**Date:17/10/2020**

**Practical no 7**

**AIM:** Write and test a program to get the number of items in a DropDownlist / combo box.

**Theory**

**WebElement** : Represents an HTML element. Generally, all interesting operations to do with interacting with a page will be performed through this interface. WebElements objects are Selenium(Java) equivalent of HTML elements in the UI such as Form, Button, Select, A, P, H0-6, etc..

**org.openqa.selenium.support.ui.Select** : Models a SELECT tag, providing helper methods to select and deselect options.

**java.util.List** : This data structure allows one two list down WebElements in Java which could be sub- elements or nested elements under the selector. The object has methods such as **object.size()** which returns the size of the list, **Selectobject.getOptions()** which returns a select option element from the object.

**Select.selectByVisibleText()** : Select all options that display text matching the argument. That is, when given ”Bar” this would select an option like: Bar .

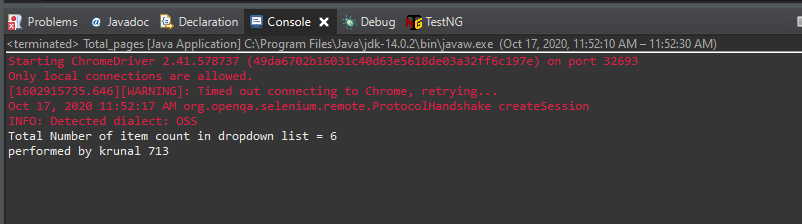
**Select.selectByIndex():** Select the option at the given index. This is done by examining the ”index” attribute of an element, and not merely by counting.

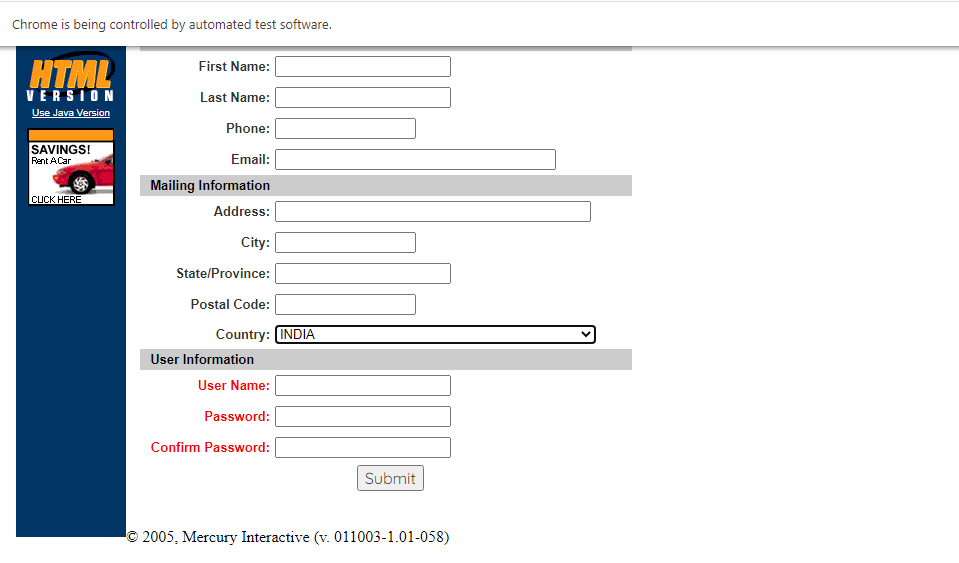
**Code:**

**Part A**

|  |
| --- |
| **package** prac77;  **import** org.openqa.selenium.By;  **import** org.openqa.selenium.WebDriver;  **import** org.openqa.selenium.chrome.ChromeDriver;  **import** org.openqa.selenium.support.ui.Select;  **import** java.util.List;  **import** org.openqa.selenium.WebElement;  **public** **class** Prac7 {  **static** String *driverPath*="E:\\tycs\\stqa prac\\prac2\\chromedriver\_win32\\chromedriver.exe";  **public** **static** **void** main(String[] args) {  System.*setProperty*("webdriver.chrome.driver", *driverPath*);  WebDriver driver= **new** ChromeDriver();  driver.get("E:/tycs/stqa%20prac/prac7/index.html");  Select selectDropdown = **new** Select(driver.findElement(By.*id*("bikes")));  List<WebElement> listOptionDropdown = selectDropdown.getOptions();  **int** dropdownCount = listOptionDropdown.size();  System.***out***.println("Total Number of item count in dropdown list = " + dropdownCount);  driver.close();  System.***out***.println("performed by krunal 713");  }  } |

**Output:-**

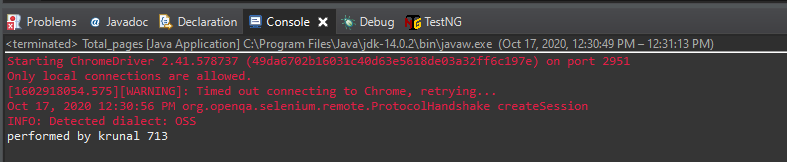


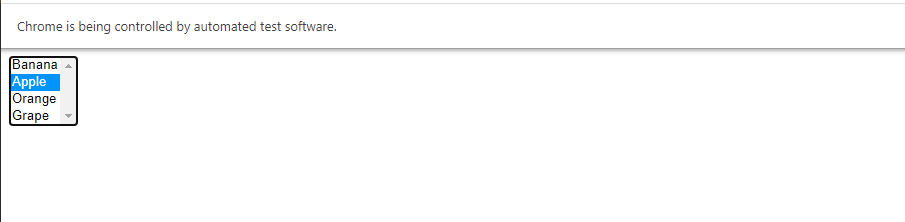
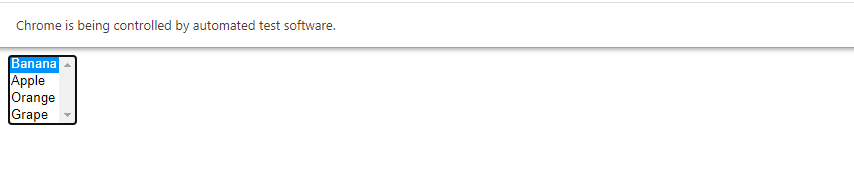


**Part-B**

|  |
| --- |
| **package** stqapracccc;  **import** org.openqa.selenium.By;  **import** org.openqa.selenium.WebDriver;  **import** org.openqa.selenium.chrome.ChromeDriver;  **import** org.openqa.selenium.support.ui.Select;  **public** **class** Total\_pages {  **static** String *driverPath*="E:\\tycs\\stqa prac\\prac2\\chromedriver\_win32\\chromedriver.exe";  **public** **static** **void** main(String[] args) **throws** InterruptedException {  //----PART2A ---------//  System.*setProperty*("webdriver.chrome.driver", *driverPath*);    String baseUrl = "http://demo.guru99.com/test/newtours/register.php";  WebDriver driver= **new** ChromeDriver();  driver.get(baseUrl);  Select drpCountry = **new** Select(driver.findElement(By.*name*("country")));  drpCountry.selectByVisibleText("INDIA");  Thread.*sleep*(8000);    //----PART2B ---------//  driver.get("http://jsbin.com/osebed/2");  Select fruits = **new** Select(driver.findElement(By.*id*("fruits")));  fruits.selectByVisibleText("Banana");  fruits.deselectByIndex(1);  Thread.*sleep*(8000);  driver.close();  System.***out***.println("performed by krunal 713");  }  } |

**Output**





**Date:20/10/2020**

**Practical no 8**

**AIM:** Write and test a program to count the number of check boxes on the page checked and unchecked count.

**Theory :**

**Xpath**

XPath stands for XML Path Language. It uses a non-XML syntax to provide a flexible

way of addressing (pointing to) different parts of an XML document. It can also be used

to test addressed nodes within a document to determine whether they match a pattern or not.

XPath is mainly used in XSLT, but can also be used as a much more powerful way of

navigating through the DOM of any XML-like language document using XPathExpression, such

as HTML and SVG, instead of relying on the Document.getElementById() or

ParentNode.querySelectorAll() methods, the Node.childNodes properties, and other

DOM Core features.

XPath uses a path notation (as in URLs) for navigating through the hierarchical

structure of an XML document. It uses a non-XML syntax so that it can be used in

URIs and XML attribute values.

**Xpath Syntax**

XPath contains the path of the element situated at the web page. Standard syntax for

creating XPath is.

Xpath = //tagname[@attribute =0 value0 ]

|  |  |
| --- | --- |
| Literal | Description |
| // : | Select current node. |
| Tagname: | Tagname of the particular node. |
| @: | Select attribute. |
| Attribute: | Attribute name of the node |
| Value: | Value of the attribute. |

**Types of Xpath**

There are two types of xpath:

**1. Absolute xpath** : It is the direct way to find the element, but the disadvantage of the absolute XPath is

that if there are any changes made in the path of the element then that XPath gets failed. The key

characteristic of XPath is that it begins with the single forward slash(/) ,which means you can select the

element from the root node.

**Example:** /html/body/div[2]/div[1]/div/h4[1]/b/html[1]/body[1]/div[2]/div[1]/div[1]/h4[1]/b[1]

**2. Relative xpath :** Relative Xpath starts from the middle of HTML DOM structure. It starts with double

forward slash (//). It can search elements anywhere on the webpage, means no need to write a long xpath

and you can start from the middle of HTML DOM structure. Relative Xpath is always preferred as it is not

a complete path from the root element. Below is the example of a relative XPath expression of the same

element shown in the below screen. This is the common format used to find element through a relative

XPath.

**Example**: //div[@class =0 featured − boxcloumnsize1 0 ]//h4[1]//b[1]

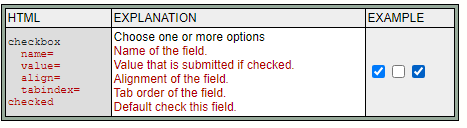
**Code:**

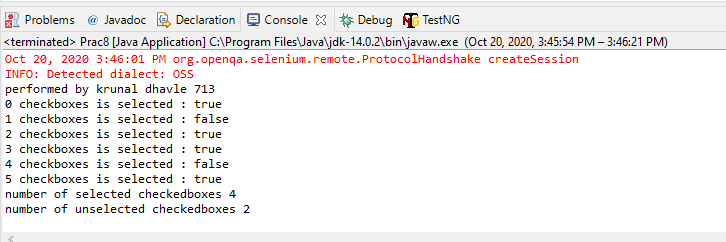
**Part A**

|  |
| --- |
| **import** java.util.List;  **import** org.openqa.selenium.By;  **import** org.openqa.selenium.WebDriver;  **import** org.openqa.selenium.WebElement;  **import** org.openqa.selenium.chrome.ChromeDriver;  **public** **class** Prac8 {  **static** String *driverPath*="E:\\tycs\\stqa prac\\prac2\\chromedriver\_win32\\chromedriver.exe";  **public** **static** **void** main(String[] args) **throws** InterruptedException {  System.*setProperty*("webdriver.chrome.driver", *driverPath*);  WebDriver driver= **new** ChromeDriver();  //driver.get("http://www.ironspider.ca/forms/checkradio.htm");  driver.get("http://www.echoecho.com/htmlforms09.htm");  //driver.get("file:///E:/tycs/stqa%20prac/prac8/radio.html");  List<WebElement> checkboxes = driver.findElements(By.*xpath*("//input[@type = 'checkbox']"));  **for**(**int** i = 0 ; i<checkboxes.size() ; i=i+1)  {  checkboxes.get(i).click();  }  System.***out***.println("performed by krunal dhavle 713");  **int** checkedCount = 0 , uncheckedCount =0;  **for**(**int** i =0 ; i < checkboxes.size() ; i++)  {  System.***out***.println(i + " " + "checkboxes is selected : "+checkboxes.get(i).isSelected());  **if**(checkboxes.get(i).isSelected())  checkedCount++;  **else**  uncheckedCount++;  }  Thread.*sleep*(5000);  System.***out***.println("number of selected checkedboxes " + checkedCount);  System.***out***.println("number of unselected checkedboxes " + uncheckedCount);  driver.close();  }  } |

**Output:-**

****

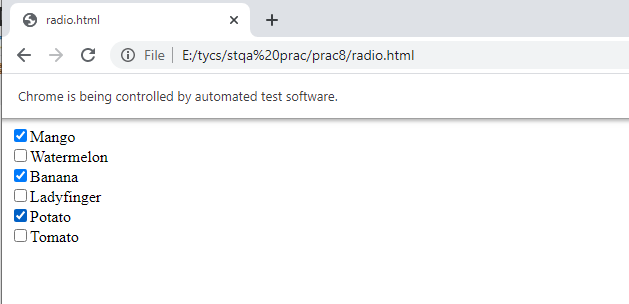


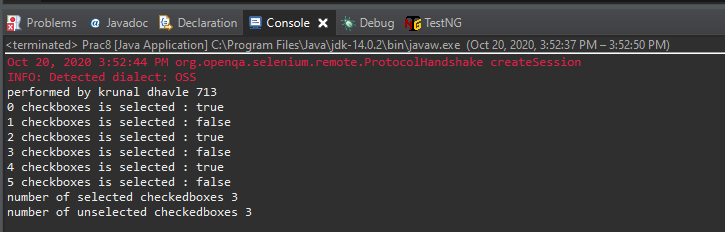


**Part -2**

|  |
| --- |
| **import** java.util.List;  **import** org.openqa.selenium.By;  **import** org.openqa.selenium.WebDriver;  **import** org.openqa.selenium.WebElement;  **import** org.openqa.selenium.chrome.ChromeDriver;  **public** **class** Prac8 {  **static** String *driverPath*="E:\\tycs\\stqa prac\\prac2\\chromedriver\_win32\\chromedriver.exe";  **public** **static** **void** main(String[] args) **throws** InterruptedException {  System.*setProperty*("webdriver.chrome.driver", *driverPath*);  WebDriver driver= **new** ChromeDriver();  driver.get("file:///E:/tycs/stqa%20prac/prac8/radio.html");  List<WebElement> checkboxes = driver.findElements(By.*xpath*("//input[@type = 'checkbox']"));  **for**(**int** i = 0 ; i<checkboxes.size() ; i=i+2)  {  checkboxes.get(i).click();  }  System.***out***.println("performed by krunal dhavle 713");  **int** checkedCount = 0 , uncheckedCount =0;  **for**(**int** i =0 ; i < checkboxes.size() ; i++)  {  System.***out***.println(i + " " + "checkboxes is selected : "+checkboxes.get(i).isSelected());  **if**(checkboxes.get(i).isSelected())  checkedCount++;  **else**  uncheckedCount++;  }  Thread.*sleep*(5000);  System.***out***.println("number of selected checkedboxes " + checkedCount);  System.***out***.println("number of unselected checkedboxes " + uncheckedCount);  driver.close();  }  } |

**Output:-**





Date:03/11/2020

**Practical no 9**

**AIM:** Load Testing using JMeter.

**Theory:-**

**What is JMeter ?**

Apache JMeter may be used to test performance both on static and dynamic resources,

Web dynamic applications. It can be used to simulate a heavy load on a server, group

of servers, network or object to test its strength or to analyze overall performance under

different load types.

**1. Thread Group :** Thread group elements are the beginning points of any test plan.

All controllers and samplers must be under a thread group. Other elements, e.g.

Listeners, may be placed directly under the test plan, in which case they will apply

to all the thread groups. As the name implies, the thread group element controls

the number of threads JMeter will use to execute your test. The controls for a

thread group allow you to:

**1.1 Number of Threads** : Each thread will execute the test plan in its entirety

and completely independently of other test threads. Multiple threads are used

to simulate concurrent connections to your server application.

**1.2 Ramp-up Period** : The ramp-up period tells JMeter how long to take to

”ramp-up” to the full number of threads chosen. If 10 threads are used, and

the ramp-up period is 100 seconds, then JMeter will take 100 seconds to get all

10 threads up and running. Each thread will start 10 (100/10) seconds after

the previous thread was begun. If there are 30 threads and a ramp-up period

of 120 seconds, then each successive thread will be delayed by 4 seconds.

**1.3 Loops :** By default, the thread group is configured to loop once through its

elements. However one can change it repeat the tests.

**2. Controllers** : JMeter has two types of Controllers: Samplers and Logical Controllers.

These drive the processing of a test.

**2.1 Samplers** : Samplers tell JMeter to send requests to a server and wait for a

response. They are processed in the order they appear in the tree. Controllers

can be used to modify the number of repetitions of a sampler. ”HTTP Request” is one of those samplers which we use to interact with HTTP protocol

to our server. **2.2 Logic Controllers :** Logic Controllers let you customize the logic that JMeter

uses to decide when to send requests. Logic Controllers can change the order

of requests coming from their child elements. They can modify the requests

themselves, cause JMeter to repeat requests, etc

**3. Listeners** : Listeners provide access to the information JMeter gathers about the test

cases while JMeter runs. The Graph Results listener plots the response times on

a graph. The ”View Results Tree” Listener shows details of sampler requests and

responses, and can display basic HTML and XML representations of the response.

Other listeners provide summary or aggregation information.

**3.1 Results Tree** : The View Results Tree shows a tree of all sample responses,

allowing you to view the response for any sample. In addition to showing the

response, you can see the time it took to get this response, and some response

codes. Note that the Request panel only shows the headers added by JMeter.

It does not show any headers (such as Host) that may be added by the HTTP

protocol implementation.

**3.2 Graph Results** : The Graph Results listener generates a simple graph that

plots all sample times. Along the bottom of the graph, the current sample (black), the

current average of all samples (blue), the current standard

deviation (red), and the current throughput rate (green) are displayed in milliseconds.

**Pre-Installation Requirements**

**JDK/JRE Installation is required(Java 8 or higher)**

Running the batch file may not work directly so one needs to add path of the java

bin folder through the terminal and then try running the batch file.

For Linux use OpenJDK package(latest available) and mark the jar file as executable

through file properties or use chmod to set execution permission.

**Steps:**

**Step-1** :- Download and Unzip Jmeter package from Apache Website.

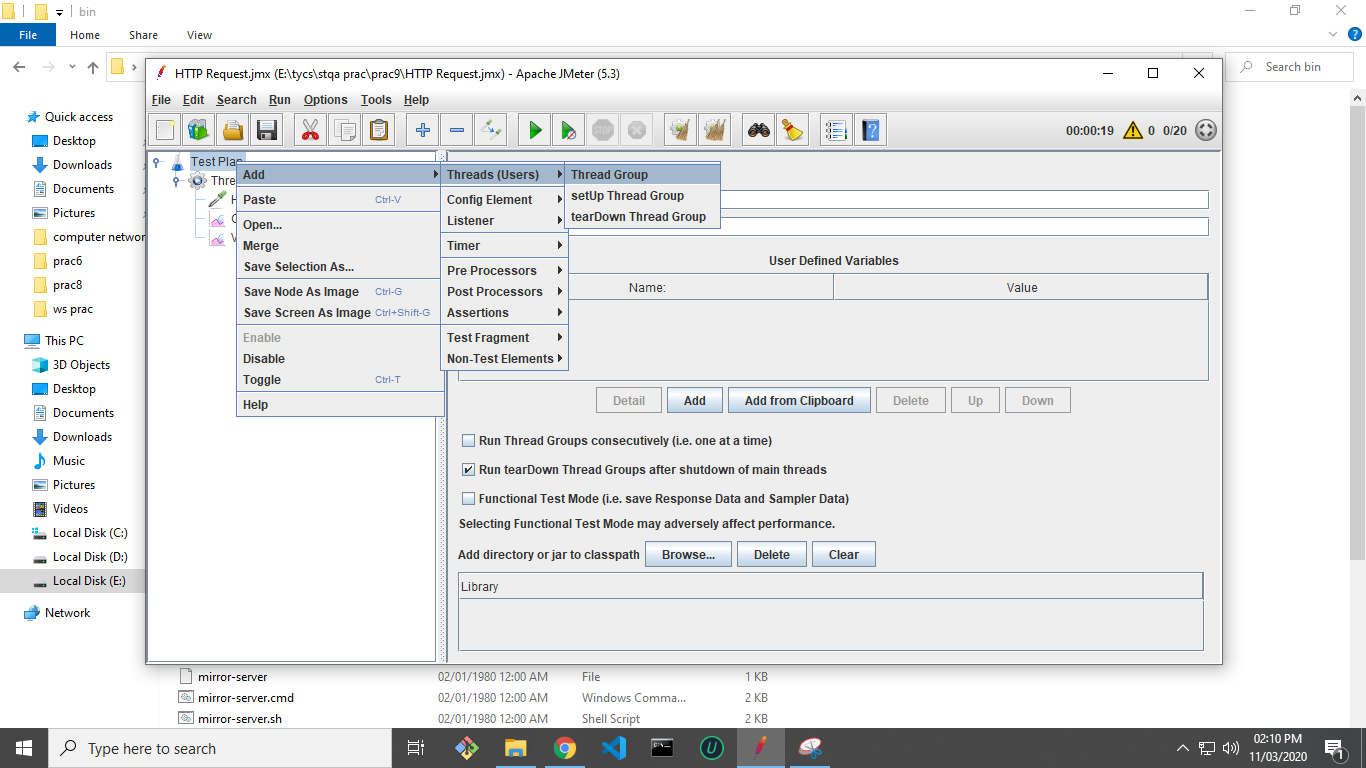
<https://jmeter.apache.org/download_jmeter.cgi>

**Step-2** :- Navigate to bin folder and run the ”ApacheJMeter.jar” or ”jmeter.bat” file

and it will launch a window. In case of error/failure refer the Pre-Installation Requirements.

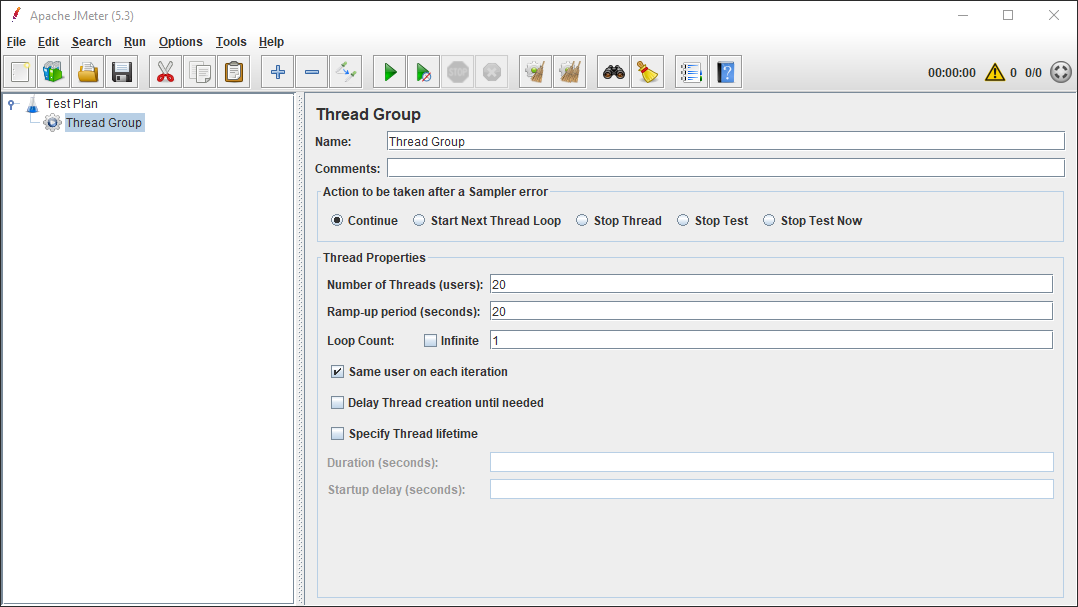
**Step-3** :- Now First step is to add an Thread Group, RightclickonT estPlan →

Add → T hreads(Users) → T hreadGroup



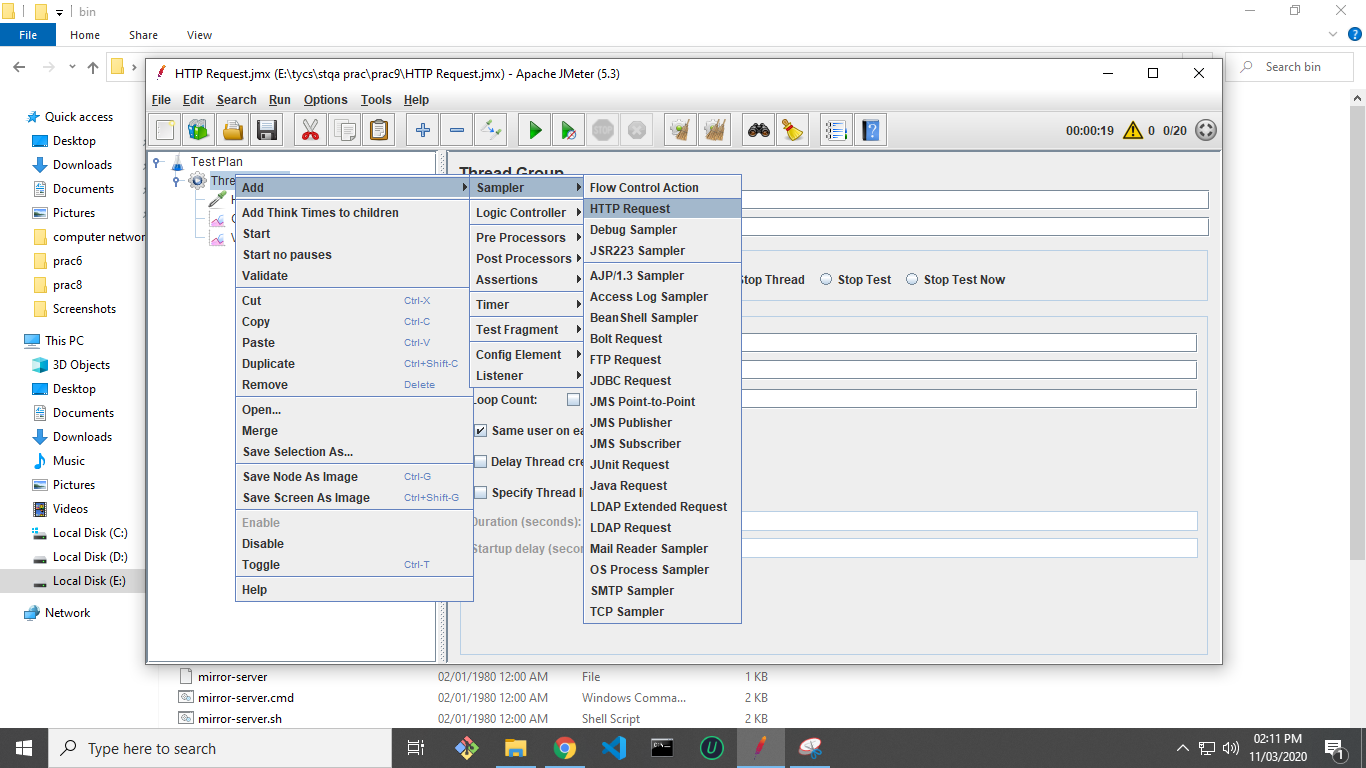
**Step-4 :-** Next rename the Thread Group and set ”Number of Threads (Users):”

as 20 and ”Ramp-up period(seconds):” as 20. Keep rest options as default.



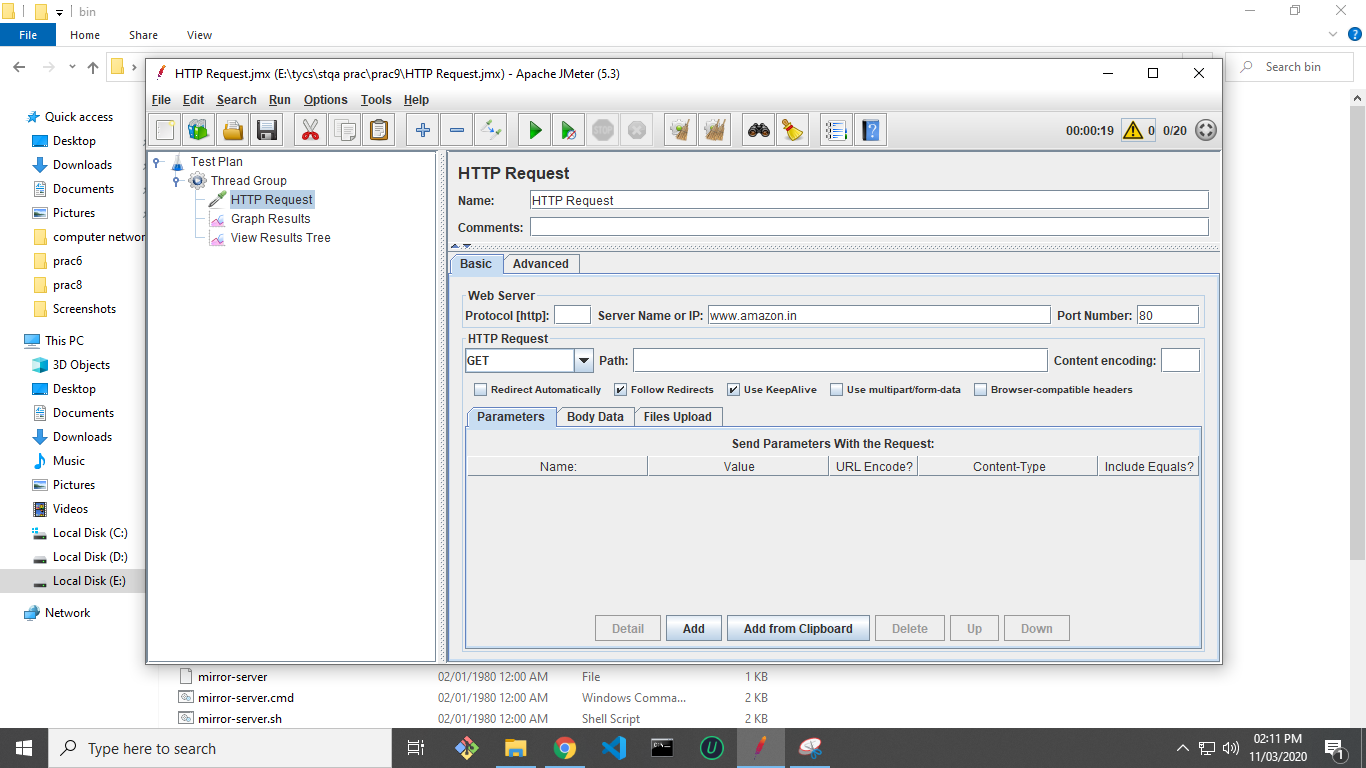
**Step-5** :- We add a HTTP Request Sampler to the Thread Group Rightclick T hreadGroup →

Add → Sampler → HT T P Request.



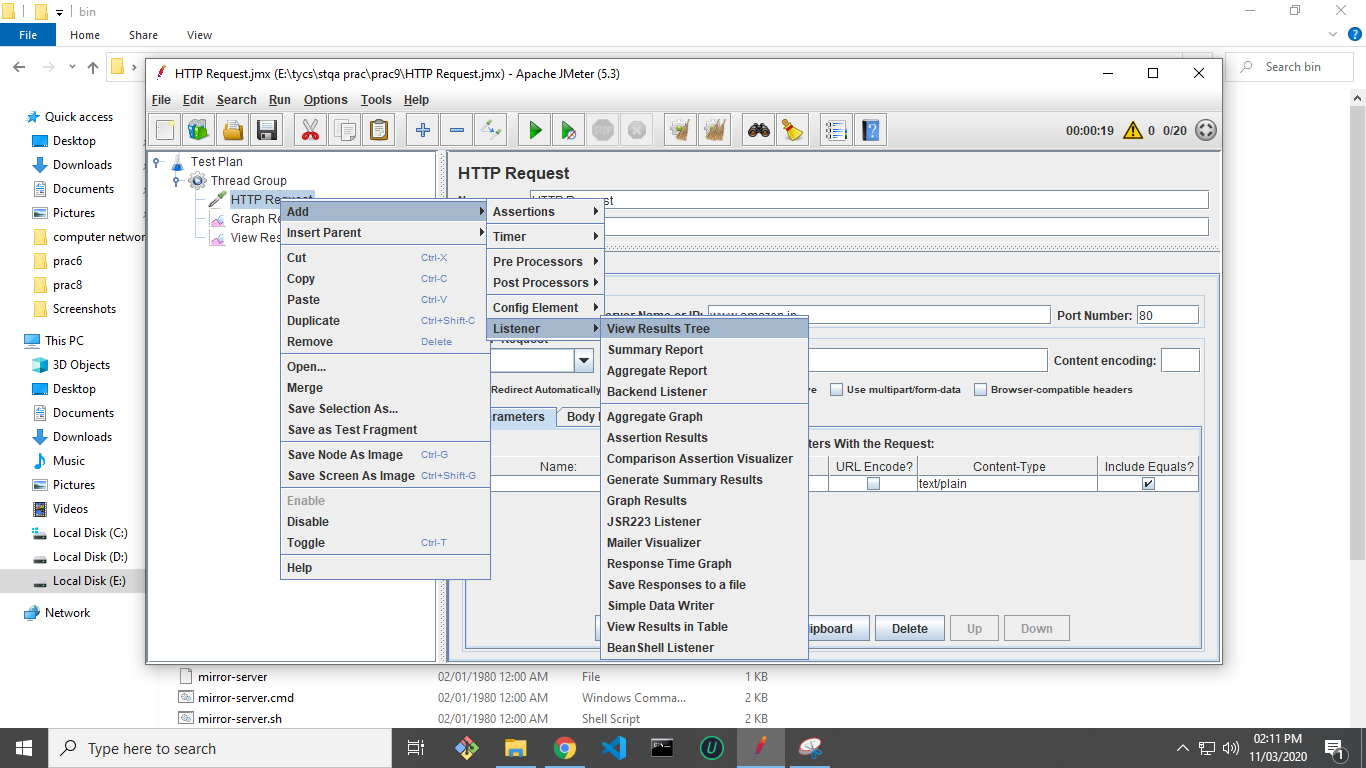
**Step-6** :- Next, Edit the ”Server Name or IP:” to the specific website or Server IP address, optionally one

configure port number and Request parameters and path as well.



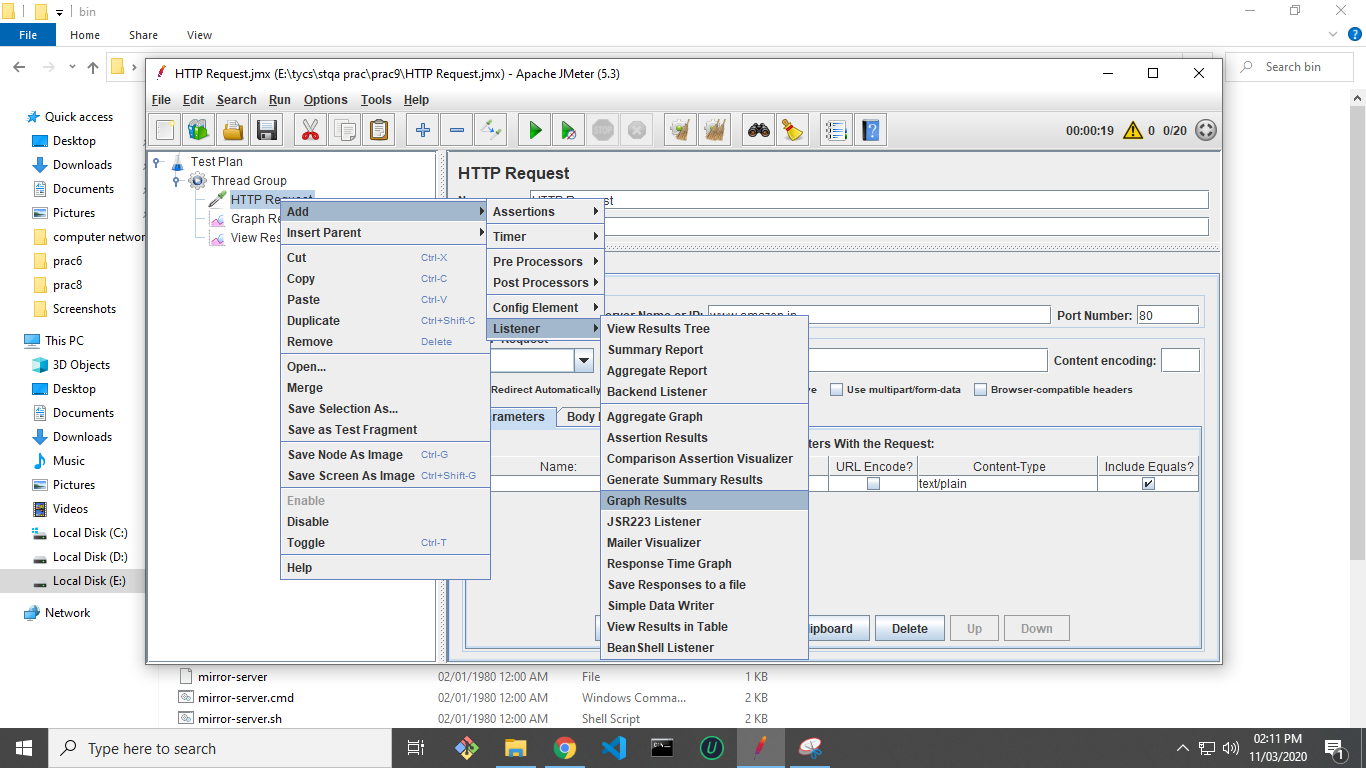
**Step-7 :-** To view the log data add a ”View Results Tree Listener” Rightclick T estPlan → Add

→ Listener → ViewResultsTree.



**Step-8 :-** To visualize the data we add a ”Graph Results Listener” Rightclick T estPlan →

Add → Listener → GraphResults

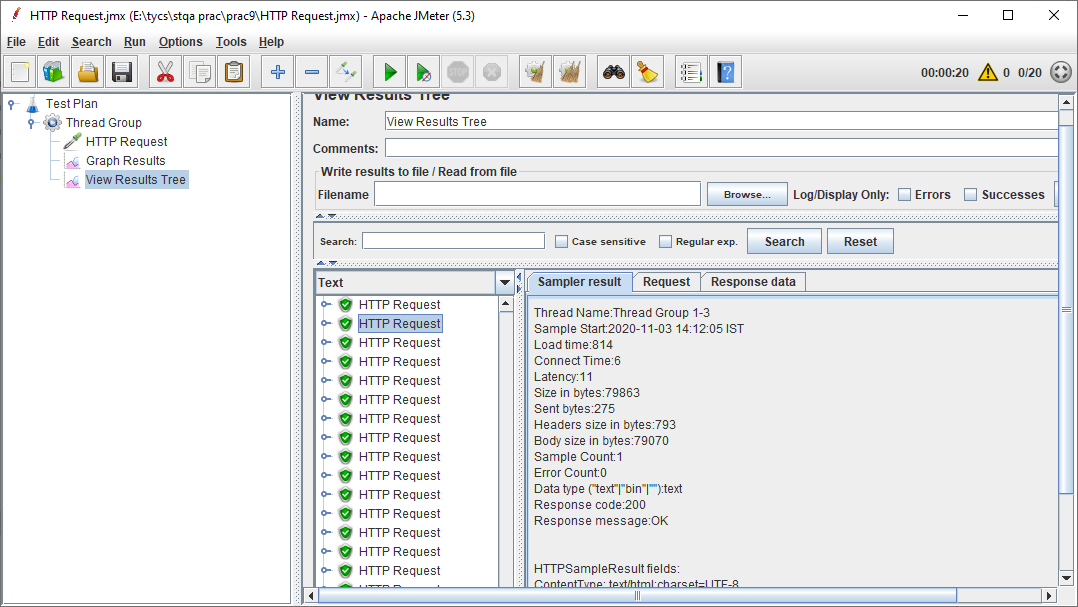


**Step-9 :-** Next save the project as some filename into the hard drive by clicking Ctrl-S.

**Step-10** :- Now Run the tests by clicking on the Green Play Button .

**Step-11** :- After the test completes successfully you can view the logged results of the test in ”View

Results Tree”



**Step-12** :- One can also visualize the parameters by looking at the ”Graph Results”.

