

1.28 Configure HTTP, FTP, Java, JDBC, and UDV in JMeter

This section will guide you to understand:

- How to use HTTP request, FTP request, Java request, JDBC connection and User Defined Variable

Development Environment:

- Apache JMeter 5.1.1 version
- MySQL 5.7
- Eclipse IDE
- Java 1.8

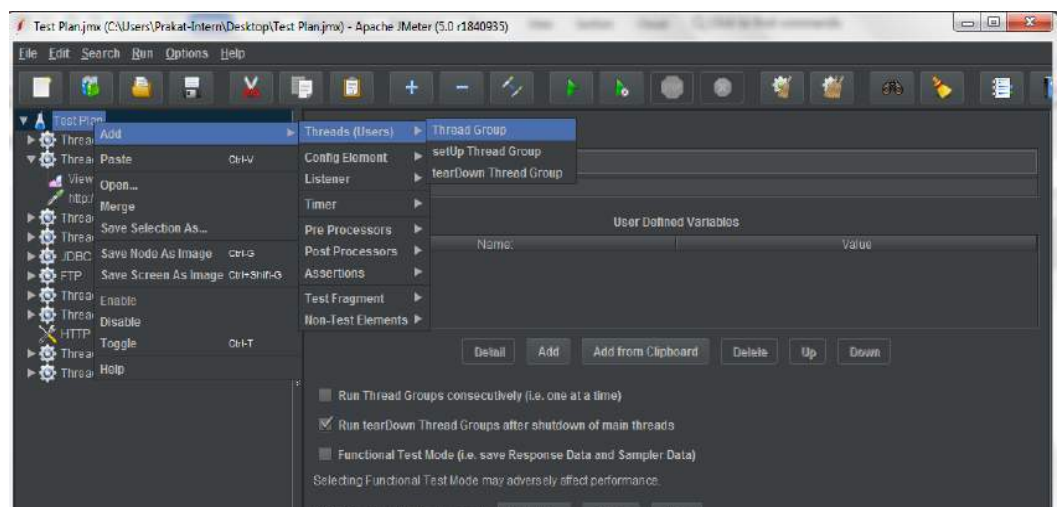
This guide has eight subsections, namely:

- 1.28.1 Using HTTP request
- 1.28.2 Using HTTP request defaults
- 1.28.3 Using FTP request
- 1.28.4 Using Java request
- 1.28.5 Adding Custom Java request
- 1.28.6 Establishing JDBC connection
- 1.28.7 Using User-Defined Variable
- 1.28.8 Pushing the code to GitHub repositories

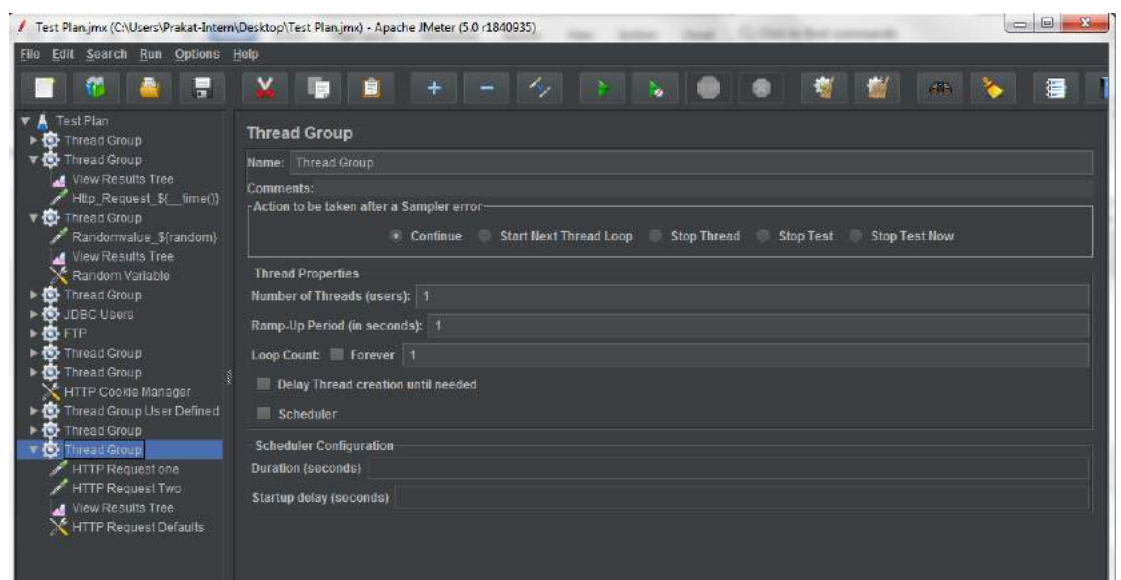
Step 1.28.1: Using HTTP request

- This HTTP request is used to send an HTTP/HTTPS request to a web server.

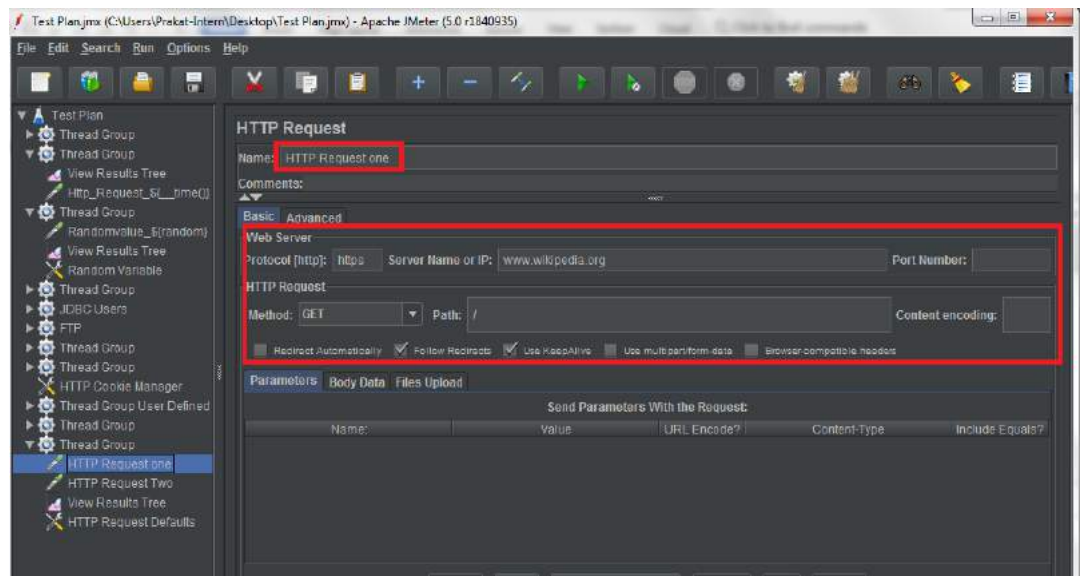
- They simulate a user request for a page from the target server. For example, you can add a HTTP request sampler if you need to perform a POST, GET, or DELETE on an HTTP service.
- Adding users in the thread group:
 1. Open JMeter.
 2. Right click on the Test Plan.
 3. Click on Add -> Threads -> Thread Group.



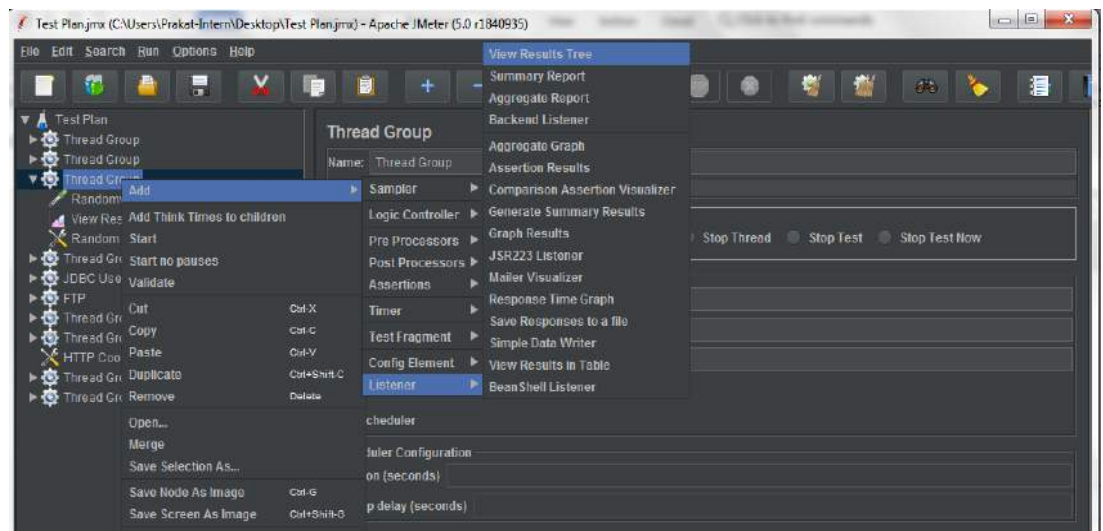
4. Set Number of Threads to 1.
5. Set Ramp-Up Period to 1.
6. Set Loop Count to 1.



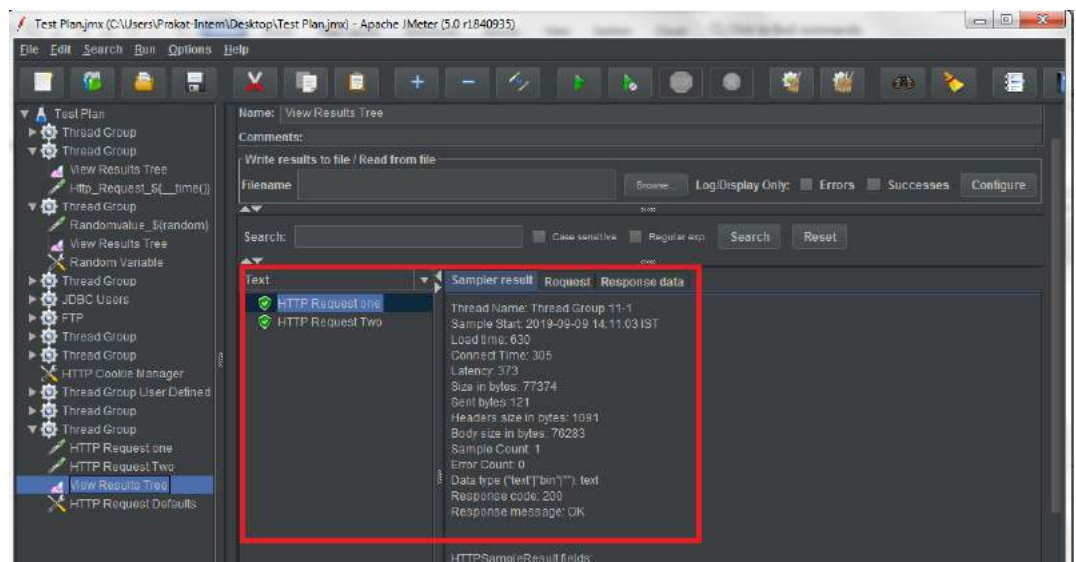
- Adding HTTP request:
 1. Right click on Thread Group.
 2. Click on Add -> Sampler -> HTTP Request.
 3. Rename HTTP Request to **HTTP Request One**.
 4. Give the protocol as “https”.
 5. Give the server name as “www.wikipedia.org”.
 6. Give the path “/”.



- Adding view results tree:
 1. Right click on Thread Group.
 2. Click on Add -> Listener -> View Results Tree.



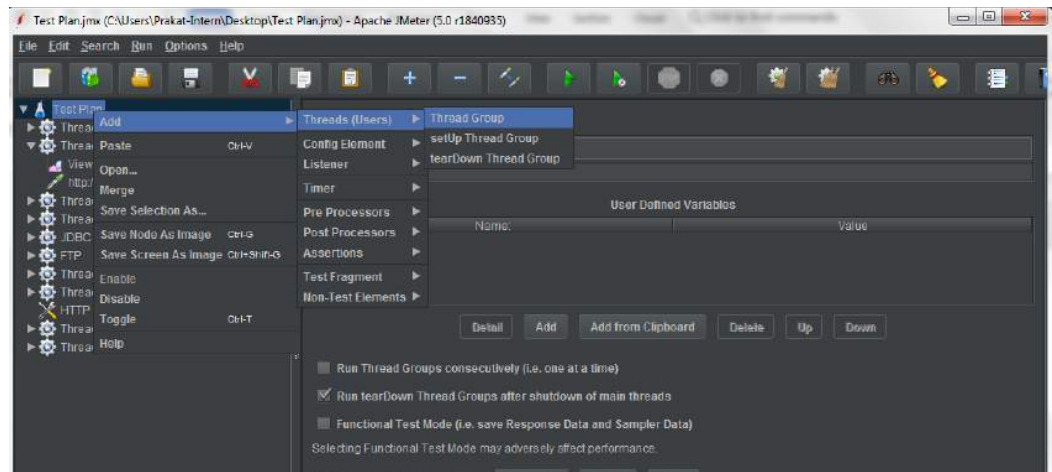
3. Run the Thread Group and open the View Results Tree to see the output.



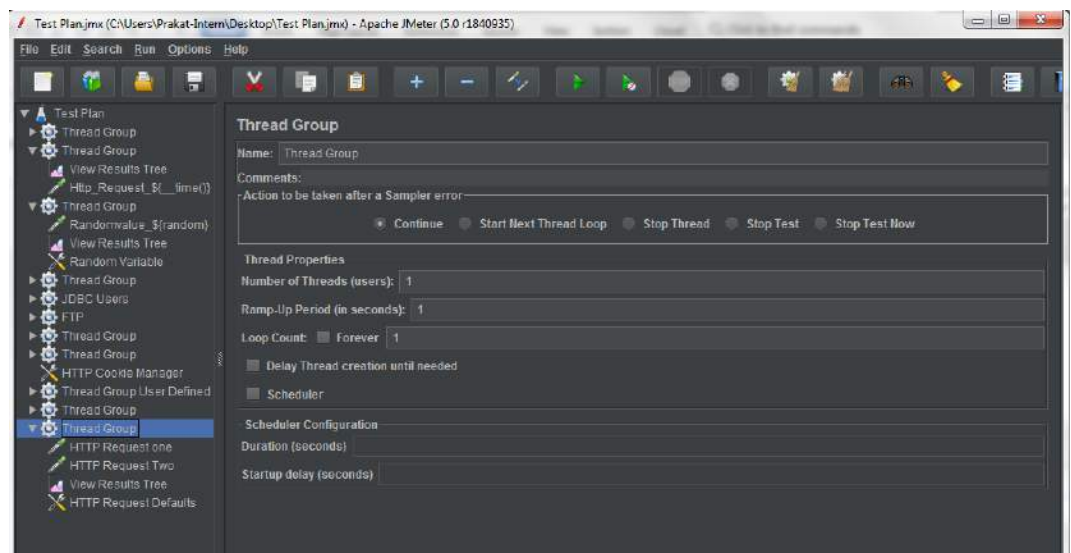
Step 1.28.2: Using HTTP request defaults

- The main goals of using the HTTP default request is to:
 - Avoid data duplication in tests
 - Make test scripts more (easily) maintainable
- For example: You have a test script that consists of 2 HTTP requests. All trigger the same test server “www.wikipedia.org”, but refer to different paths “wiki/String”, “wiki/About”, etc.
- We see elements that both HTTP requests have in common, <https://en.wikipedia.org/wiki>. Hence, we can move this common element to the HTTP request defaults field.

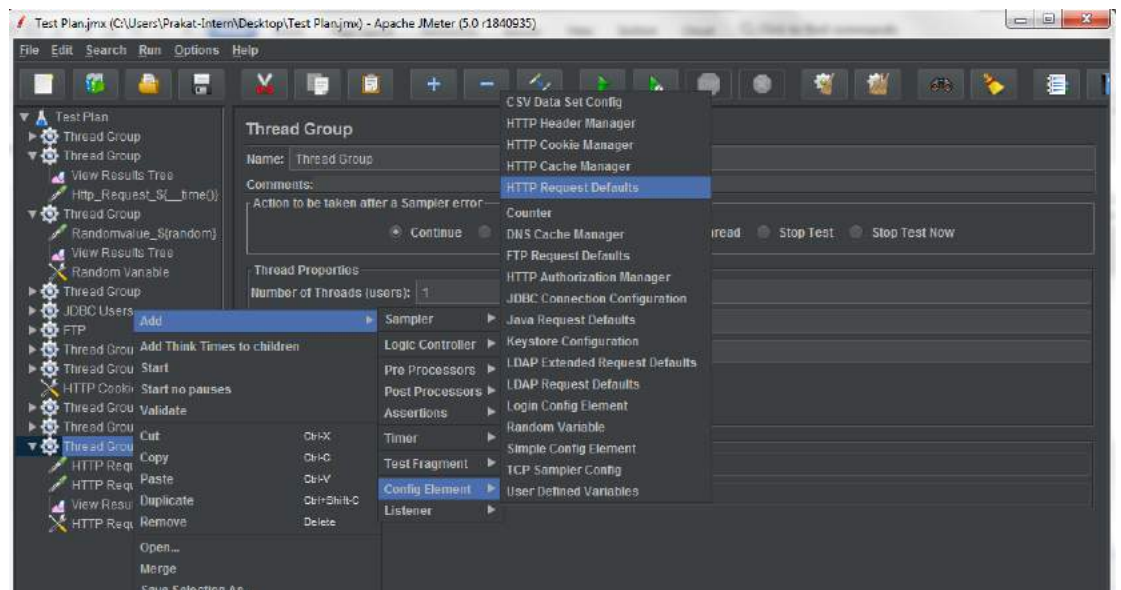
- Adding users in a thread group:
 1. Right click on the Test Plan.
 2. Click on Add -> Threads -> Thread Group.



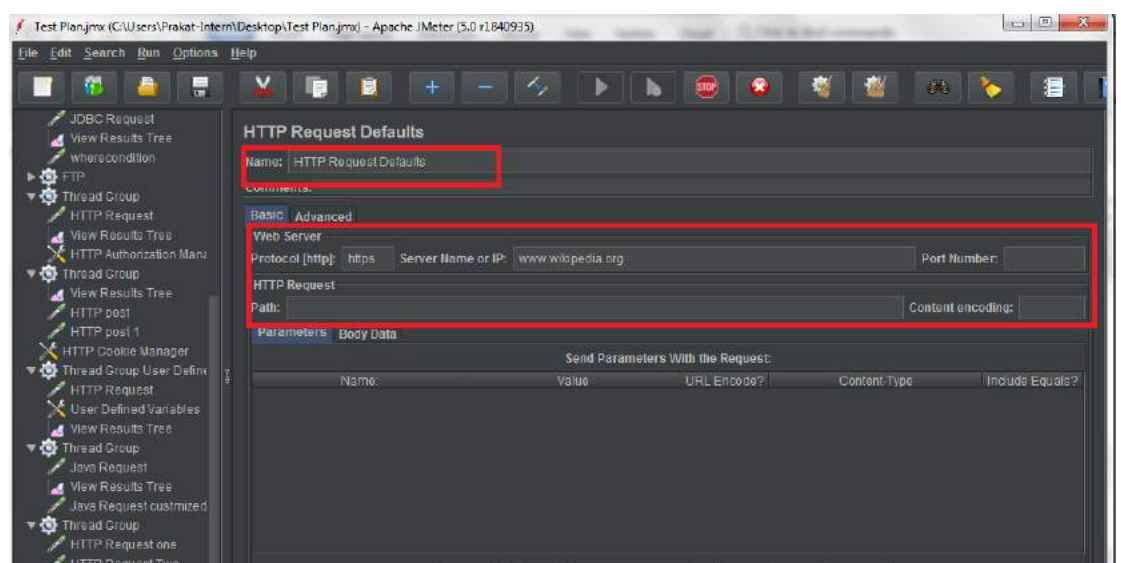
3. Set Number of Threads to 1.
4. Set Ramp-Up Period to 1.
5. Set Loop Count to 1.



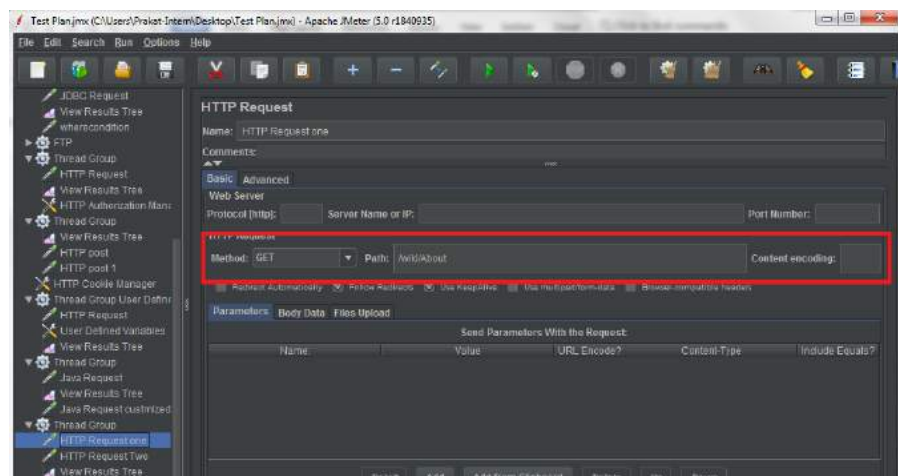
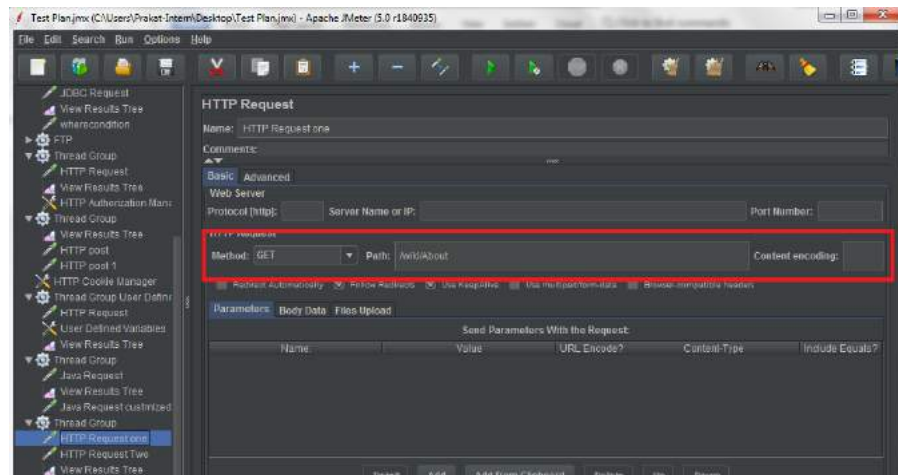
- Adding HTTP request defaults:
 1. Right click on Thread Group.
 2. Click on Add -> Config Element -> HTTP request defaults.



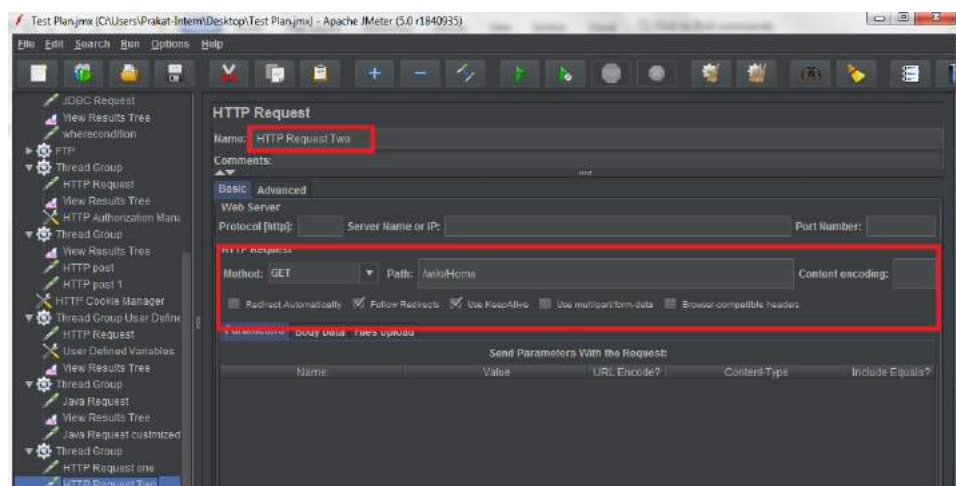
3. Give protocol as "https".
4. Give server name as "www.wikipedia.org".



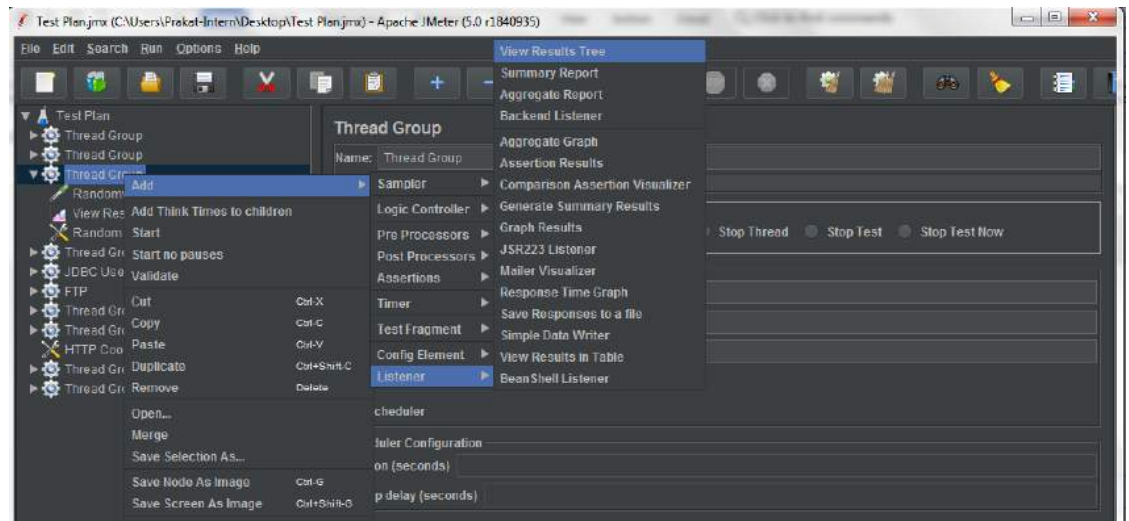
- Adding HTTP request:
 1. Right click on Thread Group.
 2. Click on Add -> Sampler -> HTTP Request.
 3. Rename HTTP Request to **HTTP Request One**.
 4. Give path "/wiki/About".



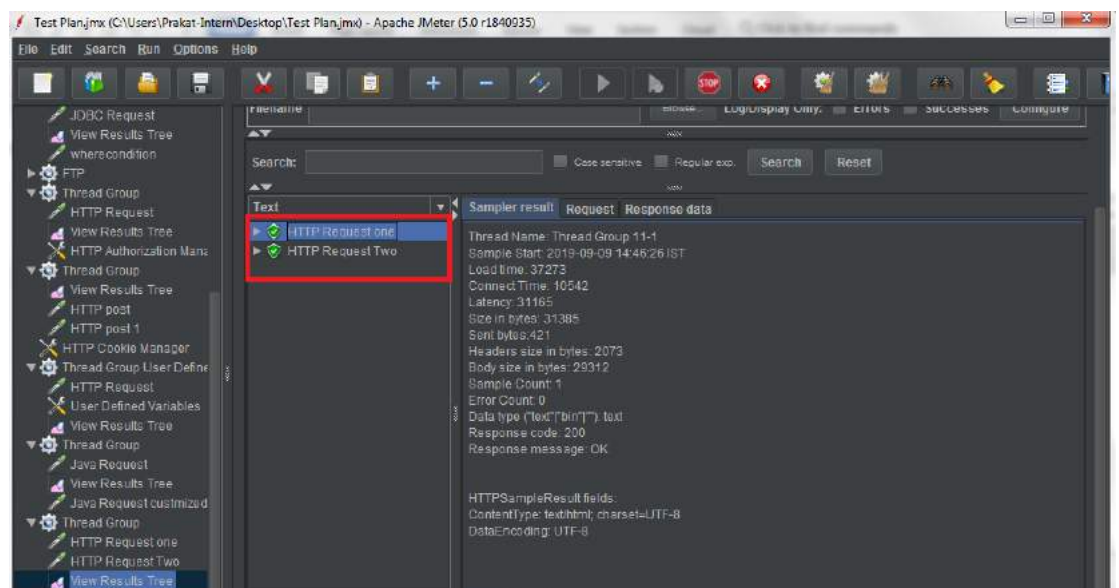
- Adding HTTP request:
 1. Right click on Thread Group.
 2. Click on Add -> Sampler -> HTTP request.
 3. Rename HTTP Request to “HTTP Request Two”.
 4. Give path “/wiki/Home”.



- Adding view results tree:
 1. Right click on Thread Group.
 2. Click on Add -> Listener -> View Results Tree.



3. Run the Thread Group and open the View Results Tree to see the output.



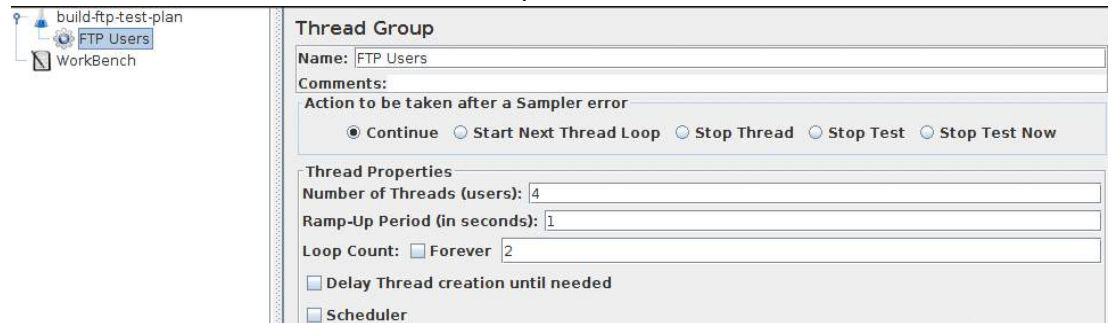
Step 1.28.3: Using FTP request

- File Transfer Protocol request is used to test the file upload or download from a specified FTP server.
- FTP request needs an FTP server with username and password to login, as without authentication no FTP server would allow you to upload or remove files from a remote server. So, the server name must be provided to make a

request followed by the file location, and then the selection of uploading (PUT) or downloading (GET) as per the scenario which you are testing.

- Adding users in a thread group:

1. Right click on Test Plan.
2. Click on Add -> Threads -> Thread Group.

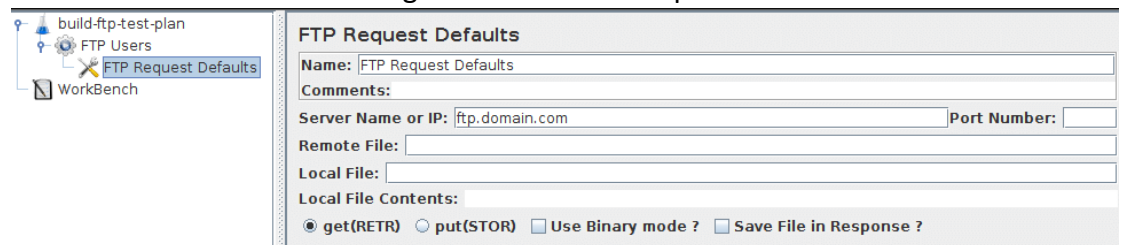


The screenshot shows the 'Thread Group' configuration dialog. On the left, a tree view shows 'build-ftp-test-plan' with 'FTP Users' and 'WorkBench' under it. The dialog has the following fields and options:

- Name:** FTP Users
- Comments:**
- Action to be taken after a Sampler error:** ☒ Continue ☐ Start Next Thread Loop ☐ Stop Thread ☐ Stop Test ☐ Stop Test Now
- Thread Properties:**
 - Number of Threads (users):** 4
 - Ramp-Up Period (in seconds):** 1
 - Loop Count:** ☐ Forever
 - ☐ Delay Thread creation until needed
 - ☐ Scheduler

- Adding default FTP request defaults:

1. Right click on Thread Group.
2. Click on Add → Config Element → FTP Request Defaults.



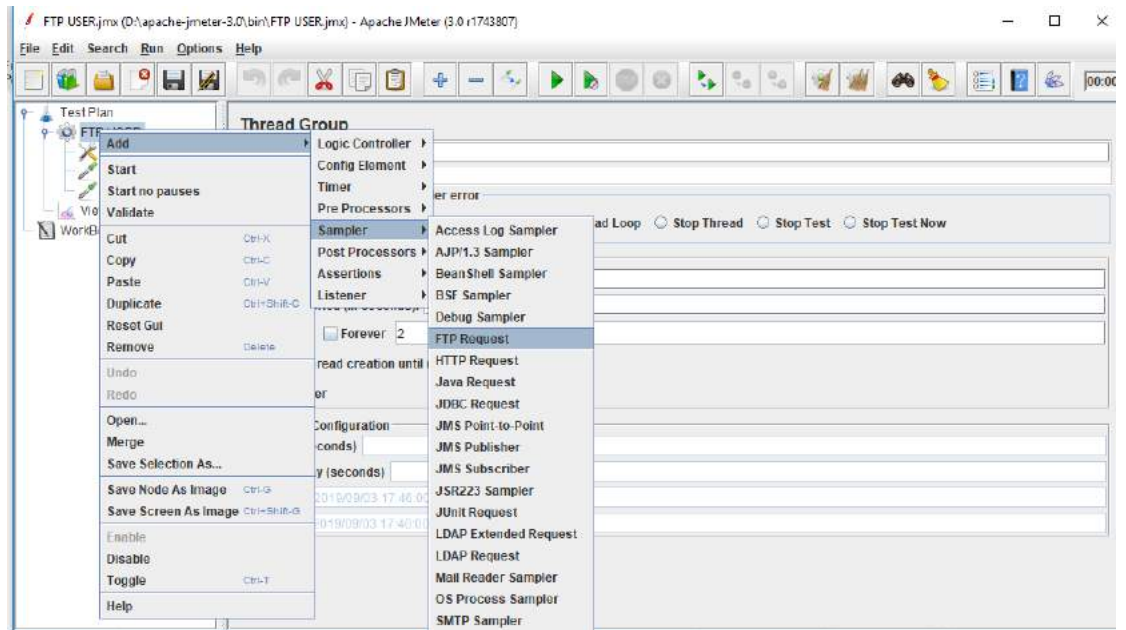
The screenshot shows the 'FTP Request Defaults' configuration dialog. On the left, the tree view shows 'FTP Request Defaults' added under 'FTP Users'. The dialog has the following fields and options:

- Name:** FTP Request Defaults
- Comments:**
- Server Name or IP:** ftp.domain.com **Port Number:**
- Remote File:**
- Local File:**
- Local File Contents:**
- Options:** ☒ get(RETR) ☐ put(STOR) ☐ Use Binary mode ? ☐ Save File in Response ?

- Adding FTP request:

1. Right click on Thread Group.

2. Click on Add → Sampler → FTP Request.

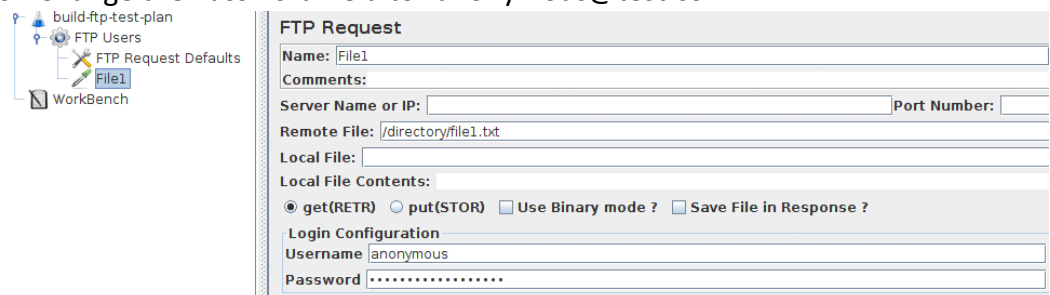


3. Change the *Name* to "File1".

4. Change the *Remote File* field to "/directory/file1.txt".

5. Change the *Username* field to "anonymous".

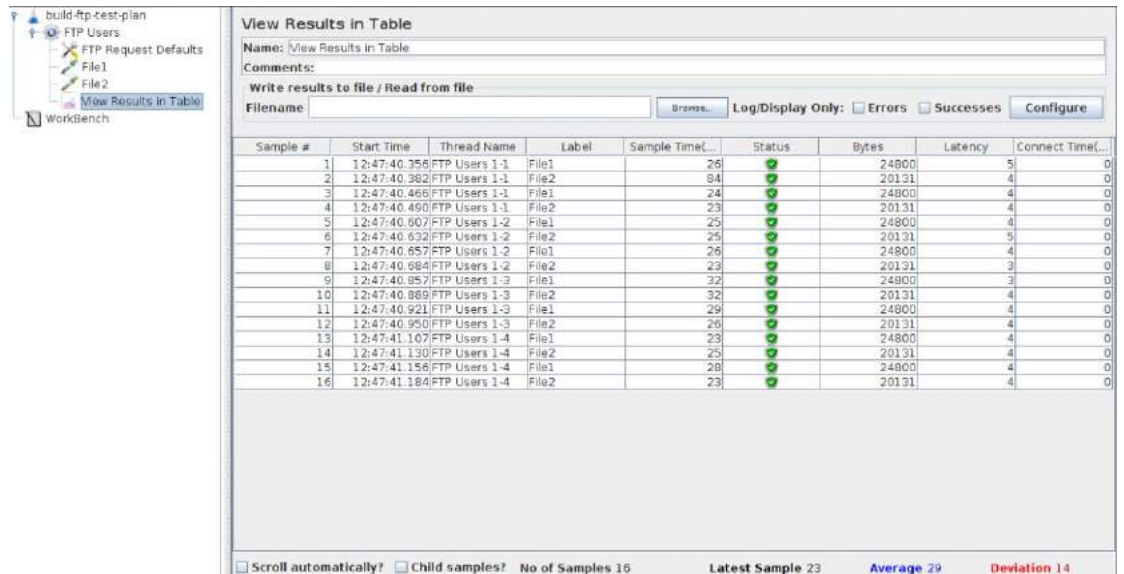
6. Change the *Password* field to "anonymous@test.com".



- Adding listeners and viewing results in table:

1. Right click on Thread Group.

2. Click on Add → Listener → View Results in the table.

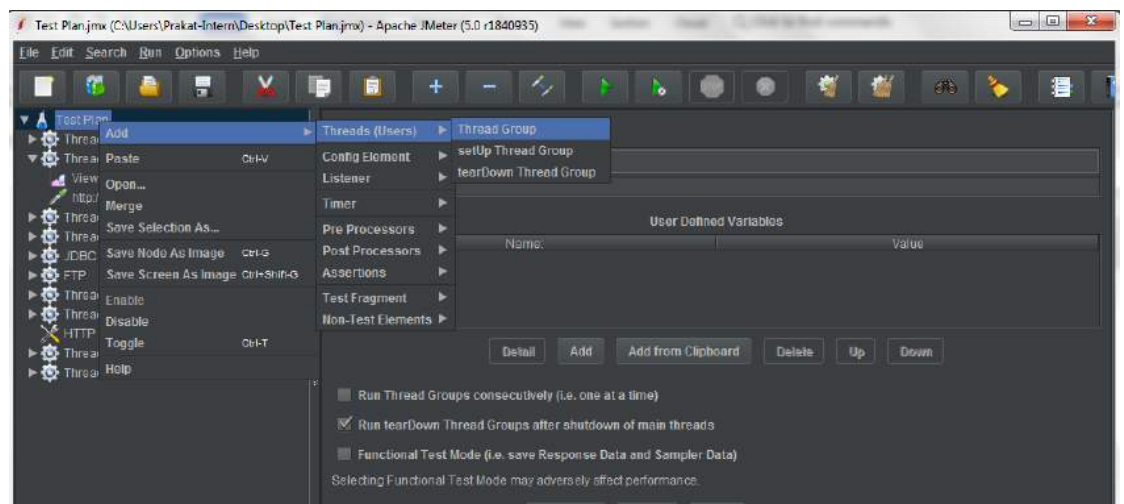


Sample #	Start Time	Thread Name	Label	Sample Time...	Status	Bytes	Latency	Connect Time...
1	12:47:40.356	FTP Users 1-1	File1	26	Success	24800	5	0
2	12:47:40.382	FTP Users 1-1	File2	84	Success	20131	4	0
3	12:47:40.466	FTP Users 1-1	File1	24	Success	24800	4	0
4	12:47:40.490	FTP Users 1-1	File2	23	Success	20131	4	0
5	12:47:40.907	FTP Users 1-2	File1	25	Success	24800	4	0
6	12:47:40.632	FTP Users 1-2	File2	25	Success	20131	5	0
7	12:47:40.657	FTP Users 1-2	File1	26	Success	24800	4	0
8	12:47:40.684	FTP Users 1-2	File2	23	Success	20131	3	0
9	12:47:40.857	FTP Users 1-3	File1	32	Success	24800	3	0
10	12:47:40.869	FTP Users 1-3	File2	32	Success	20131	4	0
11	12:47:40.921	FTP Users 1-3	File1	29	Success	24800	4	0
12	12:47:40.950	FTP Users 1-3	File2	26	Success	20131	4	0
13	12:47:41.107	FTP Users 1-4	File1	23	Success	24800	4	0
14	12:47:41.130	FTP Users 1-4	File2	25	Success	20131	4	0
15	12:47:41.156	FTP Users 1-4	File1	28	Success	24800	4	0
16	12:47:41.184	FTP Users 1-4	File2	23	Success	20131	4	0

☐ Scroll automatically?
 ☐ Child samples?
 No of Samples 16
 Latest Sample 23
 Average 29
 Deviation 14

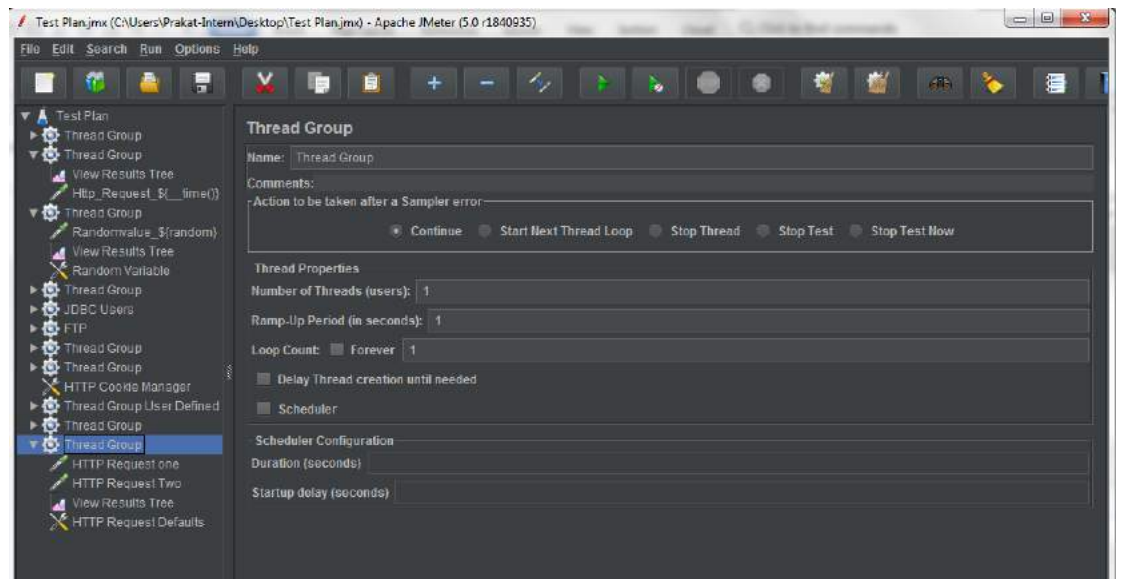
Step 1.28.4: Using Java request

- The **JavaTest** example sampler can be useful for checking test plans, because it allows one to set values in almost all the fields. These can then be used by assertions etc. The fields allow variables to be used and see the values readily.
- Adding users in a thread group:
 1. Right click on Test Plan.
 2. Click on Add → Threads → Thread Group.

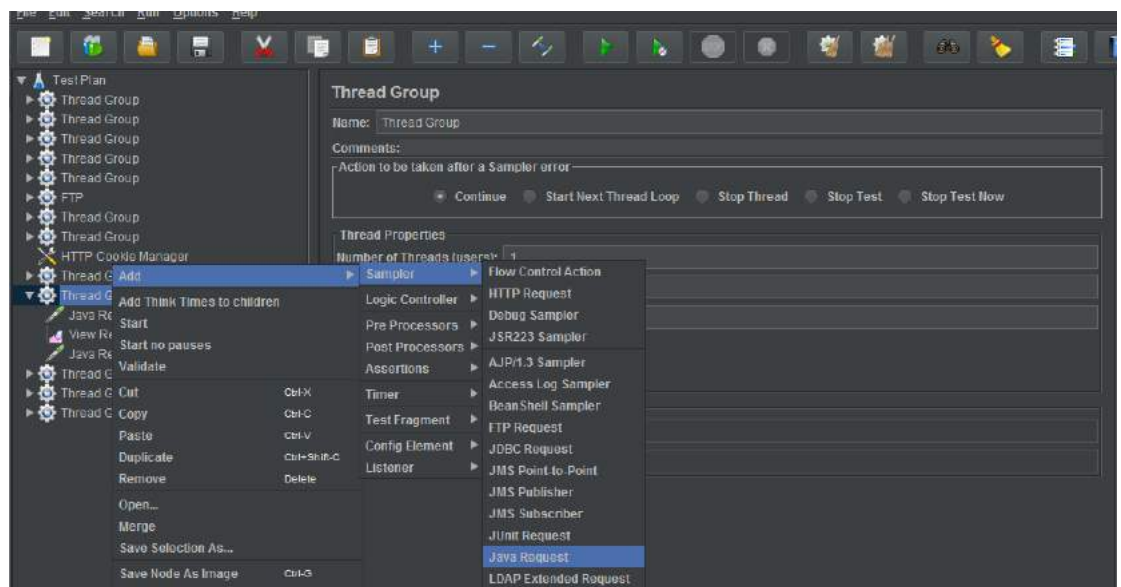


3. Set Number of Threads to 1.

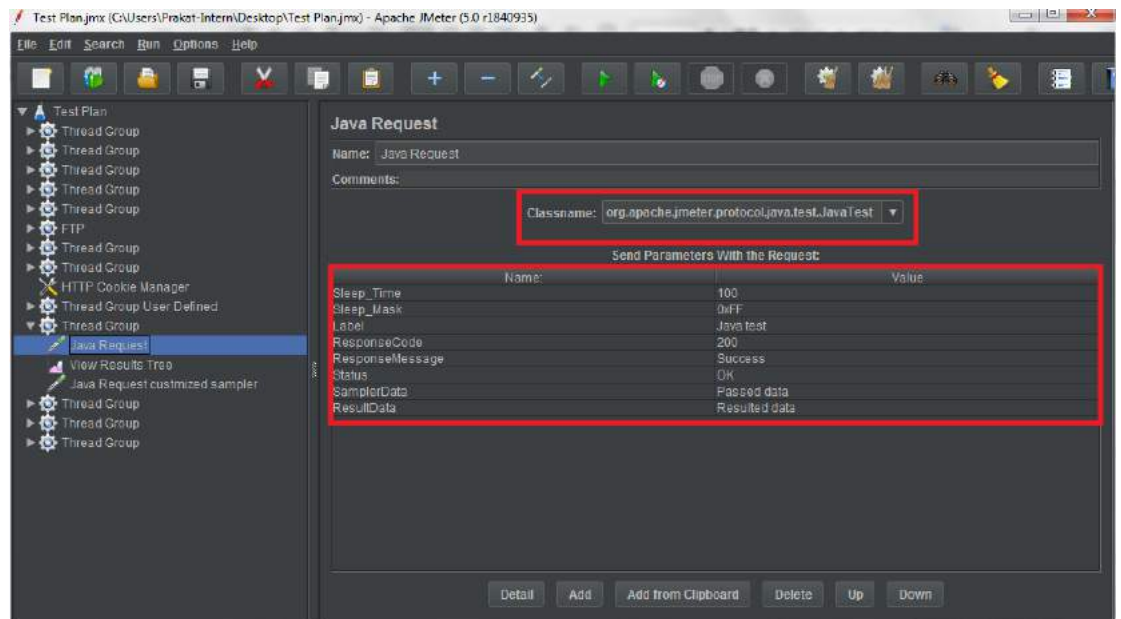
4. Set Ramp-Up Period to 1.
5. Set Loop Count to 1.



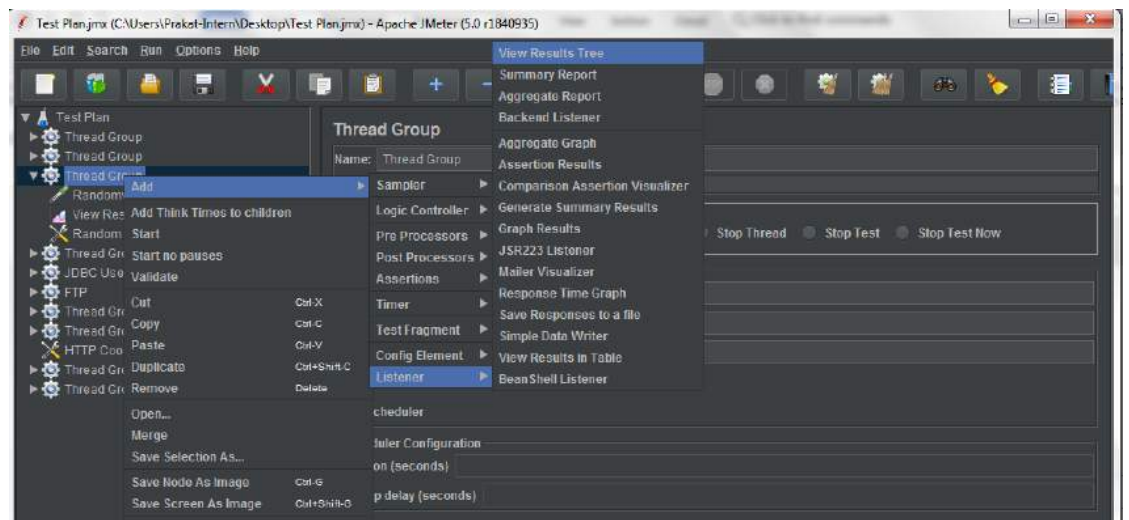
- Adding Java request:
 1. Right click on Thread Group.
 2. Click on Add -> Sampler -> Java Request.



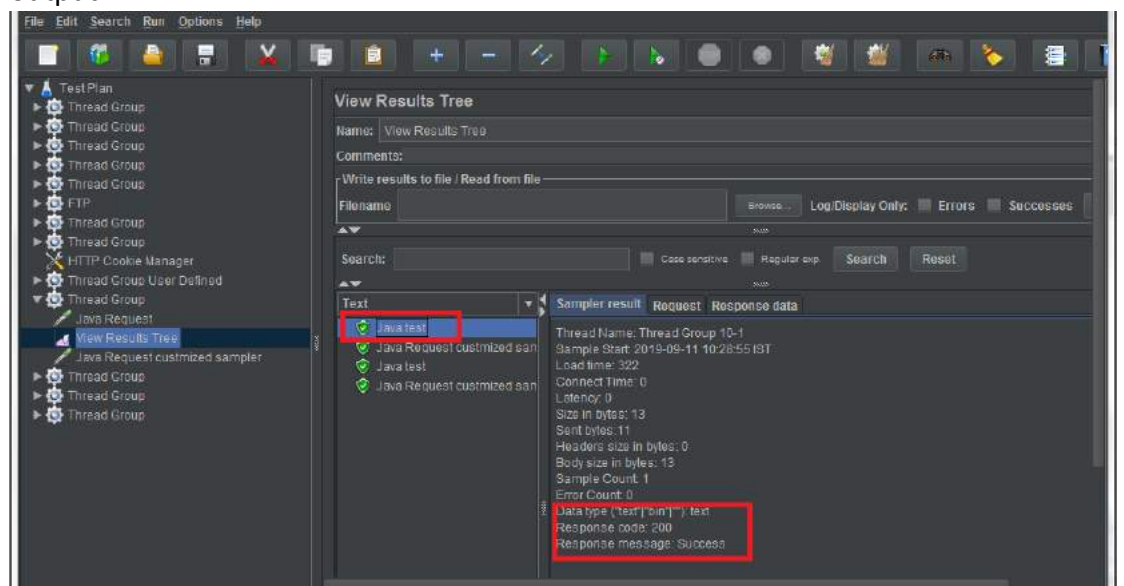
3. Enter Java request fields as follow:

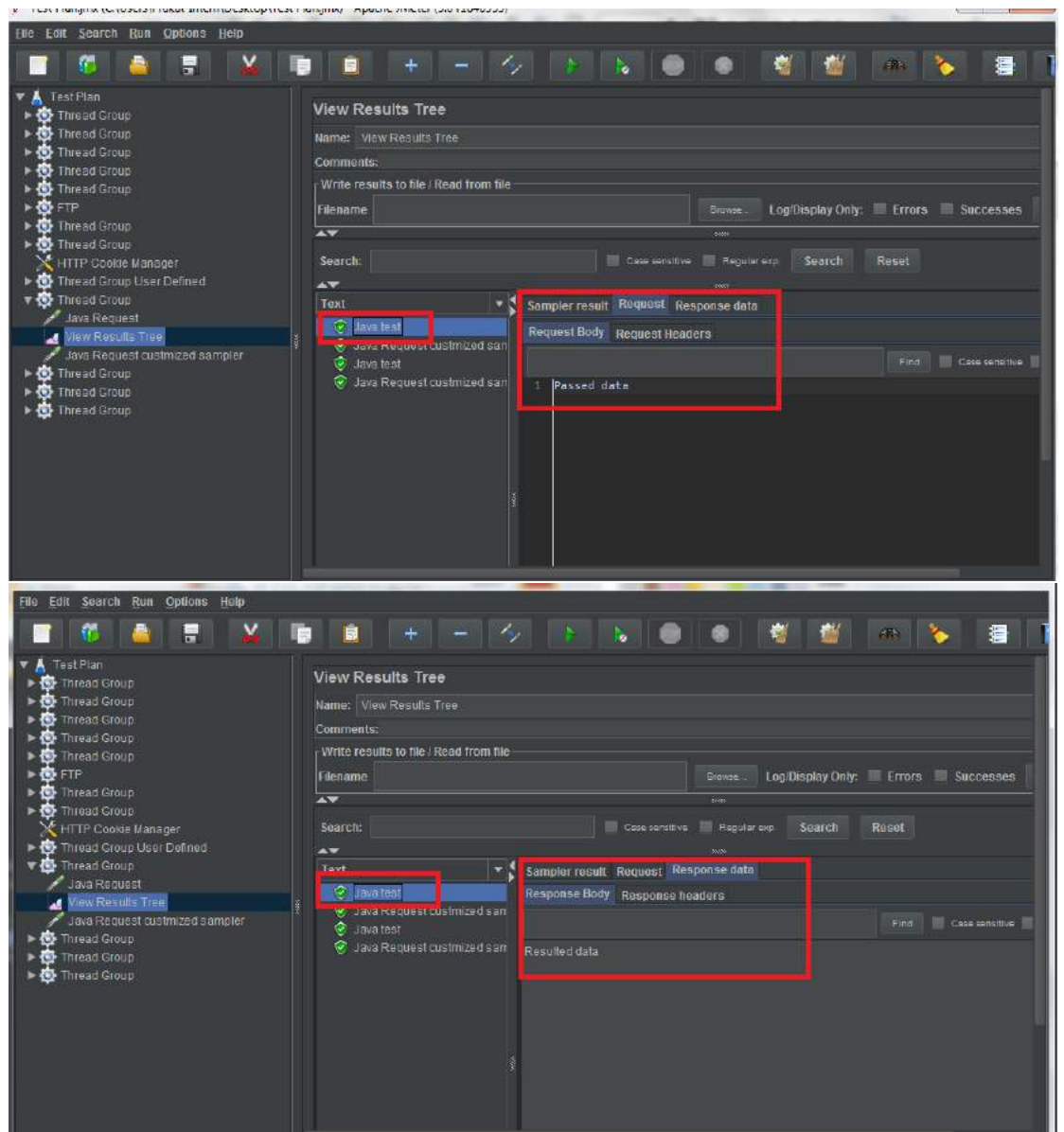


- i. **Classname:** Select “org.apache.jmeter.protocol.java.test.JavaTest”
 - ii. **Label:** The label to use, if provided, overrides Name. Ex: Java test.
 - iii. **ResponseCode:** If provided, sets the Sample Result Response Code, ex: 200.
 - iv. **ResponseMessage:** If provided, sets the Sample Result Response Message. Ex: Success.
 - v. **Status:** If provided, sets the Sample Result Status. If this equals "OK" (ignoring case), then the status is set to success, otherwise the sample is marked as failed.
 - vi. **SamplerData:** If provided, sets the Sample Result with SamplerData, ex: Passes data.
 - vii. **ResultData:** If provided, sets the Sample Result with ResultData, ex: Resulted data.
- Adding View Results Tree:
 1. Right click on Thread Group.
 2. Click on Add -> Listener -> View Results Tree.



3. Run the Thread Group and open the View Results Tree to see the output.





Step 1.28.5: Adding custom Java request

- JMeter also provides a way to create custom samplers.
- The goal is to implement a custom sampler project that will upload a test as a simple function.
- Create a Maven project for custom Java request sampler
 1. Open Eclipse.
 2. Create a Maven project (JMeterSampler).
 3. Add dependencies in the pom.xml file.
 4. the dependencies look like:

```

<!-- https://mvnrepository.com/artifact/org.slf4j/slf4j-api -->
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>1.7.28</version>
</dependency>

<!--
https://mvnrepository.com/artifact/org.apache.jmeter/ApacheJMeter_java -
-->
<dependency>
    <groupId>org.apache.jmeter</groupId>
    <artifactId>ApacheJMeter_java</artifactId>
    <version>5.1.1</version>
</dependency>

<!--
https://mvnrepository.com/artifact/org.apache.jmeter/ApacheJMeter_core -
-->
<dependency>
    <groupId>org.apache.jmeter</groupId>
    <artifactId>ApacheJMeter_core</artifactId>
    <version>5.1.1</version>
</dependency>

```

5. Create a class that extends “AbstractJavaSamplerClient” for which the code is shown below.
6. The CustomSampler class extends the AbstractJavaSamplerClient class and invokes the testFunction.
By overriding the getDefaultParameters function, we can apply default parameters that can be used with the request.

```

package jmeter;

import java.io.Serializable;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

```

```

import org.apache.jmeter.config.Arguments;

import org.apache.jmeter.protocol.java.sampler.AbstractJavaSamplerClient;

import org.apache.jmeter.protocol.java.sampler.JavaSamplerContext;

import org.apache.jmeter.samplers.SampleResult;


public class CustomSampler extends
AbstractJavaSamplerClient implements Serializable {

    private static final String METHOD_TAG = "method";

    private static final String ARG1_TAG = "arg1";

    private static final String ARG2_TAG = "arg2";

    private static final Logger LOGGER =
LoggerFactory.getLogger(CustomSampler.class);

    @Override

    public Arguments getDefaultParameters() {

        Arguments defaultParameters = new Arguments();

        defaultParameters.addArgument(METHOD_TAG, "test");

        defaultParameters.addArgument(ARG1_TAG, "arg1");

        defaultParameters.addArgument(ARG2_TAG, "arg2");

        return defaultParameters;

    }


    public SampleResult runTest
(JavaSamplerContext javaSamplerContext) {

        String method = javaSamplerContext.getParameter(METHOD_TAG);

        String arg1 = javaSamplerContext.getParameter(ARG1_TAG);

        String arg2 = javaSamplerContext.getParameter(ARG2_TAG);

        FunctionalityForSampling functionalityForSampling
= new FunctionalityForSampling();

        SampleResult sampleResult = new SampleResult();

        sampleResult.sampleStart();

        try {

```

```

String message =
    functionalityForSampling.testFunction(arg1,arg2);
sampleResult.sampleEnd();
sampleResult.setSuccessful(Boolean.TRUE);
sampleResult.setResponseCodeOK();
sampleResult.setResponseMessage(message);
} catch (Exception e) {
    LOGGER.error
    ("Request was not successfully processed",e);
    sampleResult.sampleEnd();
    sampleResult.setResponseMessage(e.getMessage());
    sampleResult.setSuccessful(Boolean.FALSE);
}
return sampleResult;
}
}

```

7. Create another class, ex: "FunctionalityForSampling", for which the code is shown below.
8. A simple function will be called by the sampler to compare passed arguments.

```

package jmeter;

public class FunctionalityForSampling {

    public String testFunction(String argument1,
String argument2) throws Exception {

        if (argument1.equals(argument2)) {

            throw new Exception();

        }

        else return argument1+argument2;

    }

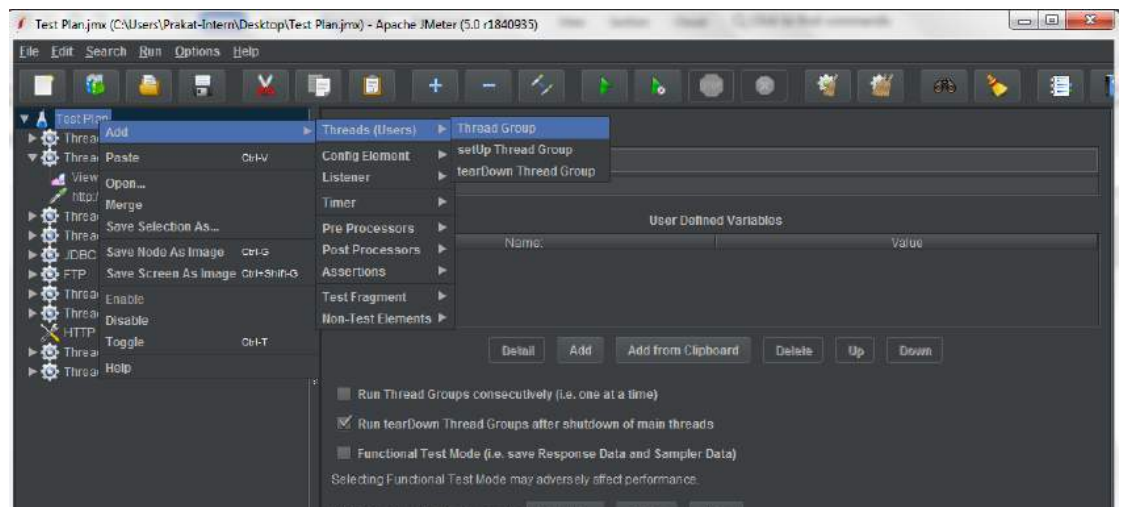
}

```

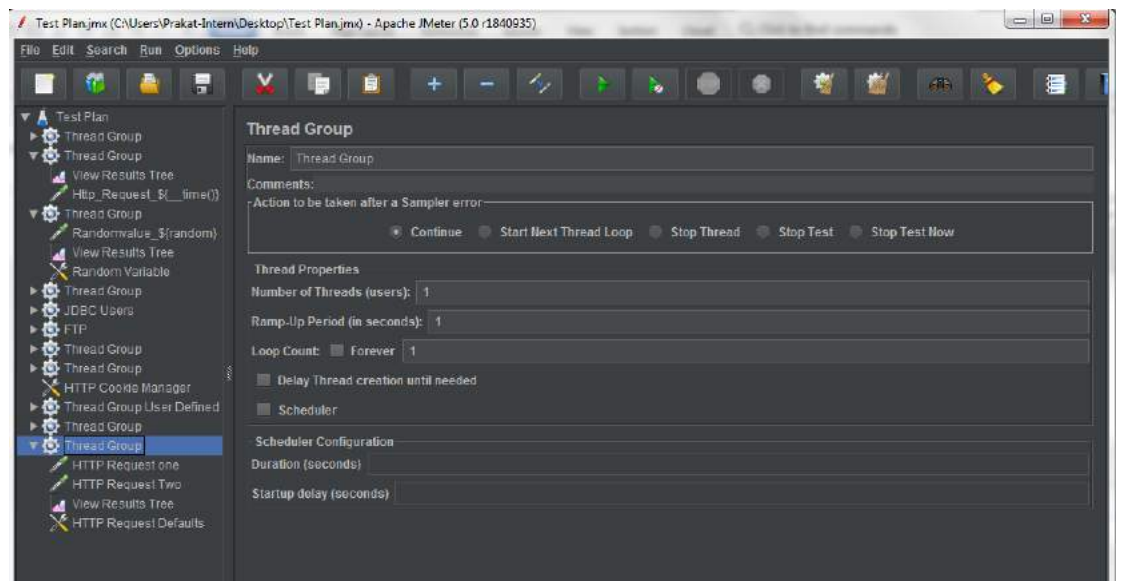
9. Maven jar files are already present in your practice labs in the /opt/ directory. Refer to the lab guide for Phase 2 for more information.

10. Add or copy the maven jar file from the directory mentioned above to the libs directory of the JMeter.

- Adding users in the thread group:
 1. Right click on Test Plan.
 2. Click on Add -> Threads -> Thread Group.

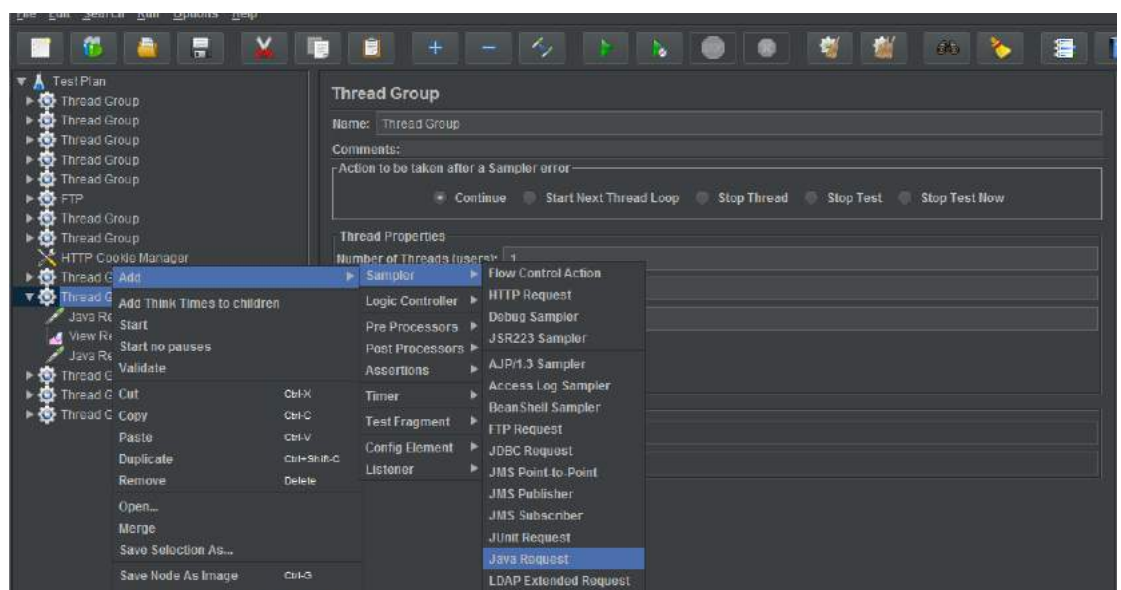


3. Set Number of Threads to 1.
4. Set Ramp-Up Period to 1.
5. Set Loop Count to 1.

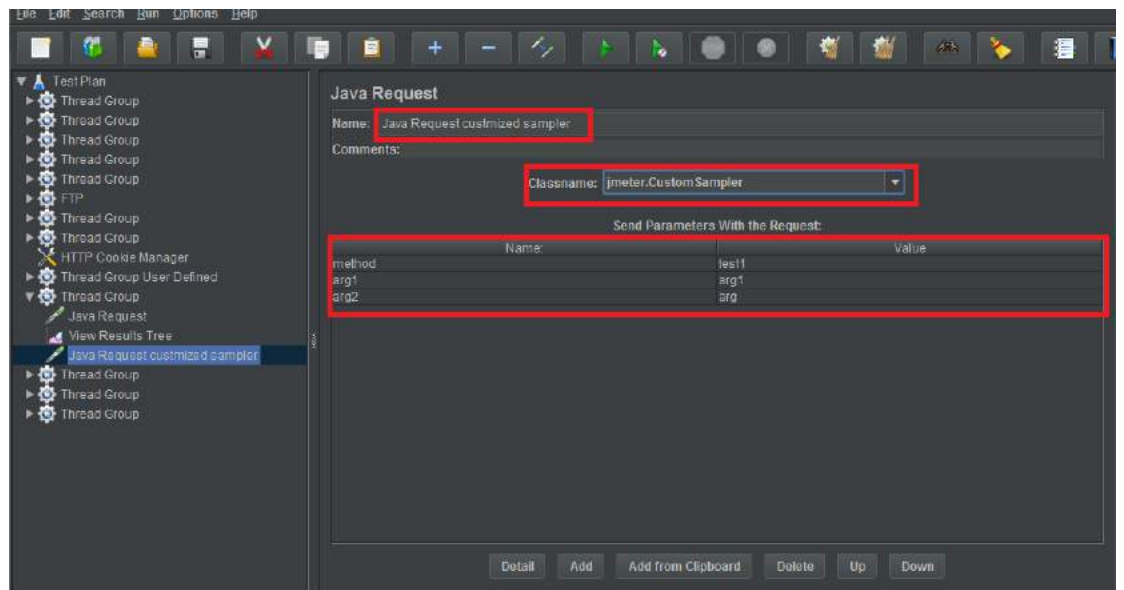


- Adding Java request:

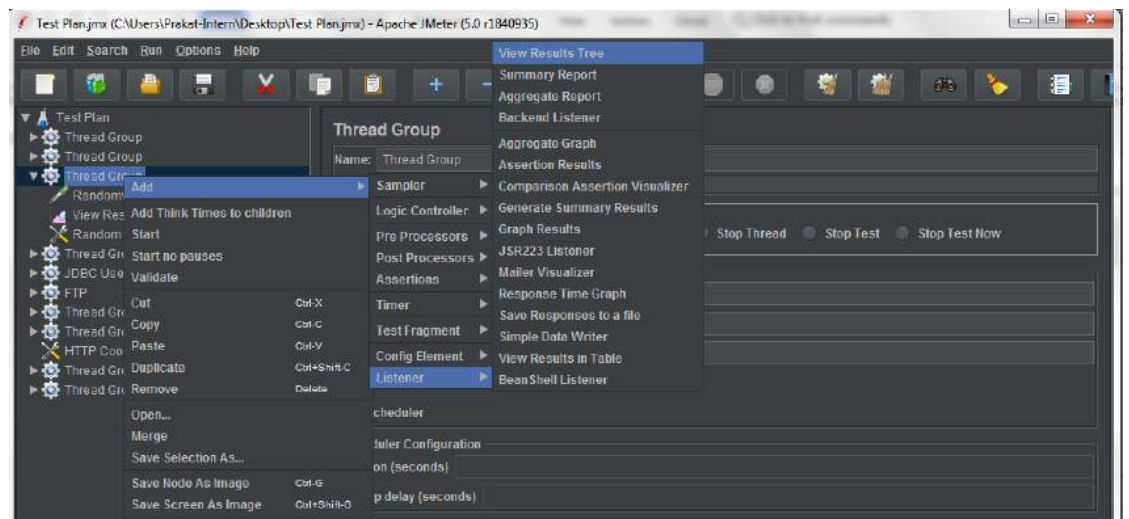
1. Right click on Thread Group.
2. Click on Add -> Sampler -> Java Request.



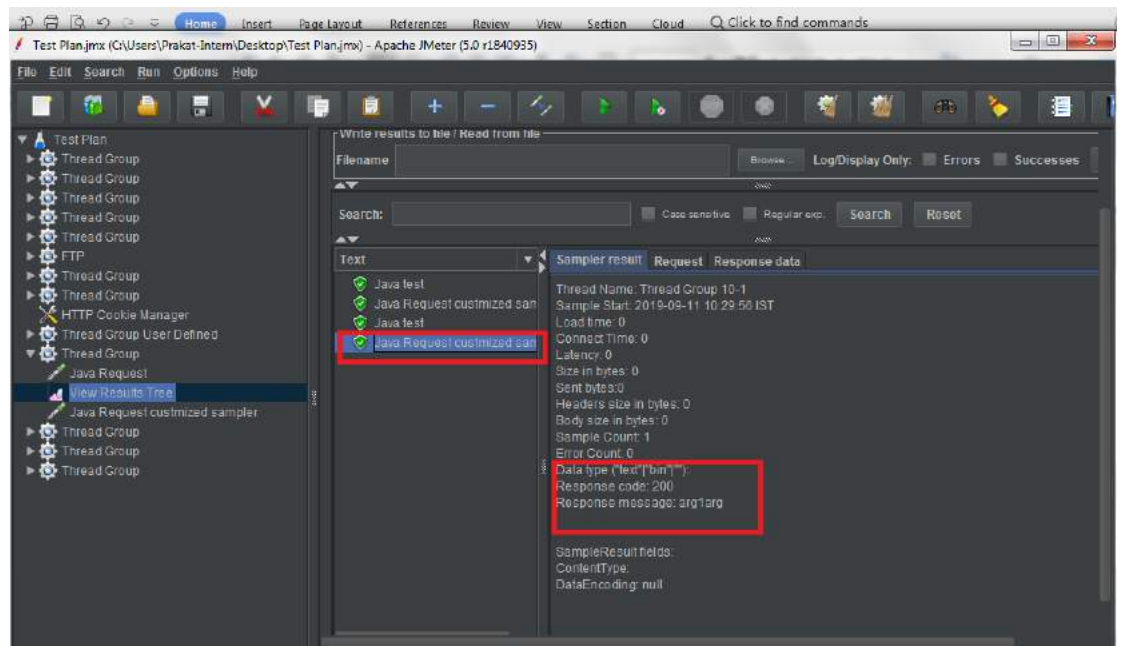
3. Enter Java Request fields as follow:



4. Name: Java Request Customized sample
 5. Change ClassName to "jmeter.CustomSampler"
- Adding View Results Tree:
 1. Right click on Thread Group.
 2. Click on Add -> Listener -> View Results Tree.

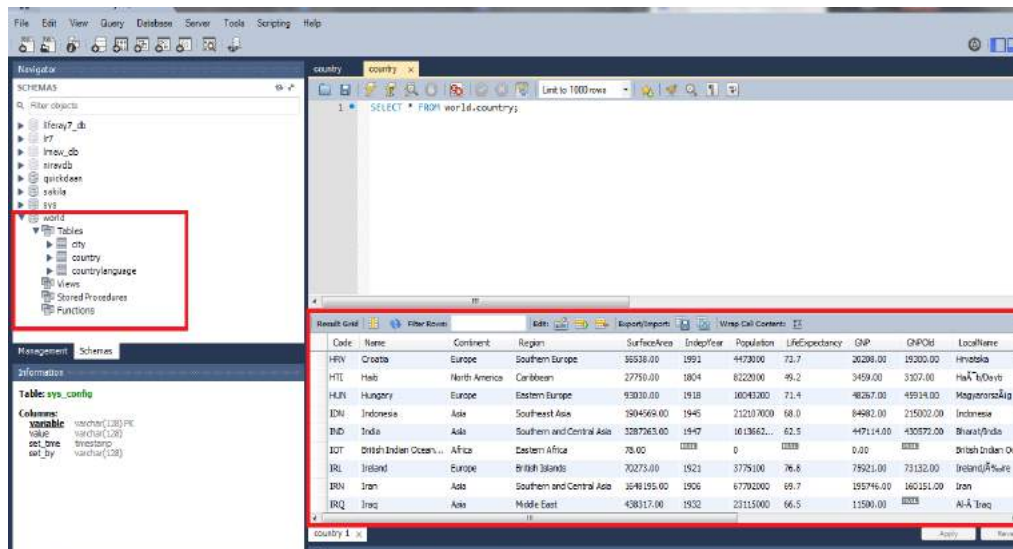


3. Run the Thread Group and open the View Results Tree to see the output.



Step 1.28.6: Establishing JDBC connection

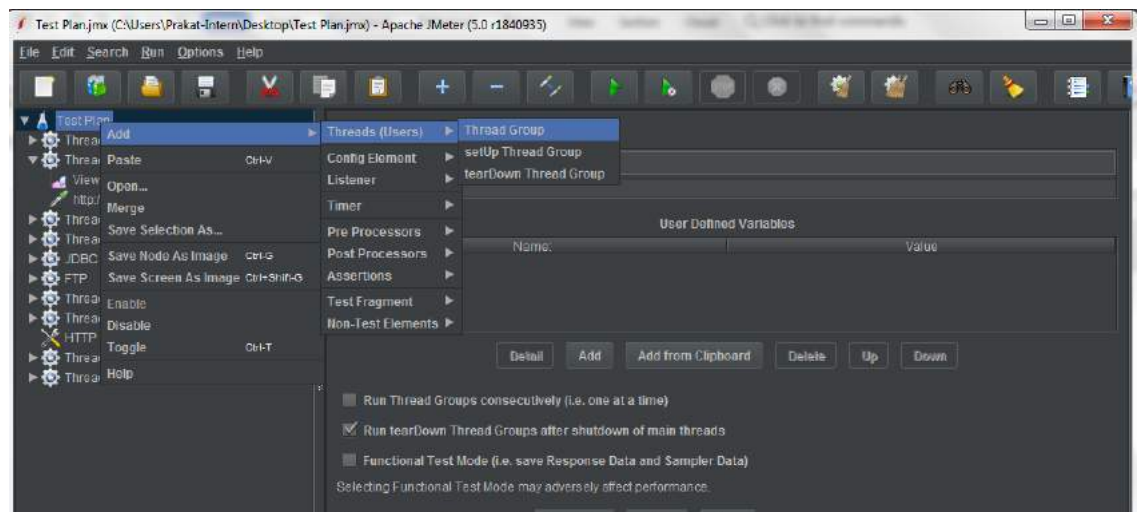
- MySQL is already installed in your practice lab. Refer to the lab guide for Phase 2 for more information.
- mysql-connector-java-5.1.38.jar file is already present in your practice lab in /usr/share/java directory.
- Before you start working with a database using JMeter, you need to do the following:
 1. Create a database called "world".
 2. Create a table called "country".
 3. Insert country details into the table "country".



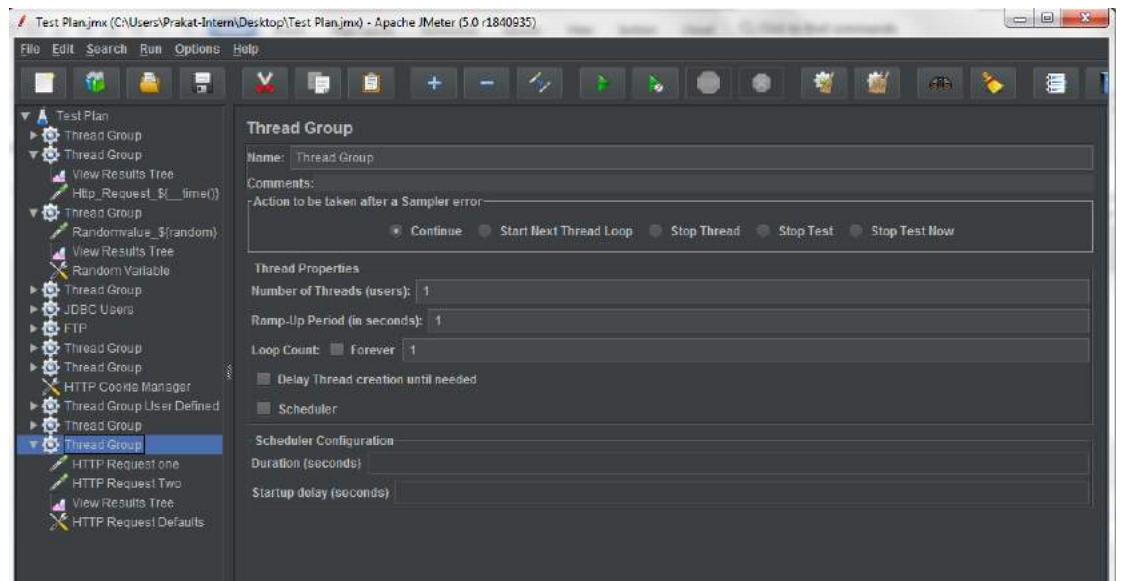
4. Copy the “mysql-connector-java-5.1.38.jar” into JMeter lib folder.

- Adding users in a thread group:

1. Right click on Test Plan.
2. Click on Add -> Threads -> Thread Group.

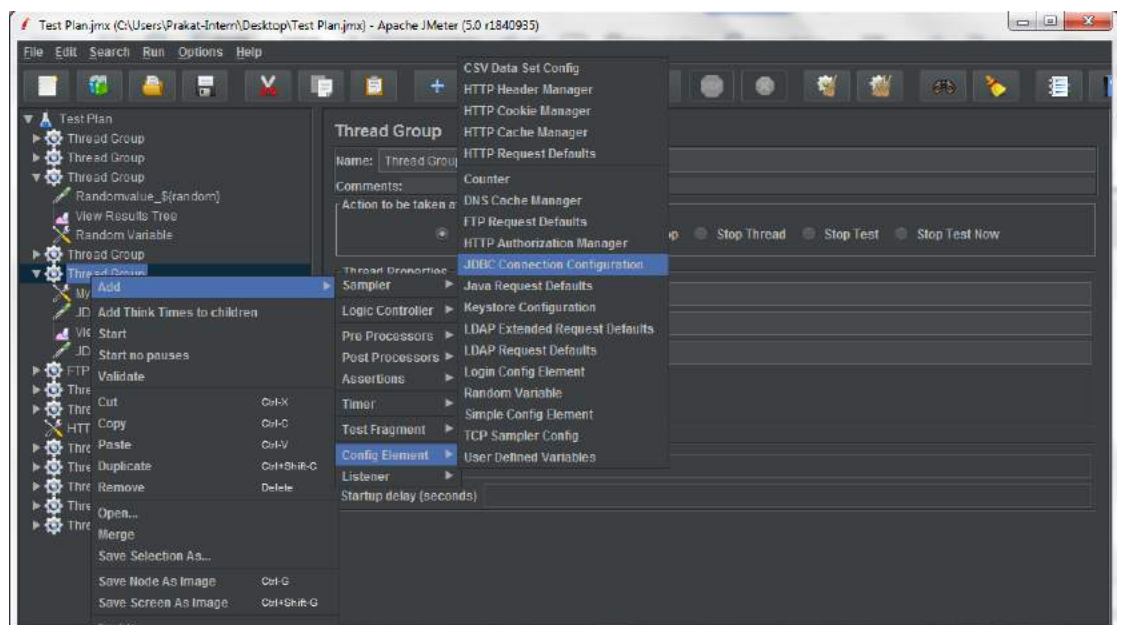


3. Set Number of Threads to 1.
4. Set Ramp-Up Period to 1.
5. Set Loop Count to 1.



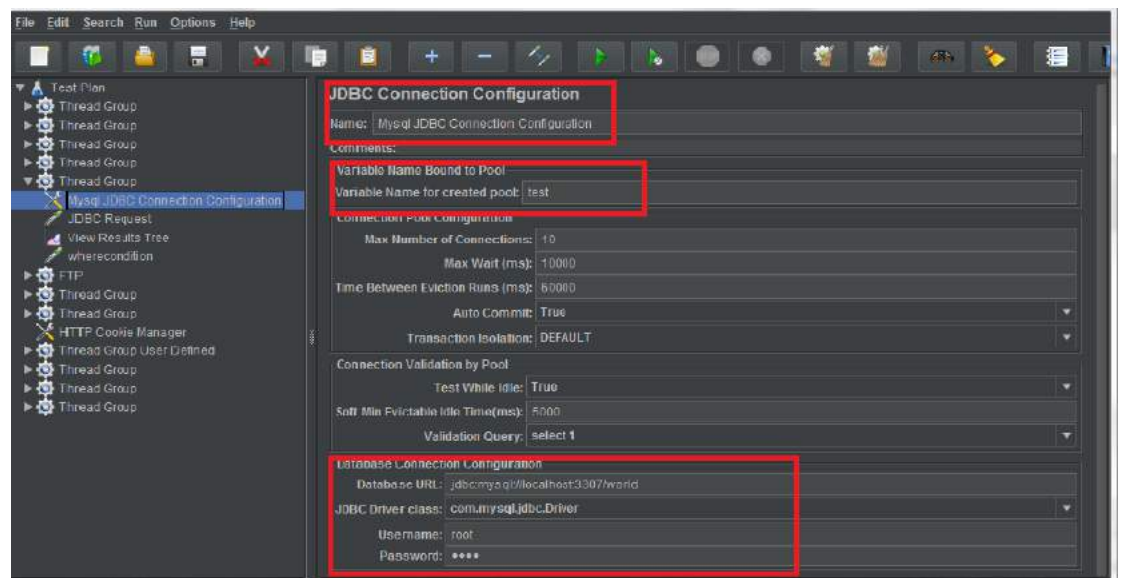
- Adding JDBC connection configuration:

1. Right click on Thread Group.
2. Click on Add -> Config Element -> JDBC Connection Configuration.

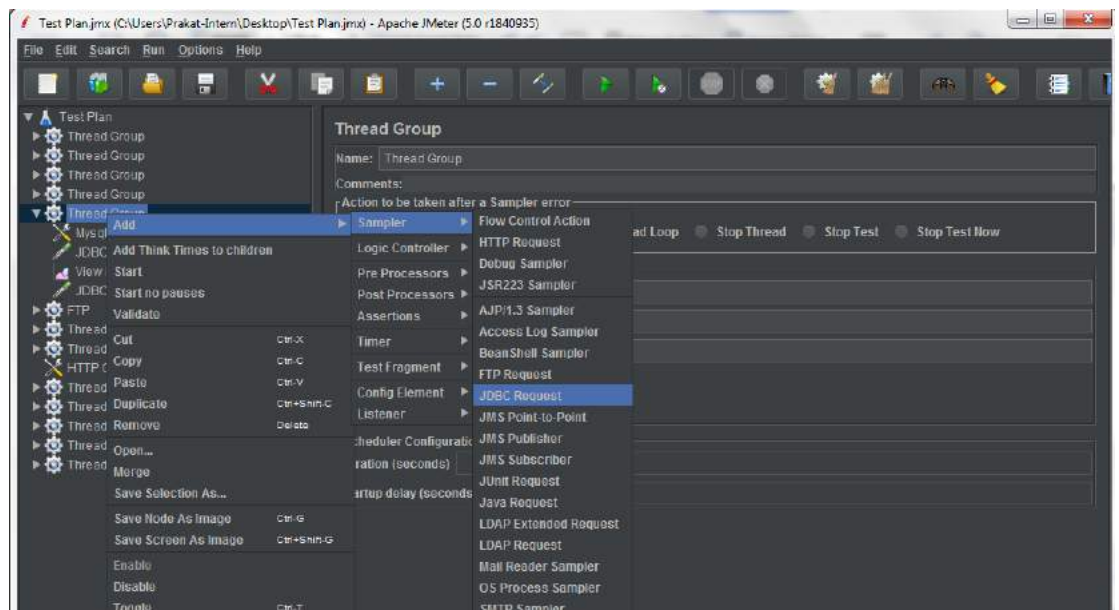


3. Set up the following fields (Here, we are using a MySQL database named 'world'):

- I. Rename JDBC Connection Configuration to “Mysql JDBC Connection Configuration”.
 - II. **Variable Name Bound to Pool:** This needs to uniquely identify the configuration. It is used by the JDBC Sampler to identify the configuration to be used, ex: “test”.
 - III. **Database URL:** jdbc:mysql://localhost:3307/world
 - IV. **JDBC Driver Class:** com.mysql.jdbc.Driver
 - V. **Username:** The username of database(here: **root**)
 - VI. **Password:** The password for the username(here: **root**)
4. The other fields on the screen can be left as the defaults.

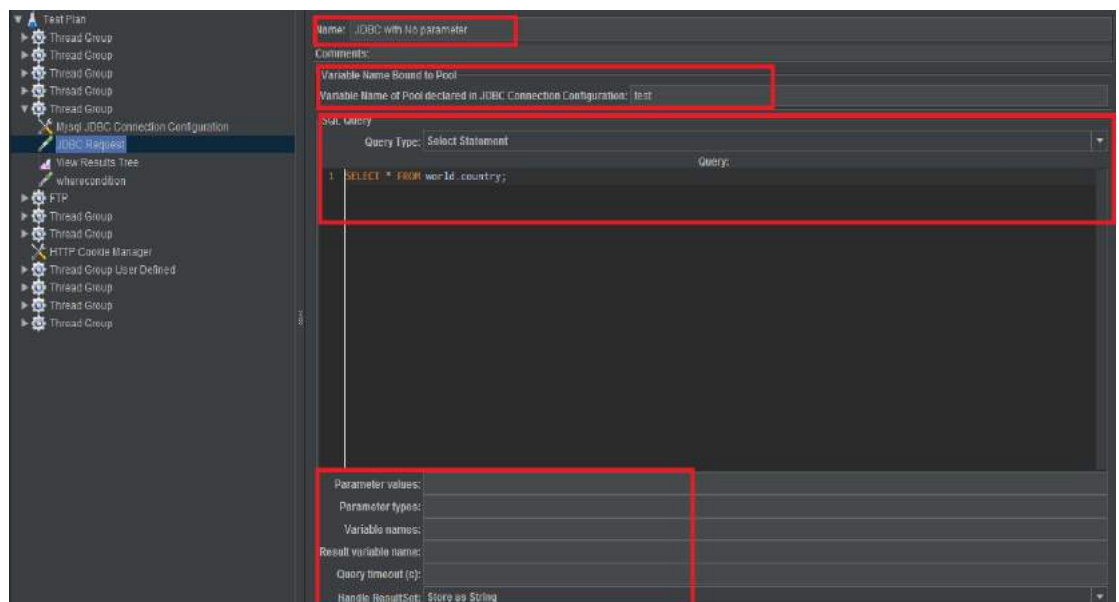


- Adding JDBC request:
 1. Right click on Thread Group.
 2. Click on Add -> Sampler -> JDBC Request.



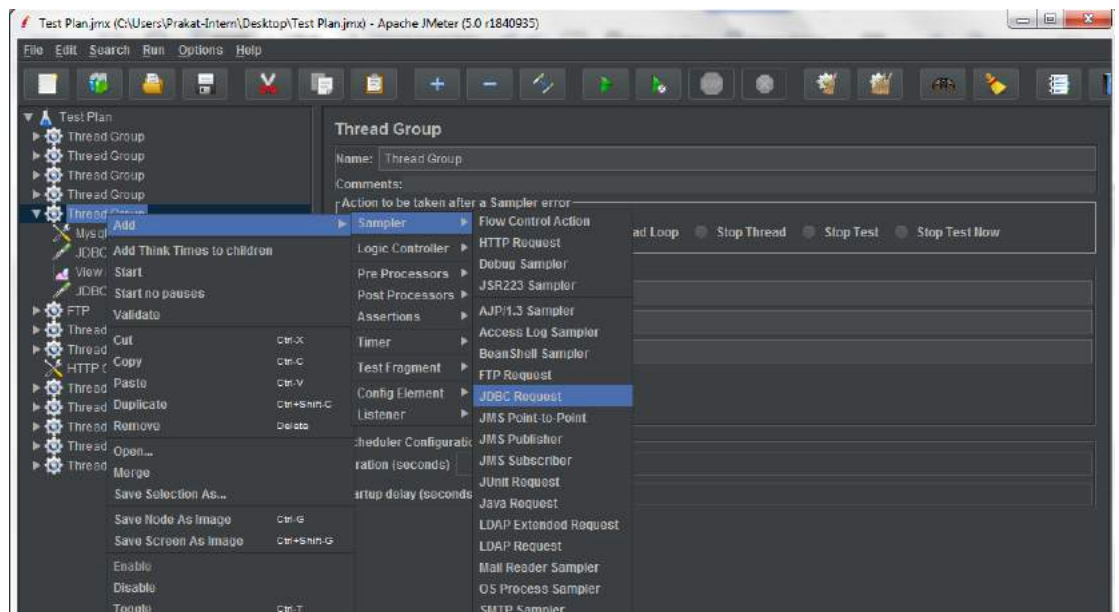
3. Enter the required fields in JDBC Request.
4. Change the Name to 'JDBC with No parameter'.
5. Enter the Pool Name: 'test' (same as in the configuration element).
6. Select Query Type as "Select Statement".
7. Enter the SQL Query String field. Ex: "SELECT * FROM world.country;"

Note: We are not passing any parameters to the SQL query.



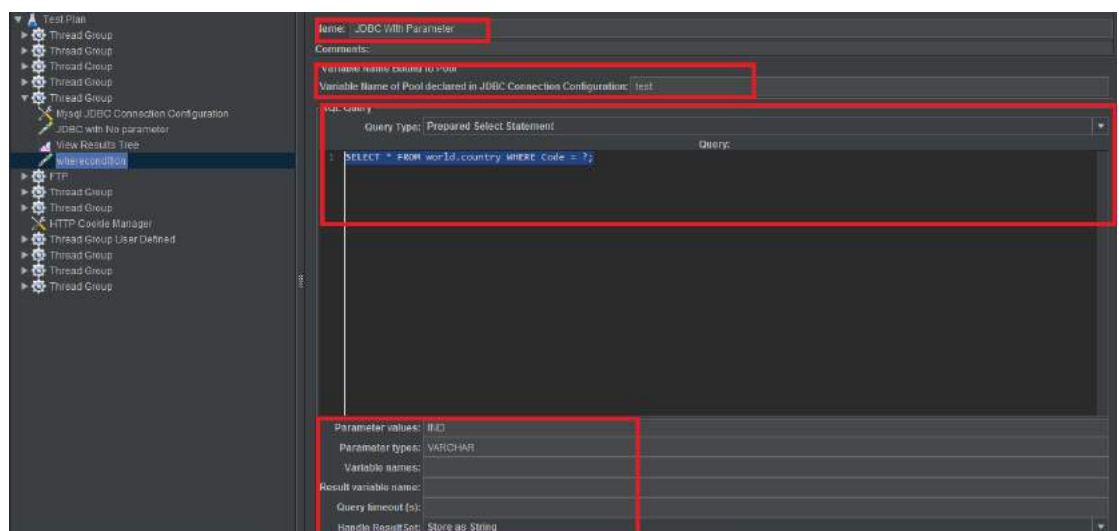
- Adding JDBC request:
 1. Right click on Thread Group.

2. Click on Add -> Sampler -> JDBC Request.



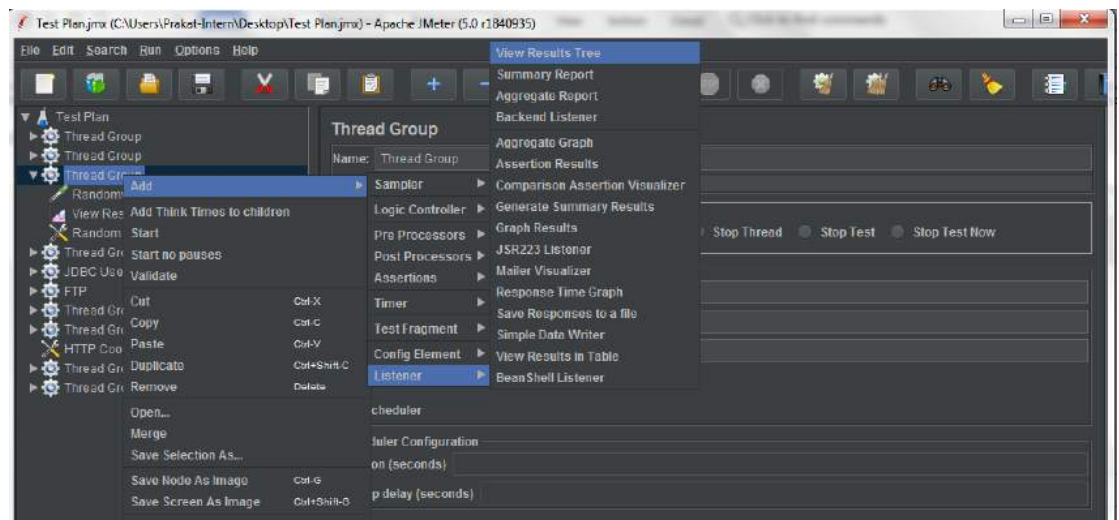
3. Enter the required fields in JDBC Request.
4. Change the Name to 'JDBC with parameter'.
5. Enter the Pool Name: 'test' (same as in the configuration element).
6. Select Query Type as "Prepared Select Statement".
7. Enter the SQL Query String field Ex: "SELECT * FROM world.country WHERE Code = ?;".
8. Enter the Parameter values field with 'IND' value.
9. Enter the Parameter types with 'VARCHAR'.

Note: We are passing value "IND" to SQL Query "WHERE Code = ?" using parameter values.

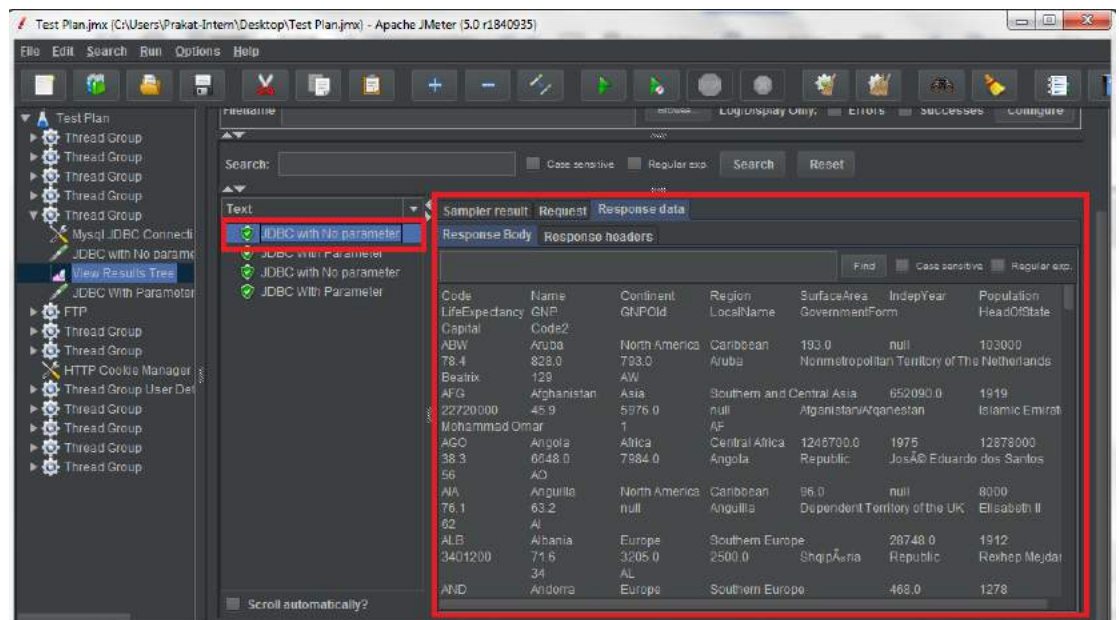


- Adding View Results Tree:

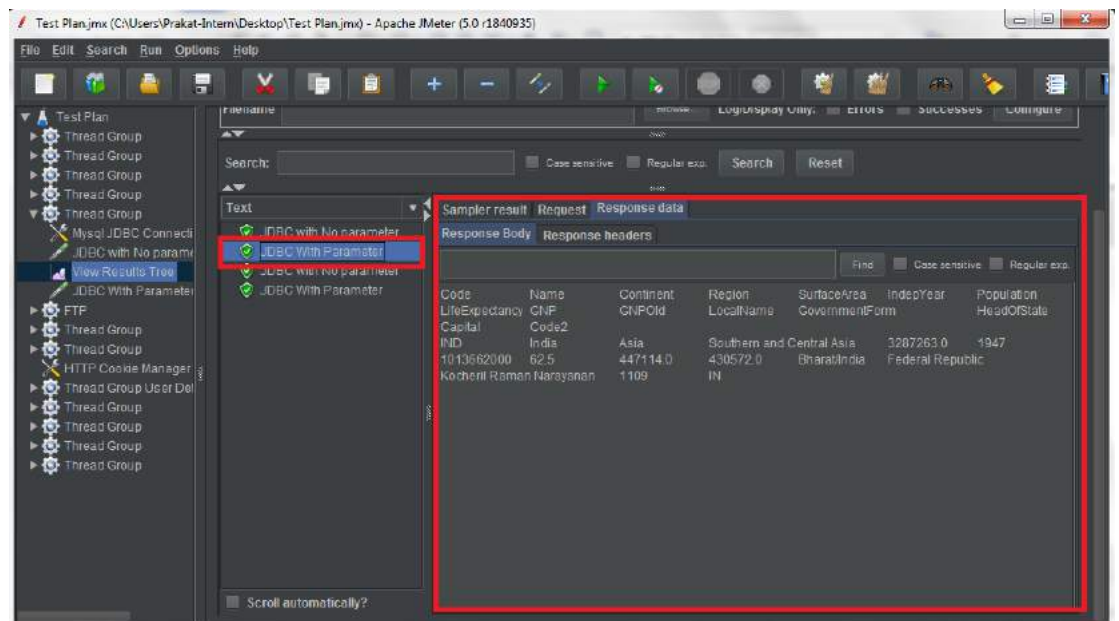
1. Right click on Thread Group.
2. Click on Add -> Listener -> View Results Tree.



3. Run the Thread Group and open the View Results Tree to see the output.
4. The output for “Select * from world.country;” query results in fetching all the details present in the country table.



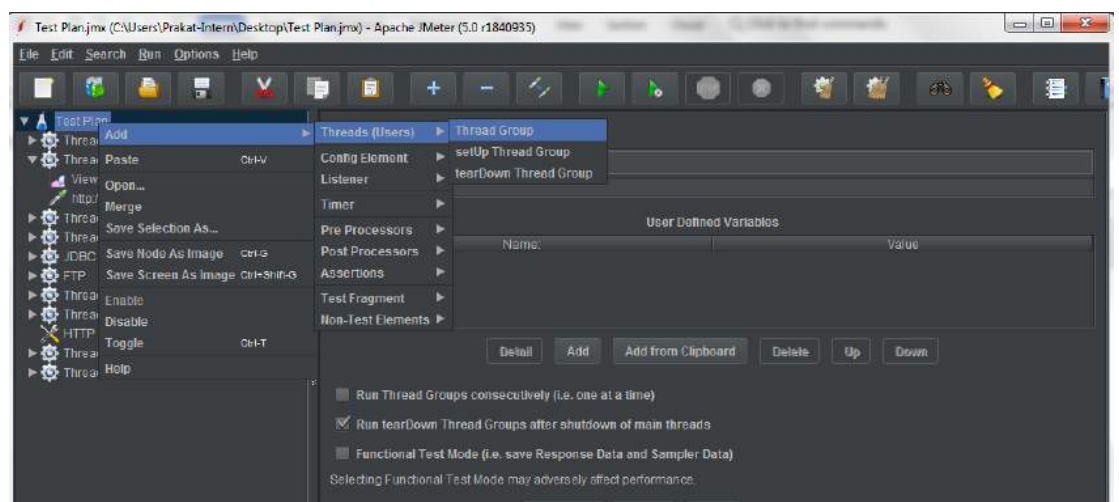
5. Output for “SELECT * FROM world.country WHERE Code = "IND";” result in fetching only one row of details for country IND present in the country table.



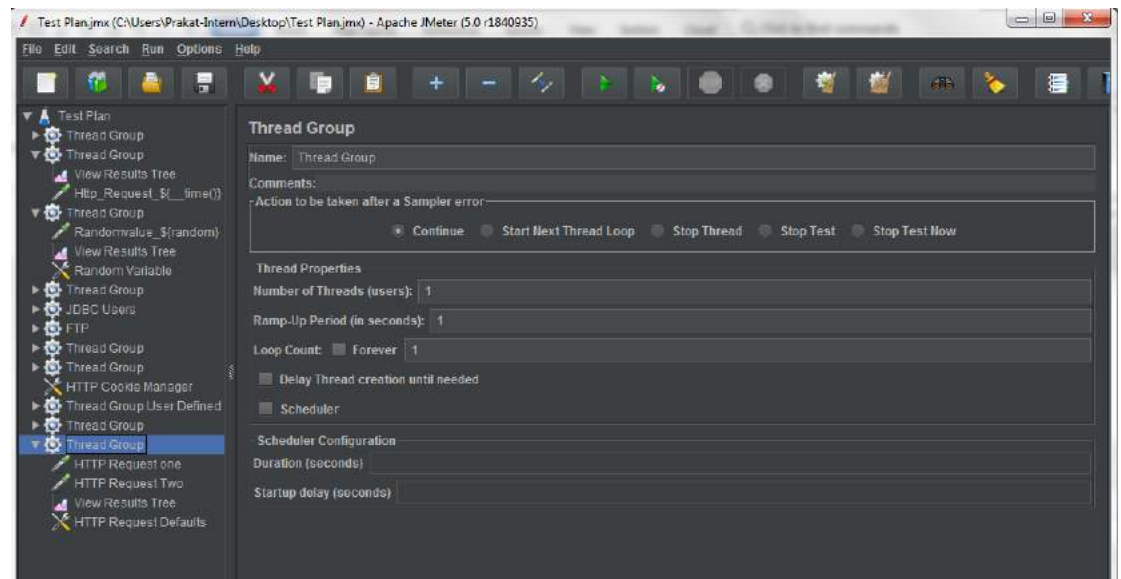
Step 1.28.7: Using User-Defined Variable

- Consider you have a number of HTTP requests that have different parameters set but the same URLs. But you know that in the future you may want to edit the URL. So, in such cases, you have to edit EVERY URL. But if you have hundreds of them, this is not a good idea.
- In this case, User Defined Variables (UDV) will be a better solution. This functionality allows changing parameters in multiple places of the Test-plan. Adding users in the thread group.

- Right click on Test Plan.
- Click on Add -> Threads -> Thread Group.

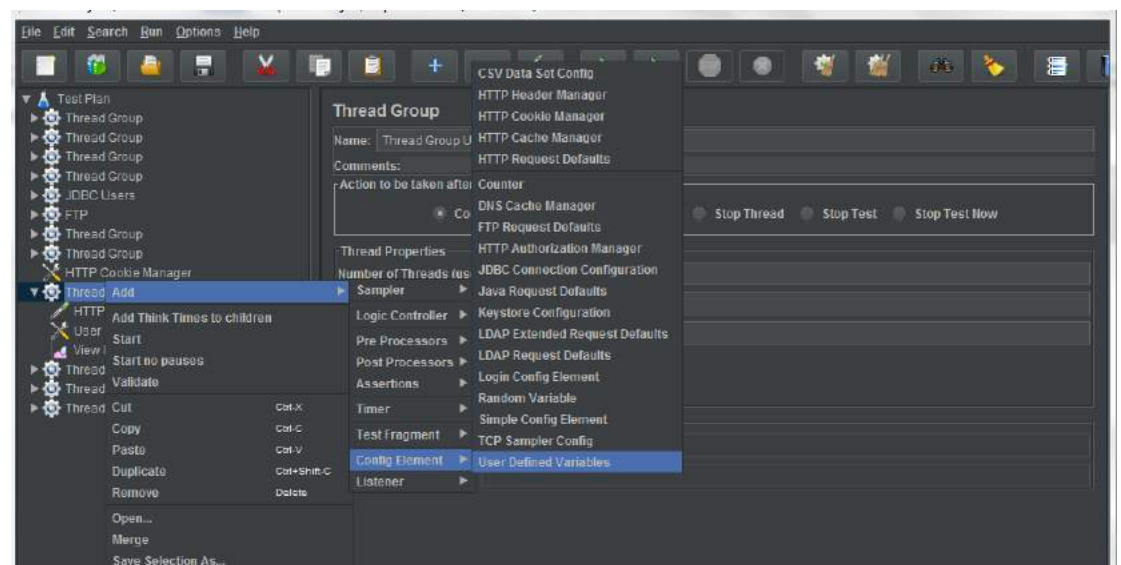


3. Set Number of Threads to 1.
4. Set Ramp-Up Period to 1.
5. Set Loop Count to 1.

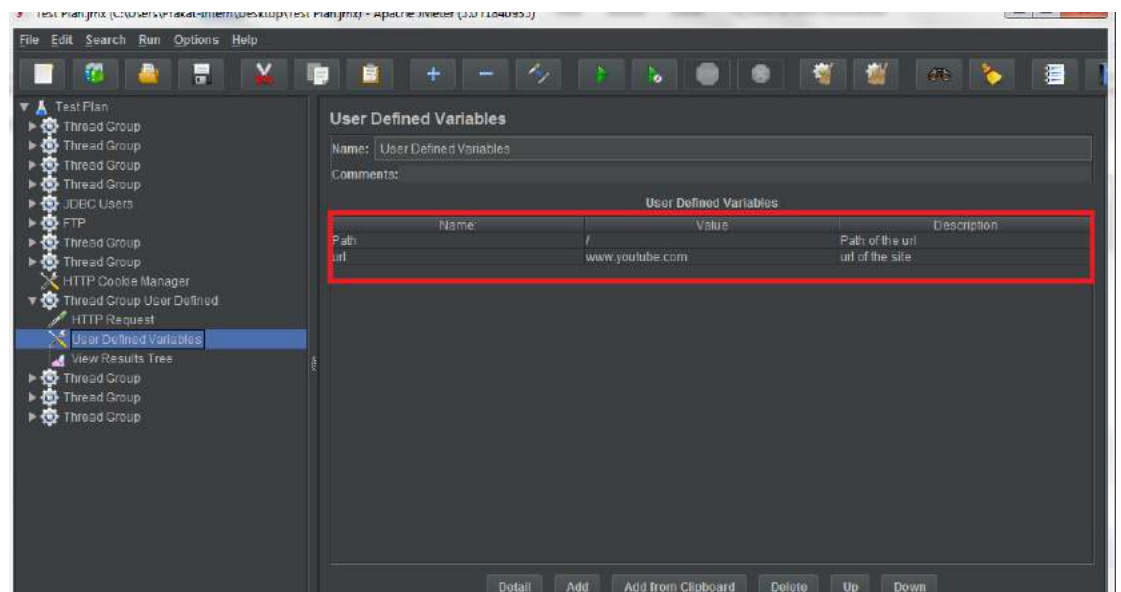


- Adding User Defined Variable:

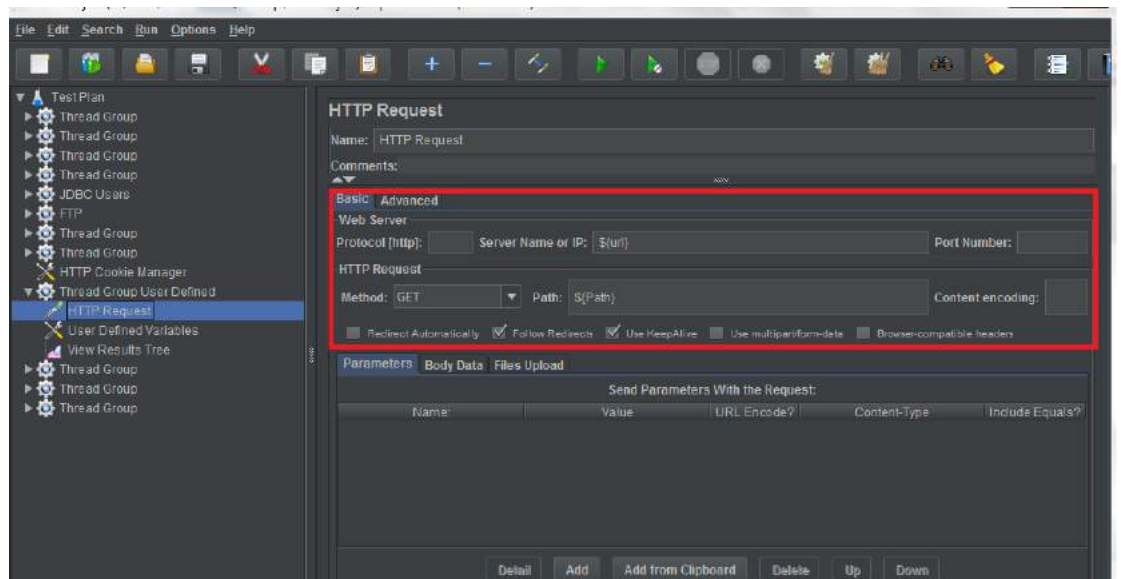
1. Right click on Thread Group.
2. Click on Add -> Config Element -> User Defined Variables.



3. Add User Defined Variables
 - I. **Name:** Reference variable name, ex: Path and URL (in this example).
 - II. **Value:** Values assigned for the reference variables Path and URL
Ex: "/" and "www.youtube.com"
 - III. **Description:** Description

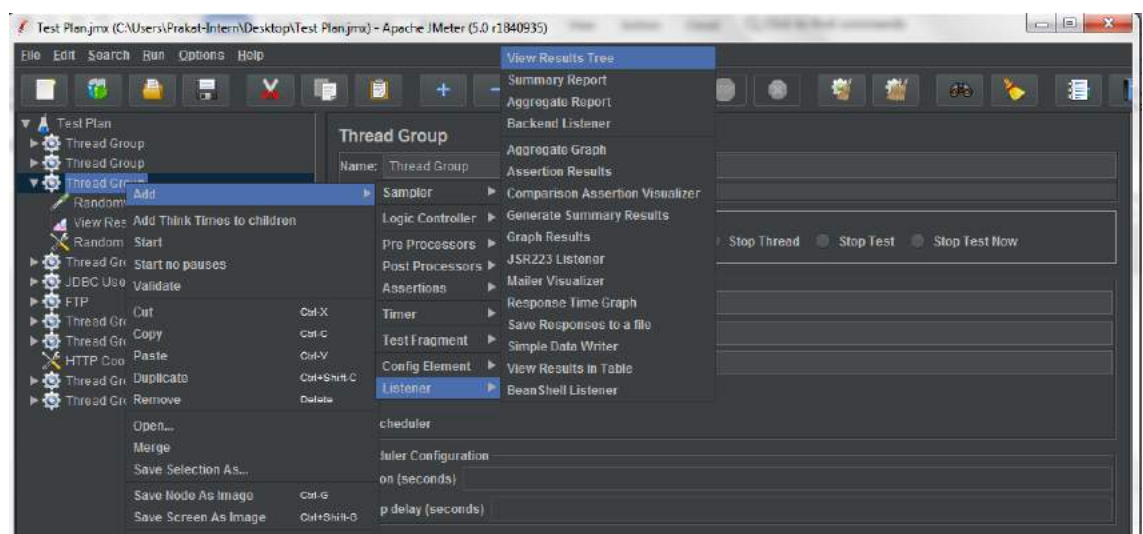


- Adding HTTP request:
 1. Right click on Thread Group.
 2. Click on Add -> Sampler -> HTTP Request.
 3. Give Server Name as `${url}`, where url is the user defined variable.
 4. Give Path `${Path}`, where Path is a user defined variable.

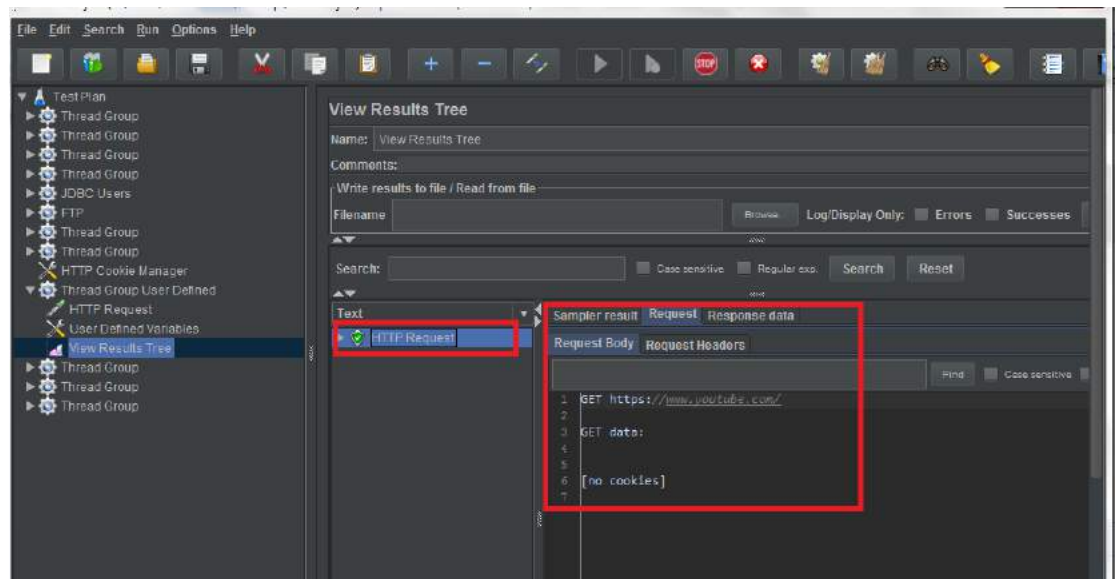


- Adding View Results Tree:

1. Right click on Thread Group.
2. Click on Add -> Listener -> View Results Tree.



3. Run Thread Group and Open View Results Tree to see the output.



Step 1.28.6: Pushing the code to GitHub repositories

Open your command prompt and navigate to the folder where you have created your files.

```
cd <folder path>
```

Initialize your repository using the following command:

```
git init
```

Add all the files to your git repository using the following command:

```
git add .
```

Commit the changes using the following command:

```
git commit . -m "Changes have been committed."
```

Push the files to the folder you initially created using the following command:

```
git push -u origin master
```