

Selenium with Eclipse

For using Eclipse you will also need the tools installed for IntelliJ, but with Eclipse most of them come with it. The only ones that don't come with Eclipse are Git and Java JDK. If you already installed Java JDK and Git then no need to reinstall them. Please refer to the other document of installing the tools if you want more **details** on how to install JDK or Git.

Installing tools necessary to create your first Functional Automation Test with Eclipse

You'll need the following tools, so install them in this order:

Java JDK

<http://www.oracle.com/technetwork/java/javase/downloads/index.html> Click on the Java Download icon and download the latest for your OS.

Eclipse

Please refer to another document about installing Eclipse if you need **details**.

If not already installed: <https://eclipse.org/downloads/> Select either "Eclipse IDE for Java Developers" or "Eclipse IDE for Java EE Developers" and Download according to your OS.

Note: the latest version of Eclipse is **Mars**, which **already has Maven installed**. If you are using Mars, verify that you have it installed by going to Help => About Eclipse, and then clicking on Installation Details.

Expand Eclipse IDE for Java EE Developer or Java Developers, expand the next folder and check if Maven is there (actual name is m2e - Maven Integration for Eclipse...).

maven - <https://maven.apache.org>

Maven is **already** in the latest version of Eclipse (Mars as of now).

If you don't see maven in eclipse, then these are the steps to install it:

1. Go to Help => Eclipse Marketplace. Search for "maven integration with eclipse", and select Maven Integration for Eclipse (Luna and newer) 1.5 in my case. Click Install.
2. Verify all three boxes are checked, then click Confirm.
3. When you're done, restart Eclipse.

Selenium - <http://www.seleniumhq.org/download/maven.jsp>

Note: Selenium should be already installed in your Eclipse by default.

JUnit - <http://junit.org/dependency-info.html>

Note: JUnit should be already installed in your Eclipse by default.

Git - install Xcode from apple store if using MAC, if using Windows, install it from here: <https://git-scm.com/download/win>. To check the git version run the following command in terminal on MAC or in Git Bash if on windows:

```
git --version
```

* It should show something like the below:

(MAC) git version 2.5.4 (Apple Git-61)

(Windows) git version 2.7.0.windows.1

After having finished all of the above steps, you are ready to start creating test scripts with Selenium, but one **important note**:

- First try with the latest version of Firefox (43 at the time), but if you see issues (I haven't found any) with ids not found or errors, you'll need to use FireFox v31 as newer versions may not work well with Selenium WebDriver 2.43 or lower.

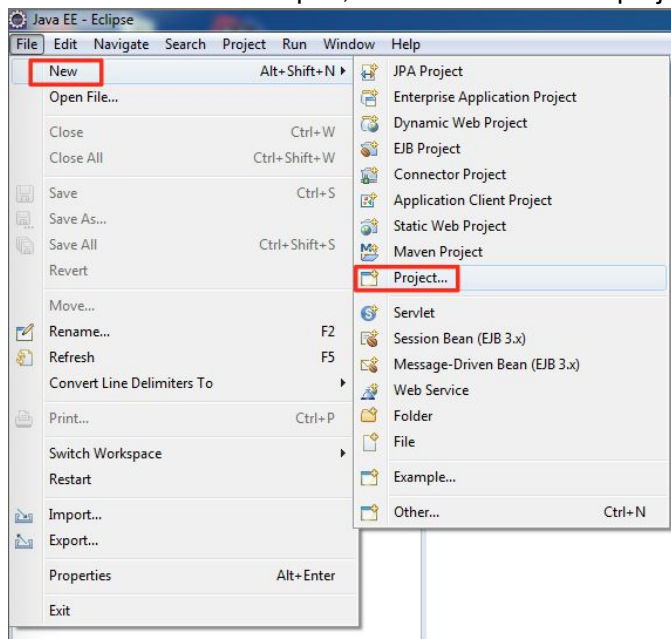
** Download version 31 from here: <https://ftp.mozilla.org/pub/firefox/releases/>

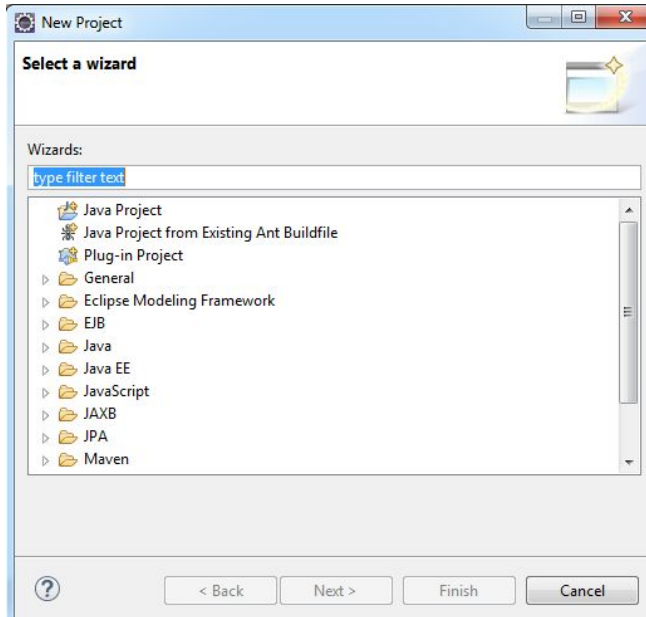
** MAC Firefox version 31: <https://ftp.mozilla.org/pub/firefox/releases/31.0/mac/en-US/>

** Windows Firefox version 31: <https://ftp.mozilla.org/pub/firefox/releases/31.0/win32/en-US/>

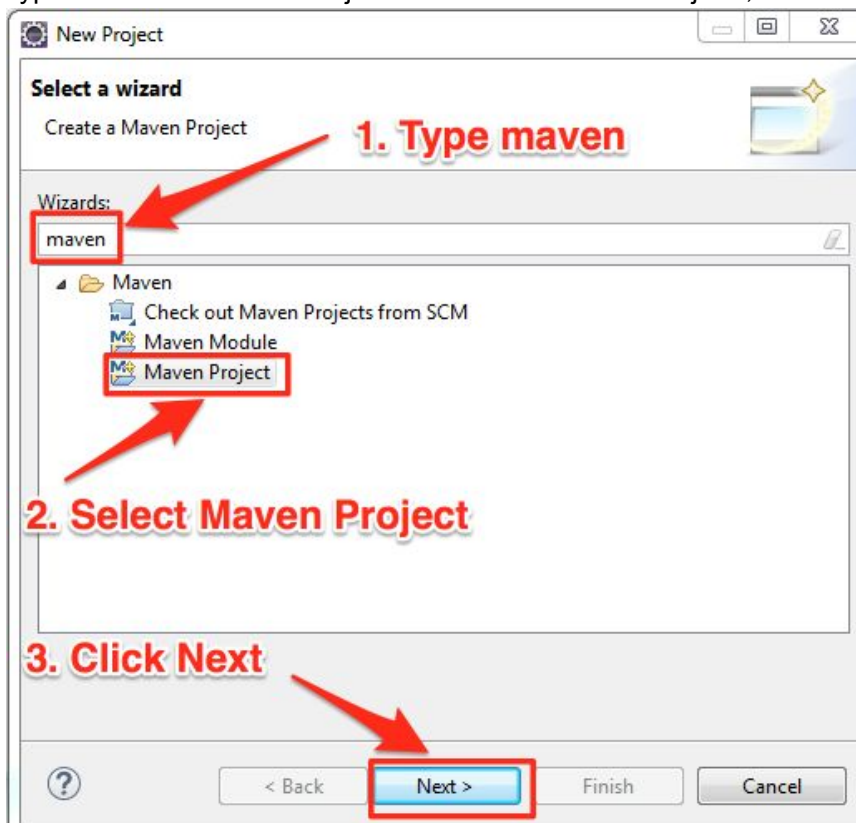
Create your first Functional Automation Test with Eclipse

To use Selenium in Eclipse, create a new Maven project. Go to File -> New -> Project...,

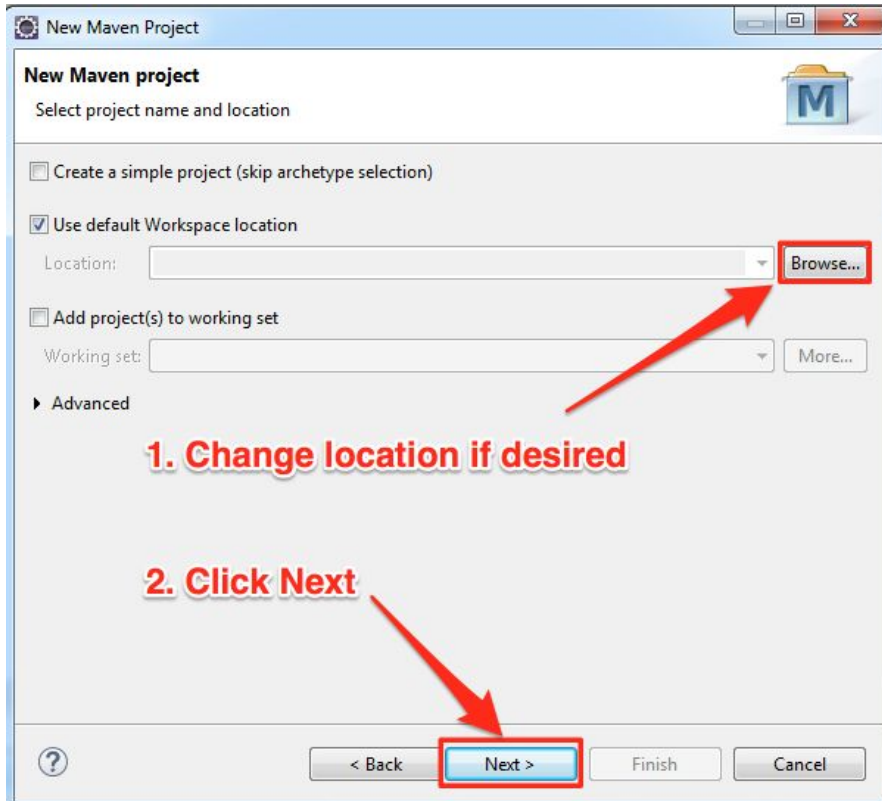




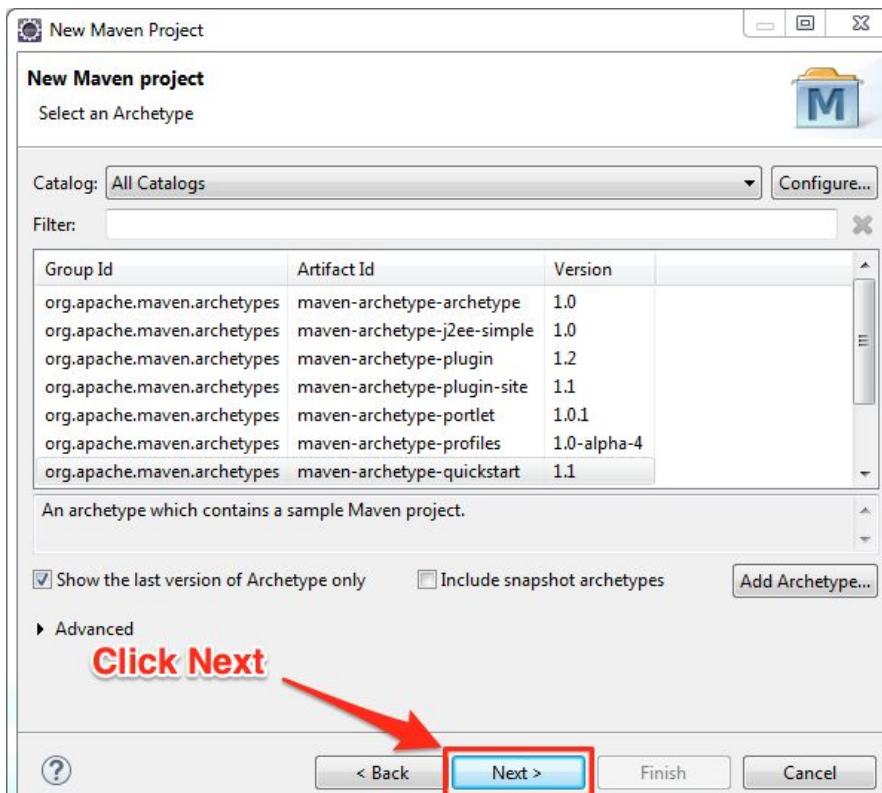
Type maven to filter the Projects and select "Maven Project", then click Next.



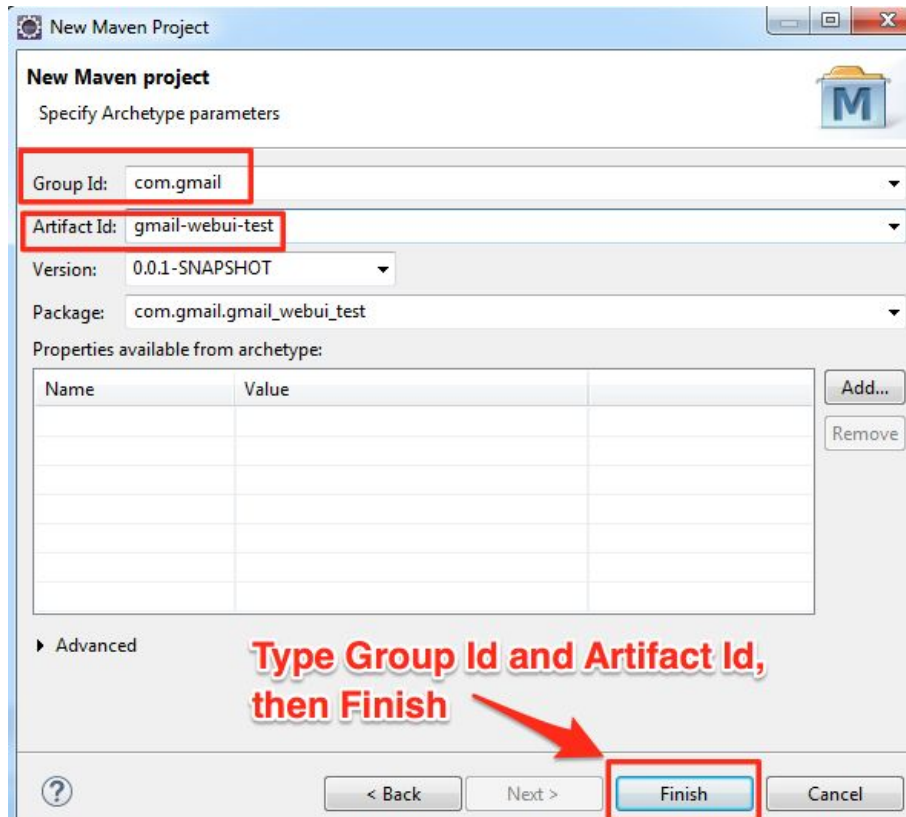
Change location if desired (default is fine to start) then Next:



Click Next:



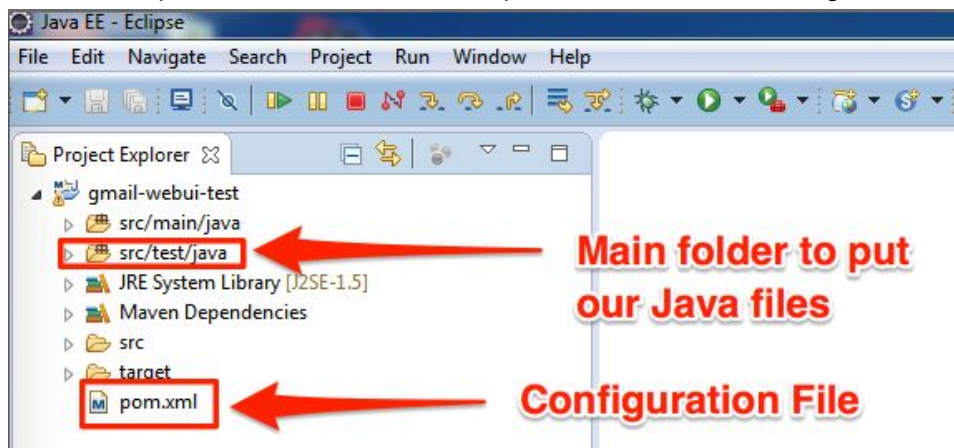
Enter com.gmail as Group Id and gmail-webui-test as Artifact Id, then click on Finish:



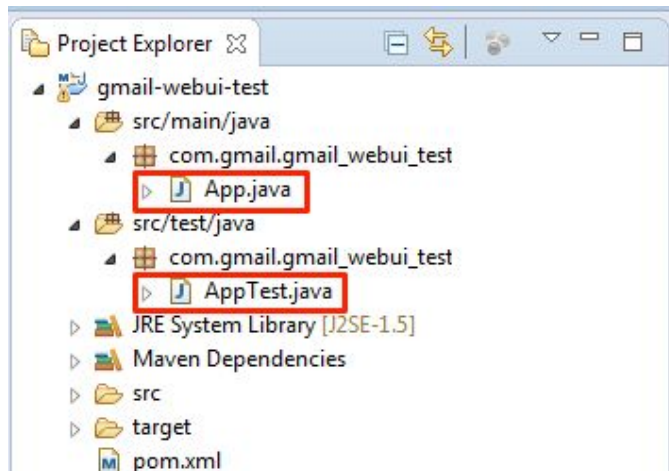
Explanation of the GUI:

On the left side there is the project name and when expanding it it contains different folders. Expand the folder gmail-webui-test. The src/test/java folder is the source folder and it is where we will be putting our Java files.

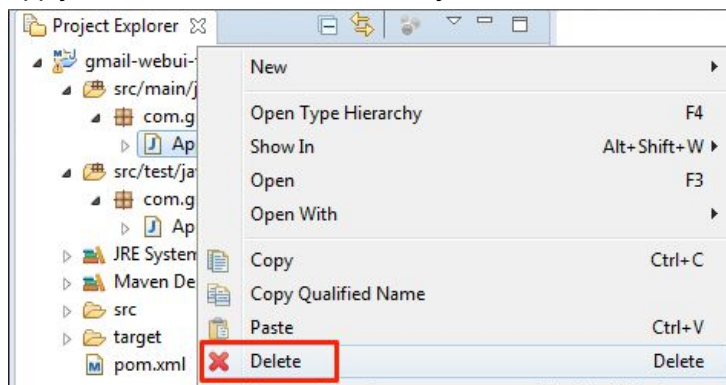
The most important file for maven is the pom.xml file. It is the configuration file.



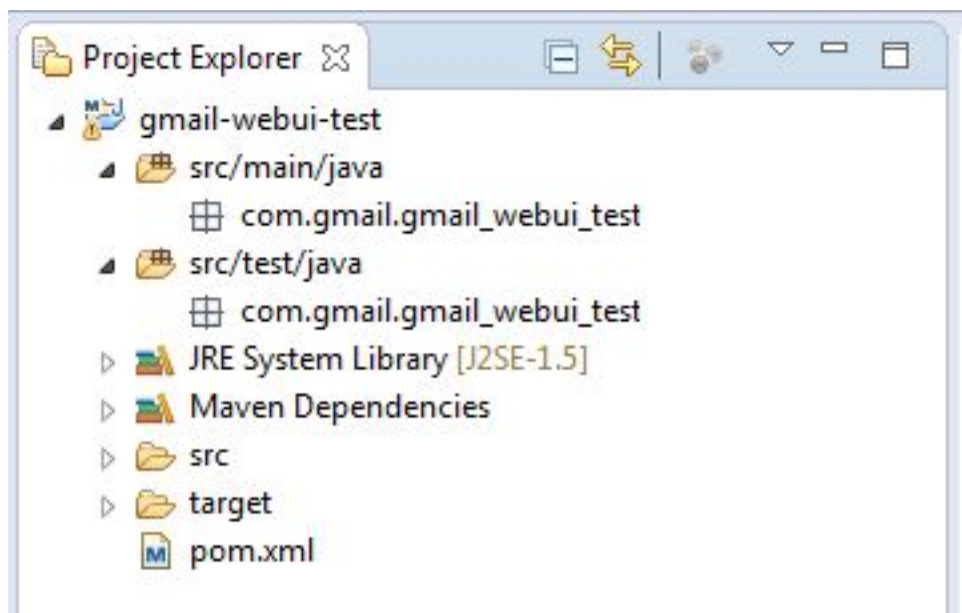
Expand src/main/java until you see the java file App.java and also expand src/test/java until you see AppTest.java.



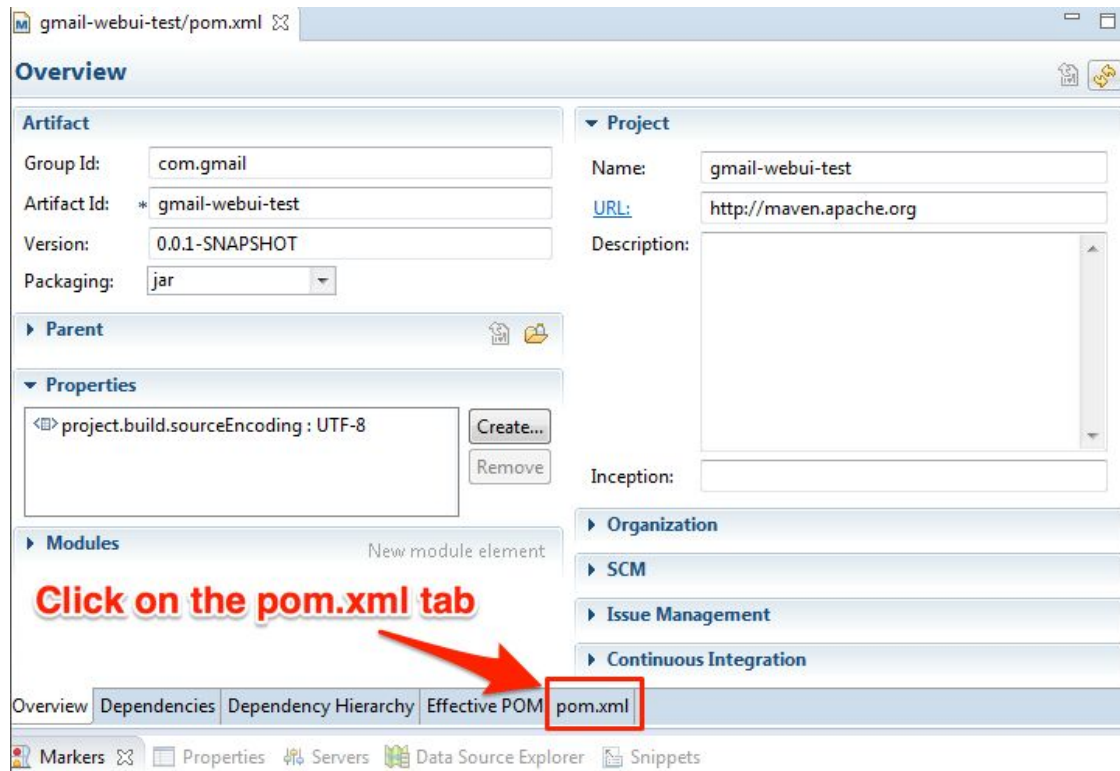
Right click and **delete** the java file AppTest.java under the folder src/test/java and also the file App.java under the folder src/main/java. This is since we'll create our own java files.

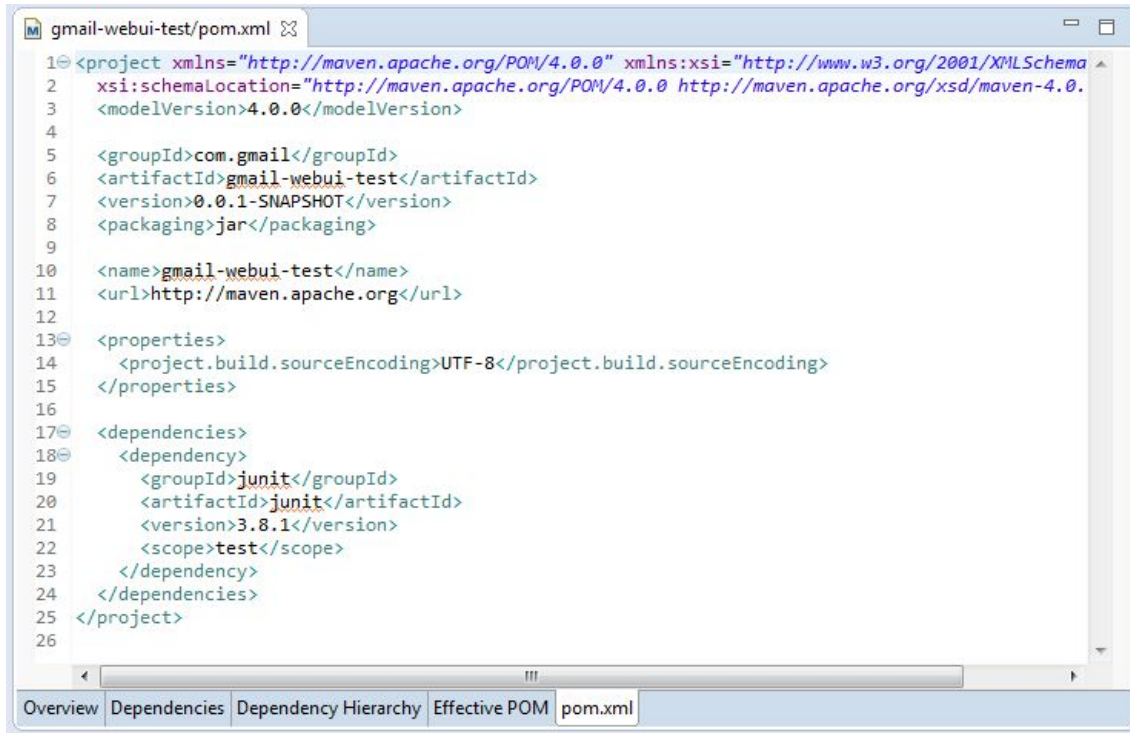


Project should look like this after deleting them:



Double-click the pom.xml and click on the pom.xml tab.





```
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema"
2   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.
3   <modelVersion>4.0.0</modelVersion>
4
5   <groupId>com.gmail</groupId>
6   <artifactId>gmail-webui-test</artifactId>
7   <version>0.0.1-SNAPSHOT</version>
8   <packaging>jar</packaging>
9
10  <name>gmail-webui-test</name>
11  <url>http://maven.apache.org</url>
12
13  <properties>
14    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15  </properties>
16
17  <dependencies>
18    <dependency>
19      <groupId>junit</groupId>
20      <artifactId>junit</artifactId>
21      <version>3.8.1</version>
22      <scope>test</scope>
23    </dependency>
24  </dependencies>
25 </project>
26
```

Overview Dependencies Dependency Hierarchy Effective POM pom.xml

Go to <http://www.seleniumhq.org/download/maven.jsp> and copy the snippet
<dependency>...</dependency>



www.seleniumhq.org/download/maven.jsp

SeleniumHQ
Browser Automation

[edit this page](#)

[Projects](#) [Download](#) [Documentation](#)

[Selenium Downloads](#)
[Latest Releases](#)
[Previous Releases](#)
[Source Code](#)
[Maven Information](#)

Donate to Selenium
with PayPal

Maven Information

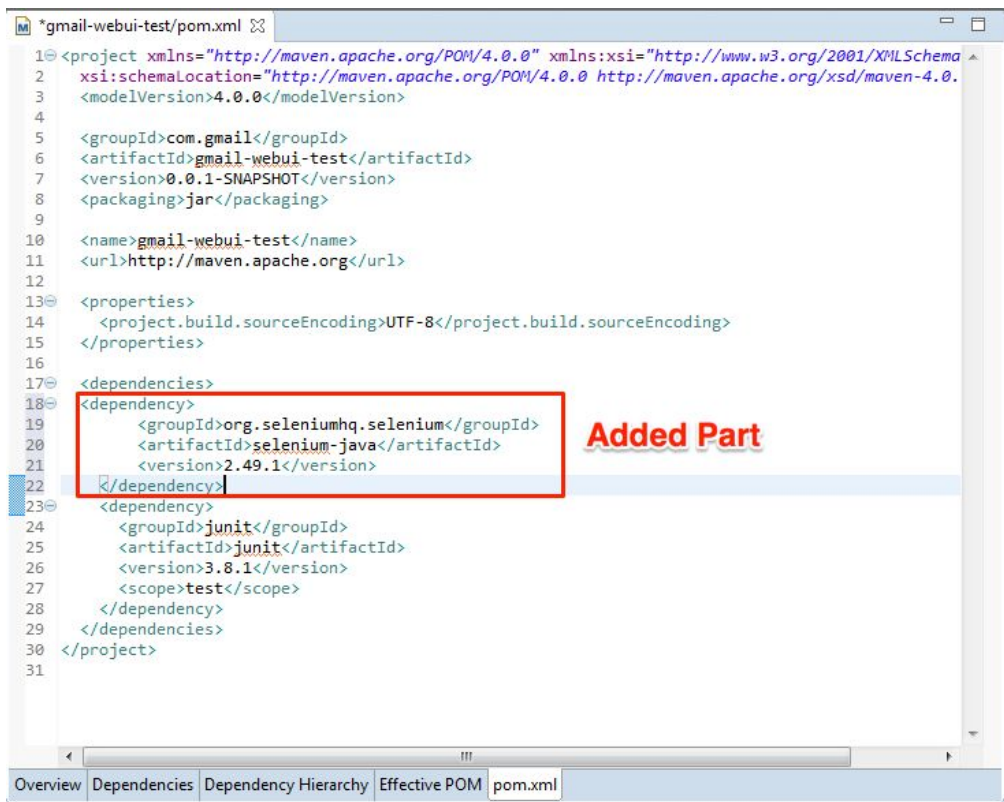
If you're using Maven, you will find all Selenium Maven artifacts directly in the repository here: <http://repo1.maven.org/maven2/org/seleniumhq/selenium/>

In order to start using DefaultSelenium or one of the new WebDriver implement project, just add the following dependency to your pom.xml:

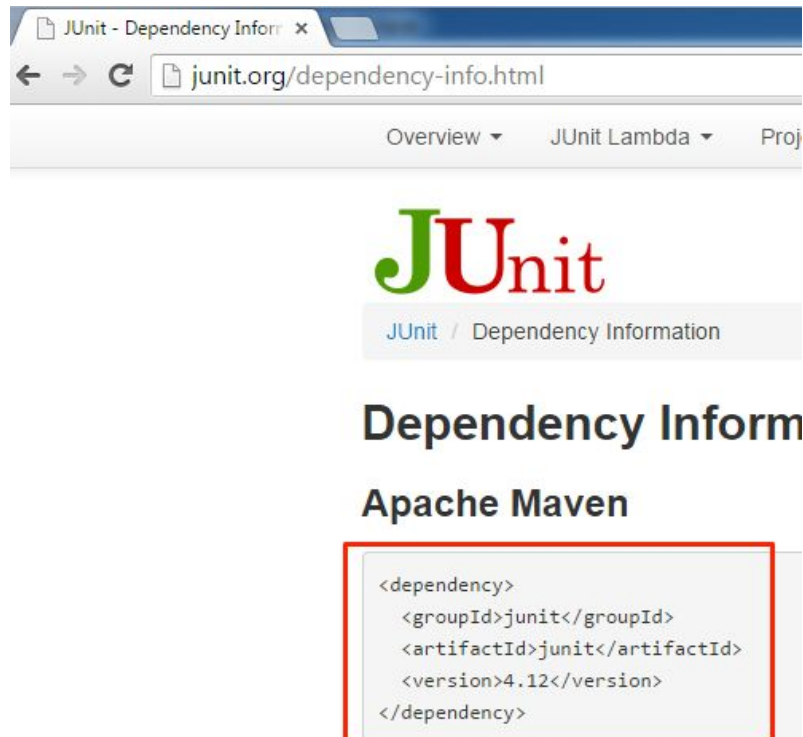
```
<dependency>
  <groupId>org.seleniumhq.selenium</groupId>
  <artifactId>selenium-java</artifactId>
  <version>2.49.1</version>
</dependency>
```

Note: Before version 2.0rc3 the artifact was named selenium-remote-control.

And paste it inside the dependencies tag.



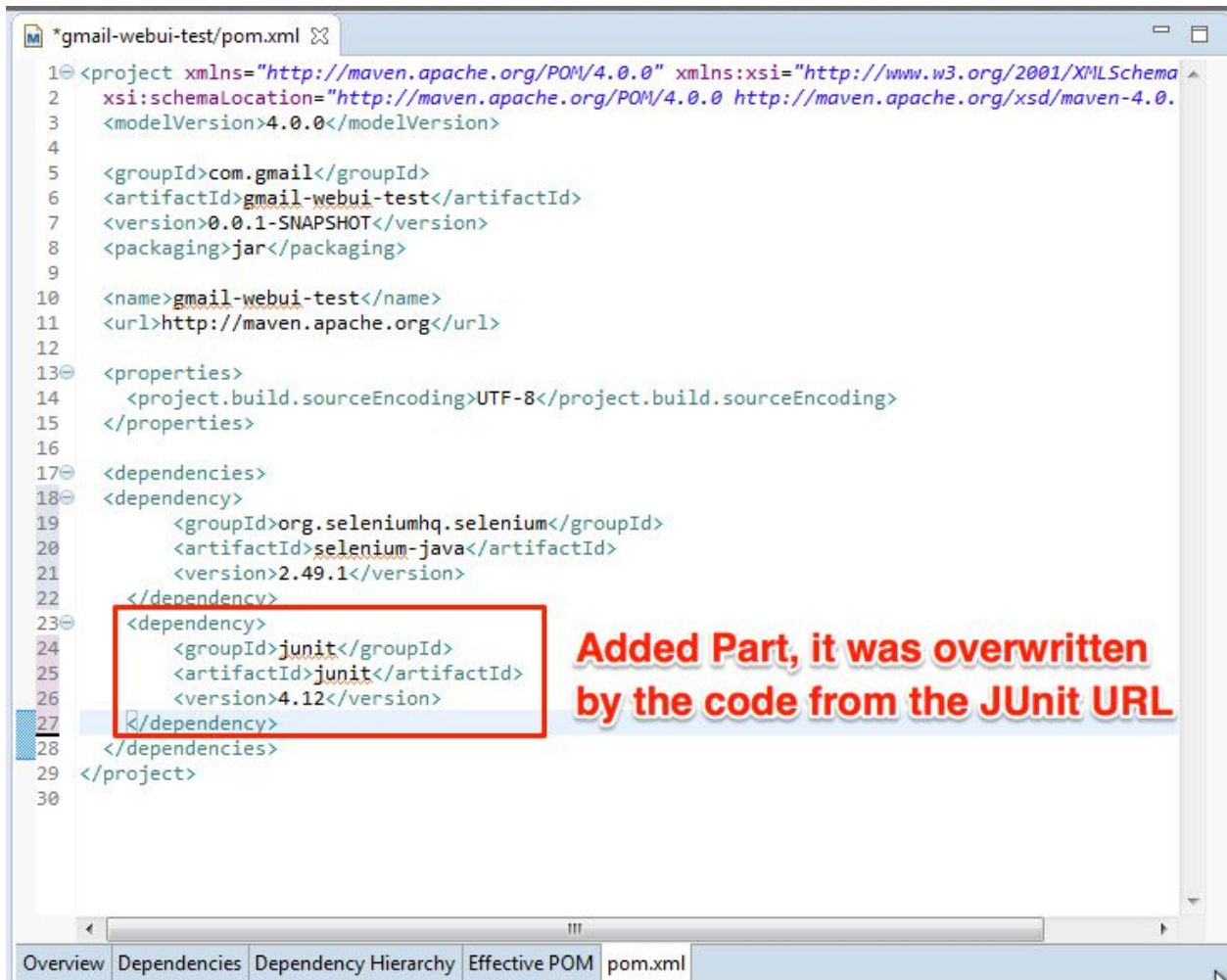
Go to <http://junit.org/dependency-info.html> and copy the snippet `<dependency>...</dependency>`



The screenshot shows a web browser window with the address bar displaying `junit.org/dependency-info.html`. The page title is "JUnit - Dependency Information". Below the title, there is a navigation bar with "Overview", "JUnit Lambda", and "Project". The main content area features the JUnit logo (a green 'J' and a red 'Unit') and a breadcrumb trail "JUnit / Dependency Information". The heading "Dependency Information" is followed by "Apache Maven". A red rectangular box highlights a code snippet for a Maven dependency:

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.12</version>
</dependency>
```

And paste it inside the dependencies tag. Since there is already one dependency for JUnit, overwrite it. The **save** the file (Ctrl + S).



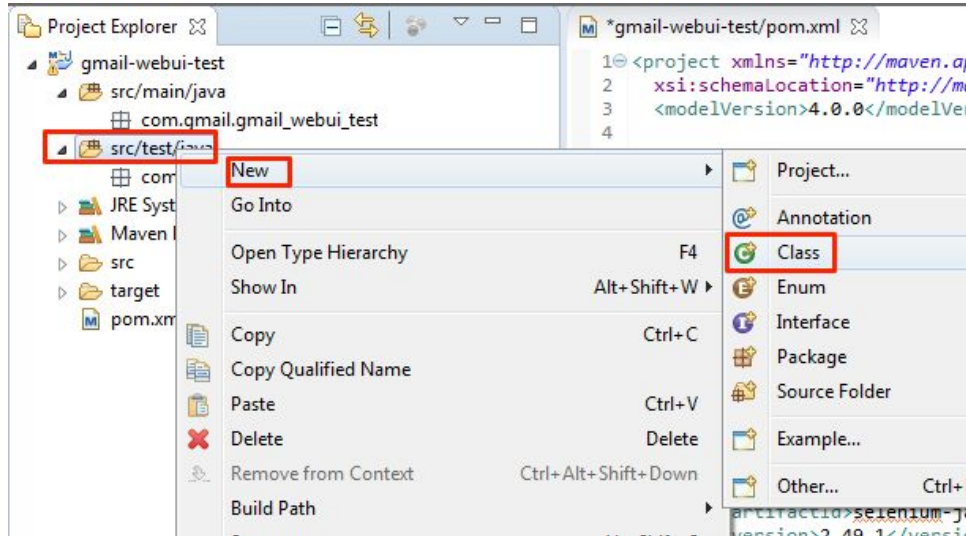
The pom.xml file should now look like the below (the **red bold** part is the two dependencies we just added).

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

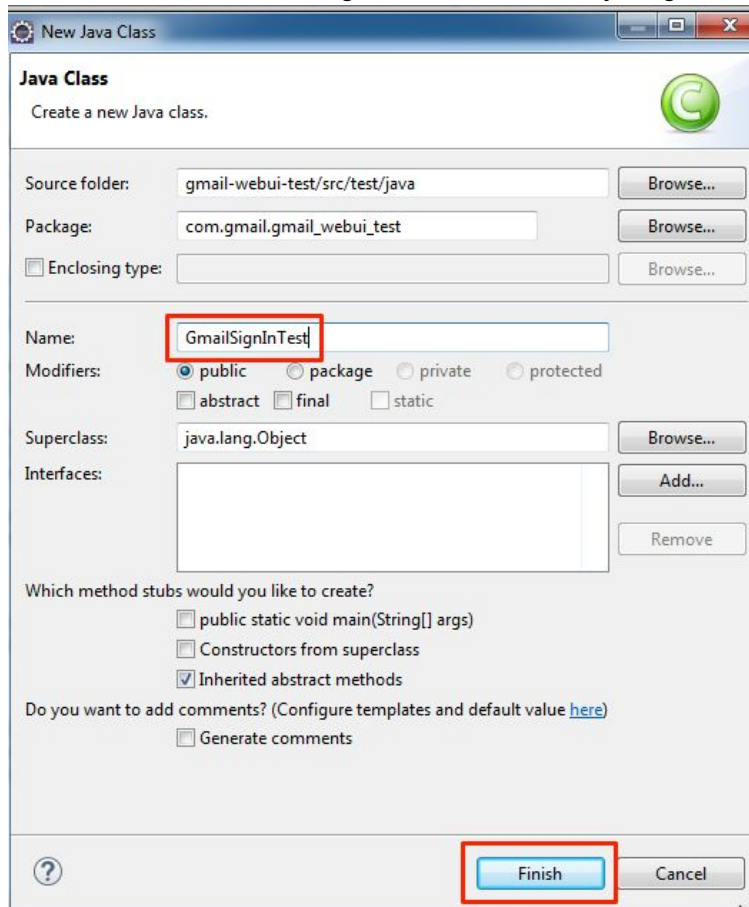
```
<dependencies>
  <dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>2.49.1</version>
  </dependency>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
  </dependency>
</dependencies>
```

```
</dependency>
</dependencies>
</project>
```

Now let's create our first java file. Right click on the folder src/test/java and go to New -> Class.

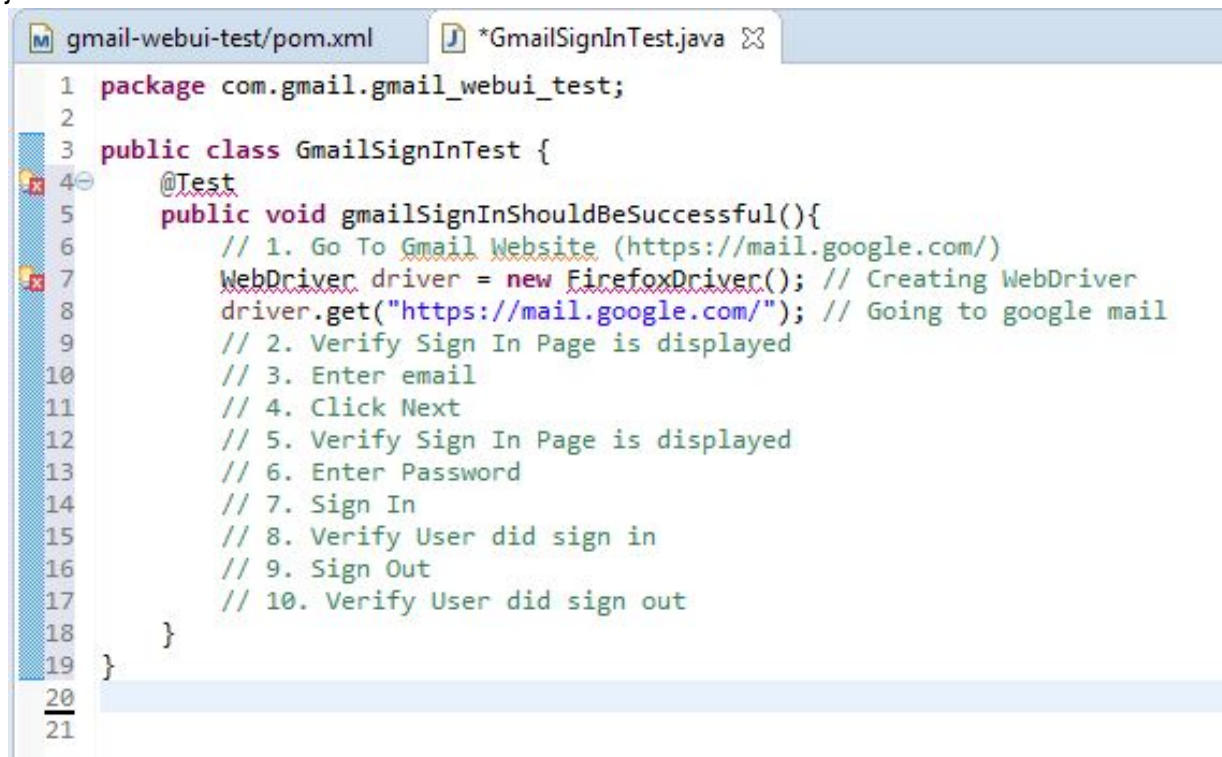


Name the class as GmailSignInTest. Leave everything else as default and click Finish:



Copy-paste the below code:

```
public class GmailSignInTest {
    @Test
    public void gmailSignInShouldBeSuccessful(){
        // 1. Go To Gmail Website (https://mail.google.com/)
        WebDriver driver = new FirefoxDriver(); // Creating WebDriver
        driver.get("https://mail.google.com/"); // Going to google mail
        // 2. Verify Sign In Page is displayed
        // 3. Enter email
        // 4. Click Next
        // 5. Verify Sign In Page is displayed
        // 6. Enter Password
        // 7. Sign In
        // 8. Verify User did sign in
        // 9. Sign Out
        // 10. Verify User did sign out
    }
}
```

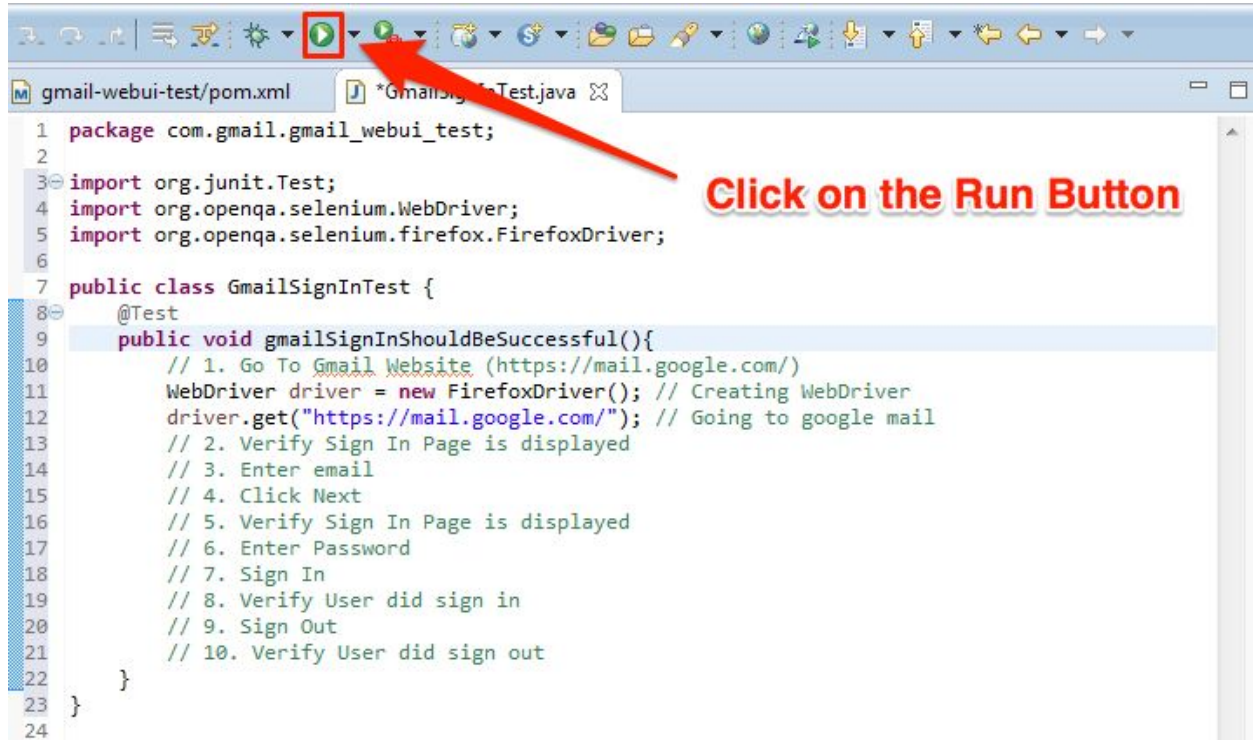


Import the necessary dependent packages. For that hover over the errors, for instance hover over the word Test, which is underlined in red and there will be a yellow pop-up, click on Import 'Test' (org.junit). Do the same for WebDriver and FirefoxDriver. Then **save** the java file (Ctrl + S).

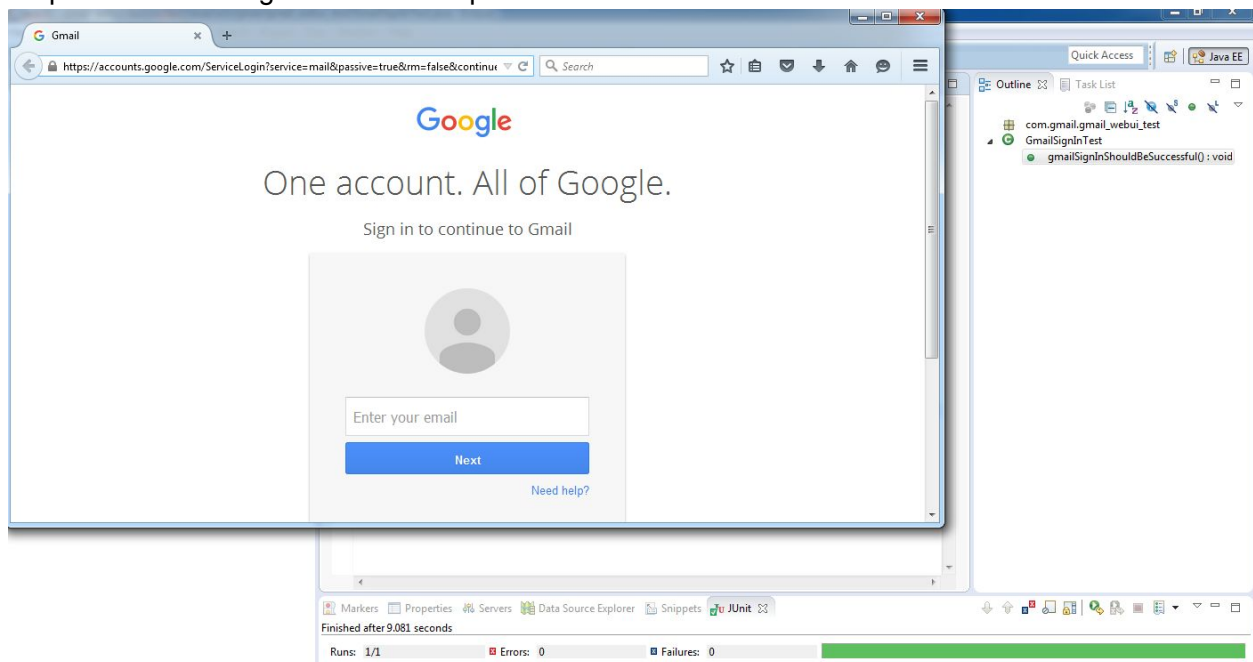
Import section should look like this after import:
import org.junit.Test;


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

Then click on the Run button to run the test.



If everything goes well, Firefox will open and it will go to the gmail Sign In page automatically and Eclipse will show as green on the left panel.



Quit the Firefox window opened by the test run.