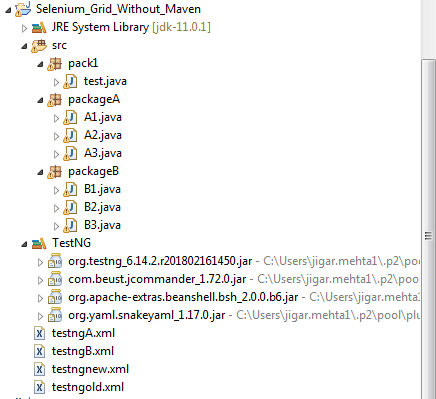
**Grid:**

TO understand Grid let’s look at below example demonstrating how to run suites in parallel i.e. testngA.xml and testngB.xml



**package** packageA;

**import** org.testng.annotations.Test;

**public** **class** A1 {

@Test

**public** **void** A1\_1() **throws** InterruptedException {

System.***out***.println("A1 start");

Thread.*sleep*(3000);

System.***out***.println("A1 ends");

}

}

**package** packageA;

**import** org.testng.annotations.Test;

**public** **class** A2 {

@Test

**public** **void** A2\_1() **throws** InterruptedException {

System.***out***.println("A2 start");

Thread.*sleep*(3000);

System.***out***.println("A2 ends");

}

}

**package** packageA;

**import** org.testng.annotations.Test;

**public** **class** A3 {

@Test

**public** **void** A3\_1() **throws** InterruptedException {

System.***out***.println("A3 start");

Thread.*sleep*(3000);

System.***out***.println("A3 ends");

}

}

**package** packageB;

**import** org.testng.annotations.Test;

**public** **class** B1 {

@Test

**public** **void** B1\_1() **throws** InterruptedException {

System.***out***.println("B1 start");

Thread.*sleep*(3000);

System.***out***.println("B1 ends");

}

}

**package** packageB;

**import** org.testng.annotations.Test;

**public** **class** B2 {

@Test

**public** **void** B2\_1() **throws** InterruptedException {

System.***out***.println("B2 start");

Thread.*sleep*(3000);

System.***out***.println("B2 ends");

}

}

**package** packageB;

**import** org.testng.annotations.Test;

**public** **class** B3 {

@Test

**public** **void** B3\_1() **throws** InterruptedException {

System.***out***.println("B3 start");

Thread.*sleep*(3000);

System.***out***.println("B3 ends");

}

}

**testngA.xml**

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

<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">

<suite name=*"SuiteA"* parallel=*"tests"*>

<test name=*"TestA1"*>

<classes>

<class name=*"packageA.A1"* />

</classes>

</test>

<test name=*"TestA2"*>

<classes>

<class name=*"packageA.A2"* />

</classes>

</test>

<test name=*"TestA3"*>

<classes>

<class name=*"packageA.A3"* />

</classes>

</test>

</suite> <!-- Suite -->

**testngB.xml**

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

<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">

<suite name=*"SuiteB"* parallel=*"tests"*>

<test name=*"TestB1"*>

<classes>

<class name=*"packageB.B1"* />

</classes>

</test>

<test name=*"TestB2"*>

<classes>

<class name=*"packageB.B2"* />

</classes>

</test>

<test name=*"TestB3"*>

<classes>

<class name=*"packageB.B3"* />

</classes>

</test>

</suite> <!-- Suite -->

**testngold.xml (this xml calls** *testngA.xml and testngB.xml)*

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

<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">

<suite name=*"Suite"* >

<suite-files>

<suite-file path=*"./testngA.xml"*></suite-file>

<suite-file path=*"./testngB.xml"*></suite-file>

</suite-files>

</suite> <!-- Suite -->

**test.java**

**package** pack1;

**import** java.util.Arrays;

**import** java.util.List;

**import** org.testng.TestNG;

**import** org.testng.collections.Lists;

**public** **class** test {

**public** **static** **void** main(String[] args) {

/\* TestNG tng = new TestNG();

List suites = Lists.newArrayList();

// suites.add(System.getProperty("user.dir") + "\\testngA.xml");

// suites.add("D:\\Eclipse SimRel WorkSpace\\Selenium\_Grid\\testngB.xml");

// OR

\*/

TestNG tng = **new** TestNG();

List<String> suites = Lists.*newArrayList*();

suites.add(System.*getProperty*("user.dir") + "\\testngold.xml");

tng.setTestSuites(suites);

tng.setSuiteThreadPoolSize(2); // this parameter is important in order to run suites in parallel

tng.run();

/\* TestNG tng = new TestNG();

tng.setTestSuites(Arrays.asList(new String[] {System.getProperty("user.dir") + "//testngold.xml"}));

tng.setSuiteThreadPoolSize(2);

tng.run();

\*/

}

}

Run test.java in Eclipse and below is the output. Since all “start” statements are printed together shows both A and B suite runs parallel.

A2 start

A1 start

B3 start

A3 start

B1 start

B2 start

A2 ends

A1 ends

B3 ends

B2 ends

A3 ends

B1 ends

===============================================

SuiteB

Total tests run: 3, Failures: 0, Skips: 0

===============================================

===============================================

SuiteA

Total tests run: 3, Failures: 0, Skips: 0

===============================================

**Selenium grid for RC and WebDriver**

Grid allows you to:

* Scale by distributing tests on several machines ( parallel execution )
* Manage multiple environments from a central point, making it easy to run the tests against a vast combination of browsers / OS.

**Step 1: Start the hub**

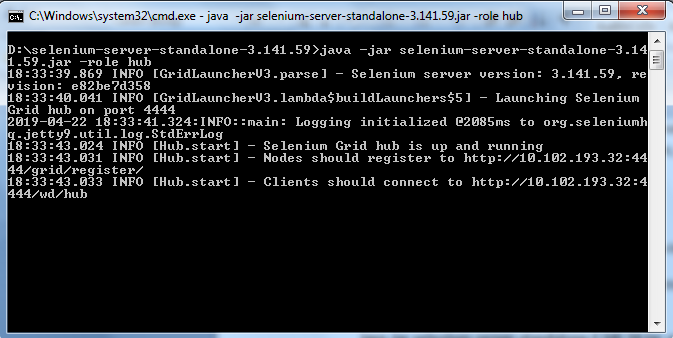
1. Download Selenium Standalone Server from below link. Current download version is 3.141.59

<https://www.seleniumhq.org/download/>

1. Open a command prompt and navigate to the directory where you copied the selenium-server-standalone file. Type the following command:

**java -jar selenium-server-standalone-3.141.59.jar -role hub**

The hub will automatically start-up using port 4444 by default. To change the default port, you can add the optional parameter -port when you run the command. You can view the status of the hub by opening a browser window and navigating to: <http://localhost:4444/grid/console>



**Step 2: Configuring the nodes by JSON**

java -jar selenium-server-standalone-3.141.59.jar -role node -nodeConfig node1.json

java -jar selenium-server-standalone-3.141.59.jar -role node -nodeConfig node2.json

A sample nodeconfig file for server version 3.x.x (>= beta4) can be seen at <https://github.com/SeleniumHQ/selenium/blob/master/java/server/src/org/openqa/grid/common/defaults/DefaultNodeWebDriver.json>

Below is sample code for node1.json / node2.json

{

"capabilities":

[

{

"browserName": "firefox",

"marionette": **true**,

"maxInstances": 5,

"seleniumProtocol": "WebDriver"

},

{

"browserName": "chrome",

"maxInstances": 5,

"seleniumProtocol": "WebDriver"

},

{

"browserName": "internet explorer",

"platform": "WINDOWS",

"maxInstances": 1,

"seleniumProtocol": "WebDriver"

},

{

"browserName": "safari",

"technologyPreview": **false**,

"platform": "MAC",

"maxInstances": 1,

"seleniumProtocol": "WebDriver"

}

],

"proxy": "org.openqa.grid.selenium.proxy.DefaultRemoteProxy",

"maxSession": 1,

"port": -1,

"register": **true**,

"registerCycle": 5000,

"hub": "http://localhost:4444",

"nodeStatusCheckTimeout": 5000,

"nodePolling": 5000,

"role": "node",

"unregisterIfStillDownAfter": 60000,

"downPollingLimit": 2,

"debug": **false**,

"servlets" : [],

"withoutServlets": [],

"custom": {}

}

**Step 3: Configuring the hub by JSON**

java -jar selenium-server-standalone.jar -role hub -hubConfig hubconfig.json

A sample hubconfig.json file can be seen at

<https://github.com/SeleniumHQ/selenium/blob/master/java/server/src/org/openqa/grid/common/defaults/DefaultHub.json>

{

"port": 4444,

"newSessionWaitTimeout": -1,

"servlets" : [],

"withoutServlets": [],

"custom": {},

"capabilityMatcher": "org.openqa.grid.internal.utils.DefaultCapabilityMatcher",

"registry": "org.openqa.grid.internal.DefaultGridRegistry",

"throwOnCapabilityNotPresent": **true**,

"cleanUpCycle": 5000,

"role": "hub",

"debug": **false**,

"browserTimeout": 0,

"timeout": 1800

}

**Using grid to run tests**

(using java as an example ) Now that the grid is in-place, we need to access the grid from our test cases.

For WebDriver nodes, you will need to use the RemoteWebDriver and the DesiredCapabilities object to define which browser, version and platform you wish to use. Create the target browser capabilities you want to run the tests against:

DesiredCapabilities capability = DesiredCapabilities.firefox();

Pass that into the RemoteWebDriver object:

WebDriver driver = new RemoteWebDriver(new URL("http://localhost:4444/wd/hub"), capability);

The hub will then assign the test to a matching node.

A node matches if all the requested capabilities are met. To request specific capabilities on the grid, specify them before passing it into the WebDriver object.

capability.setBrowserName();

capability.setPlatform();

capability.setVersion()

capability.setCapability();

Example: A node registered with the setting:

-browser browserName=firefox,version=3.6,platform=LINUX will be a match for:

capability.setBrowserName(“firefox” );

capability.setPlatform(“LINUX”);

capability.setVersion(“3.6”);

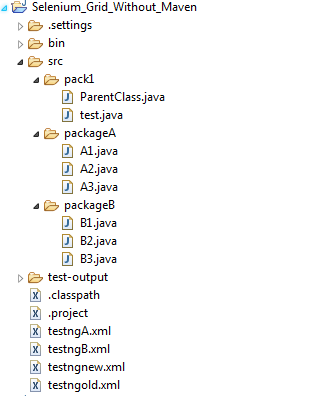
and would also be a match for

capability.setBrowserName(“firefox” );

capability.setVersion(“3.6”);

The capabilities that are not specified will be ignored. If you specify capabilities that do not exist on your grid (for example, your test specifies Firefox version 4.0, but have no Firefox 4 instance) then there will be no match and the test will fail to run.

**Create below project in Eclipse**



ParentClass.java

**package** pack1;

**import** java.net.MalformedURLException;

**import** java.net.URL;

**import** org.openqa.selenium.Platform;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.remote.DesiredCapabilities;

**import** org.openqa.selenium.remote.RemoteWebDriver;

**import** org.testng.annotations.AfterMethod;

**public** **class** ParentClass {

**public** WebDriver driver;

**public** **void** openBrowser(String browser) {

DesiredCapabilities dc = **new** DesiredCapabilities();

**if** (browser.equals("Firefox")) {

dc = DesiredCapabilities.*firefox*();

dc.setPlatform(Platform.***WINDOWS***);

System.***out***.println("Inside ParentClass and firefox browser");

} **else** **if** (browser.equals("Chrome")) {

dc = DesiredCapabilities.*chrome*();

dc.setPlatform(Platform.***WINDOWS***);

System.***out***.println("Inside ParentClass and Chrome browser");

} **else** **if** (browser.equals("IE")) {

dc = DesiredCapabilities.*internetExplorer*();

dc.setPlatform(Platform.***WINDOWS***);

System.***out***.println("Inside ParentClass and IE browser");

}

**try** {

driver = **new** RemoteWebDriver(**new** URL("http://localhost:4444/wd/hub"), dc);

} **catch** (MalformedURLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

@AfterMethod

**public** **void** quitBrowser() {

driver.quit();

}

}

test.java

**package** pack1;

**import** java.util.List;

**import** org.testng.TestNG;

**import** org.testng.collections.Lists;

**public** **class** test {

**public** **static** **void** main(String[] args) {

/\* TestNG tng = new TestNG();

List suites = Lists.newArrayList();

// suites.add(System.getProperty("user.dir") + "\\testngA.xml");

// suites.add("D:\\Eclipse SimRel WorkSpace\\Selenium\_Grid\\testngB.xml");

// OR

\*/

TestNG tng = **new** TestNG();

List<String> suites = Lists.*newArrayList*();

suites.add(System.*getProperty*("user.dir") + "\\testngold.xml");

tng.setTestSuites(suites);

tng.setSuiteThreadPoolSize(2); // this parameter is important in order to run suites in parallel

tng.run();

/\* TestNG tng = new TestNG();

tng.setTestSuites(Arrays.asList(new String[] {System.getProperty("user.dir") + "//testngold.xml"}));

tng.setSuiteThreadPoolSize(2);

tng.run();

\*/

}

}

A1.java

**package** packageA;

**import** org.testng.annotations.Test;

**import** pack1.ParentClass;

**public** **class** A1 **extends** ParentClass{

@Test

**public** **void** A1\_1() **throws** InterruptedException {

System.***out***.println("A1 start");

openBrowser("Firefox");

Thread.*sleep*(3000);

System.***out***.println("A1 ends");

}

}

A2.java

**package** packageA;

**import** org.testng.annotations.Test;

**import** pack1.ParentClass;

**public** **class** A2 **extends** ParentClass{

@Test

**public** **void** A2\_1() **throws** InterruptedException {

System.***out***.println("A2 start");

openBrowser("Chrome");

Thread.*sleep*(3000);

System.***out***.println("A2 ends");

}

}

A3.java

**package** packageA;

**import** org.testng.annotations.Test;

**import** pack1.ParentClass;

**public** **class** A3 **extends** ParentClass{

@Test

**public** **void** A3\_1() **throws** InterruptedException {

System.***out***.println("A3 start");

openBrowser("IE");

Thread.*sleep*(3000);

System.***out***.println("A3 ends");

}

}

B1.java

**package** packageB;

**import** org.testng.annotations.Test;

**import** pack1.ParentClass;

**public** **class** B1 **extends** ParentClass{

@Test

**public** **void** B1\_1() **throws** InterruptedException {

System.***out***.println("B1 start");

openBrowser("Firefox");

Thread.*sleep*(3000);

System.***out***.println("B1 ends");

}

}

B2.java

**package** packageB;

**import** org.testng.annotations.Test;

**import** pack1.ParentClass;

**public** **class** B2 **extends** ParentClass{

@Test

**public** **void** B2\_1() **throws** InterruptedException {

System.***out***.println("B2 start");

openBrowser("Chrome");

Thread.*sleep*(3000);

System.***out***.println("B2 ends");

}

}

B3.java

**package** packageB;

**import** org.testng.annotations.Test;

**import** pack1.ParentClass;

**public** **class** B3 **extends** ParentClass{

@Test

**public** **void** B3\_1() **throws** InterruptedException {

System.***out***.println("B3 start");

openBrowser("IE");

Thread.*sleep*(3000);

System.***out***.println("B3 ends");

}

}

testngA.xml

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

<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">

<suite name=*"SuiteA"* parallel=*"tests"*>

<test name=*"TestA1"*>

<classes>

<class name=*"packageA.A1"* />

</classes>

</test>

<test name=*"TestA2"*>

<classes>

<class name=*"packageA.A2"* />

</classes>

</test>

<test name=*"TestA3"*>

<classes>

<class name=*"packageA.A3"* />

</classes>

</test>

</suite> <!-- Suite -->

testngB.xml

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

<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">

<suite name=*"SuiteB"* parallel=*"tests"*>

<test name=*"TestB1"*>

<classes>

<class name=*"packageB.B1"* />

</classes>

</test>

<test name=*"TestB2"*>

<classes>

<class name=*"packageB.B2"* />

</classes>

</test>

<test name=*"TestB3"*>

<classes>

<class name=*"packageB.B3"* />

</classes>

</test>

</suite> <!-- Suite -->

Testngold.xml

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

<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">

<suite name=*"Suite"* >

<suite-files>

<suite-file path=*"./testngA.xml"*></suite-file>

<suite-file path=*"./testngB.xml"*></suite-file>

</suite-files>

</suite> <!-- Suite -->

hub.json , node1.json, node2.json should be saved in below directory. This directory should also have selenium-server-standalone-3.141.59 executable jar file

D:\selenium-server-standalone-3.141.59

hub.json

{

"host": **null**,

"port": 4444,

"newSessionWaitTimeout": -1,

"servlets" : [],

"prioritizer": **null**,

"capabilityMatcher": "jigar.mehta",

"throwOnCapabilityNotPresent": **true**,

"nodePolling": 5000,

"cleanUpCycle": 5000,

"timeout": 300000,

"browserTimeout": 0,

"maxSession": 3,

"jettyMaxThreads":-1

}

node1.json

{

"capabilities":

[

{

"browserName": "firefox",

"marionette": **true**,

"maxInstances": 5,

"seleniumProtocol": "WebDriver"

},

{

"browserName": "chrome",

"maxInstances": 5,

"seleniumProtocol": "WebDriver"

},

{

"browserName": "internet explorer",

"platform": "WINDOWS",

"maxInstances": 1,

"seleniumProtocol": "WebDriver"

},

{

"browserName": "safari",

"technologyPreview": **false**,

"platform": "MAC",

"maxInstances": 1,

"seleniumProtocol": "WebDriver"

}

],

"proxy": "org.openqa.grid.selenium.proxy.DefaultRemoteProxy",

"maxSession": 1,

"port": -1,

"register": **true**,

"registerCycle": 5000,

"hub": "http://localhost:4444",

"nodeStatusCheckTimeout": 5000,

"nodePolling": 5000,

"role": "node",

"unregisterIfStillDownAfter": 60000,

"downPollingLimit": 2,

"debug": **false**,

"servlets" : [],

"withoutServlets": [],

"custom": {}

}

node2.json

{

"capabilities":

[

{

"browserName": "firefox",

"marionette": **true**,

"maxInstances": 2,

"seleniumProtocol": "WebDriver"

},

{

"browserName": "chrome",

"maxInstances": 5,

"seleniumProtocol": "WebDriver"

},

{

"browserName": "internet explorer",

"platform": "WINDOWS",

"maxInstances": 3,

"seleniumProtocol": "WebDriver"

},

{

"browserName": "safari",

"technologyPreview": **false**,

"platform": "MAC",

"maxInstances": 4,

"seleniumProtocol": "WebDriver"

}

],

"proxy": "org.openqa.grid.selenium.proxy.DefaultRemoteProxy",

"maxSession": 1,

"port": -1,

"register": **true**,

"registerCycle": 5000,

"hub": "http://localhost:4444",

"nodeStatusCheckTimeout": 5000,

"nodePolling": 5000,

"role": "node",

"unregisterIfStillDownAfter": 60000,

"downPollingLimit": 2,

"debug": **false**,

"servlets" : [],

"withoutServlets": [],

"custom": {}

}

**Note:**

**Below link provides full list of DesiredCapabilities that can be used in node.json**

<https://github.com/SeleniumHQ/selenium/wiki/DesiredCapabilities>

**Run testngA.xml from Eclipse**

It opens Firefox and Chrome browser instance to different nodes and it closes.

Below is the output and it opens Firefox and Chrome browser and then both closes together

[RemoteTestNG] detected TestNG version 6.14.2

A1 start

A2 start

A3 start

Inside ParentClass and IE browser

Inside ParentClass and Chrome browser

Apr 25, 2019 2:03:53 AM org.openqa.selenium.remote.DesiredCapabilities chrome

INFO: Using `new ChromeOptions()` is preferred to `DesiredCapabilities.chrome()`

Apr 25, 2019 2:03:53 AM org.openqa.selenium.remote.DesiredCapabilities firefox

INFO: Using `new FirefoxOptions()` is preferred to `DesiredCapabilities.firefox()`

Inside ParentClass and firefox browser

Apr 25, 2019 2:04:51 AM org.openqa.selenium.remote.ProtocolHandshake createSession

INFO: Detected dialect: OSS

Apr 25, 2019 2:04:51 AM org.openqa.selenium.remote.ProtocolHandshake createSession

INFO: Detected dialect: W3C

A3 ends

A1 ends

Apr 25, 2019 2:05:18 AM org.openqa.selenium.remote.ProtocolHandshake createSession

INFO: Detected dialect: OSS

A2 ends

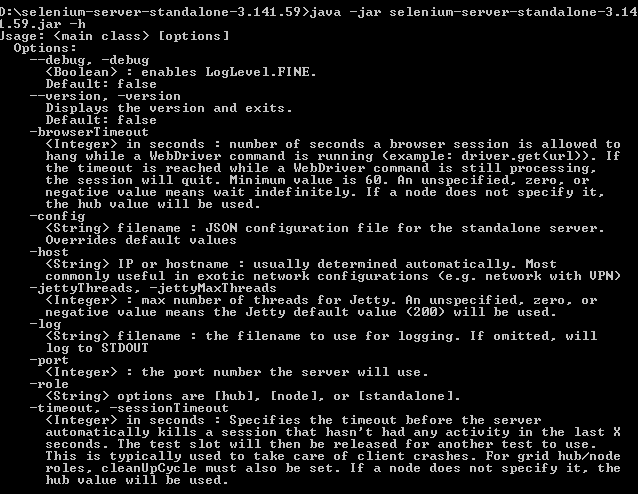
===============================================

SuiteA

Total tests run: 3, Failures: 0, Skips: 0

===============================================

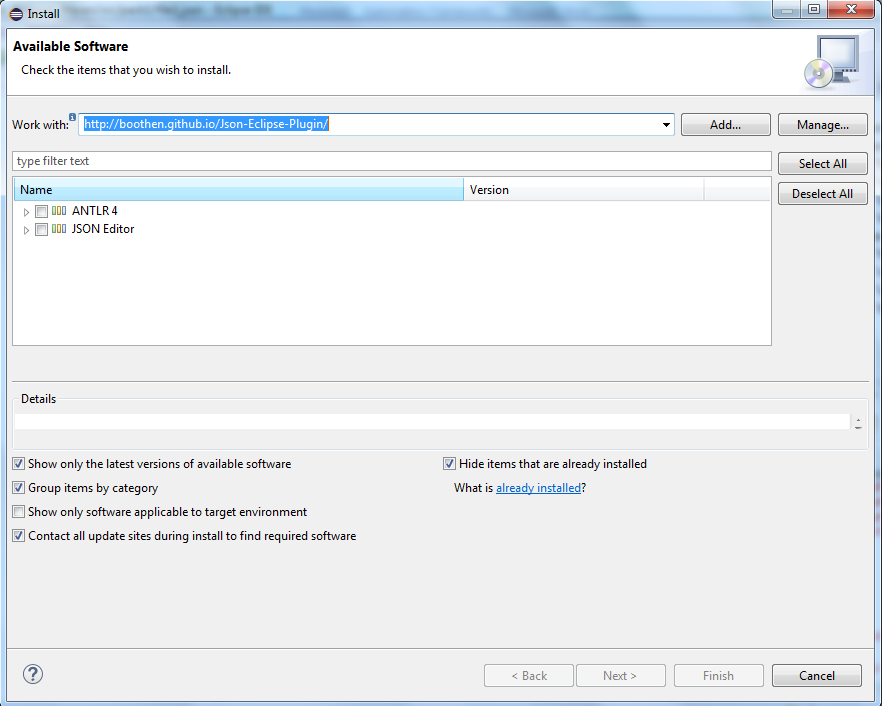
**Note to get help in CMD on Selenium Server see below**



To install json editor in Eclipse follow below steps:

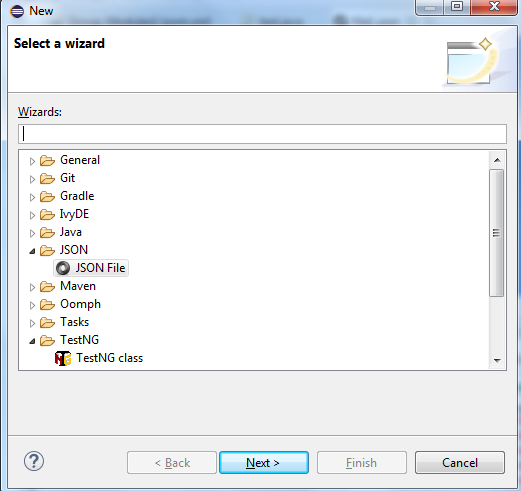
1. Goto Help->Install New Software and put below link in Work with field
2. Select ANTLR4 and JSON Editor and click Finish

<http://boothen.github.io/Json-Eclipse-Plugin/>



To create a new json file in Eclipse follow below steps

1. Navigate to File->New->Other->JSON->JSON File



Below is sample json file

{

"capabilities":

[

{

"browserName": "firefox",

"maxInstances": 5,

"seleniumProtocol": "WebDriver"

},

{

"browserName": "chrome",

"maxInstances": 3,

"seleniumProtocol": "WebDriver"

}

],

"configuration":

{

"proxy": "org.openqa.grid.selenium.proxy.DefaultRemoteProxy",

"maxSession": 2,

"port": 5523,

"host": ip,

"register": **true**,

"registerCycle": 5000,

"hubPort": 4444,

"hubHost": ip

}

}