Lab 6: Parallelizing the Tests for Selenium Grid

In this lab, we will use two techniques and describe **how to run your selenium parallel tests** by using Selenium Grid (SG) and JUnit.

Lab Solution

Lab solution is present in `C:\Users\fenago\Desktop\advanced-selenium-java\Lab06` folder.

Selenium Grid Setup for Parallel Test Execution

Our setup will be like that; we will have two nodes and one hub. Each node has got 5 Chrome, 5 Firefox and 1 Internet Explorer browser instances. The first node will use port 5555 and the second one will use 5556. Thus, we will create two node JSON files. These are **node1.json** and **node2.json.** The only difference between these JSON files is port number.

```
{
  "capabilities":
    {
      "browserName": "firefox",
      "maxInstances": 5,
      "seleniumProtocol": "WebDriver"
    },
    {
      "browserName": "chrome",
      "maxInstances": 5,
      "seleniumProtocol": "WebDriver"
    },
    {
      "browserName": "internet explorer",
      "maxInstances": 1,
      "seleniumProtocol": "WebDriver"
    }
  ],
  "proxy": "org.openqa.grid.selenium.proxy.DefaultRemoteProxy",
  "maxSession": 5,
  "port": 5555,
  "register": true,
  "registerCycle": 5000,
  "hub": "http://localhost:4444",
  "nodeStatusCheckTimeout": 5000,
  "nodePolling": 5000,
  "role": "node",
  "unregisterIfStillDownAfter": 60000,
  "downPollingLimit": 2,
  "debug": false,
  "servlets" : [],
  "withoutServlets": [],
  "custom": {}
}
```

```
{
  "capabilities":
  [
    {
      "browserName": "firefox",
      "maxInstances": 5,
      "seleniumProtocol": "WebDriver"
    },
    {
      "browserName": "chrome",
      "maxInstances": 5,
      "seleniumProtocol": "WebDriver"
    },
    {
      "browserName": "internet explorer",
      "maxInstances": 1,
      "seleniumProtocol": "WebDriver"
    }
  ],
  "proxy": "org.openqa.grid.selenium.proxy.DefaultRemoteProxy",
  "maxSession": 5,
  "port": 5556,
  "register": true,
  "registerCycle": 5000,
  "hub": "http://localhost:4444",
  "nodeStatusCheckTimeout": 5000,
  "nodePolling": 5000,
  "role": "node",
  "unregisterIfStillDownAfter": 60000,
  "downPollingLimit": 2,
  "debug": false,
  "servlets" : [],
  "withoutServlets": [],
  "custom": {}
}
```

We created two node JSON files. Our hub.JSON file remains same as shown below.

```
{
  "port": 4444,
  "newSessionWaitTimeout": -1,
  "servlets" : [],
  "withoutServlets": [],
  "custom": {},
  "capabilityMatcher": "org.openqa.grid.internal.utils.DefaultCapabilityMatcher",
  "throwOnCapabilityNotPresent": true,
  "cleanUpCycle": 5000,
  "role": "hub",
  "debug": false,
  "browserTimeout": 0,
  "timeout": 1800
}
```

In order to start Selenium Grid (hub and nodes), we should write .bat files. These are; **startnode1.bat**, **startnode2.bat**, **starthub.bat** and to trigger all of these .bat files with a single .bat file I will create a **rungrid.bat** file. All these .bat files are shown below.

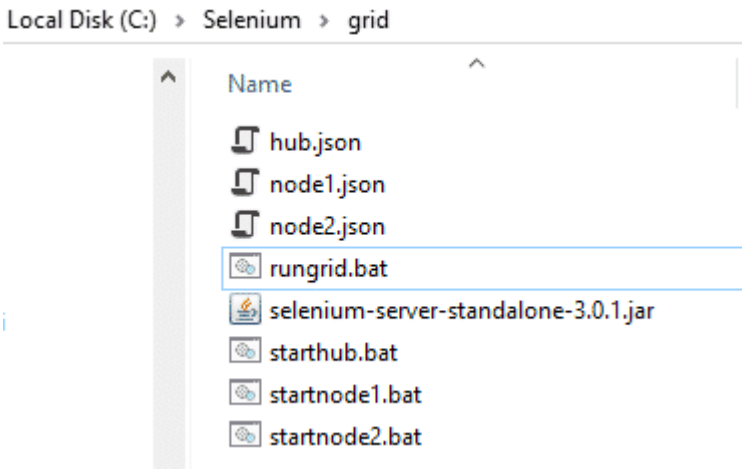
```
java -jar selenium-server-standalone-3.0.1.jar -role hub -hubConfig hub.json
```

```
java -jar -Dwebdriver.gecko.driver=C:\Selenium\drivers\firefox\geckodriver.exe -
Dwebdriver.chrome.driver=C:\Selenium\drivers\chrome\chromedriver.exe selenium-server-
standalone-3.0.1.jar -role node -nodeConfig node1.json
```

```
java -jar -Dwebdriver.gecko.driver=C:\Selenium\drivers\firefox\geckodriver.exe -
Dwebdriver.chrome.driver=C:\Selenium\drivers\chrome\chromedriver.exe selenium-server-
standalone-3.0.1.jar -role node -nodeConfig node2.json
```

```
start starthub.bat
start startnode1.bat
start startnode2.bat
```

After these settings our **C:\Selenium\Grid** folder will look like below.



When we run "**rungrid.bat**" file it starts hub a nd nodes consecutively. After that, when you go to **http://localhost:4444/grid/console** you will see that two nodes registered to one hub as shown below.

Now, we are ready to code parallel test execution with our grid setup. I will show you two techniques to run your selenium tests with JUnit.

JUnit Parallel Test Execution Techniques

1) Run Selenium Tests in Parallel using JUnit's Parallel Computer Class

```
    @Test
    public void runAllTests() {
        Class<?>[] classes = {ParallelTest1.class,ParallelTest2.class};

        // ParallelComputer(true,true) will run all classes and methods
        // in parallel.  (First arg for classes, second arg for methods)
```

```
        // I set true, true this means classes and methods runs in parallel.
        JUnitCore.runClasses(new ParallelComputer(true, true), classes);
    }
```

In above method first parameter of ParallelComputer() is for classes and the second one is for methods. Here, I will run classes and methods in parallel. Because I set both parameters as true.

Now, Let's prepare a test scenario and then write its code.

Test Scenario:

- In first test Class;
  - Open Facebook with Chrome in first test method
  - Open Amazon with Firefox in second test method
- In second test Class;
  - Open Yahoo with Chrome.

We have two test classes. In first test class, we have two test methods and in the second one, we have one test method.

Test Code:

I used 4 classes for this test. First one is **DriverManager**, it sets which browser driver will be used for the test. **GridParallelComputerTest**class modifies**ParallelComputer** class and runs the tests in parallel, the other two classes are test classes, **ParallelTest1** and **ParallelTest2**.

```
import org.openqa.selenium.Platform;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.remote.RemoteWebDriver;

import java.net.MalformedURLException;
import java.net.URL;

/**
 * Created by fenago
 */
//Driver Manager Class
public class DriverManager {

    public WebDriver driver;

    public WebDriver getDriver(String browser) throws MalformedURLException {
        //Set Browser Type
        DesiredCapabilities caps = null;
        if (browser == "chrome") {
            caps = DesiredCapabilities.chrome();
        } else if (browser == "firefox") {
            caps = DesiredCapabilities.firefox();
        }
        caps.setPlatform(Platform.WINDOWS);

        return driver = new RemoteWebDriver(new URL("http://localhost:4444/wd/hub"),
caps);
```

```
    }
}
```

```
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import org.junit.experimental.ParallelComputer;
import org.junit.runner.JUnitCore;
import org.openqa.selenium.By;
import org.openqa.selenium.Platform;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.remote.RemoteWebDriver;

import java.net.MalformedURLException;
import java.net.URL;

/**
 * Created by fenago
 */
public class GridParallelComputerTest {

    /* ~~~~~~Description~~~~~~
    Run All Test in Parallel with JUnit's ParallelComputer feature.
    By using below logic you can run your junit cases in parallel.

    Class[] cls={test1.class,test2.class,test3.class,test4.class};
    JUnitCore.runClasses(new ParallelComputer(true,true),cls);

    In above method first parameter of ParallelComputer() indicates classes and second
one is for methods.
    Here I'm running classes and methods in parallel.

    ParallelComputer Class documentation is below:
    http://junit-
team.github.io/junit/javadoc/4.10/org/junit/experimental/ParallelComputer.html
    */

    @Test
    public void runAllTests() {
        Class<?>[] classes = {ParallelTest1.class,ParallelTest2.class};

        // ParallelComputer(true,true) will run all classes and methods
        // in parallel.  (First arg for classes, second arg for methods)
        // I set true, true this means classes and methods runs in parallel.
        JUnitCore.runClasses(new ParallelComputer(true, true), classes);
    }
}
```

2) Run Selenium Tests in Parallel using JUnit’s Parametrized Class

Parallellized class is a helper class and you can define threat count in this class's **ThreadPoolScheduler** method. In **GridParallelTestBase** class, I set which browsers I will use for the test by using **@Parameterized.Parameters** annotation. Here, I added Chrome and Firefox browsers. Thus, our test will open two browsers and they will be Chrome and Firefox.

I created **DesiredCapabilities** and RemoteWebdriver below @Before annotation. All test setup is done in this Class's **setup method** and I also added a **screenshot capture method** in this class.

GridParallelTestclass is our test class and it extends **GridParallelTestBase** class. In its test method, I set the platform, go to yahoo.com, print the yahoo's title, and take a screenshot.

```
import org.junit.runners.Parameterized;
import org.junit.runners.model.RunnerScheduler;

import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.TimeUnit;

public class Parallelized extends Parameterized {

    private static class ThreadPoolScheduler implements RunnerScheduler {
        private ExecutorService executor;

        //You can set number of parallel threads in this method.
        //I set 5 and our grid will run 5 parallel test execution.
        public ThreadPoolScheduler() {
            String threads = System.getProperty("junit.parallel.threads", "5");
            int numThreads = Integer.parseInt(threads);
            executor = Executors.newFixedThreadPool(numThreads);
        }

        //@Override
        public void finished() {
            executor.shutdown();
            try {
                executor.awaitTermination(10, TimeUnit.MINUTES);
            } catch (InterruptedException exc) {
                throw new RuntimeException(exc);
            }
        }

        //@Override
        public void schedule(Runnable childStatement) {
            executor.submit(childStatement);
        }
    }

    public Parallelized(Class<?> klass) throws Throwable {
        super(klass);
        setScheduler(new ThreadPoolScheduler());
    }
}
```

```
import org.apache.commons.io.FileUtils;
import org.junit.Before;
import org.junit.runners.Parameterized;
import org.openqa.selenium.*;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.firefox.FirefoxProfile;
import org.openqa.selenium.remote.Augmenter;
import org.openqa.selenium.remote.CapabilityType;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.remote.RemoteWebDriver;

import java.io.File;
import java.io.IOException;
import java.net.URL;
import java.util.LinkedList;

/**
 * Created by fenago
 */
public class GridParallelTestBase {
    //Declare DesiredCapabilities configuration variables
    protected String browserName;
    protected Platform platformName;
    protected WebDriver driver;

    //Hold all Configuration values in a LinkedList
    //Extra Usage Information: http://www.swtestacademy.com/junit-parametrized-tests/
    @Parameterized.Parameters
    public static LinkedList<String[]> getEnvironments() throws Exception {
        LinkedList<String[]> env = new LinkedList<String[]>();
        env.add(new String[]{"firefox"});
        env.add(new String[]{"chrome"});
        //add more browsers here
        return env;
    }

    //Constructor
    public GridParallelTestBase(String browserName) {
        this.browserName = browserName;
    }

    public void setPlatform (Platform platform) {
        platformName = platform;
    }

    @Before
    public void setUp() throws Exception {
        //Set DesiredCapabilities
        DesiredCapabilities capabilities = new DesiredCapabilities();
        //Firefox Profile Settings
        if (browserName.equals("firefox")) {
            FirefoxProfile profile = new FirefoxProfile();
```

```
            //Accept Untrusted Certificates
            profile.setAcceptUntrustedCertificates(true);
            profile.setAssumeUntrustedCertificateIssuer(false);
            //Use No Proxy Settings
            profile.setPreference("network.proxy.type", 0);
            //Set Firefox profile to capabilities
            capabilities.setCapability(FirefoxDriver.PROFILE, profile);
        }
        //Set Platform
        capabilities.setPlatform(platformName);
        //Set BrowserName
        capabilities.setCapability("browserName", browserName);
        capabilities.setCapability("build", "JUnit-Parallel");
        driver = new RemoteWebDriver(new URL("http://localhost:4444/wd/hub"),
capabilities);
    }

    //TakeScreenShot
    public void takeScreenShot () {
        driver = new Augmenter().augment(driver);
        File srcFile = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);
        String screenshotName = getClass().getSimpleName();
        System.out.println("ScreenShotName: " + screenshotName);
        try {
            FileUtils.copyFile(srcFile, new File("screenshotName.png"));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

}
```

```
import org.junit.After;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.openqa.selenium.Platform;

@RunWith(Parallelized.class)
public class GridParallelTest extends GridParallelTestBase{

    //Constructor
    public GridParallelTest(String browserName) {
        super(browserName);
    }

    @Test
    public void parallelGridTest() throws Exception {
        //Set Platform Name
        setPlatform(Platform.WIN10);

        //Go to Amazon.com
        System.out.println("Test is started for: "+ browserName);
```

```
        driver.get("http://www.yahoo.com");
        System.out.println("Page title is: " + driver.getTitle());
        System.out.println("Test is finished for: "+ browserName);

        //ScreenShot Section
        takeScreenShot();
    }

    @After
    public void tearDown() throws Exception {
        driver.quit();
    }
}
```

After run this test you will see that two browsers (Chrome & Firefox) will open in parallel and tests will be passed.