Selenium Grid

- Selenium Grid is a concept/set of configurations, using which we can distribute the multiple tests accross multiple machines,having multiple operating systems and multiple browsers for execution.
- Purpose:
  - For Saving Execution Time
  - When there are huge number of tests taking long time for execution, Selenium Grid can distribute these tests across multiple machines and execute them simultaneously for reducing the execution time.
  - Example: Lets say, we have 10 tests where each test takes 1 minute each. If you are not using Selenium Grid, then one test will be executed after another in the same machine and hence takes 10 minutes of time for executing 10 tests. But if we use Selenium Grid to distribute these tests across 5 machines, then 5 tests will be executed at the same time and hence it takes only 2 minutes for executing all the 10 tests.
- Hub and Node Machines
  - In Hub, we will place the centralized code
  - From Hub, the tests will be distributed accross multiple node machines
  - Node machines can belong to different operating systems like Windows, Linux or Mac
  - Also we can distribute the tests across multiple browsers in the same node machine too
- Role of TestNG in Parallelism
  - Create a Maven Project
  - Create a package say testcases and under it and create Test classes say TestA, TestB and TestC
  - Create test methods inside all the Test classes - TestA (View here), TestB (View here) and TestC (View here)
  - Create a new TestNG suite contain all the above Test classes - View here
  - Execute the testng.xml using TestNG Suite and observe that the tests are executed one after the other
  - Mention parallel="tests" in the suite tag of testng.xml and execute it, all the tests will be run in parallel
- TestNG Versus Grid
  - The purpose of TestNG is to execute the tests parallely
  - Where as Grid will distribute accross multiple machines
  - Using TestNG + Grid, we can distribute the tests across multiple machines run them parallely
- Running TestNG Suites in parallel
  - Create a Maven Project
  - Create suiteOne.xml and suiteTwo.xml, where suiteOne.xml is referring to TestA, TestB and TestC and suiteTwo.xml is referring to TestD, TestE and TestF - view suiteOne.xml here and view suiteTwo.xml file here
  - Create a master suite say testng.xml referring to suiteOne.xml and suiteTwo.xml, and execute it to see that the prioritization is not happending at suite level - view testng.xml here
  - Execute the master suite and observe that the prioritization is not happening at master suite level, instead it is happening at individual suite level
  - Inorder to run the TestNG suites in parallel, create a package say 'parallel' and create a class under is say RunParallelSuites with this code and execute - View code here

---