References

[1] http://maven.apache.org/maven-1.x/maven.pdf

[2] http://jakarta.apache.org/jmeter/

[3] http://www.adventnet.com/products/qengine/index.html

[4] http://java.sun.com

[5] http://java.sun.com/products/archive/j2se/5.0_06/index.html

[6] http://www.badboy.com.au

Appendix A: CODE SEGMENTS

XSL Code sample from report

```
<xsl:template name="pagelist">
    <table width="100%" border="0" cellpadding="0" cellspacing="0">
        <tr>
            <td align="left" width="7" height="7" colspan="2"><img
src="images/top_left.gif" alt=""></img></td>
            <td background="images/top_line.gif"></td>
            <td align="right" width="7" height="7" colspan="2"><img
src="images/top_right.gif" alt=""></img></td>
        </tr>
        <tr>
            <td bgcolor="#ABABAB" width="1"></td>
            <td width="6"></td>
            <td align="center">
                <table border="0" cellpadding="0" cellspacing="1" width="95%"
bgcolor="#FFFFFF">
                    <tr>
                        <th align="left" valign="top" height="25"><img
src="images/testcases.png" alt="Test Scenarios"></img></th>
                    </tr>
                    <tr valign="top" bgcolor="#999999">
                        <th>TEST Case</th>
                        <th>Status</th>
                        <th>Tests</th>
                        <th>Failures</th>
                        <th>Success Rate</th>
                        <th>Total Time</th>
                        <th>Average Time</th>
                        <th>Details</th>
                    </tr>

                    <xsl:for-each select="/testResults/*[not(@tn = preceding::*/@tn)]">
                        <xsl:variable name="testcase_pre" select="@tn" />
                        <xsl:variable name="testcase" select="substring-before($testcase_pre, '
')" />
                        <xsl:variable name="count" select="count(../*[@tn = current()/@tn])"
/>
                        <xsl:variable name="failureCount" select="count(../*[@tn =
current()/@tn][attribute::s='false'])" />
                        <xsl:variable name="successCount" select="count(../*[@tn =
current()/@tn][attribute::s='true'])" />
                        <xsl:variable name="successPercent" select="$successCount div
$count" />
```

```
                        <xsl:variable name="totalTime" select="sum(../*[@tn =
current()/@tn]/@t)" />
                        <xsl:variable name="averageTime" select="$totalTime div $count" />



                        <tr bgcolor="#E8E8E8">
                            <xsl:attribute name="class">
                                <xsl:choose>
                                    <xsl:when test="$failureCount &gt; 0">Failure</xsl:when>
                                </xsl:choose>
                            </xsl:attribute>
                            <th>
                                 <xsl:value-of select="$testcase" />
                            </th>
                            <th>
                                <xsl:if test="$failureCount > 0">Failed</xsl:if>
                                <xsl:if test="0 >= $failureCount">Passed</xsl:if>
                            </th>
                            <th><xsl:value-of select="$count" /></th>
                            <th><xsl:value-of select="$failureCount" /></th>
                            <th>
                                <xsl:call-template name="display-percent">
                                    <xsl:with-param name="value" select="$successPercent" />
                                </xsl:call-template>
                            </th>
                            <th>
                                <xsl:call-template name="display-time">
                                    <xsl:with-param name="value" select="$totalTime" />
                                </xsl:call-template>
                            </th>
                            <th>
                                <xsl:call-template name="display-time">
                                    <xsl:with-param name="value" select="$averageTime" />
                                </xsl:call-template>
                            </th>
                            <th>
                                <xsl:if test="$failureCount > 0">
                                    <a href="">
                                     <xsl:attribute
name="href"><xsl:text/>javascript:change('page_details_<xsl:value-of select="position()"
/>')</xsl:attribute>
                                        <img src="images/expand_f.png"
alt="expand/collapse"><xsl:attribute name="id"><xsl:text/>page_details_<xsl:value-of
select="position()" />_image</xsl:attribute></img>
                                        </a>
                                </xsl:if>
                                <xsl:if test="0 >= $failureCount">
                                    <a href="">
                                     <xsl:attribute
name="href"><xsl:text/>javascript:change('page_details_<xsl:value-of select="position()"
/>')</xsl:attribute>
                                        <img src="images/expand_p.png"
alt="expand/collapse"><xsl:attribute name="id"><xsl:text/>page_details_<xsl:value-of
select="position()" />_image</xsl:attribute></img>
                                        </a>
                                </xsl:if>
                            </th>
                        </tr>

                        <!--  Test Case Details goes here  -->

                         <tr class="page_details" bgcolor="#FFFFFF">
                                 <xsl:attribute name="id"><xsl:text/>page_details_<xsl:value-of
select="position()" /></xsl:attribute>
                                 <td colspan="8" bgcolor="#FFFFFF">
                                    <div align="center">
                                    <br/>
                               <!--b>Details for Page "<xsl:value-of select="$testcase" />"</b--
>
                               <table border="0"  cellpadding="0" cellspacing="1"
width="95%" bgcolor="#FFFFFF">
                               <tr bgcolor="#63BB8A">
                                 <th>Request URL</th>
                                 <th>Time (milliseconds)</th>
                                 <th>Bytes</th>
                                 <th>Success</th>
                               </tr>

                               <xsl:for-each select="../*[@tn = $testcase_pre]">

                                 <tr bgcolor="#E0EFE7">
                                    <td><xsl:value-of select="@lb" /></td>
                                 <td align="right"><xsl:value-of select="@t" /></td>
                                    <td align="right"><xsl:value-of select="@by" /></td>
                                    <td align="center"><xsl:value-of select="@s" /></td>
                                 </tr>
                               </xsl:for-each>

                               </table>
                             <br/>
                             </div>
```

```
                                </td>
                              </tr>
                         </xsl:for-each>
                      </table>
                      <br/>
                  </td>
                  <td width="6"></td>
                  <td bgcolor="#ABABAB" width="1"></td>
              </tr>
              <tr>
                  <td align="left" width="7" height="7" colspan="2"><img
src="images/bot_left.gif" alt=""></img></td>
                  <td background="images/bot_line.gif"></td>
                  <td align="right" width="7" height="7" colspan="2"><img
src="images/bot_right.gif" alt=""></img></td>
              </tr>
          </table>
</xsl:template>
```

Java Code Segment of the "executesTests" method

```
          public void executeTests(String mappingFilePath,String testMainDir,String
testModules,String testMode,String users,String iterations,String rampUp,String
testCases) {
       try {
          System.out.println("----------------------------------------------------------");
          System.out.println("|  Generating {test.plan} and {userdefined.variables} files");
          System.out.println("----------------------------------------------------------");

          String[]
module=commonUtils.commaSeparatedStringToStringArray(testModules);
          String[] testcase=commonUtils.commaSeparatedStringToStringArray(testCases);

          PropertyEditor.writePropertyToFile(Constants.TESTPLAN_FILENAME,
Constants.TEST_MODE, testMode);
          PropertyEditor.writePropertyToFile(Constants.TESTPLAN_FILENAME,
Constants.NUMBER_USERS, users);
          PropertyEditor.writePropertyToFile(Constants.TESTPLAN_FILENAME,
Constants.NUMBER_ITERATIONS, iterations);
          PropertyEditor.writePropertyToFile(Constants.TESTPLAN_FILENAME,
Constants.RAMPUP_TIME, rampUp);

          XMLParser xmlParser=new XMLParser(mappingFilePath);

          System.out.println("----------------------------------------------------------");
          System.out.println("|  TEST PLAN");
```

```
        System.out.println("--------------------------------------------------------");


if(testMode.equals(Constants.MODE_LOAD)||testMode.equals(Constants.MODE_REGR
ESSION)){
            for(int i=0; i<module.length; i++){
               Iterator
moduleIterator=(xmlParser.getNodeList(Constants.XPATH_MODULE)).iterator();
               while (moduleIterator.hasNext()){

                  Element moduleElement=(Element)moduleIterator.next();
                  String
moduleId=xmlParser.getChildNodeValue(moduleElement,Constants.ELEMENT_MODU
LE_MODULEID);
                  String
moduleStatus=xmlParser.getChildNodeValue(moduleElement,Constants.ELEMENT_MO
DULE_ACTIVE);

                  if(moduleStatus.equals(Constants.STR_TRUE) &&
moduleId.equals(module[i].trim())){

                     System.out.println("| + "+moduleId);

PropertyEditor.writePropertyToFile(Constants.TESTPLAN_FILENAME,
Constants.MODULE_CODE, moduleId);

                     Element
testcasesElement=moduleElement.getChild(Constants.ELEMENT_MODULE_TESTCAS
ES);
                     Iterator
testcaseIterator=(testcasesElement.getChildren(Constants.ELEMENT_TESTCASES_TES
TCASE)).iterator();

                     while(testcaseIterator.hasNext()){

                        Element testcaseElement=(Element)testcaseIterator.next();
                        String
testcaseId=xmlParser.getChildNodeValue(testcaseElement,Constants.ELEMENT_TESTC
ASE_ID);
                        String
testcaseStatus=xmlParser.getChildNodeValue(testcaseElement,Constants.ELEMENT_TE
STCASE_ACTIVE);
                        String
testcaseType=xmlParser.getChildNodeValue(testcaseElement,Constants.ELEMENT_TES
TCASE_TYPE);
```

```
                        if(testcaseStatus.equals(Constants.STR_TRUE) &&
testcaseType.equals(testMode)){
                           System.out.println("| |_+ "+testcaseId);

PropertyEditor.writePropertyToFile(Constants.TESTPLAN_FILENAME,
Constants.TESTCASE_CODE,testcaseId);

                           String
propertyFileName=testMainDir+System.getProperty("file.separator")+moduleId+System.
getProperty("file.separator")+Constants.FOLDER_JMETER+System.getProperty("file.se
parator")+Constants.FOLDER_PROPERTIES+System.getProperty("file.separator")+testc
aseId+Constants.PROPERTIES_FILE_EXT;

PropertyEditor.writeToPropertyFile(propertyFileName,Constants.PROPERTY_FILENA
ME);

                           Element
inputsElement=testcaseElement.getChild(Constants.ELEMENT_TESTCASE_INPUTS);
                           Iterator
inputIterator=(inputsElement.getChildren(Constants.ELEMENT_INPUTS_INPUT)).iterat
or();

                           while(inputIterator.hasNext()){
                              Element inputElement=(Element)inputIterator.next();
                              String input=inputElement.getText();
                              System.out.println("|   |_ "+input);

PropertyEditor.writePropertyToFile(Constants.TESTPLAN_FILENAME,
Constants.INPUT_CODE,input);
                           }
                        }
                     }
                  }
               }
               moduleIterator.remove();
            }
         }
         else if(testMode.equals(Constants.MODE_CUSTOM)){
           for(int j=0; j<testcase.length; j++){
              Iterator
moduleIterator=(xmlParser.getNodeList(Constants.XPATH_MODULE)).iterator();
              while (moduleIterator.hasNext()){
                 Element moduleElement=(Element)moduleIterator.next();
```

```
                    String
moduleId=xmlParser.getChildNodeValue(moduleElement,Constants.ELEMENT_MODU
LE_MODULEID);
                    String
moduleStatus=xmlParser.getChildNodeValue(moduleElement,Constants.ELEMENT_MO
DULE_ACTIVE);

                    if(moduleStatus.equals(Constants.STR_TRUE)){
                       Element
testcasesElement=moduleElement.getChild(Constants.ELEMENT_MODULE_TESTCAS
ES);
                       Iterator
testcaseIterator=(testcasesElement.getChildren(Constants.ELEMENT_TESTCASES_TES
TCASE)).iterator();
                       while(testcaseIterator.hasNext()){
                           Element testcaseElement=(Element)testcaseIterator.next();
                           String
testcaseId=xmlParser.getChildNodeValue(testcaseElement,Constants.ELEMENT_TESTC
ASE_ID);
                           String
testcaseStatus=xmlParser.getChildNodeValue(testcaseElement,Constants.ELEMENT_TE
STCASE_ACTIVE);

                           if(testcaseStatus.equals(Constants.STR_TRUE) &&
testcase[j].equals(testcaseId.trim())){

PropertyEditor.writePropertyToFile(Constants.TESTPLAN_FILENAME,
Constants.MODULE_CODE, moduleId);

PropertyEditor.writePropertyToFile(Constants.TESTPLAN_FILENAME,
Constants.TESTCASE_CODE,testcaseId);
                              System.out.println("| + "+testcaseId);
                              String
propertyFileName=testMainDir+System.getProperty("file.separator")+moduleId+System.
getProperty("file.separator")+Constants.FOLDER_JMETER+System.getProperty("file.se
parator")+Constants.FOLDER_PROPERTIES+System.getProperty("file.separator")+testc
aseId+Constants.PROPERTIES_FILE_EXT;

PropertyEditor.writeToPropertyFile(propertyFileName,Constants.PROPERTY_FILENA
ME);

                              Element
inputsElement=testcaseElement.getChild(Constants.ELEMENT_TESTCASE_INPUTS);
                              Iterator
inputIterator=(inputsElement.getChildren(Constants.ELEMENT_INPUTS_INPUT)).iterat
or();
```

```
                            while(inputIterator.hasNext()){

                                Element inputElement=(Element)inputIterator.next();
                                String input=inputElement.getText();
                                System.out.println("| |_ "+input);

PropertyEditor.writePropertyToFile(Constants.TESTPLAN_FILENAME,
Constants.INPUT_CODE,input);
                            }
                        }
                    }
                }
            }
            moduleIterator.remove();
        }
    }

    System.out.println("------------------------------------------------------------");

} catch (Exception e) {
    System.out.println("****** Exception executeTests ****** "+ e);
}
}
```

Appendix B: Environment Configuration

Setting-up the environment for both “User” and “Developer” is the initial workaround for using Test Automation Framework. Following table shows the required installations.

	User	**Developer**
jdk-1_5_0_06	Compulsory	Compulsory
maven-1.0.2	Compulsory	Compulsory
jakarta-jmeter-2.2	Optional	Compulsory
Badboy-2.0.1	N/A	Compulsory

J2SE Development Kit [4]

Both “User” & “Developer” should install the **jdk-1_5_0_06** in environment. The exact version is available at [5]

JAVA_HOME environment variable should be added and **%JAVA_HOME%\bin** should append to the Path variable.

Apache Maven [1]

Both “User” & “Developer” should install the **maven-1.0.*2*** in environment. The exact version is available at [1]

MAVEN_HOME environment variable should be added and **%MAVEN_HOME%\bin** should append to the Path variable.

Apache JMeter [2]

"Developer" needs to install **jakarta-jmeter-2.2** in the environment in order to create and edit automated test scripts.

When "User" executes a test run, there might be some failed test cases in the final analysis. In that case, user can perform a separate test run for particular test cases with JMeter GUI. Test Automation Framework itself has an archive of "ready –to-run" test cases. So installing **jakarta-jmeter-2.2** in "User's" machine is optional (no dependency in executing test run) but it can be very useful.

jakarta-jmeter-2.2 binary is available at [2]

JMETER_HOME environment variable should be added
and **%JMETER_HOME%\bin** should append to the Path variable.

Badboy [6]

"Developer" needs to install **Badboy-2.0.1** in order to record test cases. So "User" don't need this to be installed in the environment. The exact version is available @ [6]

Appendix C: User Manual

Installation & Configuration

In order to use the Test Automation Framework, "User" has to download "maven-mecp-test-plugin" into the environment.

Download "maven-mecp-test-plugin"

Run the following command in the command line & it will automatically download and install "maven-mecp-test-plugin" in the environment.

```
> maven plugin:download
        -DgroupId=mecp
        -DartifactId=maven-mecp-test-plugin
        -Dversion=2.0.4
```

Configuration of "build.properties" file

This property file contains the properties which are specific to the user. Therefore this file should be placed in the "user's home directory" (e.g. C:\Documents and Settings\{user name} in windows and /home/{user name} in Linux)

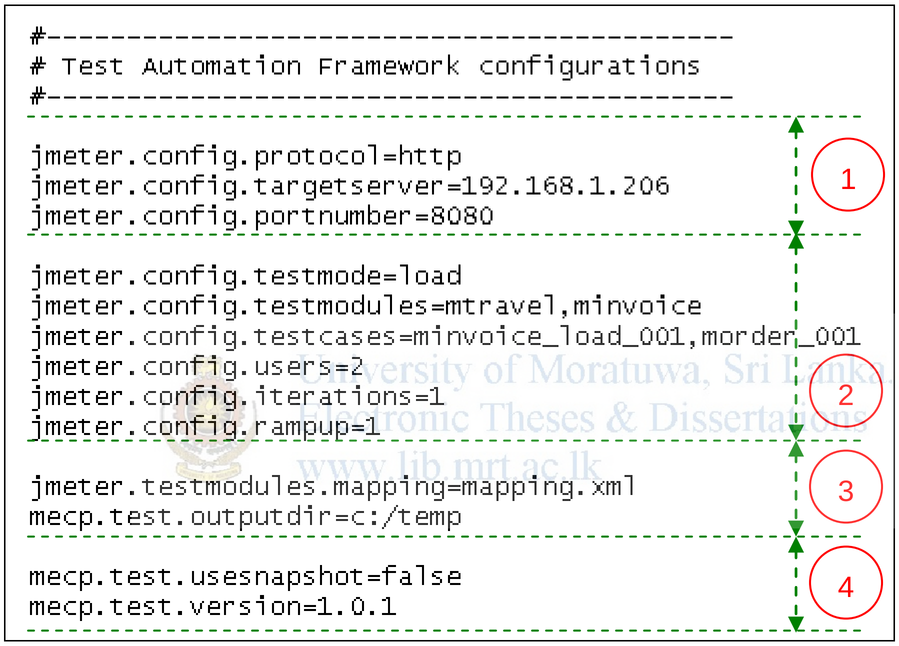
Test Automation Framework specific properties are listed as follows (with sample property values);

```
jmeter.config.protocol=http
jmeter.config.targetserver=192.168.1.206
jmeter.config.portnumber=8680

jmeter.config.testmode=regression
jmeter.config.testmodules=mtravel
jmeter.config.testcases=mtravel_025,mtravel_049
jmeter.config.users=1
jmeter.config.iterations=1
jmeter.config.rampup=1

jmeter.testmodules.mapping=mapping.xml
mecp.test.outputdir=c:/temp

mecp.test.usesnapshot=false
mecp.test.version=1.0.4
```

Apart from these Test Automation Framework specific properties, “build.properties” file should contain following general properties.

maven.repo.remote=http://jupiter:9090/MavenCache/repository/
maven.jboss.home=D:/INSTALLATIONS/jboss-4.0.3SP1
maven.jboss.deploy.dir=${maven.jboss.home}/server/default/deploy

Test Automation Framework Features



```
#-------------------------------------------
# Test Automation Framework configurations
#-------------------------------------------

jmeter.config.protocol=http
jmeter.config.targetserver=192.168.1.206
jmeter.config.portnumber=8080

jmeter.config.testmode=load
jmeter.config.testmodules=mtravel,minvoice
jmeter.config.testcases=minvoice_load_001,morder_001
jmeter.config.users=2
jmeter.config.iterations=1
jmeter.config.rampup=1

jmeter.testmodules.mapping=mapping.xml
mecp.test.outputdir=c:/temp

mecp.test.usesnapshot=false
mecp.test.version=1.0.1
```

Feature Configurations

1

```
jmeter.config.protocol=http
jmeter.config.targetserver=192.168.1.206
jmeter.config.portnumber=8080
```

jmeter.config.protocol can define the Protocol which user is going to test. In most of our scenarios, it will be HTTP.

jmeter.config.targetserver can define the Server which is going to test. If user is going to test an application running on a local machine, the property value should be **localhost** and otherwise **Server-IP** should be used.

jmeter.config.portnumber can simply define the Port which, Application is running on.

2

```
jmeter.config.testmode=regression
jmeter.config.testmodules=mtravel,minvoice
jmeter.config.testcases=minvoice_load_001,morder_001
jmeter.config.users=2
jmeter.config.iterations=3
jmeter.config.rampup=1
```

This part of the configuration handles all test cases in the **mecp-test** module. System designed to handle test cases in very flexible manner.

jmeter.config.testmode property can be used to define the test mode. There are 3 test modes called **regression**, **load** and **custom**. One of the test modes can be set to this property.

jmeter.config.testmodules property can be used to define the product(s) or project(s), the user is going to test. Each project or product should be separated by a comma (,).

jmeter.config.testcases property can be used to define specific test case(s). Each test case should be separated by a comma (,).

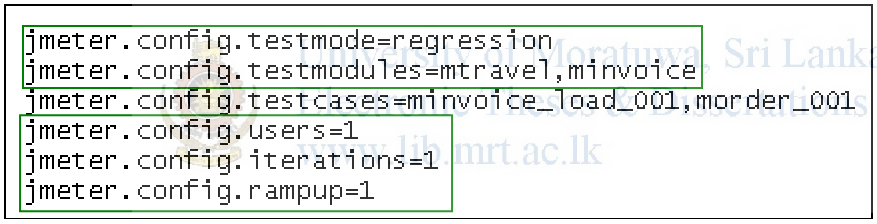
jmeter.config.users property can be used to define the number of threads(users) for the test case(s). Each thread will execute the test case in its entirety and completely independently of other test threads. Multiple threads are used to simulate concurrent connections to server application.

jmeter.config.iterations property can be used to define the number of iterations(loops) for the test case(s). Each thread will execute this number of iterations.

jmeter.config.rampup property can be used to define the ramp-up period. The ramp-up period tells the system 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 the system 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.

Regression, Load and Custom mode configurations and their capabilities as follows;

Regression

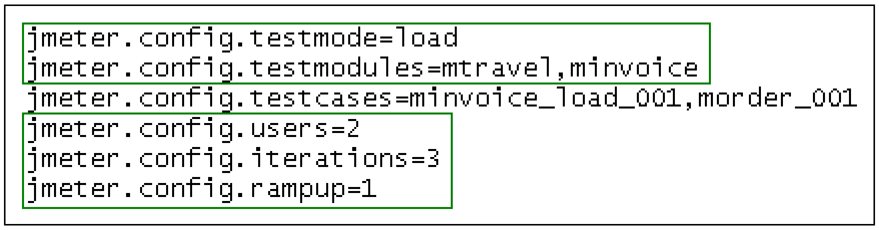


```
jmeter.config.testmode=regression
jmeter.config.testmodules=mtravel,minvoice
jmeter.config.testcases=minvoice_load_001,morder_001
jmeter.config.users=1
jmeter.config.iterations=1
jmeter.config.rampup=1
```

This configuration run all the **regression** test cases under the given projects or products in one by one manner. All the highlighted properties are used in this configuration.

jmeter.config.testmode must set to “**regression**”.
jmeter.config.testmodules should set with project(s) or product(s) list.
jmeter.config.users should be set to “**1**”.
jmeter.config.iterations should be set to “**1**”.
jmeter.config.rampup should be set to “**1**”.

Load



```
jmeter.config.testmode=load
jmeter.config.testmodules=mtravel,minvoice
jmeter.config.testcases=minvoice_load_001,morder_001
jmeter.config.users=2
jmeter.config.iterations=3
jmeter.config.rampup=1
```

This configuration run all the **load** test cases under the given project(s) or product(s) in one by one manner. All the highlighted properties are used in this configuration.

jmeter.config.testmode must set to "**load**".

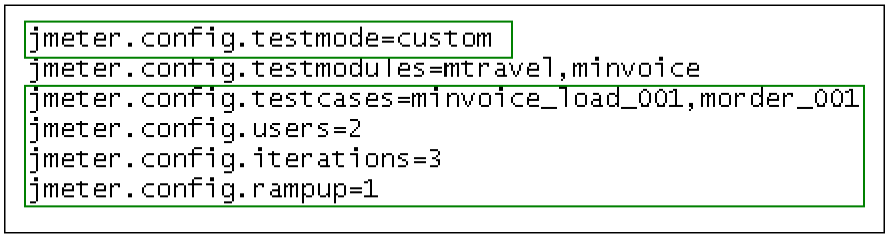
jmeter.config.testmodules should set with project(s) or product(s) list.

jmeter.config.users can be set to any number.

jmeter.config.iterations can be set to any number.

jmeter.config.rampup can be set to any number.

Custom



```
jmeter.config.testmode=custom
jmeter.config.testmodules=mtravel,minvoice
jmeter.config.testcases=minvoice_load_001,morder_001
jmeter.config.users=2
jmeter.config.iterations=3
jmeter.config.rampup=1
```

This configuration gives more flexibility of running test cases. One or few specific test cases can run with this configuration. In most of the load testing scenarios this will be very much useful. All the highlighted properties are used in this configuration.

jmeter.config.testmode must set to "**custom**".

jmeter.config.testcases can be used to define specific test case(s).

jmeter.config.users can be set to any number.

jmeter.config.iterations can be set to any number.

jmeter.config.rampup can be set to any number.

3

```
jmeter.testmodules.mapping=mapping.xml
mecp.test.outputdir=c:/temp
```

jmeter.testmodules.mapping property contains the mapping file name which is using to configure and maintain all Test Cases included in the “mecp-test” module. (There is nothing to change in this property right now)

mecp.test.outputdir property is there to configure Output Directory of the Test Automation Framework. Test Automation Framework is designed to use this directory for extract “mecp-test” module, save test results & archive test scripts.

4

```
mecp.test.usesnapshot=false
mecp.test.version=1.0.1
```

mecp.test.usesnapshot with this configuration user can switch between the locally build “mecp-test” module version and downloaded “mecp-test” version. If the user change configuration to “**true**”, the system will build “mecp-test” module locally and use that **jar** file to test run. If the user changes it to “**false**” the system will take the “mecp-test” jar file version from the mecp.test.version property, then automatically download it from the maven repository and use it with the test run.

Test Automation Framework Commands

Test Automation Framework has two main functionalities.

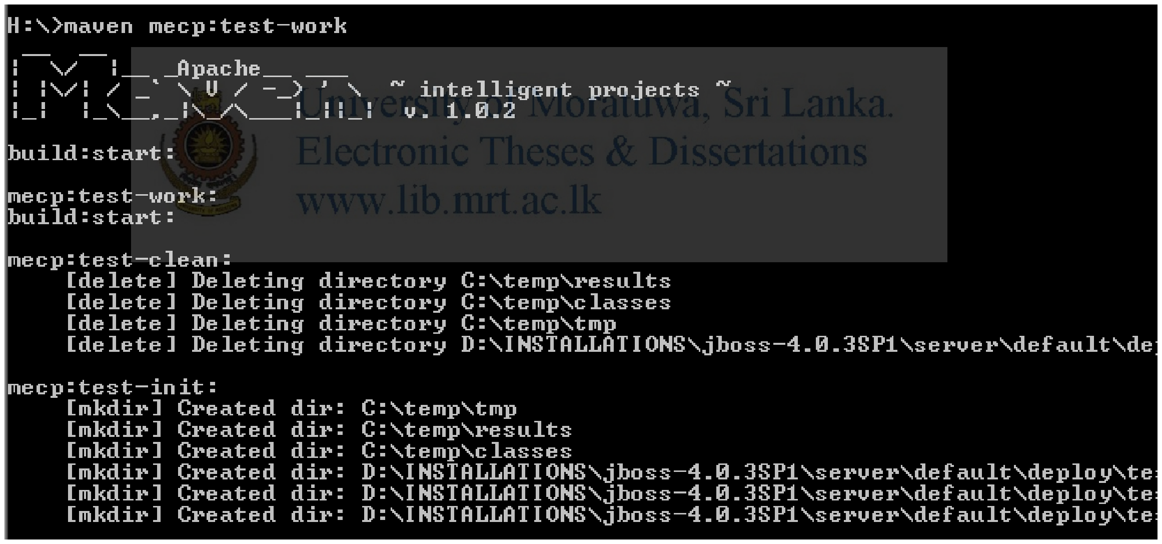
1. Execute Test Run
2. View Test Map

Execute Test Run

In order to execute a test run, “User” needs to run following command in the command-line.

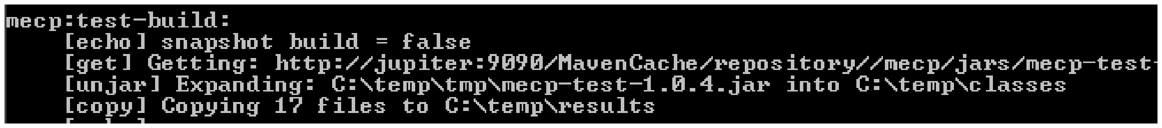
```
> maven mecp:test-work
```

Using command-line console log, “User” can get the clear idea of the ongoing process of the test run.



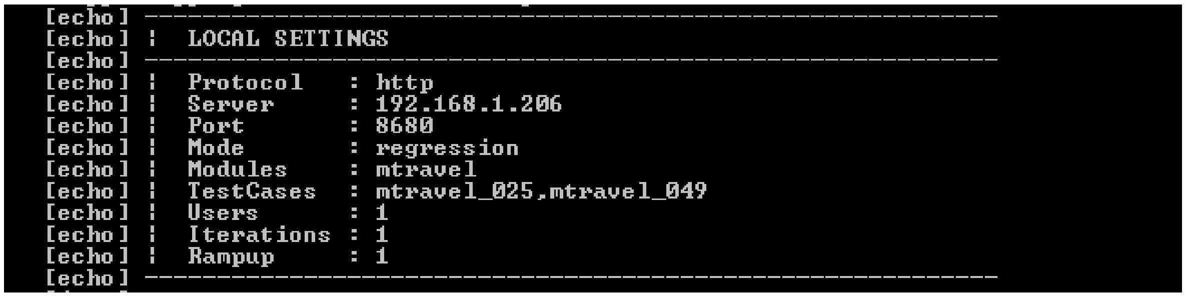
```
H:\>maven mecp:test-work
 __  __
|  \/  |__ _Apache__ ___
| |\/| / _` \ V / -_) ' \  ~ intelligent projects ~
|_|  |_\__,_|\_/\___|_||_|  v. 1.0.2

build:start:

mecp:test-work:
build:start:

mecp:test-clean:
    [delete] Deleting directory C:\temp\results
    [delete] Deleting directory C:\temp\classes
    [delete] Deleting directory C:\temp\tmp
    [delete] Deleting directory D:\INSTALLATIONS\jboss-4.0.3SP1\server\default\de

mecp:test-init:
    [mkdir] Created dir: C:\temp\tmp
    [mkdir] Created dir: C:\temp\results
    [mkdir] Created dir: C:\temp\classes
    [mkdir] Created dir: D:\INSTALLATIONS\jboss-4.0.3SP1\server\default\deploy\te
    [mkdir] Created dir: D:\INSTALLATIONS\jboss-4.0.3SP1\server\default\deploy\te
    [mkdir] Created dir: D:\INSTALLATIONS\jboss-4.0.3SP1\server\default\deploy\te
```

As depicted in the figure, Test Automation Framework cleans all the temporary files and makes necessary new files in the environment.



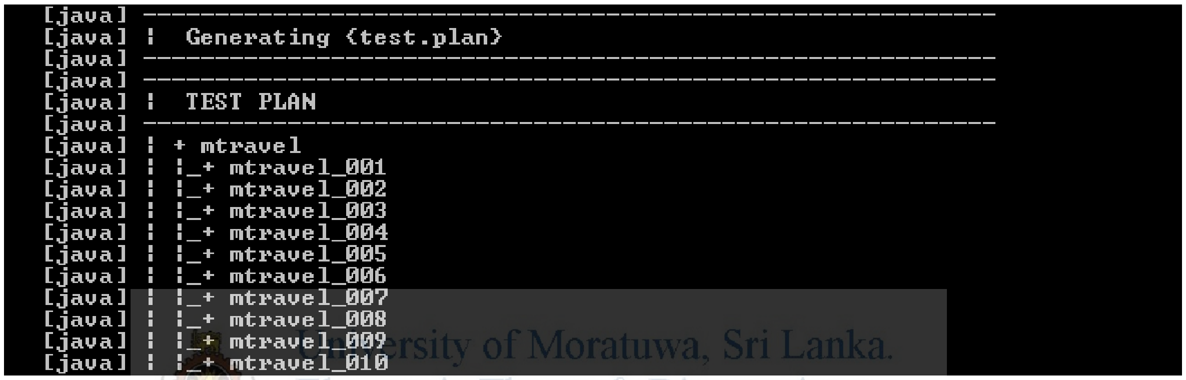
```
mecp:test-build:
    [echo] snapshot build = false
    [get] Getting: http://jupiter:9090/MavenCache/repository//mecp/jars/mecp-test-
    [unjar] Expanding: C:\temp\tmp\mecp-test-1.0.4.jar into C:\temp\classes
    [copy] Copying 17 files to C:\temp\results
```

Then it downloads exact version of the “mecp-test” module from the maven repository and extract it to the given location.



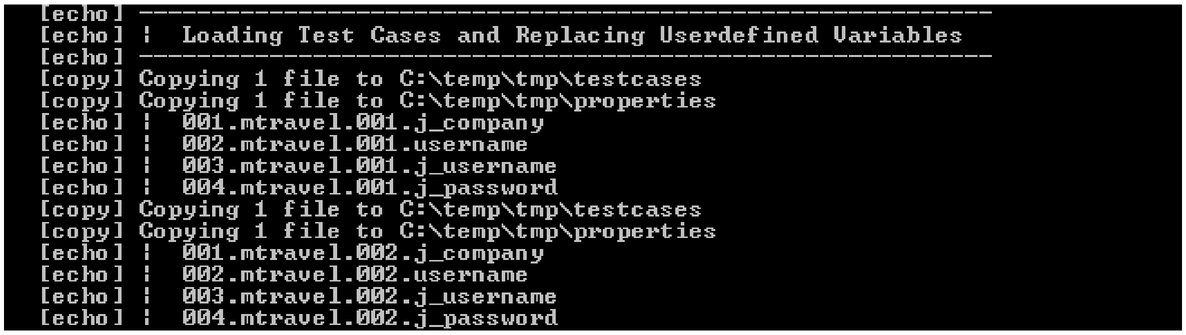
```
[echo] ------------------------------------------------------------
[echo] |  LOCAL SETTINGS
[echo] ------------------------------------------------------------
[echo] |  Protocol    : http
[echo] |  Server      : 192.168.1.206
[echo] |  Port        : 8680
[echo] |  Mode        : regression
[echo] |  Modules     : mtravel
[echo] |  TestCases   : mtravel_025,mtravel_049
[echo] |  Users       : 1
[echo] |  Iterations  : 1
[echo] |  Rampup      : 1
[echo] ------------------------------------------------------------
```

It shows the “User” configuration which is going to follow.



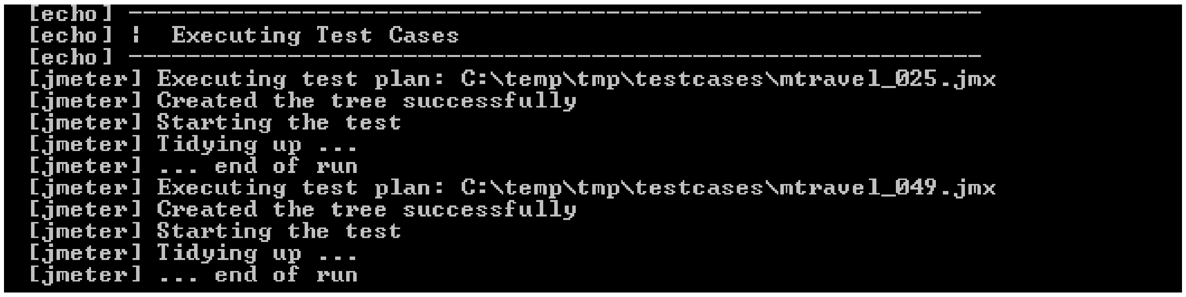
```
[java] ------------------------------------------------------------
[java] |  Generating (test.plan)
[java] ------------------------------------------------------------
[java] ------------------------------------------------------------
[java] |  TEST PLAN
[java] ------------------------------------------------------------
[java] | + mtravel
[java] | |_+ mtravel_001
[java] | |_+ mtravel_002
[java] | |_+ mtravel_003
[java] | |_+ mtravel_004
[java] | |_+ mtravel_005
[java] | |_+ mtravel_006
[java] | |_+ mtravel_007
[java] | |_+ mtravel_008
[java] | |_+ mtravel_009
[java] | |_+ mtravel_010
```

According to the “User” configuration, test plan generates and it shows the “tree-view” of test plan in the console.



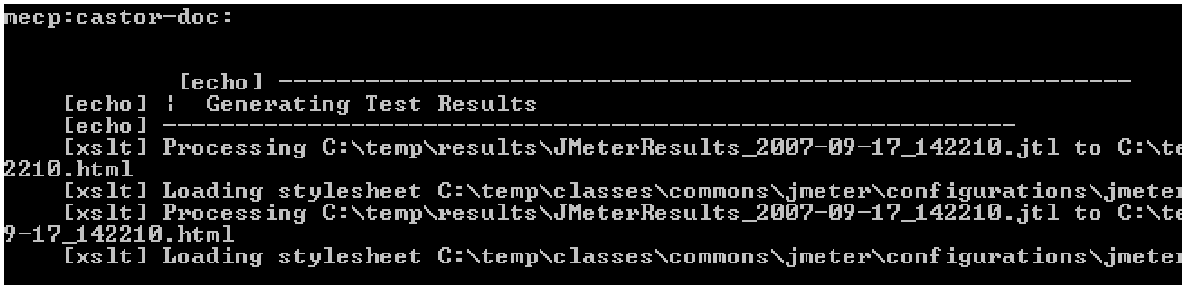
```
[echo] ------------------------------------------------------------
[echo] |  Loading Test Cases and Replacing Userdefined Variables
[echo] ------------------------------------------------------------
[copy] Copying 1 file to C:\temp\tmp\testcases
[copy] Copying 1 file to C:\temp\tmp\properties
[echo] |  001.mtravel.001.j_company
[echo] |  002.mtravel.001.username
[echo] |  003.mtravel.001.j_username
[echo] |  004.mtravel.001.j_password
[copy] Copying 1 file to C:\temp\tmp\testcases
[copy] Copying 1 file to C:\temp\tmp\properties
[echo] |  001.mtravel.002.j_company
[echo] |  002.mtravel.002.username
[echo] |  003.mtravel.002.j_username
[echo] |  004.mtravel.002.j_password
```

According to the test plan, all the test case files, property files and input files are loading into the environment. When Test Automation Framework loads each test case, it will automatically populate the user defined variables.



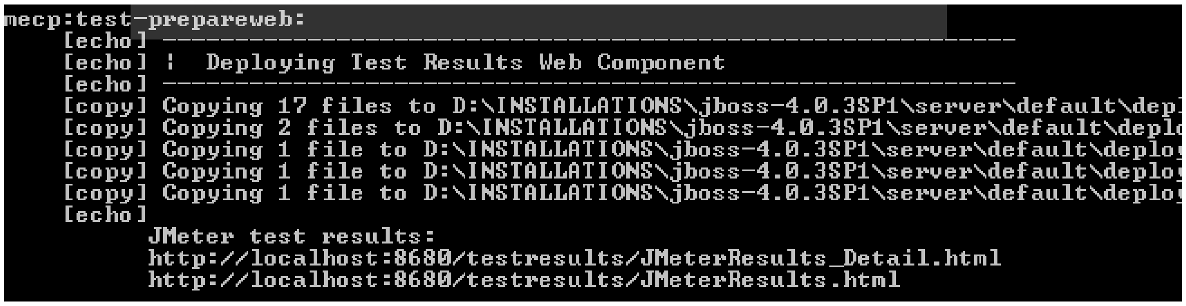
```
  [echo] -----------------------------------------------------------
  [echo] !  Executing Test Cases
  [echo] -----------------------------------------------------------
  [jmeter] Executing test plan: C:\temp\tmp\testcases\mtravel_025.jmx
  [jmeter] Created the tree successfully
  [jmeter] Starting the test
  [jmeter] Tidying up ...
  [jmeter] ... end of run
  [jmeter] Executing test plan: C:\temp\tmp\testcases\mtravel_049.jmx
  [jmeter] Created the tree successfully
  [jmeter] Starting the test
  [jmeter] Tidying up ...
  [jmeter] ... end of run
```

Then Test Automation Framework executes all the loaded scripts one by one and collects the results.



```
mecp:castor-doc:


            [echo] ------------------------------------------------------------
    [echo] !  Generating Test Results
    [echo] ------------------------------------------------------------
    [xslt] Processing C:\temp\results\JMeterResults_2007-09-17_142210.jtl to C:\te
2210.html
    [xslt] Loading stylesheet C:\temp\classes\commons\jmeter\configurations\jmeter
    [xslt] Processing C:\temp\results\JMeterResults_2007-09-17_142210.jtl to C:\te
9-17_142210.html
    [xslt] Loading stylesheet C:\temp\classes\commons\jmeter\configurations\jmeter
```

Once it finished with executing all the test cases, the finalized test results transformed into two separate HTML reports.



```
mecp:test-prepareweb:
    [echo] ------------------------------------------------------------
    [echo] !  Deploying Test Results Web Component
    [echo] ------------------------------------------------------------
    [copy] Copying 17 files to D:\INSTALLATIONS\jboss-4.0.3SP1\server\default\depl
    [copy] Copying 2 files to D:\INSTALLATIONS\jboss-4.0.3SP1\server\default\deplo
    [copy] Copying 1 file to D:\INSTALLATIONS\jboss-4.0.3SP1\server\default\deplo
    [copy] Copying 1 file to D:\INSTALLATIONS\jboss-4.0.3SP1\server\default\deplo
    [copy] Copying 1 file to D:\INSTALLATIONS\jboss-4.0.3SP1\server\default\deplo
    [echo]
          JMeter test results:
          http://localhost:8680/testresults/JMeterResults_Detail.html
          http://localhost:8680/testresults/JMeterResults.html
```

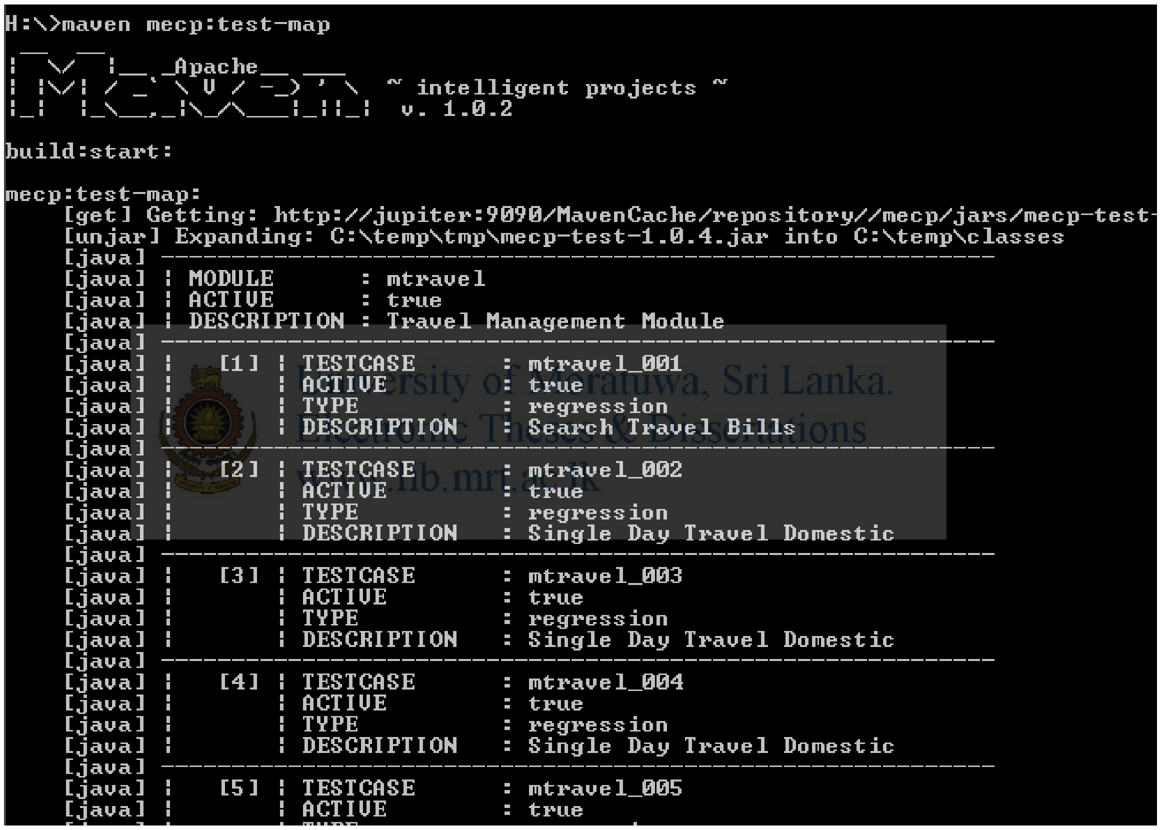
Finally it creates a web-component based on test results and deploys it into the application server.

View Test Map

This is a minor functionality which gives some visibility to the “mecp-test” module. This enables “User” to view the existing test cases and their descriptions in relevant module(s). “User” needs to run following command in the command-line to view the test map.

```
> maven mecp:test-map
```

It gives results in the console, as depicted in the following figure.

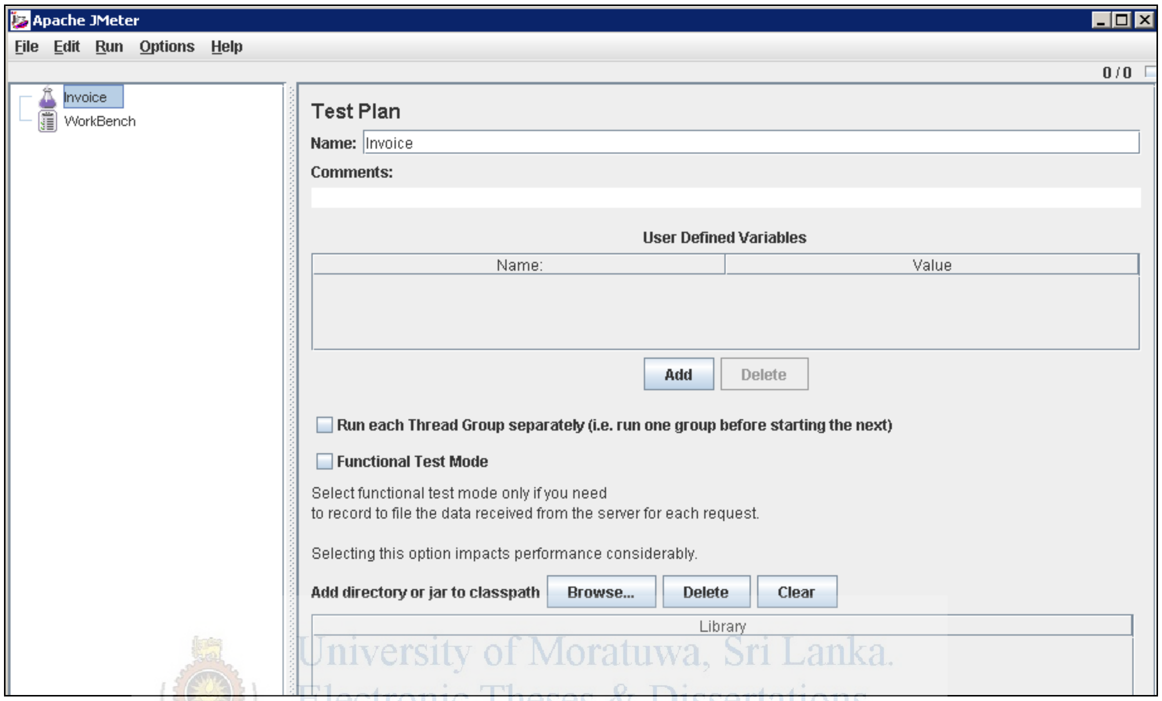


```
H:\>maven mecp:test-map
 __  __
¦  \/  ¦__ _Apache__ ___
¦ ¦\/¦ / _` \ V / -_) ' \  ~ intelligent projects ~
¦_¦  ¦_\__,_¦\_/\___¦_¦¦_¦  v. 1.0.2

build:start:

mecp:test-map:
    [get] Getting: http://jupiter:9090/MavenCache/repository//mecp/jars/mecp-test-
    [unjar] Expanding: C:\temp\tmp\mecp-test-1.0.4.jar into C:\temp\classes
    [java] -----------------------------------------------------------
    [java] ¦ MODULE      : mtravel
    [java] ¦ ACTIVE      : true
    [java] ¦ DESCRIPTION : Travel Management Module
    [java] -----------------------------------------------------------
    [java] ¦   [1] ¦ TESTCASE      : mtravel_001
    [java] ¦       ¦ ACTIVE        : true
    [java] ¦       ¦ TYPE          : regression
    [java] ¦       ¦ DESCRIPTION   : Search Travel Bills
    [java] -----------------------------------------------------------
    [java] ¦   [2] ¦ TESTCASE      : mtravel_002
    [java] ¦       ¦ ACTIVE        : true
    [java] ¦       ¦ TYPE          : regression
    [java] ¦       ¦ DESCRIPTION   : Single Day Travel Domestic
    [java] -----------------------------------------------------------
    [java] ¦   [3] ¦ TESTCASE      : mtravel_003
    [java] ¦       ¦ ACTIVE        : true
    [java] ¦       ¦ TYPE          : regression
    [java] ¦       ¦ DESCRIPTION   : Single Day Travel Domestic
    [java] -----------------------------------------------------------
    [java] ¦   [4] ¦ TESTCASE      : mtravel_004
    [java] ¦       ¦ ACTIVE        : true
    [java] ¦       ¦ TYPE          : regression
    [java] ¦       ¦ DESCRIPTION   : Single Day Travel Domestic
    [java] -----------------------------------------------------------
    [java] ¦   [5] ¦ TESTCASE      : mtravel_005
    [java] ¦       ¦ ACTIVE        : true
```

Creating Test Automation Framework Compatible Test Cases

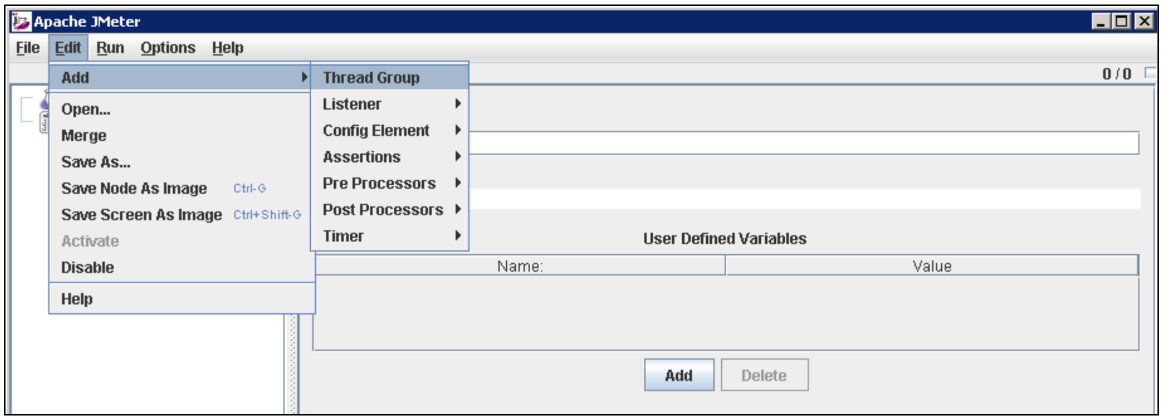
Creating Test Automation Framework specific scripts needs few steps to follow. JMeter GUI should be used to create these scripts effectively. Starting JMeter, pop-ups following GUI and what user should do is create the script with essential elements and save it according to the specific naming convention.

As the first thing change the Test Plan Name with relevant module name (e.g. Invoice, Travel, Logistics).

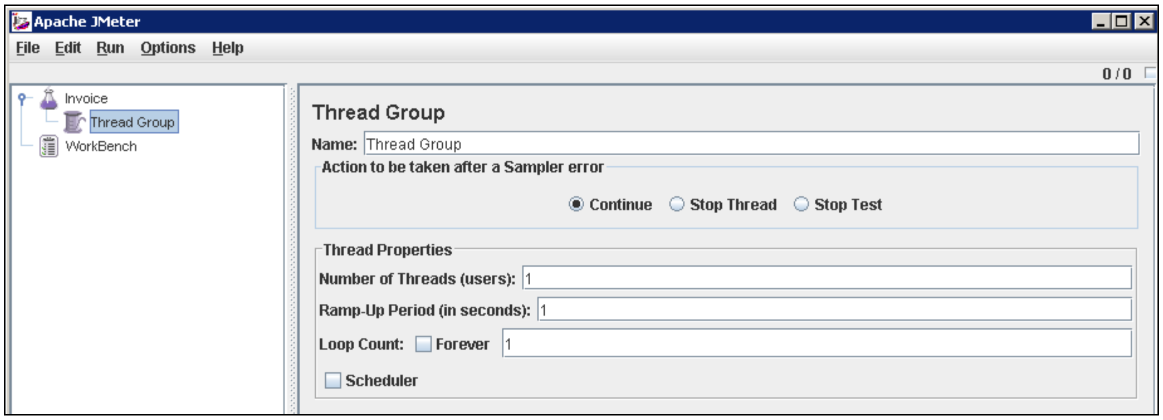

Apache JMeter
File Edit Run Options Help
0 / 0
Invoice
WorkBench
Test Plan
Name: Invoice
Comments:
User Defined Variables
Name:
Value
Add
Delete
Run each Thread Group separately (i.e. run one group before starting the next)
Functional Test Mode
Select functional test mode only if you need
to record to file the data received from the server for each request.
Selecting this option impacts performance considerably.
Add directory or jar to classpath
Browse...
Delete
Clear
Library

Thread Group

There are essential elements in each and every script. Thread Group is the main element in the Test Plan (Test Case or Script). Adding Thread Group to the Test Plan can be done by the JMeter GUI as follows.



Added Thread Group has following default settings with it. But for Preparing it to compatible with Test Automation Framework, there should be changes in the default settings.


Apache JMeter

File Edit Run Options Help

Invoice
Thread Group
WorkBench

Thread Group

Name: Thread Group

Action to be taken after a Sampler error

Continue [✓] Stop Thread Stop Test

Thread Properties

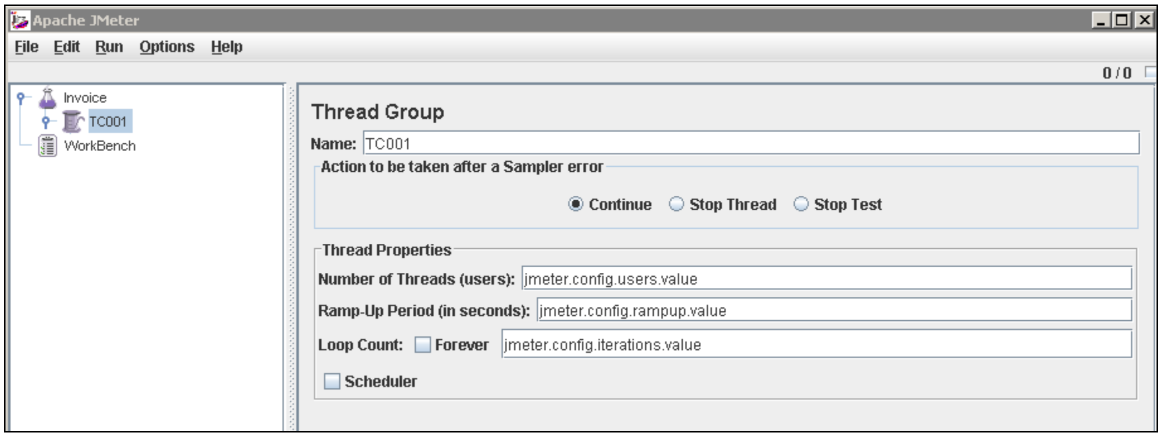
Number of Threads (users): 1

Ramp-Up Period (in seconds): 1

Loop Count: Forever 1

Scheduler

jmeter.config.users.value variable value replaced Number of Threads, jmeter.config.rampup.value variable value replaced Ramp-Up Period and jmeter.config.iterations.value variable value replaced Loop Count . Thread Group Name can be set to the Test Case ID. Following figure shows the exact settings that need to have.


Apache JMeter

File Edit Run Options Help

Invoice
TC001
WorkBench

Thread Group

Name: TC001

Action to be taken after a Sampler error

Continue [✓] Stop Thread Stop Test

Thread Properties

Number of Threads (users): jmeter.config.users.value

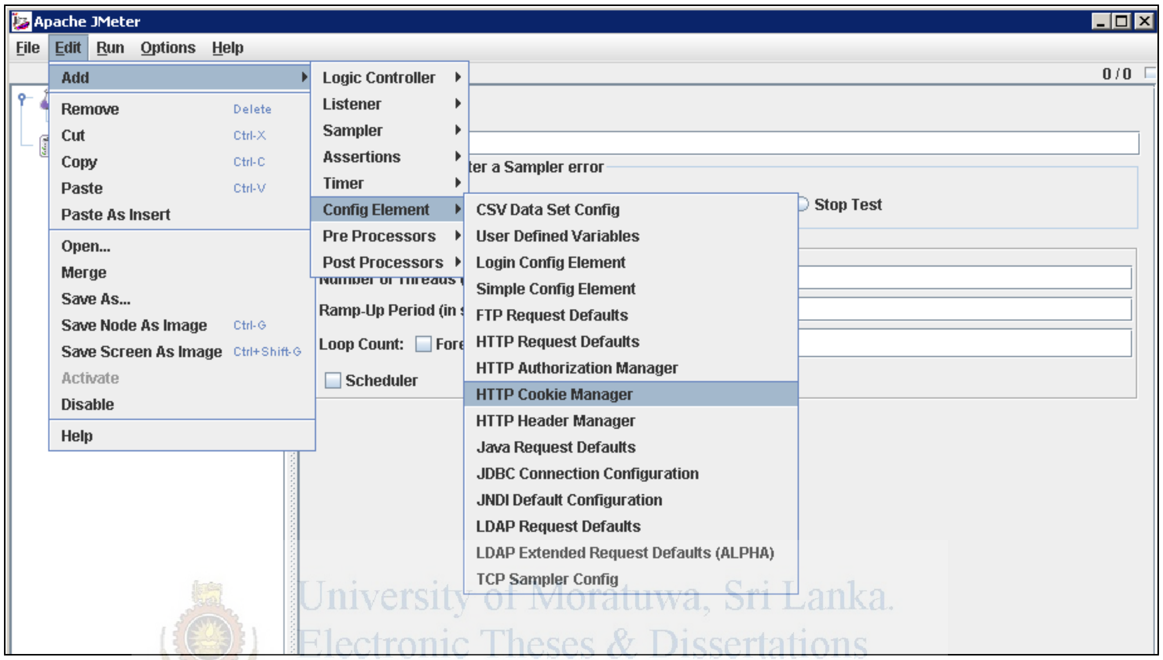
Ramp-Up Period (in seconds): jmeter.config.rampup.value

Loop Count: Forever jmeter.config.iterations.value

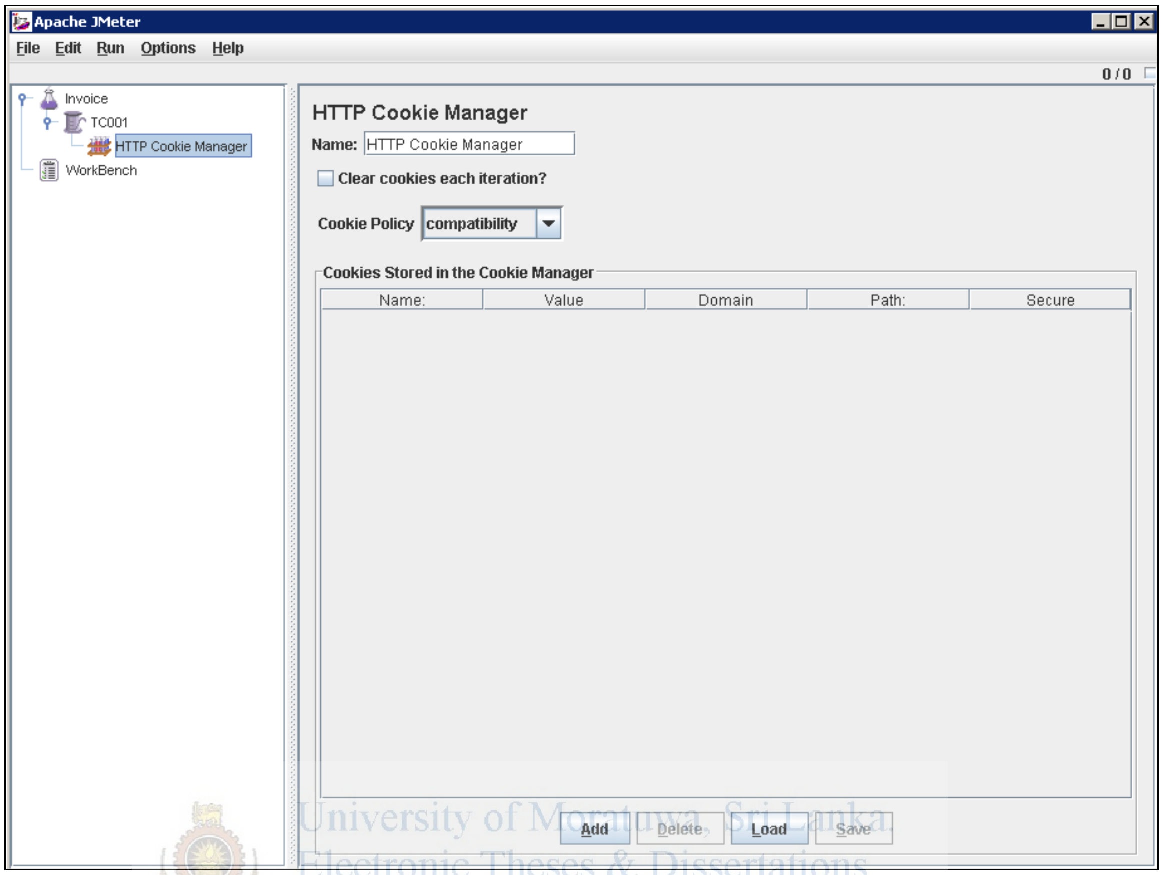
Scheduler

HTTP Cookie Manager

All the other elements that need to have in the script (Test Plan), must be under the added Thread Group. Next step is adding HTTP Cookie Manager into the Thread Group. As in following figure we can add HTTP Cookie Manager into the Thread Group.



No changes for default settings.



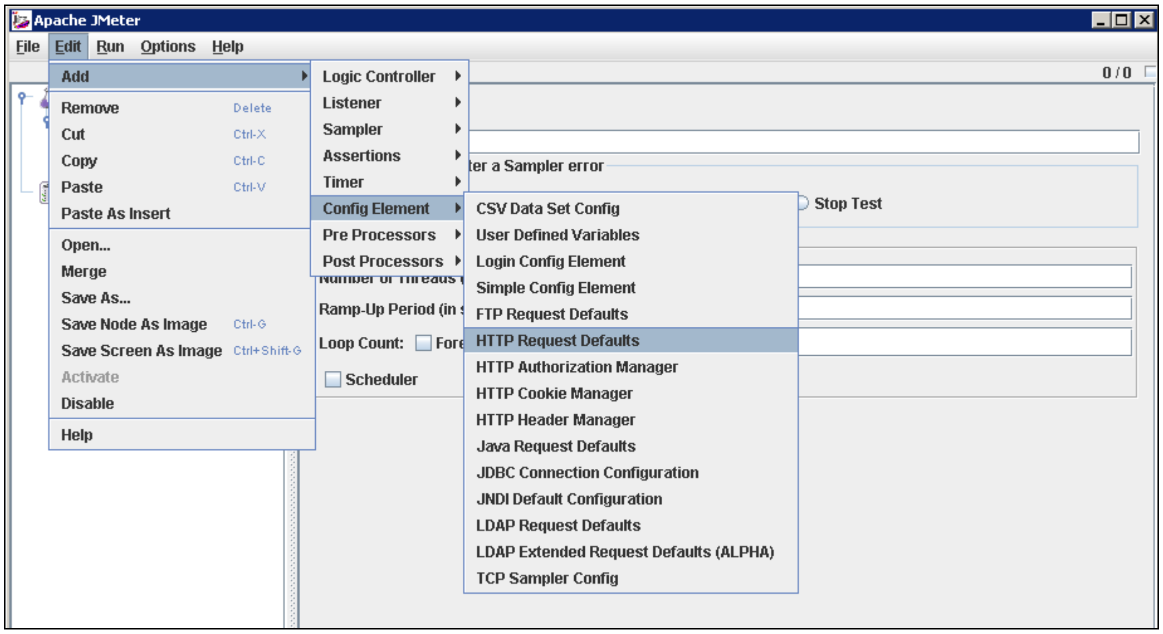
Name: HTTP Cookie Manager

Clear cookies each iteration?

Cookie Policy: compatibility

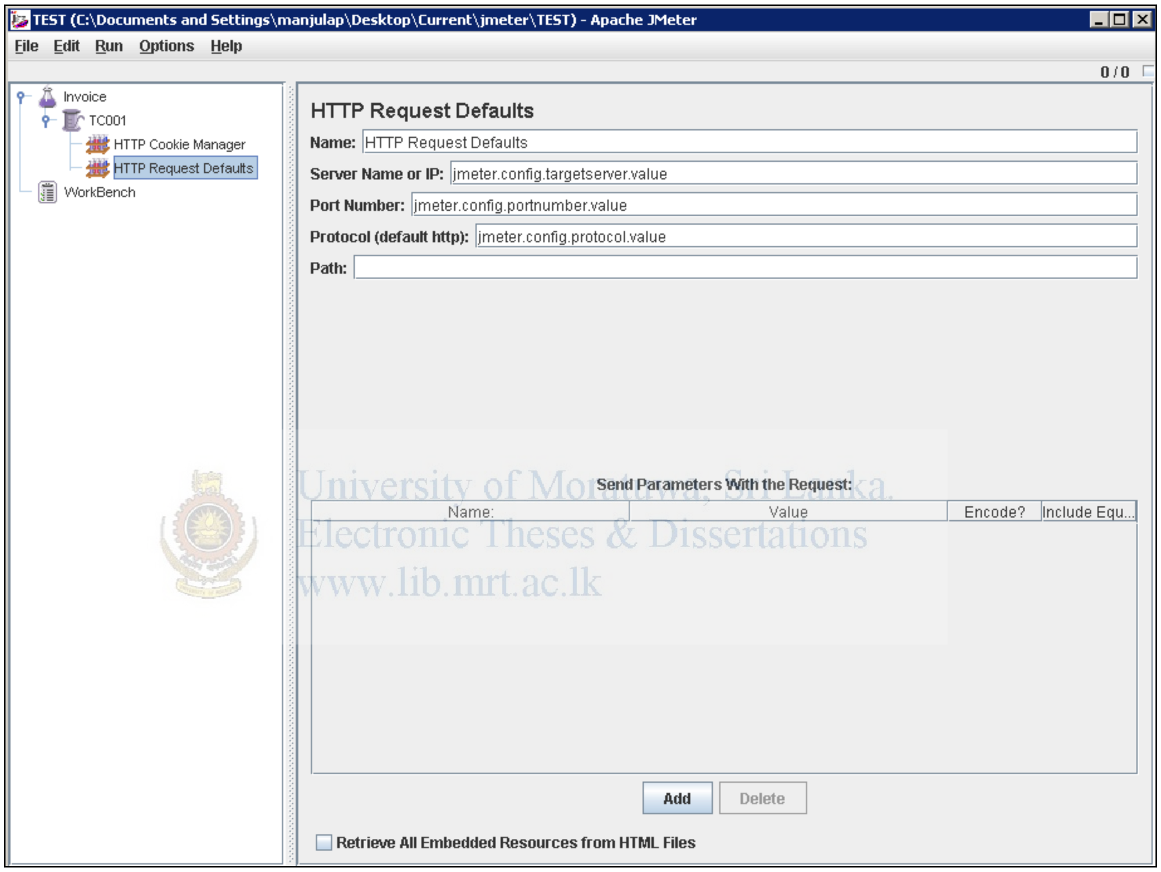
HTTP Request Defaults

Add HTTP Request Defaults into the Thread Group. Figure shows the exact way of adding it to the Thread Group.



]

Here there should be changes for the default settings. Server Name or IP set with jmeter.config.targetserver.value, Port Number set with jmeter.config.portnumber.value and Protocol set with jmeter.config.protocol.value. Following figure shows the configuration.


TEST (C:\Documents and Settings\manjulap\Desktop\Current\jmeter\TEST) - Apache JMeter

HTTP Request Defaults

Name: HTTP Request Defaults

Server Name or IP: jmeter.config.targetserver.value

Port Number: jmeter.config.portnumber.value

Protocol (default http): jmeter.config.protocol.value

Path:

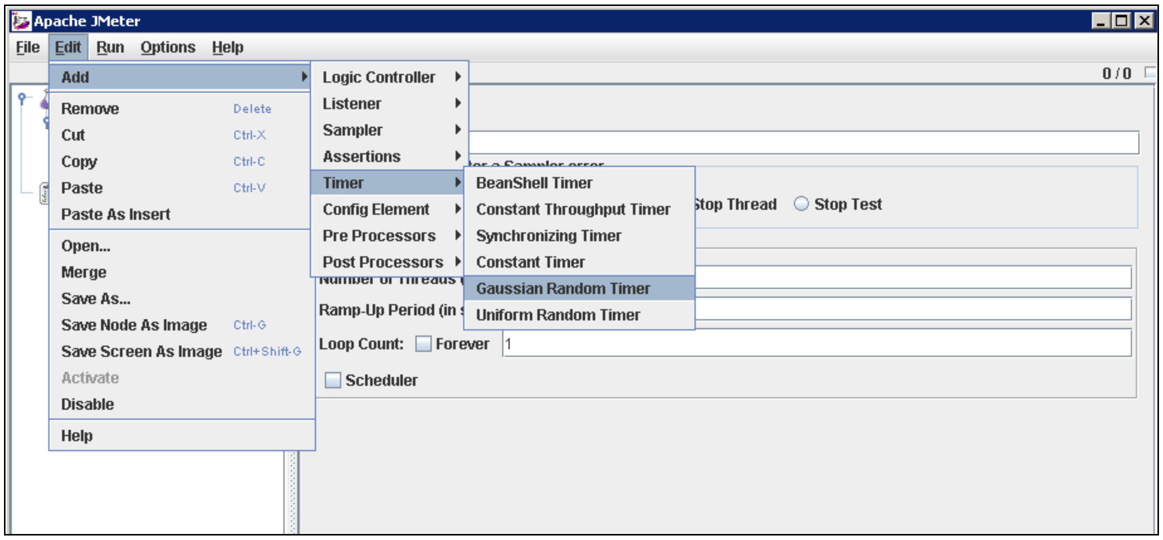
Send Parameters With the Request:

Name:	Value	Encode?	Include Equ...

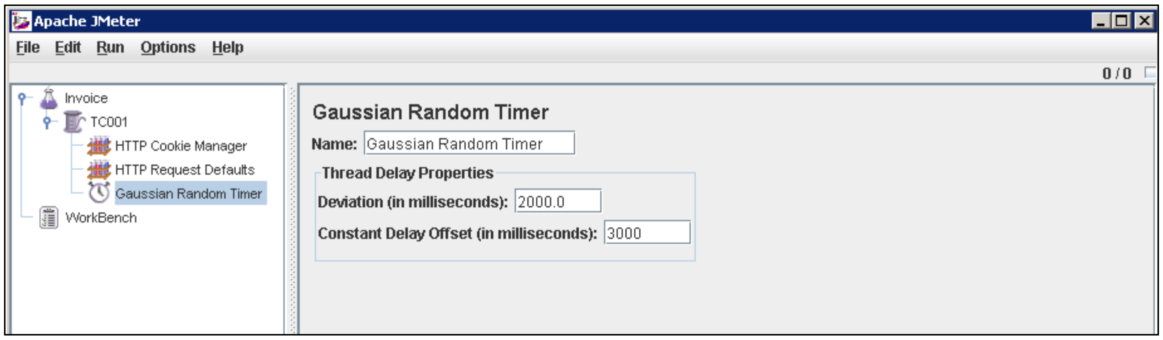
Retrieve All Embedded Resources from HTML Files

Gaussian Random Timer

Adding Gaussian Random Timer into the Thread Group can e done as follows.



Default settings of the Deviation should be changed to 2000 and Constant Delay Offset should be changed to 3000.



Gaussian Random Timer

Name: Gaussian Random Timer

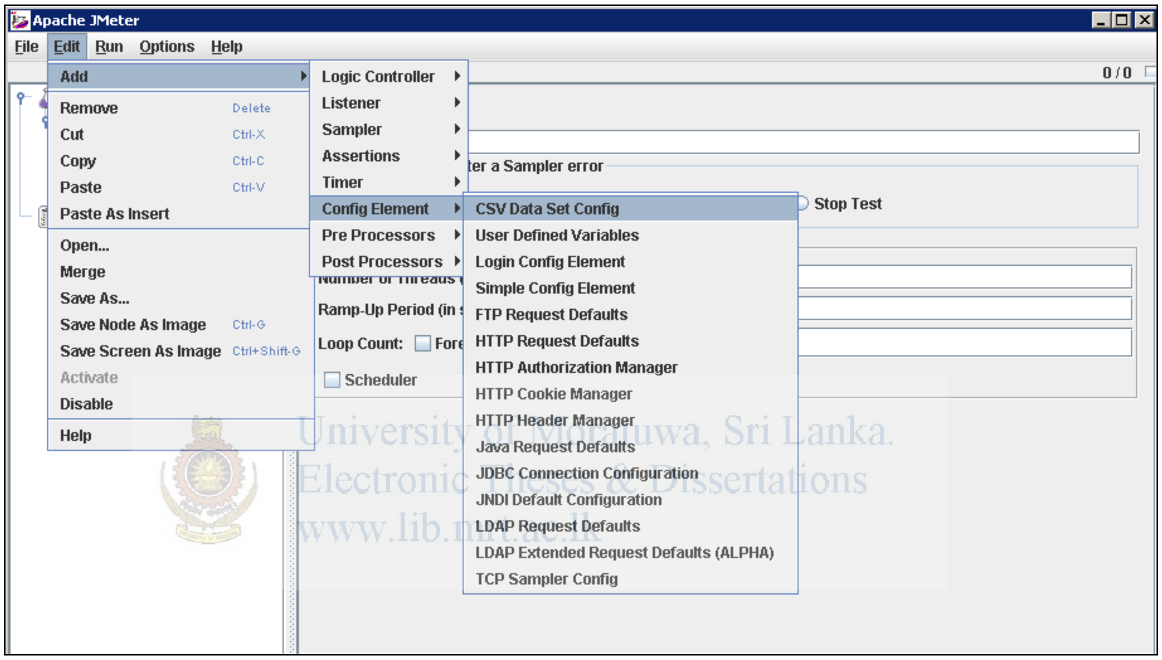
Thread Delay Properties

Deviation (in milliseconds): 2000.0

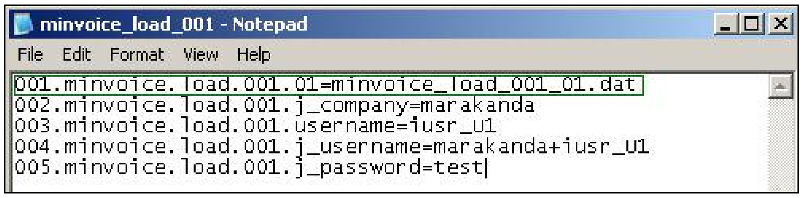
Constant Delay Offset (in milliseconds): 3000

CSV Data Set Config

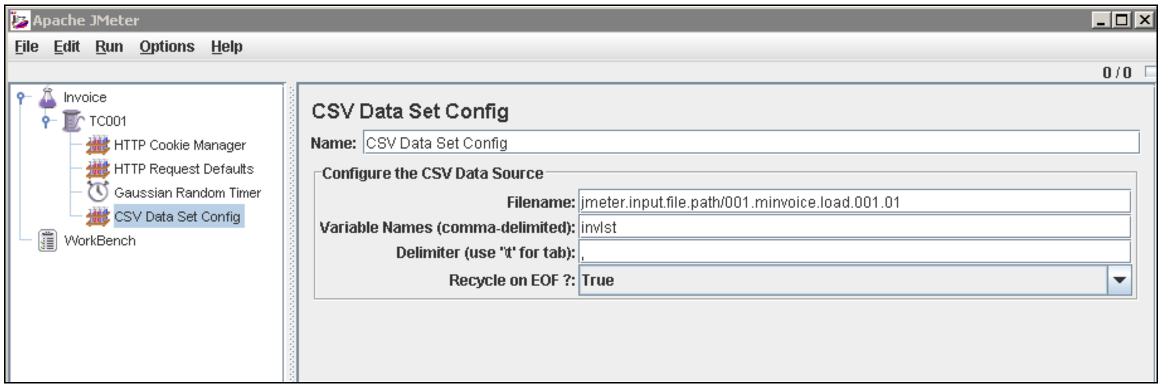
This Element is optional for the scripts. If there is a need of feeding data from a CSV file (can be used in most of the load testing test cases), this element can be used to fulfill that need. Adding element to the Thread Group as follows.



Changes need to the default settings. Filename need to set as jmeter.input.file.path/{INPUT_FILENAME}. Input file name should be declared in the relevant property file and {INPUT_FILENAME} should be replaced by the variable which holds that input file name. Variable Names can be filled with one or more variables and values in the CSV file can be access by these variables. Following example clearly illustrates the scenario.



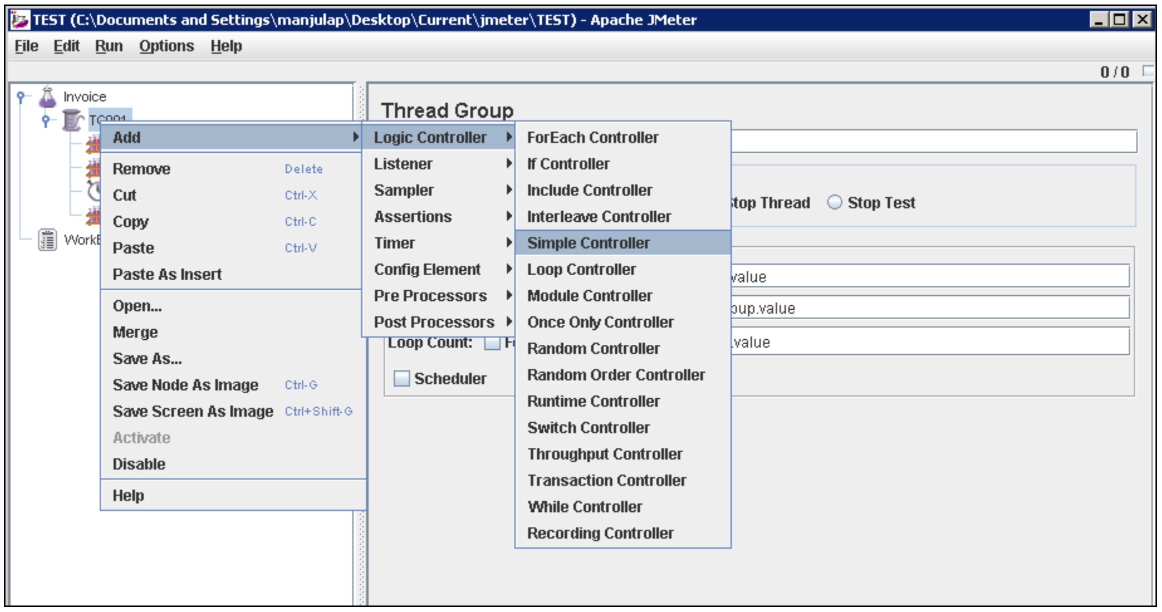
```
001.minvoice.load.001.01=minvoice_load_001_01.dat
002.minvoice.load.001.j_company=marakanda
003.minvoice.load.001.username=iusr_U1
004.minvoice.load.001.j_username=marakanda+iusr_U1
005.minvoice.load.001.j_password=test
```



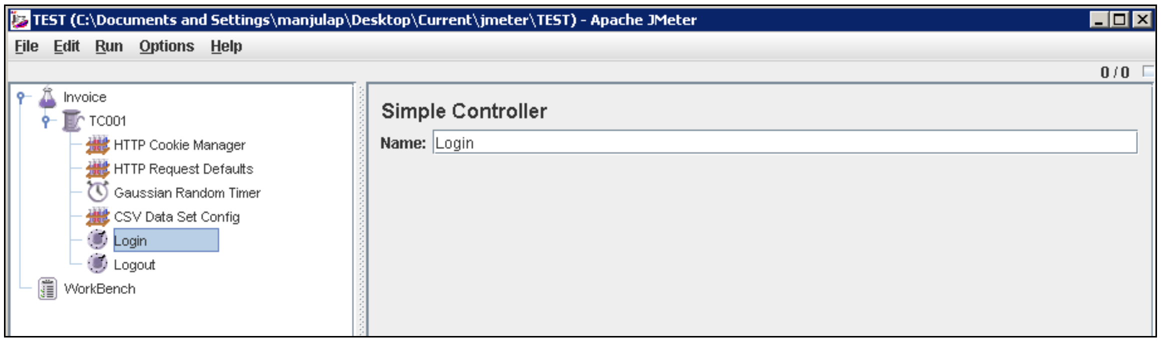
Name: CSV Data Set Config
Filename: jmeter.input.file.path/001.minvoice.load.001.01
Variable Names (comma-delimited): invlst
Delimiter (use '\t' for tab): ,
Recycle on EOF ?: True

Simple Controller

Adding a Simple Controller into the Thread Group is as follows.



Test Automation Framework specific scripts using two Simple Controllers for both Login and Logout.

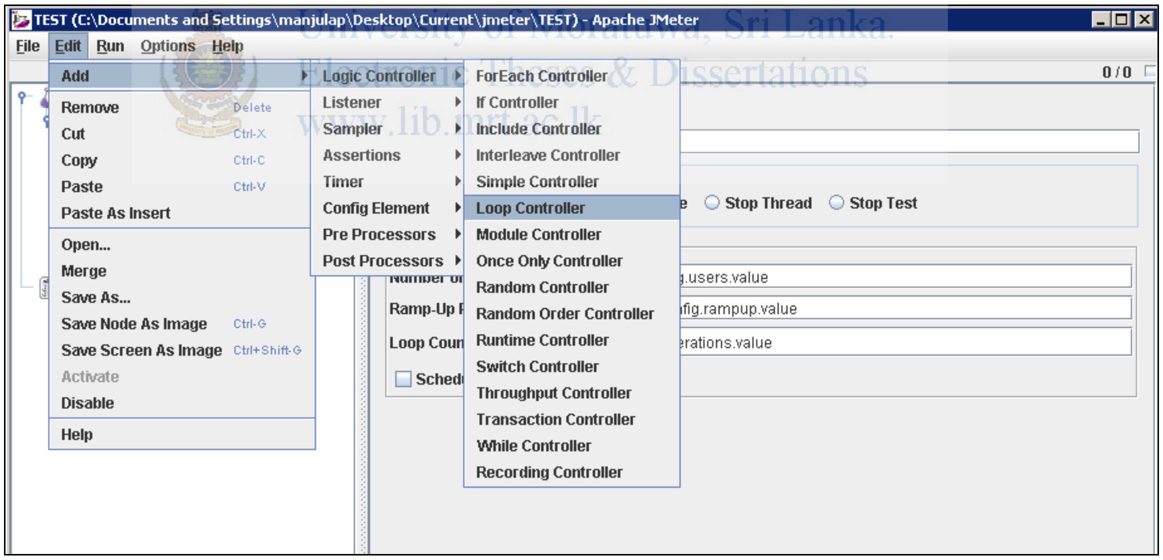

TEST (C:\Documents and Settings\manjulap\Desktop\Current\jmeter\TEST) - Apache JMeter

Simple Controller

Name: Login

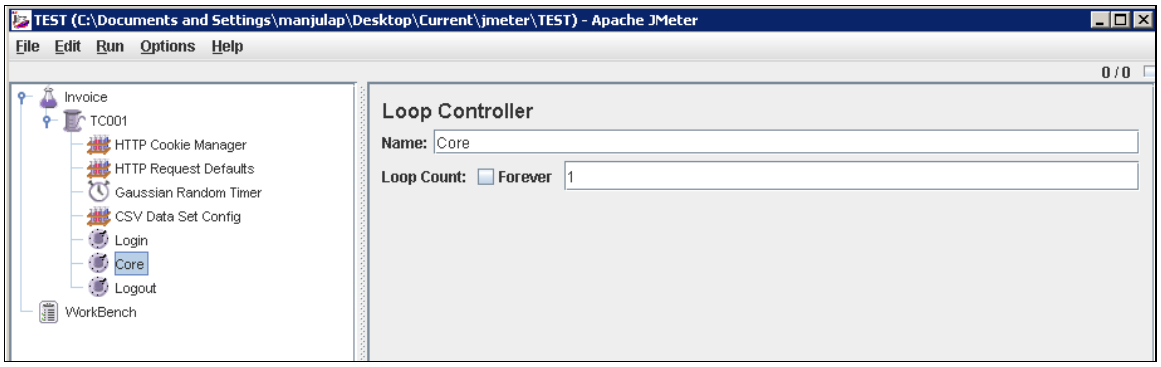
Loop Controller

Adding a Loop Controller into the Thread Group is as follows.


TEST (C:\Documents and Settings\manjulap\Desktop\Current\jmeter\TEST) - Apache JMeter

.users.value
fig.rampup.value
erations.value

Test Automation Framework specific scripts using a loop controller to handle the core of the test case.



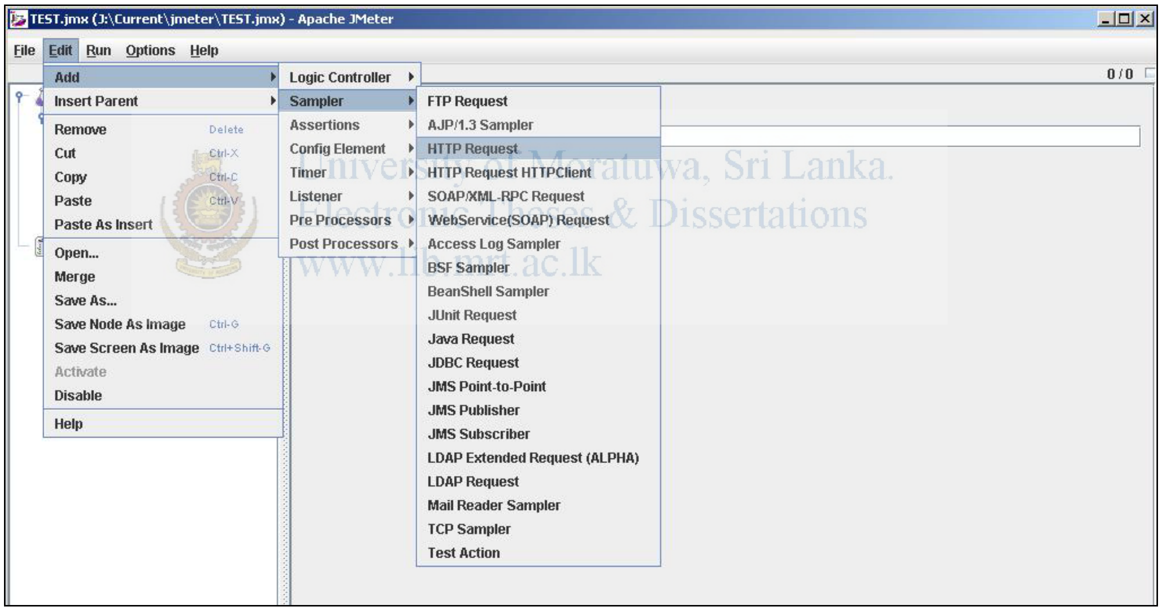
Loop Controller

Name: Core

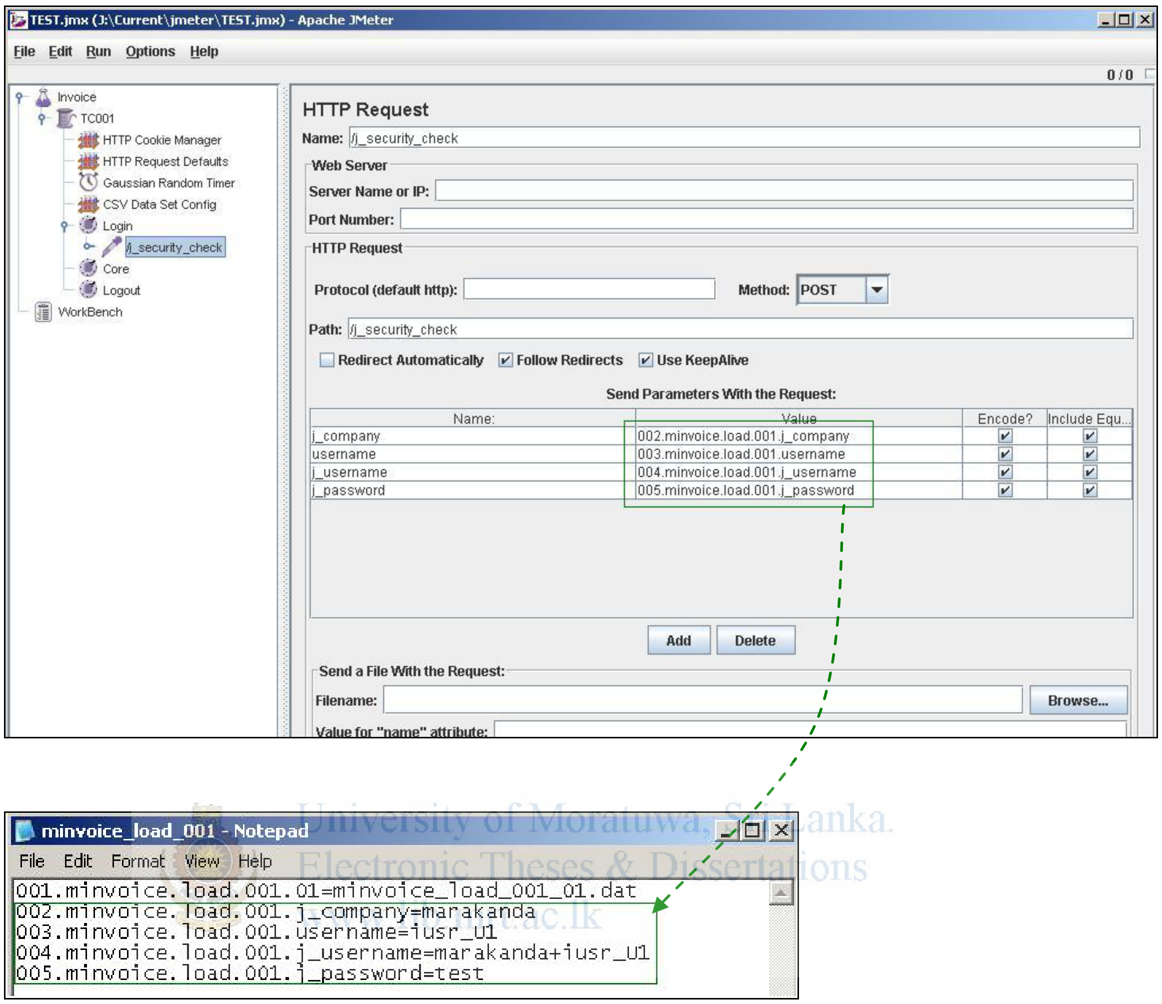
Loop Count: ☐ Forever 1

HTTP Request

Adding a HTTP Request into a Simple Controller or Loop Controller is as follows.



HTTP Request has lots of configurations and list of parameters. “Follow Redirects” and “Use KeepAlive” checkboxes should be checked. Important parameters can be replaced by variables which are defined in the relevant **properties file**.



HTTP Request

Name: /j_security_check

Web Server

Server Name or IP:

Port Number:

HTTP Request

Protocol (default http): Method: POST

Path: /j_security_check

☐ Redirect Automatically ☑ Follow Redirects ☑ Use KeepAlive

Send Parameters With the Request:

Name:	Value	Encode?	Include Equ...
j_company	002.minvoice.load.001.j_company	✓	✓
username	003.minvoice.load.001.username	✓	✓
j_username	004.minvoice.load.001.j_username	✓	✓
j_password	005.minvoice.load.001.j_password	✓	✓

Send a File With the Request:

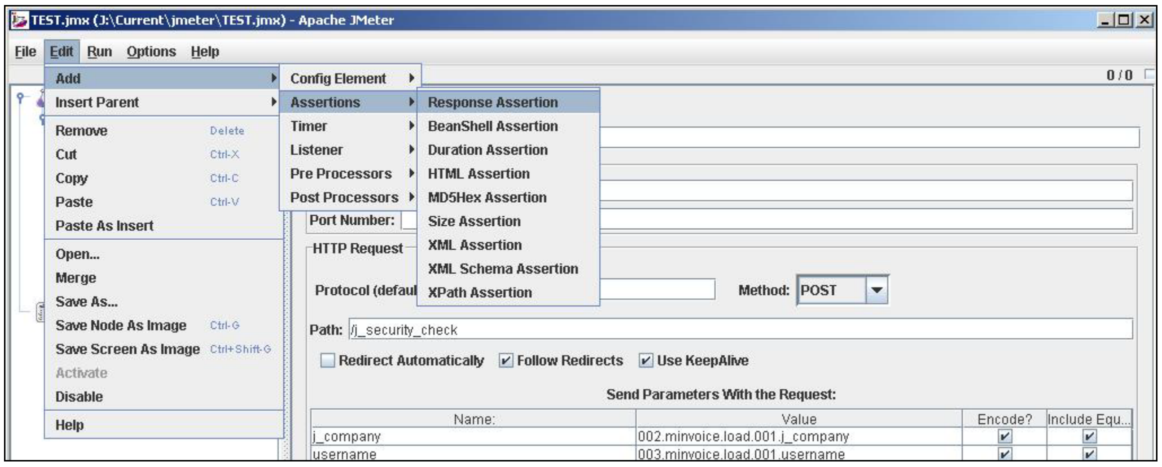
Filename:

Value for "name" attribute:

```
001.minvoice.load.001.01=minvoice_load_001_01.dat
002.minvoice.load.001.j_company=marakanda
003.minvoice.load.001.username=iusr_U1
004.minvoice.load.001.j_username=marakanda+iusr_U1
005.minvoice.load.001.j_password=test
```

Response Assertion

As a practice add Response Assertions to each and every HTTP Requests. Adding a Response Assertion is as follows.



Port Number:

HTTP Request

Protocol (defaul Method: POST

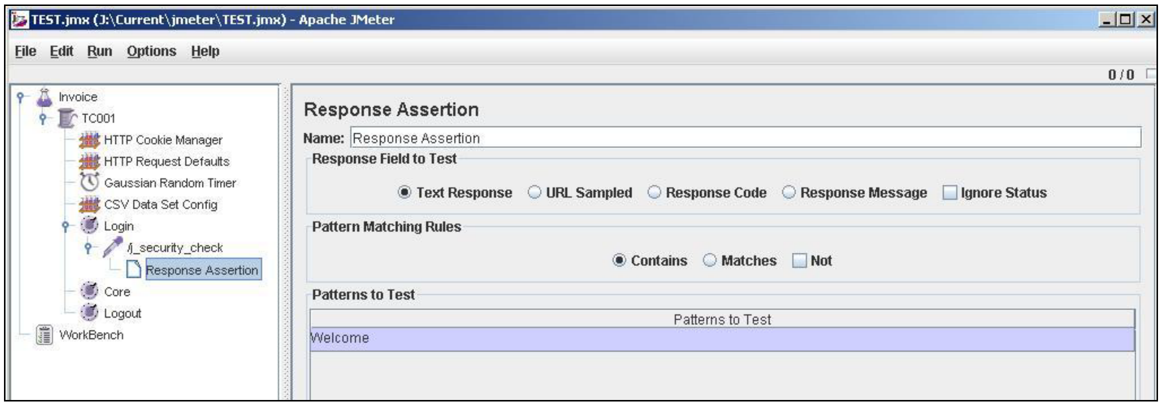
Path: /j_security_check

☐ Redirect Automatically ☑ Follow Redirects ☑ Use KeepAlive

Send Parameters With the Request:

Name:	Value	Encode?	Include Equ...
j_company	002.minvoice.load.001.j_company	✓	✓
username	003.minvoice.load.001.username	✓	✓

It is possible to add one or more patterns to a Response Assertion. Different configurations can be used to perform checks on the response.



Response Assertion

Name: Response Assertion

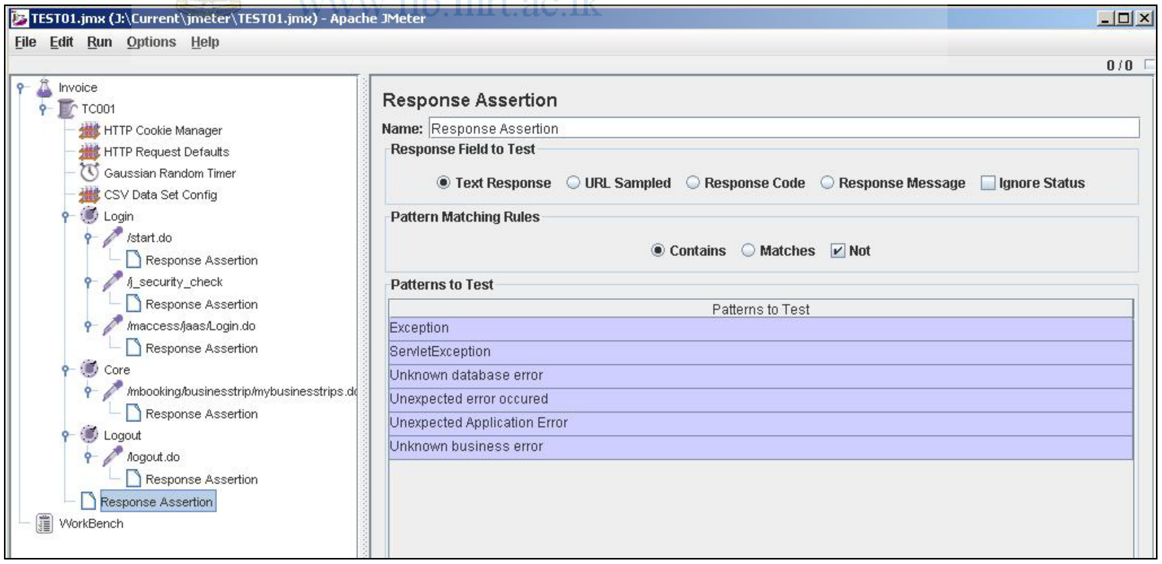
Response Field to Test: [✓ Text Response] URL Sampled Response Code Response Message Ignore Status

Pattern Matching Rules: [✓ Contains] Matches Not

Patterns to Test

Patterns to Test
Welcome

The most important assertion is adding to the Thread group as follows. It is always checks for the common system errors which can be arise in any of the HTTP request. Add all patterns According to the figure below.



Response Assertion

Name: Response Assertion

Response Field to Test: [✓ Text Response] URL Sampled Response Code Response Message Ignore Status

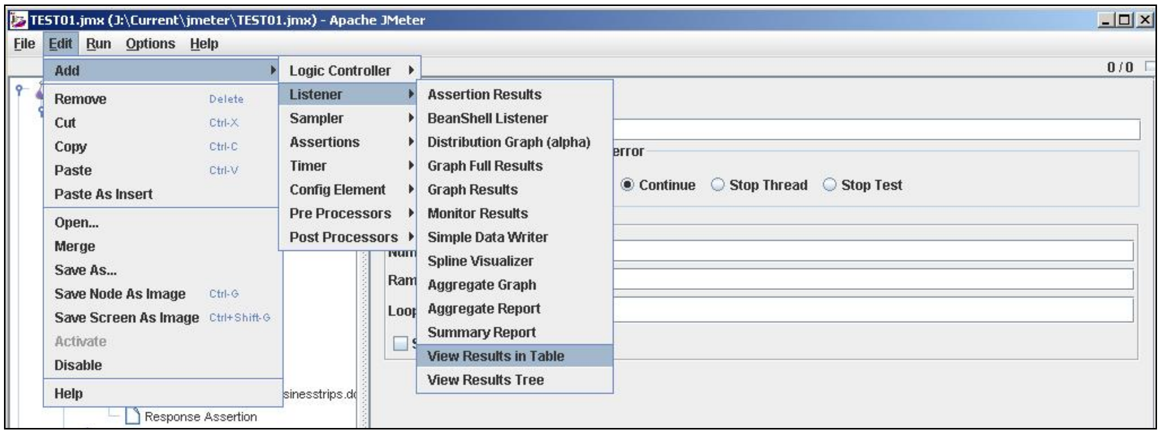
Pattern Matching Rules: [✓ Contains] Matches [✓ Not]

Patterns to Test

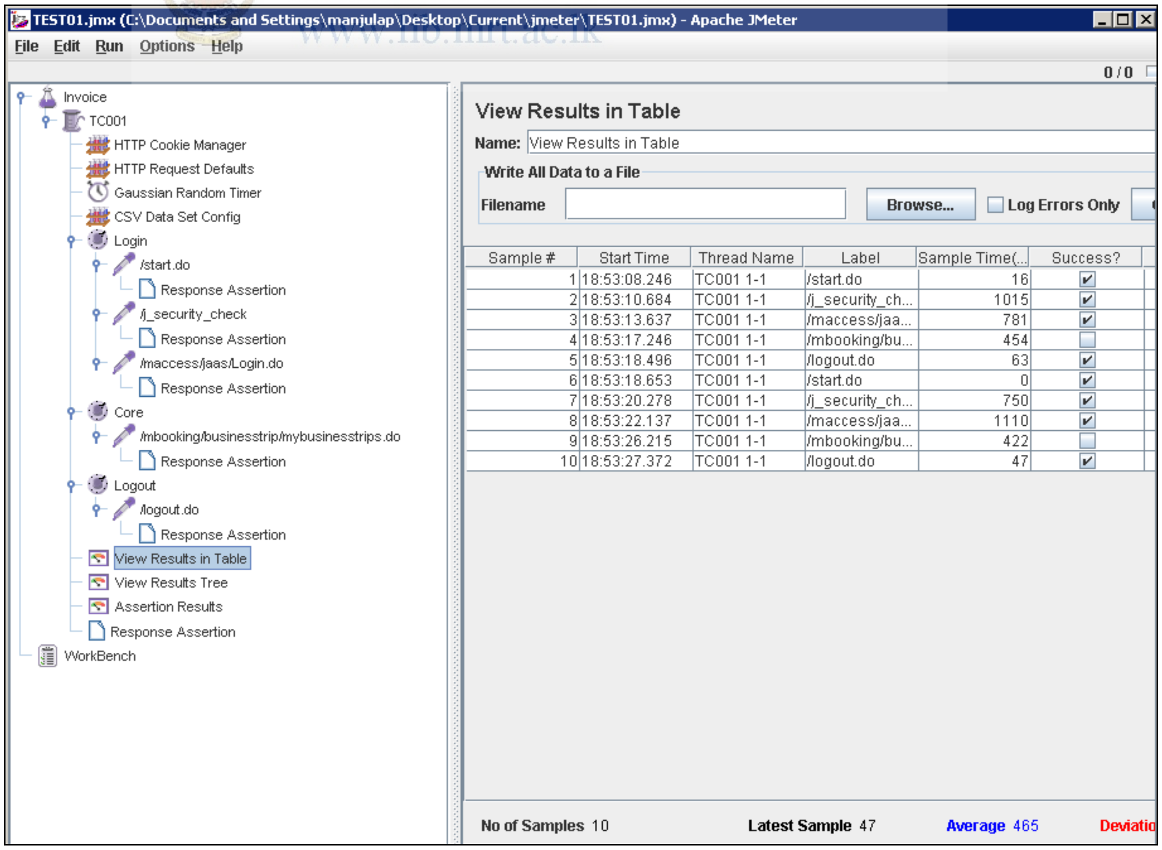
Patterns to Test
Exception
ServletException
Unknown database error
Unexpected error occured
Unexpected Application Error
Unknown business error

View Results in Table

View Results in Table is using to take all time measurements of the executed test cases. This element should be added to the Thread group as follows.



Running a test case in JMeter GUI mode will give an output of "View Results in Table" as shown in the following figure. This listener is very useful to get the time taken for each and every request and so many other important counts.



View Results in Table

Name: View Results in Table

Write All Data to a File

Filename

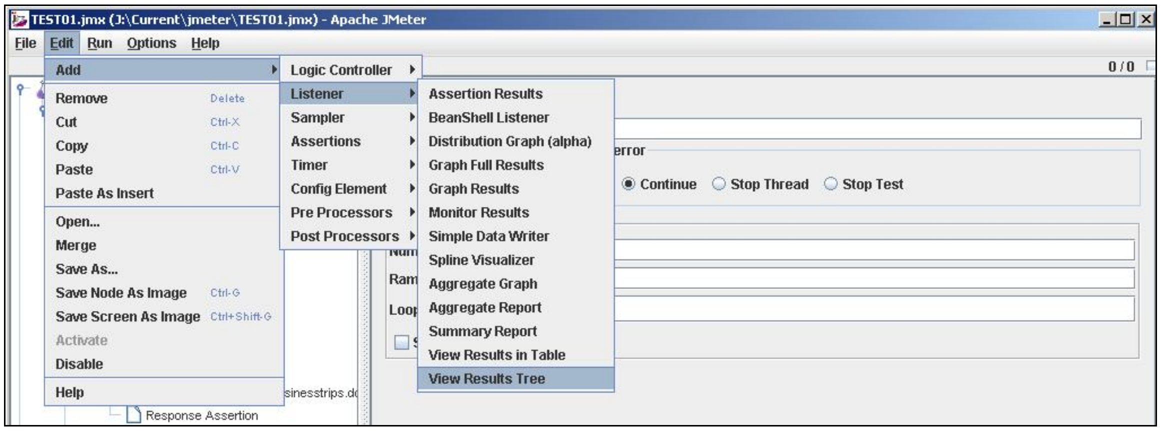
Log Errors Only

Sample #	Start Time	Thread Name	Label	Sample Time(...	Success?
1	18:53:08.246	TC001 1-1	/start.do	16	✓
2	18:53:10.684	TC001 1-1	/j_security_ch...	1015	✓
3	18:53:13.637	TC001 1-1	/maccess/jaa...	781	✓
4	18:53:17.246	TC001 1-1	/mbooking/bu...	454	☐
5	18:53:18.496	TC001 1-1	/logout.do	63	✓
6	18:53:18.653	TC001 1-1	/start.do	0	✓
7	18:53:20.278	TC001 1-1	/j_security_ch...	750	✓
8	18:53:22.137	TC001 1-1	/maccess/jaa...	1110	✓
9	18:53:26.215	TC001 1-1	/mbooking/bu...	422	☐
10	18:53:27.372	TC001 1-1	/logout.do	47	✓

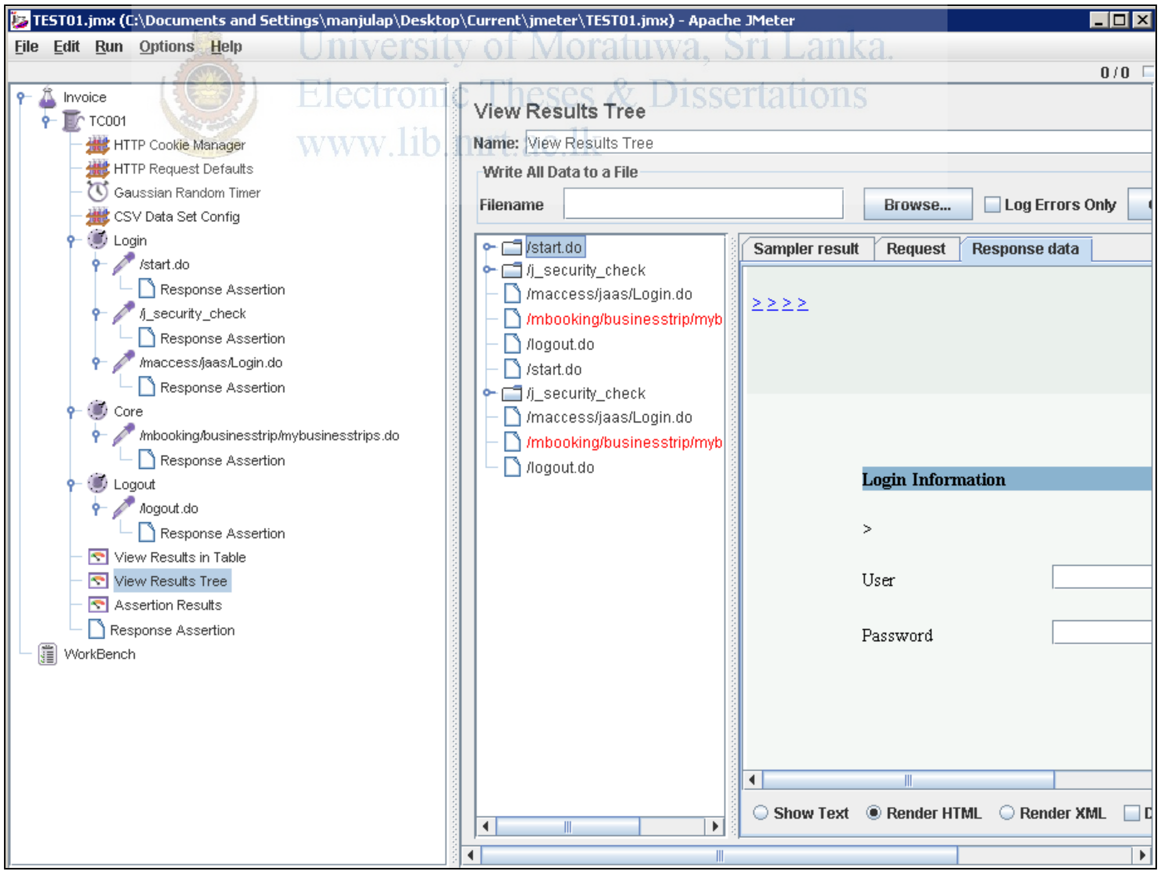
No of Samples 10 Latest Sample 47 Average 465 Deviatio

View Results Tree

Adding View Results Tree into the Thread Group as follows.

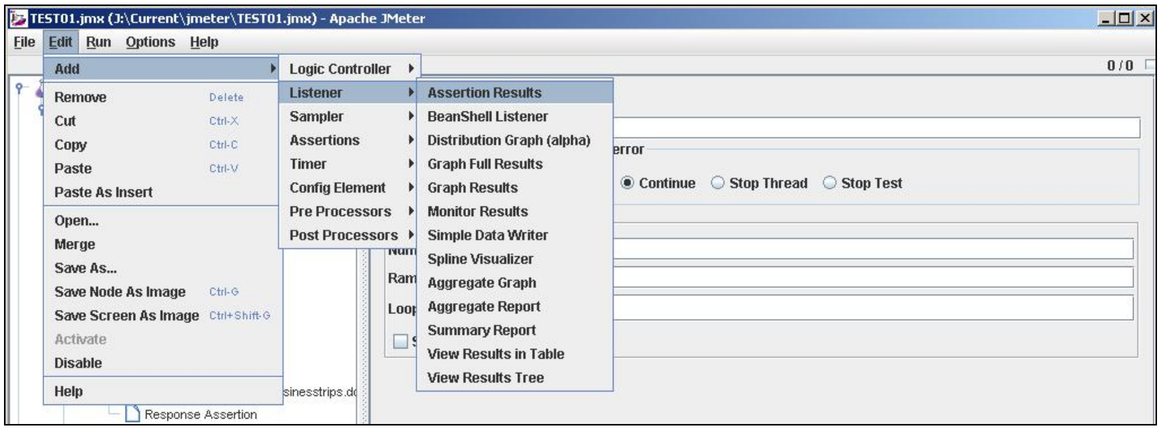


This listener has two panes to show the requests tree and the output of each request. This is the most important listener, when it comes to debug failed test cases.

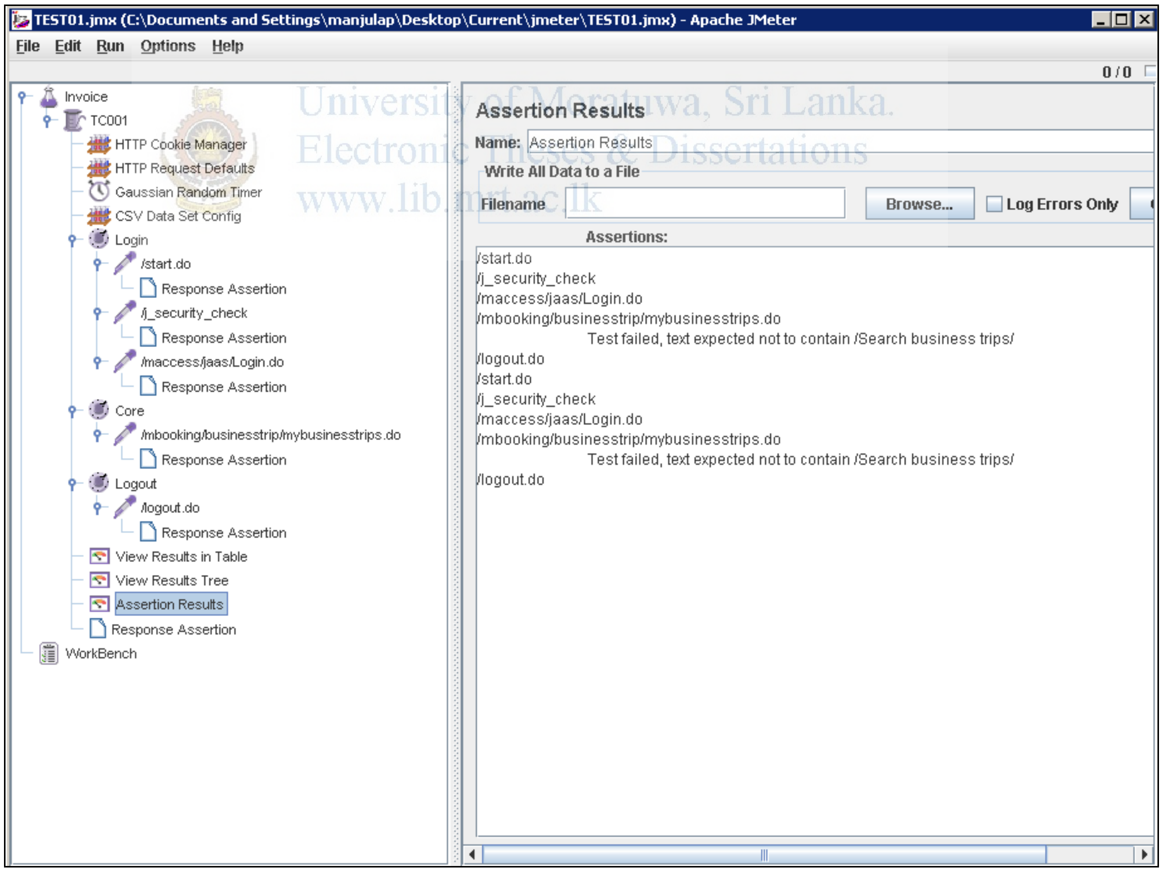


Assertion Results

Adding Assertion Results into the Thread Group is as follows.



This listener is very useful in debugging failed test cases.



```
/start.do
/j_security_check
/maccess/jaas/Login.do
/mbooking/businesstrip/mybusinesstrips.do
                    Test failed, text expected not to contain /Search business trips/
/logout.do
/start.do
/j_security_check
/maccess/jaas/Login.do
/mbooking/businesstrip/mybusinesstrips.do
                    Test failed, text expected not to contain /Search business trips/
/logout.do
```