Cucumber Setup: Maven, Extent Reports, Cucumber and Eclipse Plugin

This section will guide you to:

* Set up Cucumber eclipse plugin
* Set up Maven project
* Set up Extent report

**Development Environment:**

* JRE: OpenJDK Runtime Environment 11.0.2
* Eclipse IDE for Enterprise Java Developers v2019-03 (4.11.0)
* TestNG
* Selenium jars
* Cucumber jars

This guide has four subsections, namely:

1.1 Configuring Cucumber Eclipse

1.2 Configuring Cucumber with Maven project

1.3 Configuring Cucumber extent report

**Step 1.1:** Configuring Cucumber Eclipse

Cucumber jar files are already present in your practice lab. Refer to the lab guide for Phase 2 for more information.

* Launch Eclipse.
* Create Java Project.
* Go to Java Project and create a folder “jars.”
* Add all the downloaded jars to the “Jars” folder.
* Select the Created Project → Right Click → Build path → Configure Build path →Click on Libraries → Add JARs →Select all the jars from “Jars” folder → Click on Apply → Add Library → Select TestNG → Click on Apply → Click on Finish.

**Step 1.2:** Configuring Cucumber with MavenProject

* Create a Maven Project.
* Go to File → New → Others → Maven → Maven Project → Next.
* Provide group ID (group ID will identify your project uniquely across all projects).
* Provide artifact ID (artifact ID is the name of the jar without version. You can choose any name, which is in lowercase). Click on Finish.
* Open pom.xml.
* Go to the package explorer on the left-hand side of Eclipse.
* Expand the created project.
* Locate the pom.xml file.
* Right-click and select the option, open with “Text Editor.”
* Adding dependencies to the project: This will indicate Maven about the jar files to be downloaded from the central repository to the local repository.
* Open the pom.xml in the edit mode, create dependencies tag (<dependencies></dependencies>), inside the project tag.
* Inside the dependencies tag, create dependency tag (<dependency></dependency>).
* Copy the dependency tag for the following from Maven Repository.
* Selenium Webdriver
* Cucumber-Core jar
* Cucumber-Java jar
* Cucumber-TestNG jar
* Provide the information of all copied dependencies within the dependency tag.
* Verify binaries.
* Once the pom.xml is edited successfully, save it.
* Go to Project → Clean. After a few minutes, you will be able to see a Maven repository.

**Step 1.3:** ConfiguringCucumber Extent Report

* Add below Cucumber Extent Report library to Maven project.

<dependency>

<groupId>com.aventstack</groupId>

<artifactId>extentreports</artifactId>

<version>3.0.6</version>

</dependency>

<dependency>

<groupId>com.vimalselvam</groupId>

<artifactId>cucumber-extentsreport</artifactId>

<version>3.0.2</version>

</dependency>

* Add Extent Config to the project.

Extent Config is required by the Cucumber Extent Report plugin to read the report configuration. This gives the capability to set many useful settings to the report from the *XML* configuration file.

* Create a ***New File*** and name it as ***extent-config.xml*** by right clicking on the ***configs*** folder in the project.
* Read the extent-config.xml path.
* Make an entry for the path of the config in the ***Configuration.properties*** file.

*reportConfigPath=C:/ToolsQA/CucumberFramework/configs/extent-config.xml*

* Write a method ***getReportConfigPath()*** in the ***ConfigFileReader***class to return the extent report config file path.

public String getReportConfigPath(){

String reportConfigPath = properties.getProperty("reportConfigPath");

if(reportConfigPath!= null) return reportConfigPath;

else throw new RuntimeException("Report Config Path not specified in the Configuration.properties file for the Key:reportConfigPath");

}

* Modify TestRunner to Implement Cucumber Extent Reporter
* Modify the runner class and add *com.cucumber.listener.ExtentCucumberFormatter:output/report.html* as a plugin followed by the report file as input. This should be done within the *@CucumberOptions* annotation.

***@CucumberOptions( plugin = {“com.cucumber.listener.ExtentCucumberFormatter:target/cucumber- reports/report.html”})***

The above setup will generate the report in the output directory with the name report.html.

* Write extent reports

Add a method *writeExtentReport()* in the *TestRunner* class to write the report.

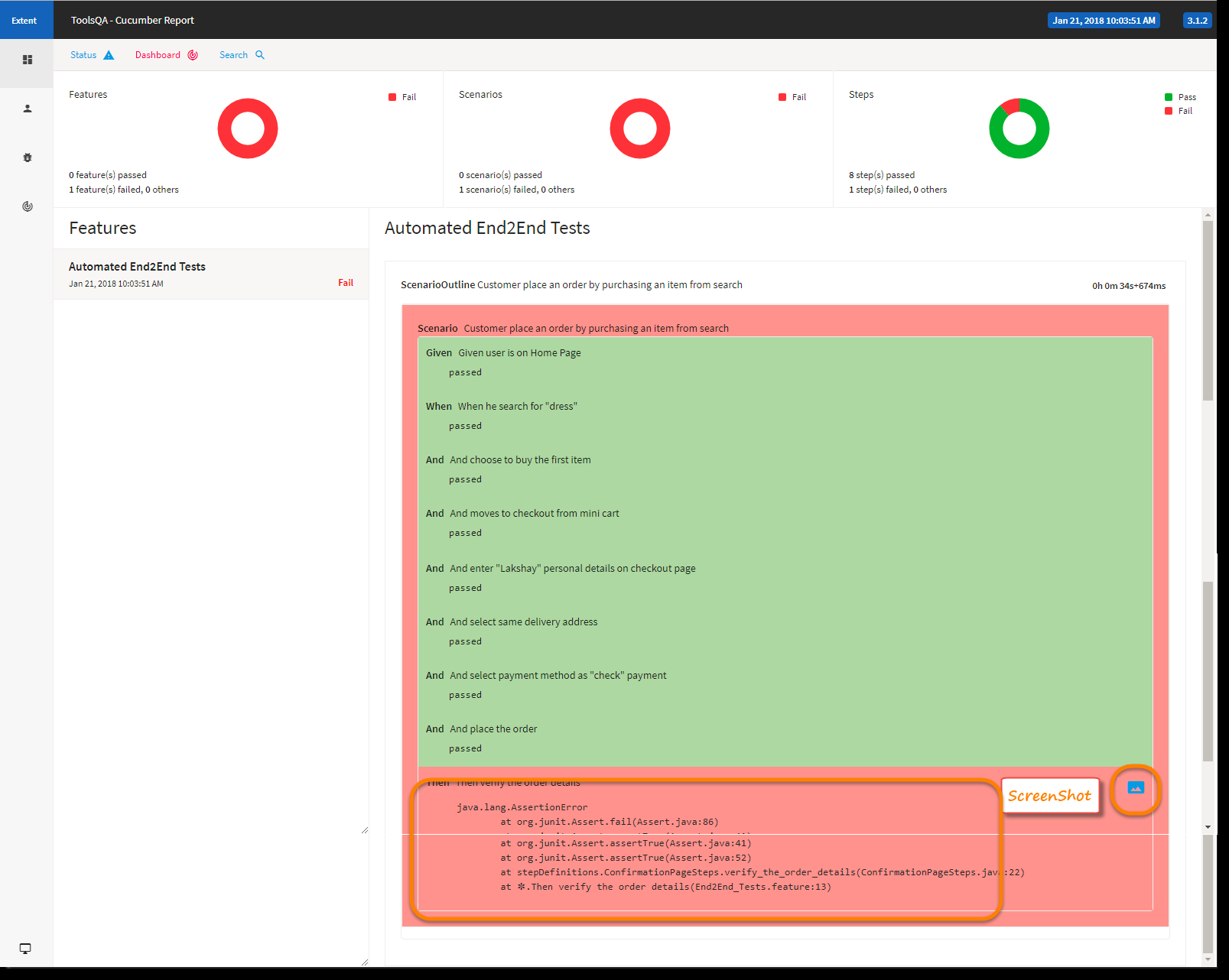
@AfterClass

public static void writeExtentReport() {

Reporter.loadXMLConfig(new File(FileReaderManager.getInstance().getConfigReader().getReportConfigPath()));

}

* Sample extent report



TestNG:

This section will guide you to:

* Implement @Test and other related annotations

**Development Environment**

* Eclipse IDE for Enterprise Java Developers v2019-03 (4.11.0)
* JRE: OpenJDK Runtime Environment 11.0.2
* TestNG

This guide has six sub-sections, namely:

2.1 Creating a simple Java project

* 1. Installing TestNG
  2. Adding TestNG libraries to the Class-Path
  3. Creating a class file named TestAnnotations
  4. Running the project as TestNG

**Step 2.1:** Creating a simple Java project

* Open Eclipse.
* Go to the **File** menu and select **New->Java Project.**
* Enter the project name as **Annotations**. Click on **Next.**
* This will create the project files in the Project Explorer.

**Step 2.2:** Installing TestNG

* TestNG is already installed in your Practice labs. To learn about its directory details you can refer to the lab guide for Phase 2.

**Step 2.3:** Adding TestNG libraries to the Class Path

* In the **Project Explorer**, right-click on **Annotations.**
* Select **Properties**. Select **Java Build Path** from the list. Go to **Libraries.**
* Click on **Add Library.** Select **TestNG**. Click on **Next**. Now, click on **Finish.**
* Finally, click on **Apply and Close.**

**Step 2.4:** Creating a class file named TestAnnotations

* In the Project Explorer, expand **Annotations->Java Resources.**
* Right-click on **src** and choose **New->Class.**
* In **Class Name,** enter **TestAnnotations**. In **Package Name,** enter **com.testannotations** and click on **Finish.**
* Enter the following code:

**package** com.testannotations;

**import** org.testng.annotations.\*;

**public** **class** TestAnnotations {

@Test

**public** **void** Test1() {

System.***out***.println("Test1 Executed");

}

@Test

**public** **void** Test2() {

System.***out***.println("Test2 Executed");

}

@BeforeTest

**public** **void** beforeTest() {

System.***out***.println("BeforeTest Executed");

}

@AfterTest

**public** **void** AfterTest() {

System.***out***.println("AfterTest Executed");

}

@BeforeMethod

**public** **void** beforeMethod() {

System.***out***.println("BeforeMethod Executed");

}

@AfterMethod

**public** **void** afterMethod() {

System.***out***.println("AfterMethod Executed");

}

@BeforeClass

**public** **void** beforeClass() {

System.***out***.println("BeforeClass Executed");

}

@AfterClass

**public** **void** afterClass() {

System.***out***.println("AfterClass Executed");

}

}

**Step 2.5:** Running the project as TestNG

* Right-click on **TestAnnotations** class. Click on **TestNG->Convert to TestNG.**
* Click on **Finish.** It will create a **TestNG.xml** file. Open that file.
* Right click on the screen. Select **Run As ->TestNG Suite.**

Recording JMeter Script

This section will guide you to:

* Record a script in JMeter

**Development Environment:**

* Apache JMeter 5.3 Version
* Open JDK Runtime Environment 11.0.2

This guide has five subsections, namely:

3.1 Configuring HTTP Script Recorder to your JMeter

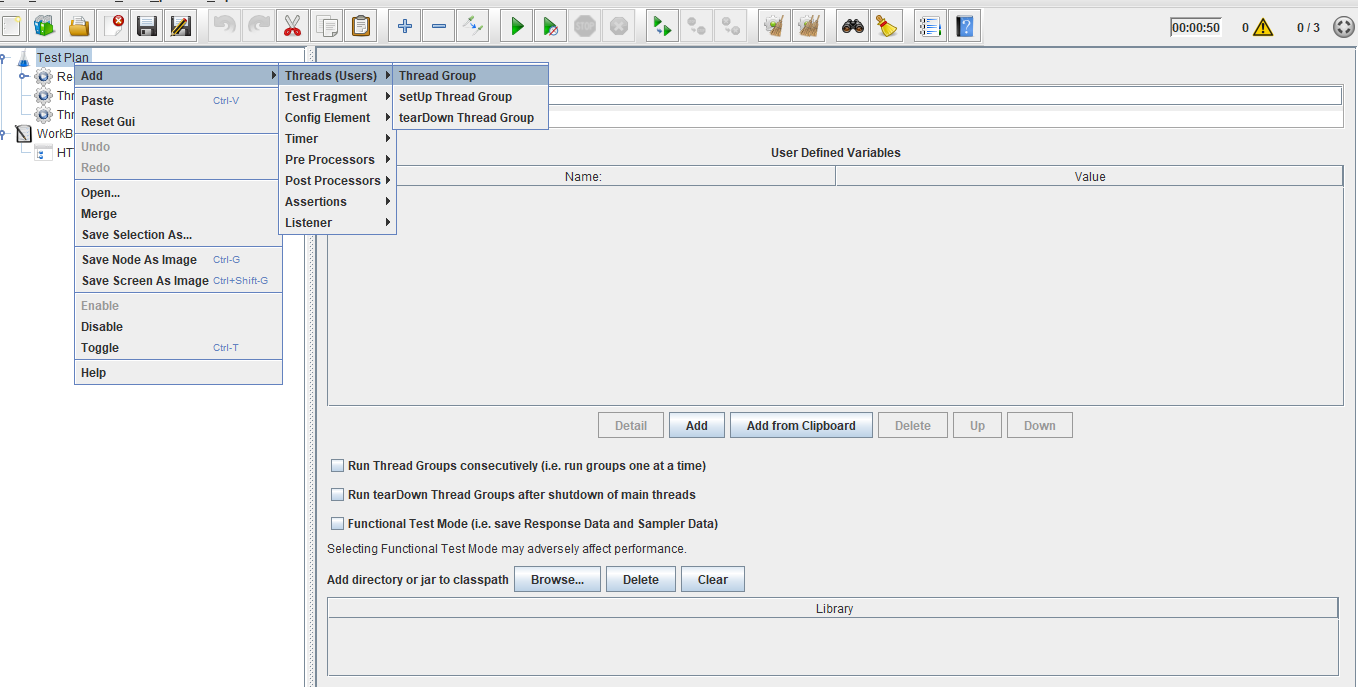
3.2 Doing proxy settings in JMeter

3.3 Configuring JMeter in browsers (using Firefox browser here)

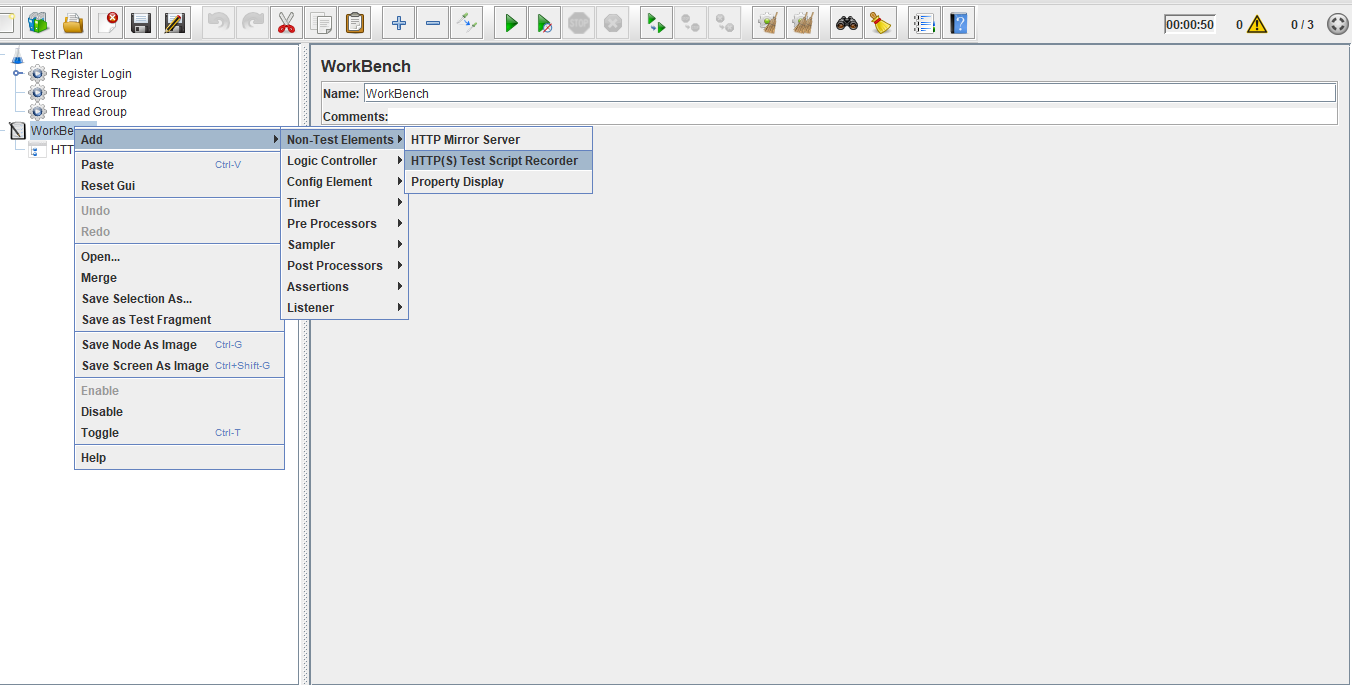
3.4 Recording the script

**Step 3.1:** Configuring HTTP Script Recorder to your JMeter

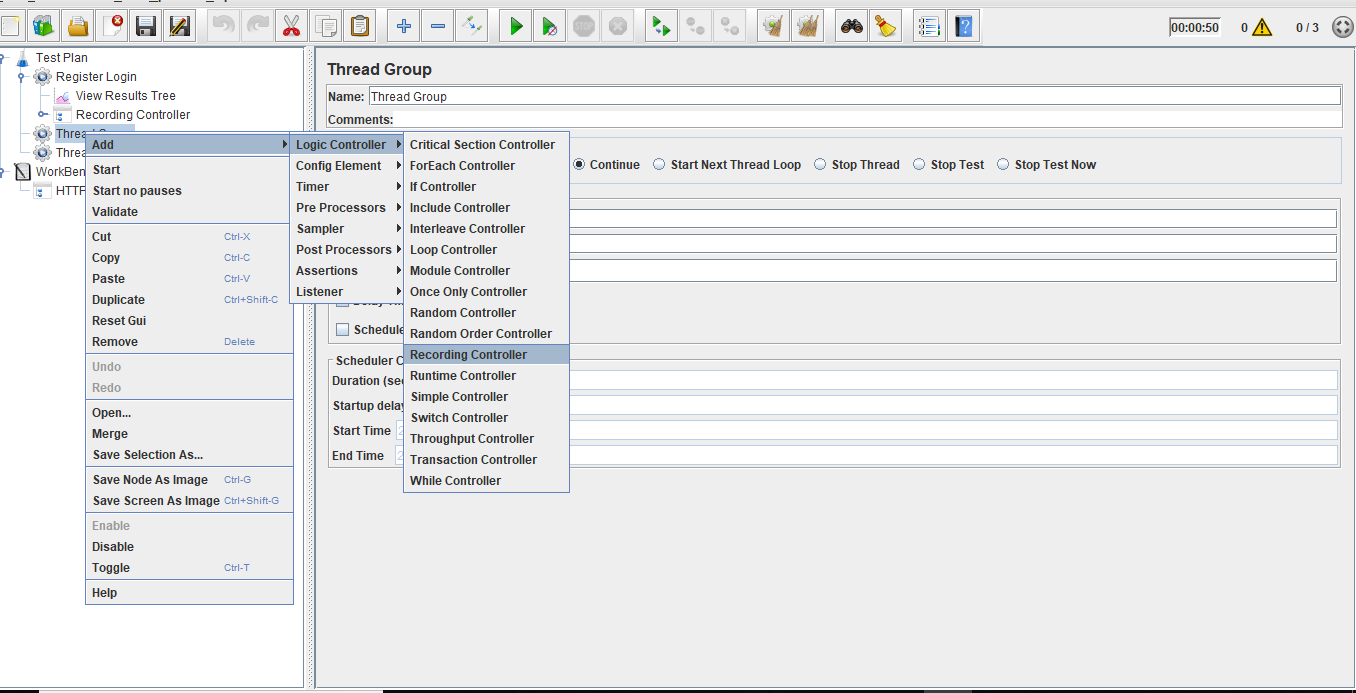
* Open Jmeter by clicking on the JMeter icon on the desktop in your practice lab.
* Right click on Test Plan.
* Click on Thread(Users) -> Thread Group.



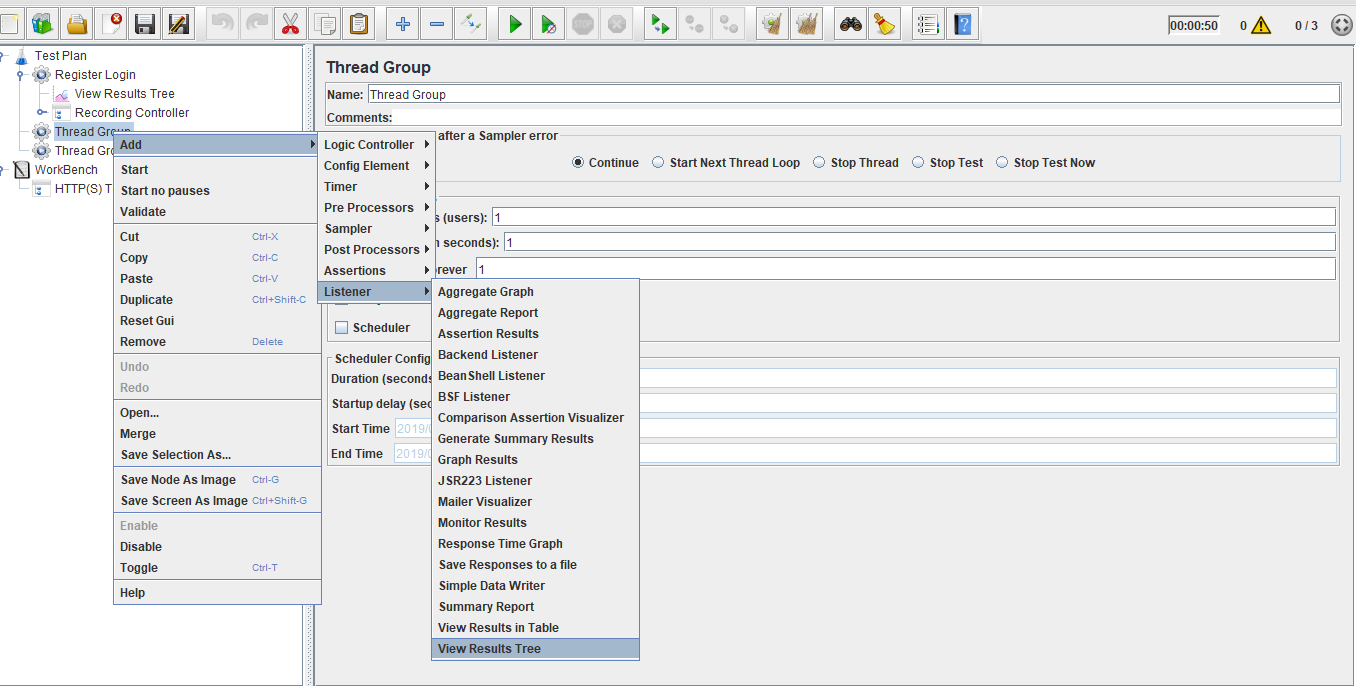
* Right click on Workbench.
* Click on Add -> NonTest Elements -> Http(s) Test Script Recorder.



* Right click on Thread Group.
* Click on Add -> Logic Controller -> Recording Controller.

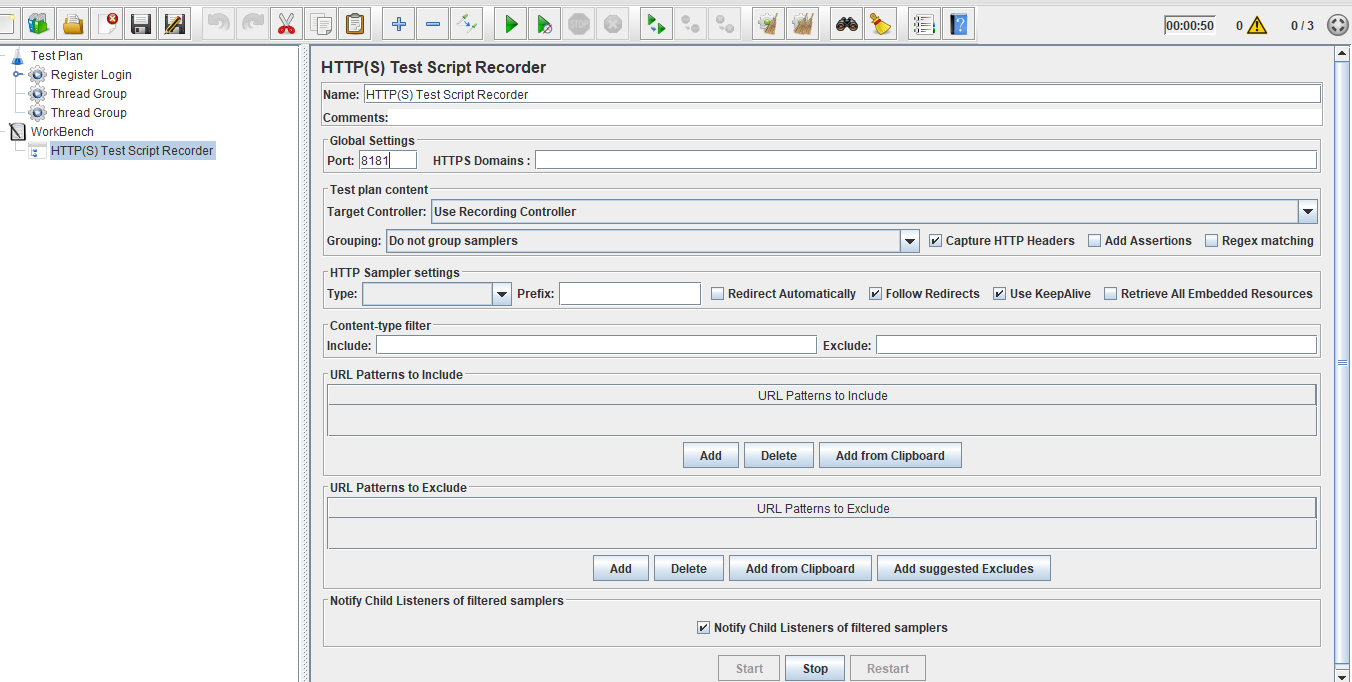


* Right click on Thread Group.
* Click on Add -> Listener -> View Result in Tree.



**Step 3.2:**  Doing proxy settings in JMeter

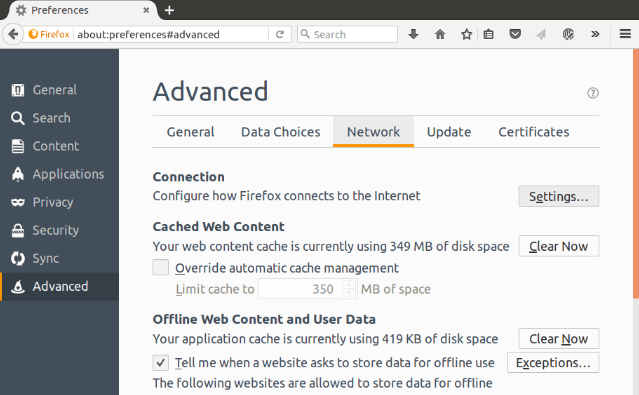
* Click on HTTP(S) Script Recorder.
* Set port number (optional).
* Click on the Start Button.



**Note:** TheDefault Certificate will be generated in your local machine.

**Step 3.3:** Configuring JMeter in browsers (using Firefox browser here)

* Start Firefox, but do not close JMeter.
* From the toolbar, click *Edit* → *Preferences* (or *Tools* → *Preferences* or type **about:preferences#advanced** as the URL). This should bring up the options.



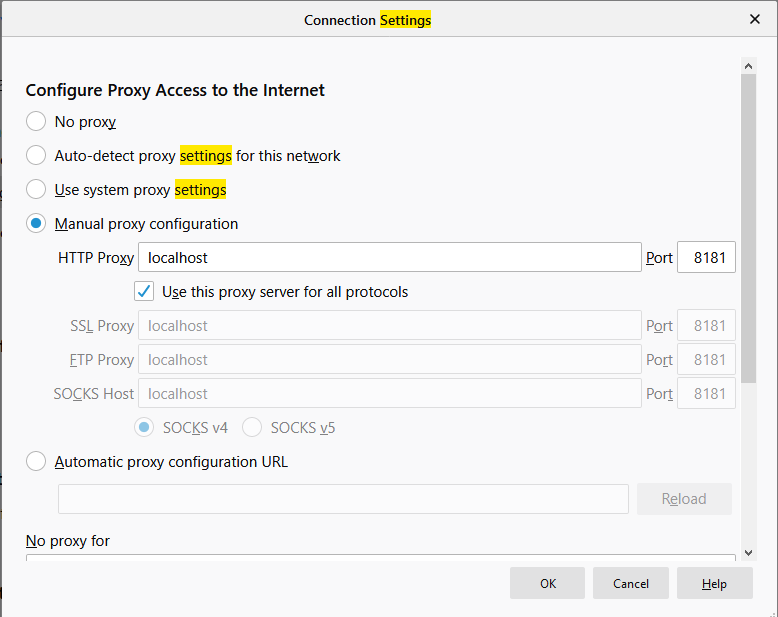
* Select the Advanced tab and Network tab.
* Click Settings Button near the top.
* On the new POP-UP, check Manual Proxy Configuration. The address and port fields should be enabled now.

Address:

Enter localhost or the IP address of your system.

Port:

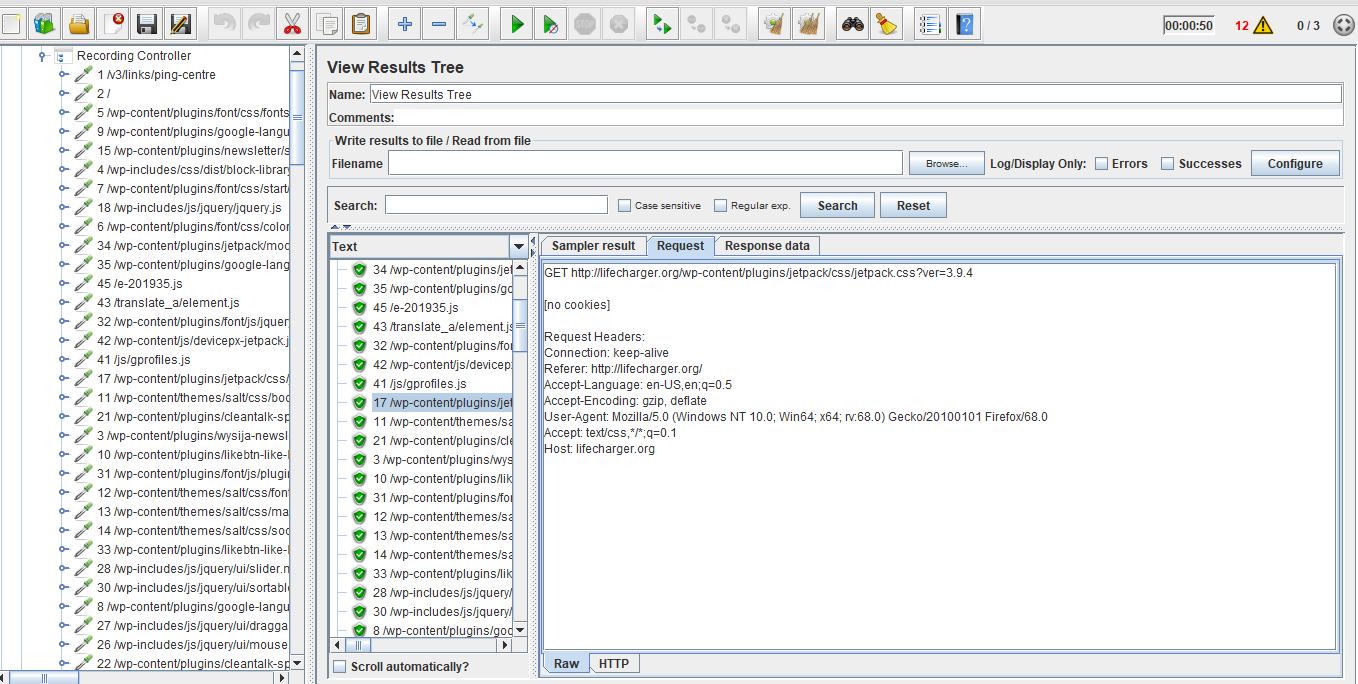
Enter 8181(Samp port in HTTP(S) Script Recorder)



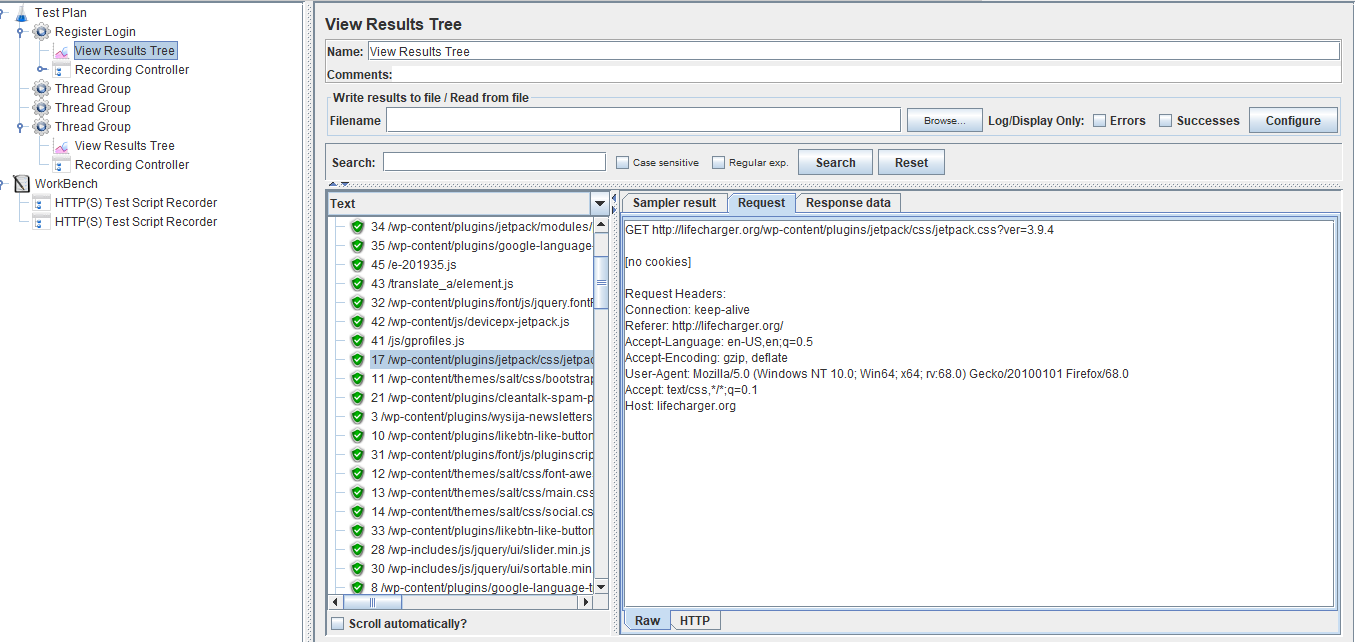
* Check the **Use this proxy server for all protocols** checkbox.
* Click the **Ok** button and that should return you to the browser.

**Step 3.4:** Recording the script

* Open the website you want to check in a browser, example: Firefox.
* Recording Controller records the script.



* View the results in **View Result Tree**.



Postman Setup

This section will guide you to understand:

* Postman
* Importance of Postman
* Installation of Postman

This guide has two subsections, namely:

4.1 Installing Postman

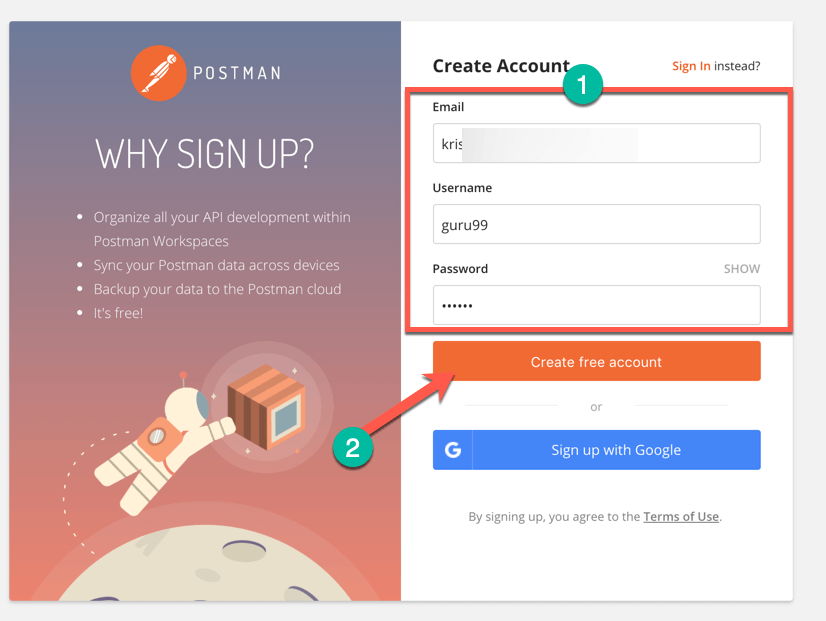
* 1. Setting up Postman

**Step 4.1: Installing Postman**

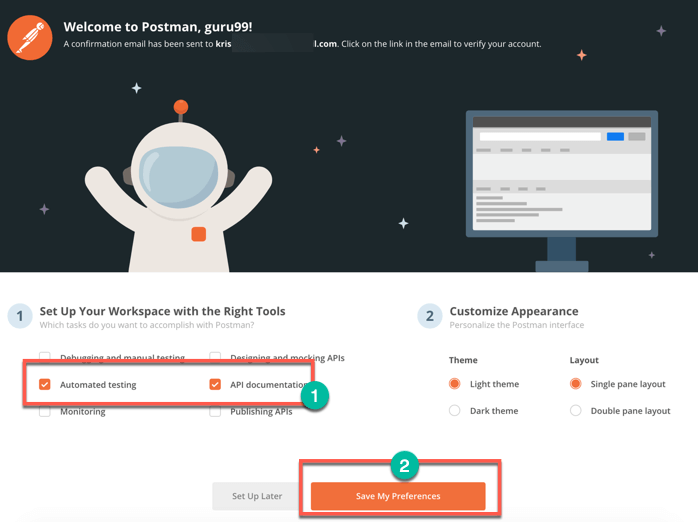
* Postman is already installed in your practice lab. Refer QA to QE lab guide -- Phase 3 for more information.

**Step 4.2: Setting up Postman**

* Open Postman and click on the sign up button to create a Postman Account.



* NOTE: There are two ways to sign up for a Postman account. One is to create your own Postman account, and the other is to use a Google account. Though Postman allows users to use the tool without logging in, signing up ensures that your collection is saved and can be accessed for later use.
* Select the required workspace tools.
* Click on Save My Preferences.



* You will see this Startup Screen below:

