if-then-else or switch-case
You can also perform looping like do-while.
Following programming languages are supported by WebDriver

Java, .Net, PHP, Python, Perl, Ruby

Need for test automation

- Frequent regression testing
- Rapid feedback to developers during the development process
- Virtually unlimited iterations of test case execution
- Customized reporting of application defects
- Support for Agile and eXtreme development methodologies
- Disciplined documentation of test cases
- Finding defects missed by manual testing

How Selenium WebDriver fulfills them?

- Support for iPhone and Android testing
- Implementation of listeners
- Better features for Ajax testing.
- You can easily simulate clicking on front and back button of browser.
- Unlike RC you don’t have to start a server in webdriver.
- You can simulate movement of a mouse using selenium.
- Tabs and pops are more or less the same.
- You can find coordinates of any object using Webdriver.

Cross browser testing

Cross browser testing is to test your website or application in multiple browsers- and making sure that it works consistently and as in intended without any dependencies, or compromise in Quality.

This is applicable to both web and mobile applications.

Cross Browser Testing using Selenium WebDriver

- Run Same Test Case using different Drivers (Browsers)
- This can be achieved using TestNG
- Also this can be achieved using Selenium Grid

Introducing WebDriver History

In 2006 a plucky engineer at Google named Simon Stewart started work on a project he called WebDriver. Google had long been a heavy user of Selenium, but testers had to work around the limitations of the product. Simon wanted a testing tool that spoke directly to the browser using the ‘native’ method for the browser and operating system, thus avoiding the restrictions of a sandboxed Javascript environment. The WebDriver project began with the aim to solve the Selenium’ pain-points.

Introducing WebDriver History

Jump to 2008. The Beijing Olympics mark China’s arrival as a global power, massive mortgage default in the United States triggers the worst international recession since the Great Depression, The Dark Knight is viewed by every human (twice), still reeling from the untimely loss of Heath Ledger. But the most important story of that year was the merging of Selenium and WebDriver. Selenium had massive community and commercial support, but WebDriver was clearly the tool of the future. The joining of the two tools provided a common set of features for all users and brought some of the brightest minds in test automation under one roof. Perhaps the best explanation for why WebDriver and Selenium are merging was detailed by Simon Stewart, the creator of WebDriver, in a joint email to the WebDriver and Selenium community on August 6, 2009.

Setting up Java

Download

http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html

Double click it

Setting up Eclipse

Download http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/mars2

Extract and Open Eclipse

Setting up Gradle (Build Tool)

Download https://gradle.org/gradle-download/

Extract it and Put bin/gradle in the PATH

Setting up TestNG and WebDriver using Gradle

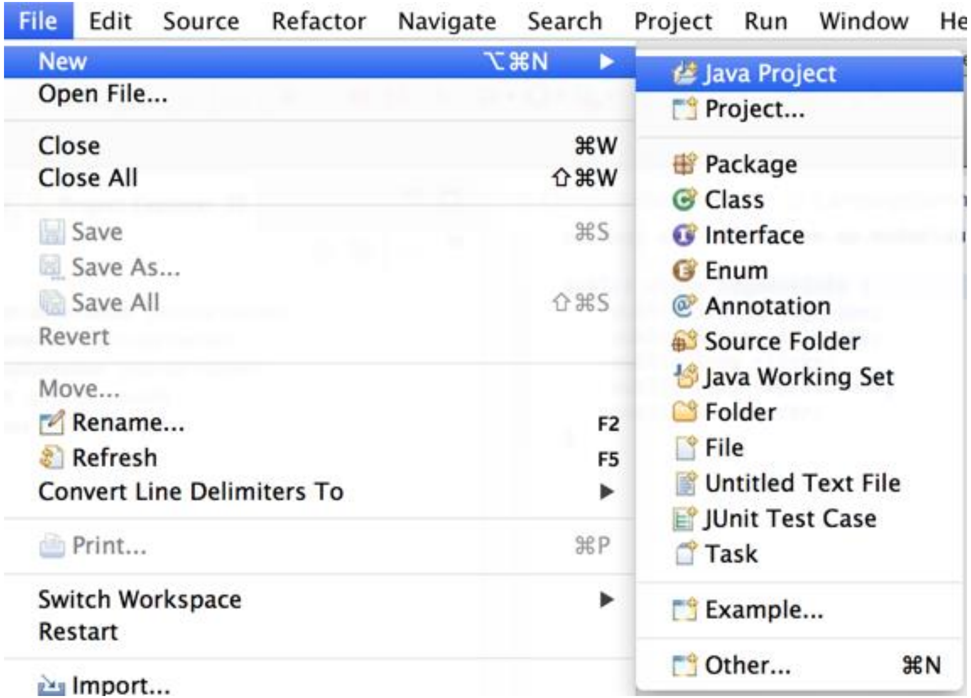
```
apply plugin: 'java'
apply plugin: 'eclipse'

repositories {
    mavenCentral()
}

dependencies {
   compile 'mysql:mysql-connector-java:5.1.29'
   compile "org.testng:testng:6.8.7"
   compile("org.uncommons:reportng:1.1.2") {
      exclude group: "org.testng", module: "testng"
   }
   compile "com.jcraft:jsch:0.1.44-1"
   compile "org.seleniumhq.selenium:selenium-java:2.40.0"
   compile "javax.servlet:servlet-api:2.5"
   compile "net.sourceforge.jexcelapi:jxl:2.6.12"
}
```

```
version = '1.0'

jar {
    manifest.attributes provider: 'gradle'
}

task run(type: JavaExec, dependsOn: classes) {
   main = 'org.testng.TestNG'
   args = ['build/resources/main/testng.xml'].toList()
   classpath configurations.runtime
   classpath sourceSets.main.output
}
```

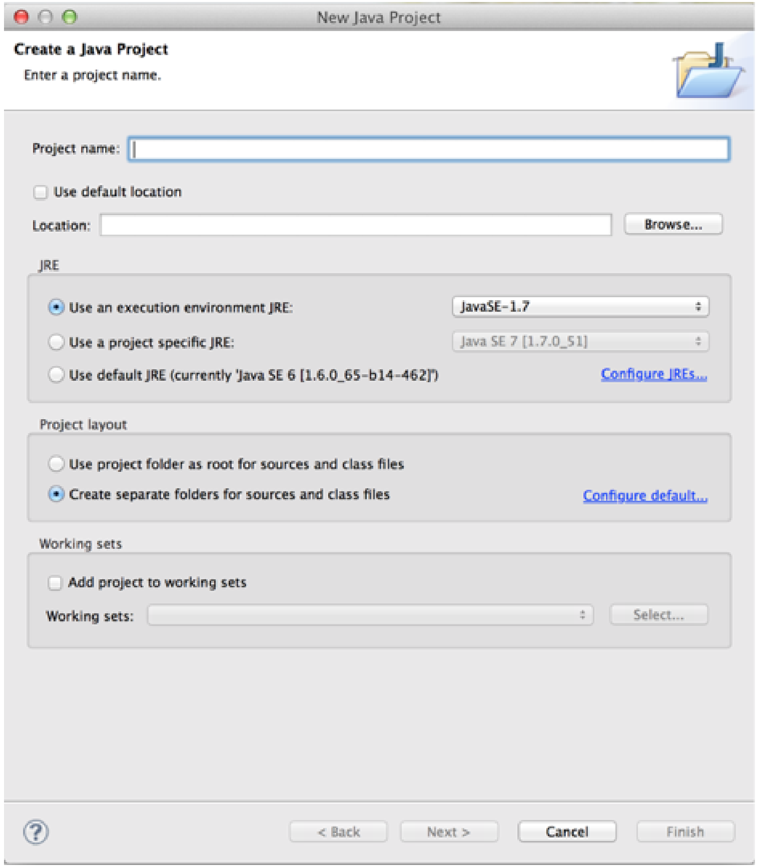
Creating and running tests on the setup

- Create Folder Structure as follows
- Project Folder -> src ->main -> java
- Create build.gradle file in Project Folder
- Run gradle cleanEclipse eclipse

Import Project into Eclipse



Importing Project



New Java Project

Create a Java Project

Enter a project name.

Project name:

□ Use default location

Location:

JRE

● Use an execution environment JRE: JavaSE-1.7

○ Use a project specific JRE: Java SE 7 [1.7.0_51]

○ Use default JRE (currently 'Java SE 6 [1.6.0_65-b14-462]')

Project layout

○ Use project folder as root for sources and class files

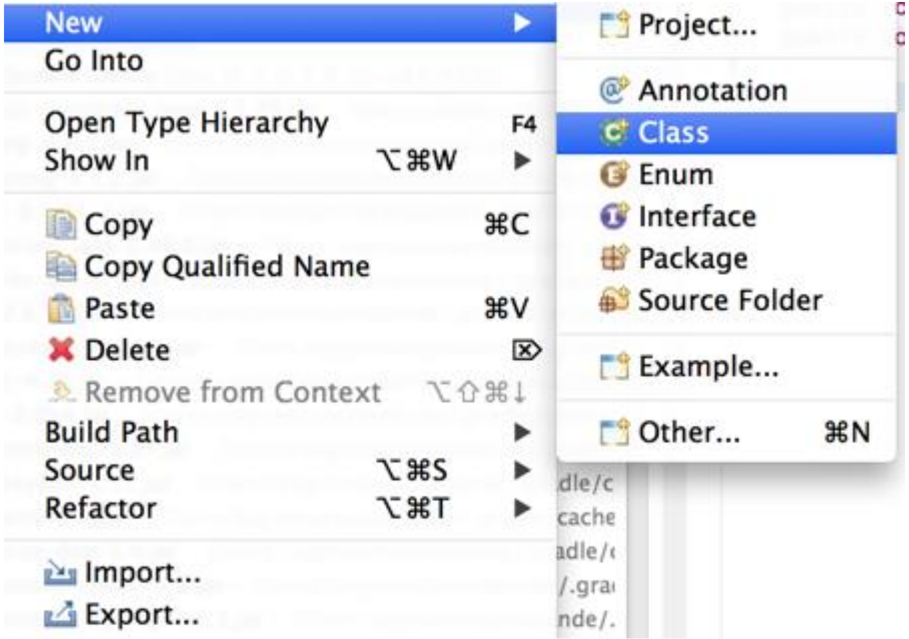
● Create separate folders for sources and class files

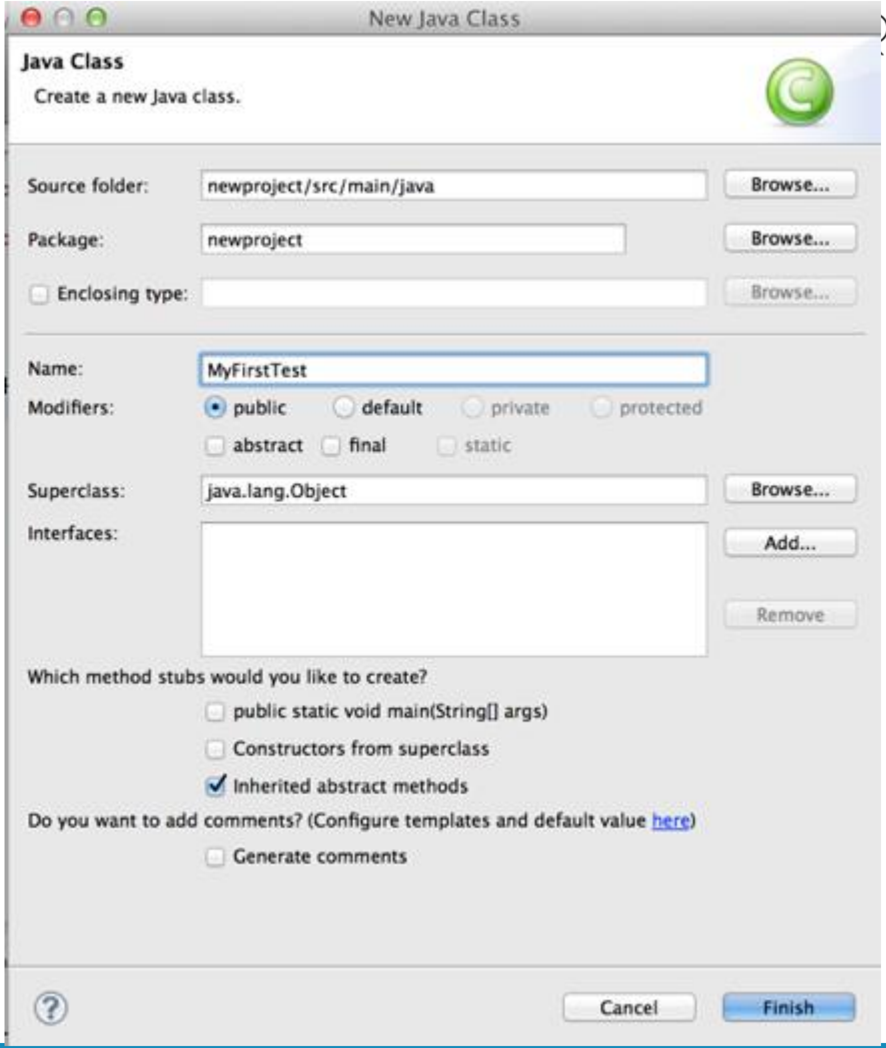
Working sets

□ Add project to working sets

Working sets:

Creating New Class





Java Class

Create a new Java class.

Source folder: newproject/src/main/java

Package: newproject

□ Enclosing type:

Name: MyFirstTest

Modifiers: ● public ○ default ○ private ○ protected

□ abstract □ final □ static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

□ public static void main(String[] args)

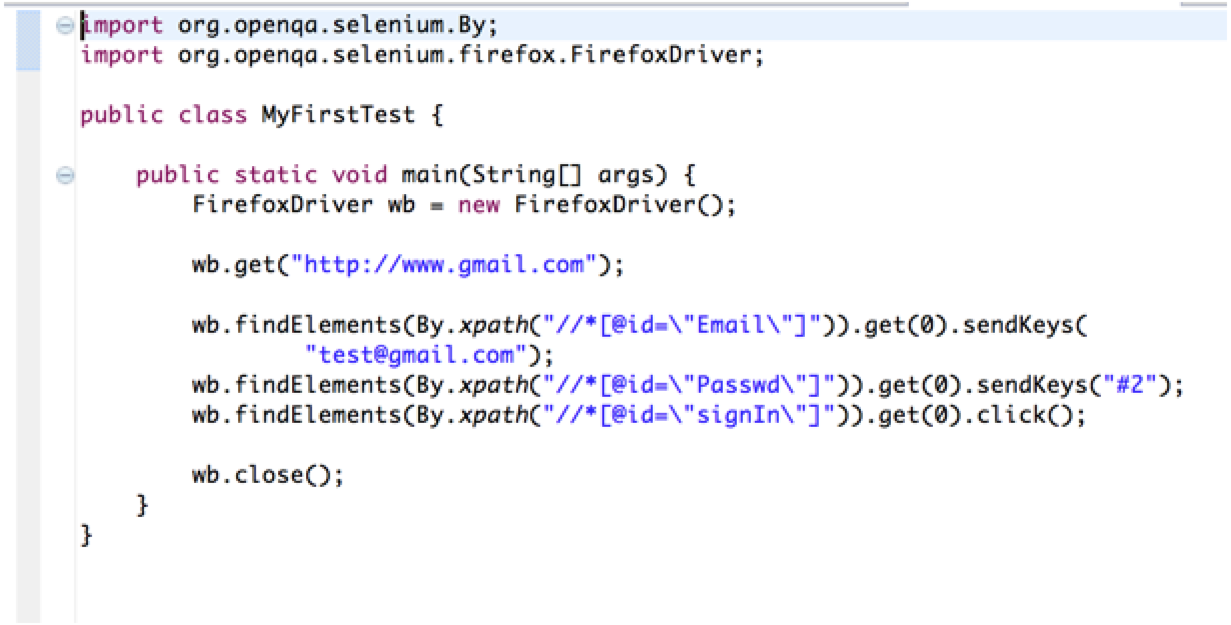
□ Constructors from superclass

☑ Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

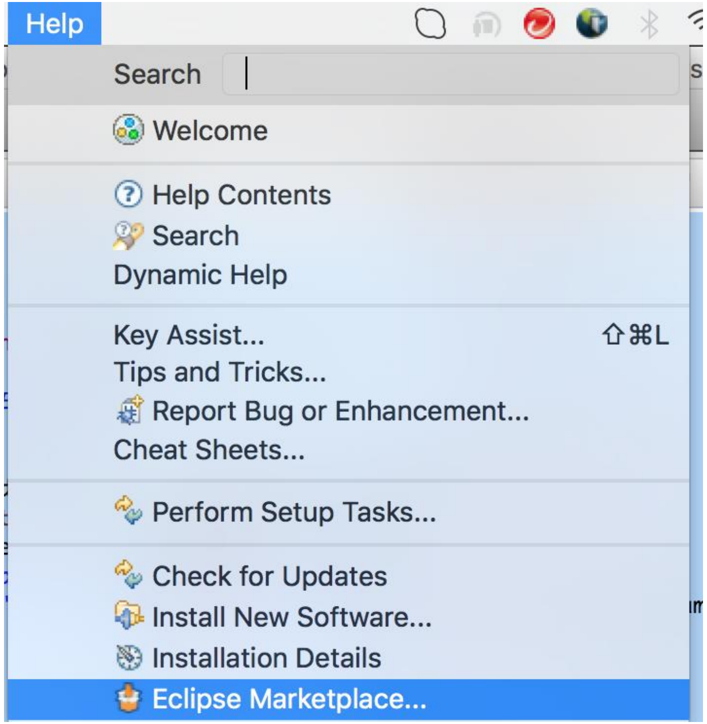
□ Generate comments

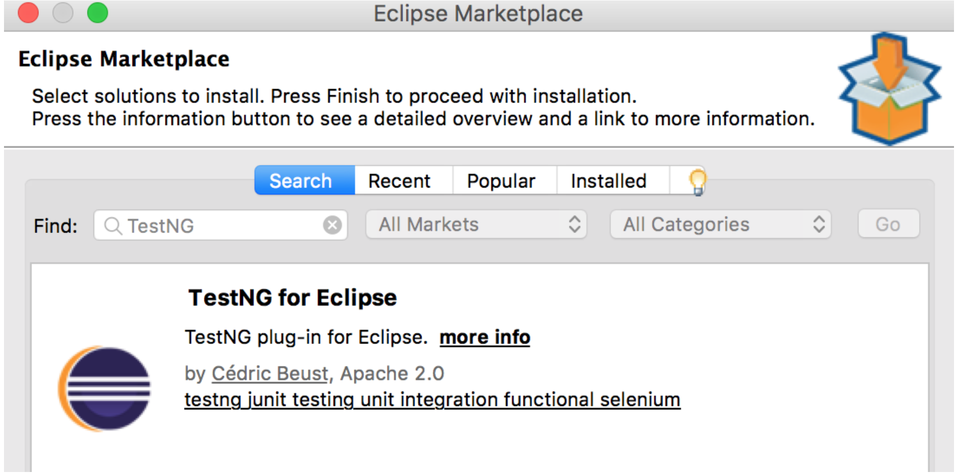
Writing First Test



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

public class MyFirstTest {

    public static void main(String[] args) {
        FirefoxDriver wb = new FirefoxDriver();

        wb.get("http://www.gmail.com");

        wb.findElements(By.xpath("//*[@id=\"Email\"]")).get(0).sendKeys(
                "test@gmail.com");
        wb.findElements(By.xpath("//*[@id=\"Passwd\"]")).get(0).sendKeys("#2");
        wb.findElements(By.xpath("//*[@id=\"signIn\"]")).get(0).click();

        wb.close();
    }
}
```

Install TestNG Plugin for Eclipse





Eclipse Marketplace

Select solutions to install. Press Finish to proceed with installation.
Press the information button to see a detailed overview and a link to more information.

Find: TestNG

TestNG for Eclipse

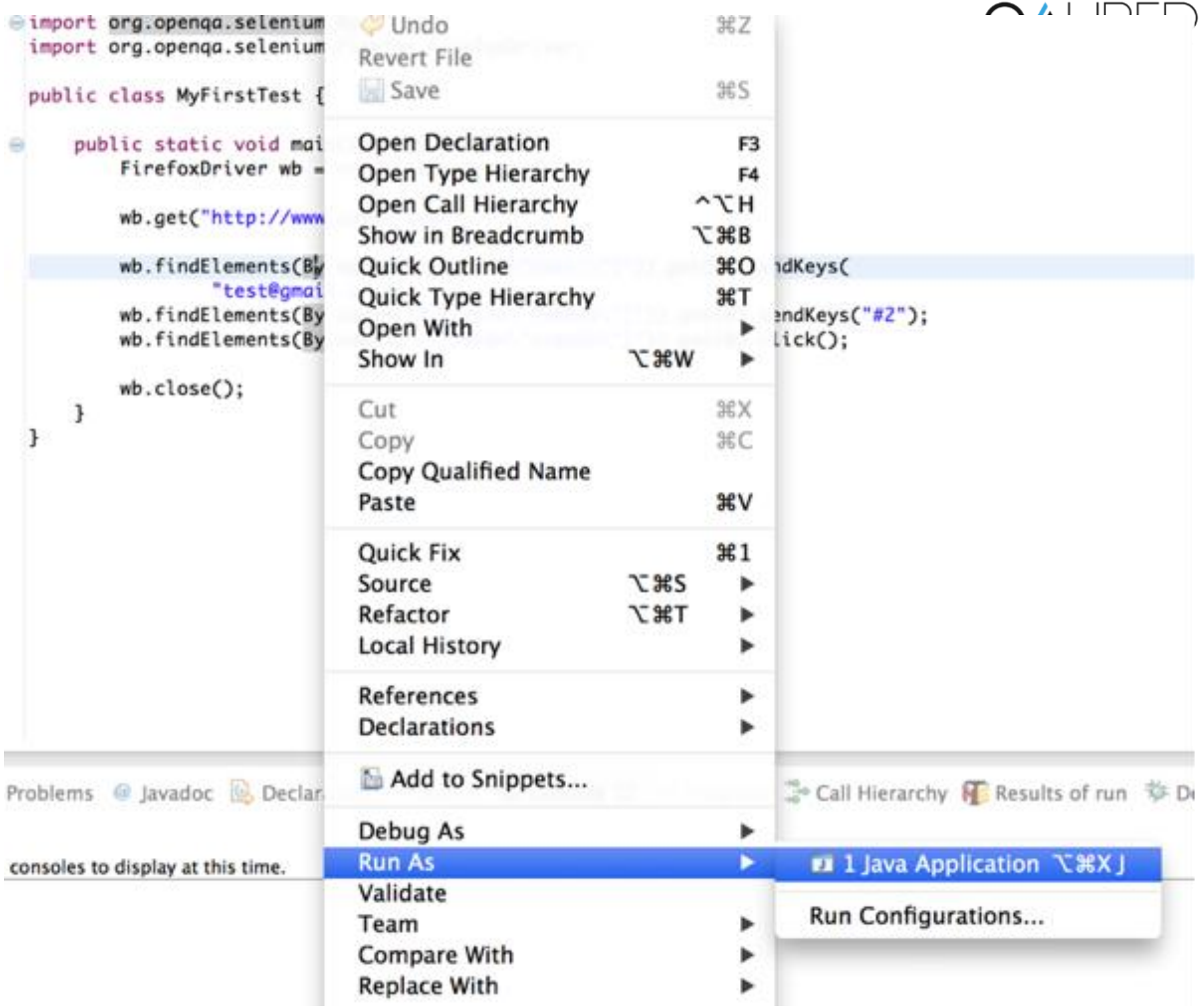
TestNG plug-in for Eclipse. **more info**

by Cédric Beust, Apache 2.0

testng junit testing unit integration functional selenium

Install and Restart Eclipse

Running Test



```
import org.openqa.selenium
import org.openqa.selenium

public class MyFirstTest {

    public static void mai
        FirefoxDriver wb =

        wb.get("http://www

        wb.findElements(By                    ndKeys(
                "test@gmai
        wb.findElements(By                    endKeys("#2");
        wb.findElements(By                    lick();

        wb.close();
    }
}
```

consoles to display at this time.