Learning Objectives

Learners will be able to...

- **Differentiate between performance and stress testing**
- **Identify the purposes of performance and stress testing**
- **Differentiate between sniffers and fiddlers**
- **Install JMeter**
- **Create performance and stress tests with JMeter**

info

Make Sure You Know

This assignment uses JMeter which is an external program. Instructional content is in Codio, but you will be asked to download and install JMeter (and Java) so you can follow along with the examples.

Limitations

This is not an in-depth lesson on all of the features of JMeter.

About Performance Testing

Performance Testing

Performance testing is an important part of the software development process. It helps to ensure that the system meets its performance requirements and that it is able to handle the expected workload. Performance testing can be used to identify potential problems before they become major issues, and it can help to improve the overall quality of the system.

If you want to identify potential bottlenecks in an application you can also use performance testing. Bottlenecks are areas of an application that are causing performance issues. They can be caused by inefficient code, inadequate hardware, or a lack of resources. Performance testing can help to identify these bottlenecks and provide solutions for improving the performance of the system.

Here are some common forms of performance testing:

- **Load testing** – is used to measure the performance of a system under normal conditions. It is used to determine how the system behaves when it is subjected to a normal workload.
- **Stress testing** – is used to measure the performance of a system under peak conditions. It is used to determine how the system behaves when it is subjected to an unusually high workload.
- **Endurance testing** – is done to make sure the software can handle the expected load over a long period of time.
- **Spike testing** – tests the software's reaction to sudden large spikes in the load generated by users.
- **Volume testing** – Under Volume Testing, a large amount of data is populated in a database, and the overall software system's behavior is monitored. The objective is to check a software application's performance under varying database volumes.
- **Scalability testing** – The objective of scalability testing is to determine the software application's effectiveness in "scaling up" to support an increase in user load. It helps plan capacity addition to your software system.

Performance testing can be conducted in a variety of ways. It can be done manually, using scripts, or using automated tools. Manual performance testing involves manually running tests and recording the results. Scripted performance testing involves writing scripts to automate the process of

running tests and recording the results. Automated performance testing involves using tools to automate the process of running tests and recording the results.

Performance testing can be used to measure a variety of metrics, including response time, throughput, resource utilization, and scalability. Response time is the amount of time it takes for a system to respond to a request. Throughput is the amount of data that can be processed in a given period of time. Resource utilization is the amount of resources (e.g. memory, CPU, disk space) that are being used by the system. Scalability is the ability of a system to handle an increasing workload without degrading performance.

Performance Testing Tools

Here are some common tools used for performance testing. We will be using JMeter in this assignment as it is commonly used and freely available.

Apache JMeter is an open source performance testing tool used to test the performance of web applications. It is a Java-based application designed to measure the performance of web applications and services. It can be used to simulate a heavy load on a server, network or object to test its strength or to analyze overall performance under different load types. It can also be used to measure the response time of web applications and services.

It is a powerful tool, as it allows users to create complex test scenarios with multiple threads, timers, and other elements. It also provides a wide range of reporting options, including graphs and tables. It is easy to use and can be used to test applications on different platforms, including Windows, Linux, and Mac OS X. It is also compatible with most web browsers, including Chrome and Firefox.

LoadRunner is a performance testing tool from HP that helps you to test the performance of your applications and websites. It can simulate thousands of users accessing your application simultaneously and measure the response time, throughput, and other performance metrics. LoadRunner can also be used to identify bottlenecks in your application and help you to optimize it for better performance.

BlazeMeter – was designed and built by engineers who are passionate about open source. BlazeMeter gets you massive scale load and performance testing directly from your IDE. Plus, see what your user sees under load with combined UX & load testing. And the best part? It's all in there: performance, functional, scriptless, API testing and monitoring, test data, and mock services.

NeoLoad is a performance testing tool that helps you to identify and diagnose performance issues in your web applications. It allows you to simulate real user scenarios and measure the performance of your application under load, can be used to test the performance of web

applications, mobile applications, it is a great tool for performance testing and can be used to ensure that your application is running optimally. It is easy to use and provides detailed reports and graphs that help you identify and diagnose performance issues.

Also several other tools should be mentioned, like Gatling, ReadyAPI, Appvance UTP, Silk Performer, HP Performance Center - this list is not comprehensive.

Stress Testing

Stress testing is a type of software testing that evaluates the reliability and stability of a system under conditions that exceed the limits of normal operation. Stress testing is usually better at detecting stability, availability, and exception handling by a system under heavy load than what is considered correct behavior under normal conditions.

The term “Stress testing” is often used synonymously with “Load testing” as well as “Performance testing,” which is incorrect because these types of testing answer different business questions and use different methodologies.

Basic Features of Stress Testing

- Determine the stability of the work;
- The main purpose - determine the strength limit of the server;
- Target parameter for the test - throughput;
- Stress test - a comprehensive test that includes load testing;
- Stress testing is testing for excessive load, including an unexpectedly large number of users.

Differences Between the Stress and Load Testing

Often terms **Stress testing** and **Load testing** are thought to mean the same thing. This is not true. Yes, stress and load testing have some common features, for example, a smooth increase of load. But in fact, both concepts are two different processes.

Load testing increases the load to a peak value at which the system will start to slow down. For the stress test, the test is to constantly overload the resource until the server completely “lags”.

Stress testing shows the response rate of your resource when the load is significantly higher than the norm specified in the requirements for the project.

When to Stress Test?

If you allow your user base to expand. It will show how the site behaves when there is a sudden increase in traffic. It is extremely important that the resource will still work, even if the load is much more than the stability of the server.

Example:

The web service is designed to process and display user-created pages, each of which may consist of plain text and dynamic controls. It is known that one user creates 1 page per day, which contains an average of 1,000 characters of text and one form. It is also known that the system receives 1 request per minute to display the original page. In this case, the speed of page display is a critical business process.

A stress test based on this workload model does not address the risks associated with the fact that the system will cease to meet performance requirements if its usage scenario changes. For example, the speed of display of a page can drop significantly if users add dozens of forms to it instead of one.

info

Distributed Systems

In the case of distributed systems testing, it is necessary to consider not only the actual load volume, but also their proportions in the total volume.

Sniffers and Fiddlers

Testing Performance with Sniffers and Fiddlers

Sniffers and **fiddlers** are two essential tools for network analysis and monitoring. Sniffers are used to capture and analyze network traffic while Fiddler is a debugging proxy that can be used to monitor, modify, and debug web traffic.

Sniffers

A sniffer (also known as a packet analyzer or protocol analyzer) is a type of software that captures and records all data packets that pass through a network. It works by monitoring the entire network traffic, capturing and analyzing all the packets that traverse the network. It can be used to monitor network traffic, detect and identify malicious network activity, troubleshoot network problems, and analyze the performance of applications.

Sniffers usually come in two forms: hardware and software. Hardware sniffers are physical devices that are plugged into the network and capture all data packets that pass through it. Software sniffers, on the other hand, are installed on a computer and capture data packets that are sent and received by the computer.

Fiddlers

Fiddler is a debugging proxy that can be used to monitor, modify, and debug web traffic. It works by intercepting, logging, and manipulating HTTP requests and responses between the client and the server. It can be used to analyze and modify web requests, debug Ajax requests, examine and modify the contents of webpages, and trace the communication between the client and the server. It also supports HTTPS, allowing developers to debug secure web traffic.

Fiddler is also a powerful tool for web developers. It is used to diagnose and debug problems related to web applications, and can be used to replicate user actions for testing purposes. It can also be used to create automated tests for web applications and generate performance reports.

Sniffers and Fiddler are essential tools for network analysis, monitoring, and debugging. They allow developers to capture and analyze network traffic, detect and identify malicious network activity, and debug and

modify web requests. They are powerful tools for web developers and network administrators, and can be used to diagnose and debug problems related to web applications and network performance.

Tools for Sniffing and Fiddling

Popular Tools

- **Wireshark:** Wireshark is a popular open-source network sniffer that can be used to capture and analyze network traffic. It supports a wide range of protocols and provides comprehensive statistics and analysis capabilities.
- **Microsoft Message Analyzer:** Microsoft Message Analyzer is a powerful packet sniffer and network analyzer that can be used to capture and analyze network traffic. It supports a wide range of protocols and provides comprehensive statistics and analysis capabilities.
- **NetworkMiner:** NetworkMiner is a free, open-source network sniffer that can be used to capture and analyze network traffic. It supports a wide range of protocols and provides comprehensive statistics and analysis capabilities.
- **Fiddler:** Fiddler is a debugging proxy that can be used to monitor, modify, and debug web traffic. It can be used to analyze and modify web requests, debug Ajax requests, examine and modify the contents of webpages, and trace the communication between the client and the server.

Sniffer and Fiddler Examples

Use Cases for Sniffers and Fiddlers

- **Debugging a web application:** Fiddler can be used to debug web applications by capturing and analyzing the HTTP requests and responses between the client and the server. It can be used to replicate user actions for testing purposes and generate performance reports.
- **Detecting malicious network activity:** Sniffers can be used to detect and identify malicious network activity by capturing and analyzing all the packets that traverse the network. It can be used to monitor network traffic, detect suspicious activity, and analyze the performance of applications.
- **Troubleshooting network problems:** Sniffers can be used to troubleshoot network problems by capturing and analyzing all the packets that traverse the network. It can be used to monitor network traffic, identify the source of the problem, and diagnose the issue.
- **Testing Web Applications:** Fiddler can be used to test web applications by capturing and analyzing the requests and responses between the client and the server. To do this, you first need to set up Fiddler as a proxy on the client computer. Then, run the web application in the browser and Fiddler will start capturing and analyzing the HTTP requests and responses. You can then use Fiddler to modify the requests and responses and test different scenarios. You can also use Fiddler to generate performance reports to measure the performance of the web application.

Installing Apache JMeter and Plugins

Install JMeter

JMeter is an application that runs outside of Codio and the browser. You will need to manually install it on your system.

To install JMeter we need to install Java on our computer first. To see the version of Java, we can check in the terminal (Linux and Mac) or the command prompt (Windows). Running the following command should return the version of Java installed on your machine. JMeter requires Java version 8 or later.

```
java -version
```

If Java is not installed on your machine, you will need to download and install it on your machine.

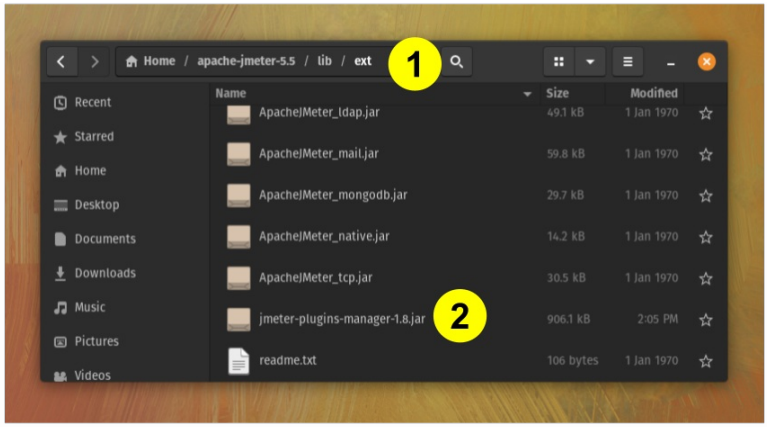
Now that we are ready to install JMeter, download the binary. **Important**, you will need to scroll down to get the latest version of JMeter. At that time of writing, that would be 5.5. If you are on a Mac or Linux machine download the file with the `.tgz` extension. If you are Windows, download the file with the `.zip` extension. Then unzip the archive. Please note where the unzipped directory resides on your system.

Inside the Apache JMeter directory is another directory named `bin`. In the terminal, navigate to this location. To start JMeter, run the command `./jmeter` on Mac or Linux. On Windows you can run the `ApacheJMeter` file.

JMeter Plugins Manager

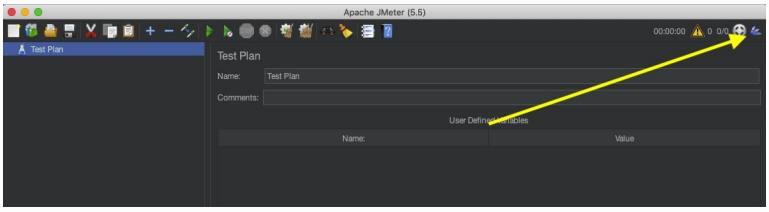
We must also install the JMeter Plugins Manager. The idea of JMeter Plugins Manager is simple: instead of installing various plugins manually, it will do it for you through a nice UI. You no longer have to handle JAR files. You can now use a GUI to install, remove, and upgrade plugins.

To download and install the plugin manager, you first need to download this JAR file. Then, move the JAR file(2) to the `/lib/ext/` directory(1) inside the parent Apache JMeter directory.



The image depicts a file manger in Linux with the plugin JAR file in the /lib/ext/ directory.

You must restart JMeter in order to see the plugin manager. If everything was successful, you will see an icon in the top-left corner.



The image depicts the JMeter plugin manager icon in the top-right corner of the application.

Click on the plugin icon to open the manager in a new window. The default view(1) is the list of currently installed plugins. If you click on `Available Plugins`(2), you can search for plugins to install. Any new plugins are compiled into a list(3). Finally, click the button in the bottom-right corner(4) to install the plugins and restart JMeter.

The image depicts the plugin manager interface.

Let's install some of the more commonly used plugins. Search for each of the following plugins in the `Available Plugins` tab. Check the box next to each plugin you want to install. Once all of them are checked, click the button in the bottom-right corner.

1. 3 Basic Graphs
2. BlazeMeter Step-by-step Debugger
3. Custom JMeter Functions
4. Custom Thread Groups
5. Distribution/Percentile Graphs
6. Dummy Sampler
7. Flexible File Writer
8. KPI vs KPI graphs
9. PerfMon (Servers Performance Monitoring)
10. Throughput Shaping Timer

The JMeter plugins project provides documentation for each plugin if you are curious about what they do.

Creating a Load Test

info

JMeter

The activities on this page require JMeter. Make sure it is running before continuing.

Test Plans

We are going to create a load test, which simulates actions that would occur during normal usage. JMeter uses the term “test plan” to describe the tests that it performs. On the left we see our `Test Plan` tree(1). Click on it and rename it `Google Test`(2). This test visits Google’s homepage.

We need to add users who will visit the page. Right-click on `Google Test`(1). Then select `Add`(2). In the next menu, hover over `Threads (Users)`(3). Finally, select `Thread Group`(4).

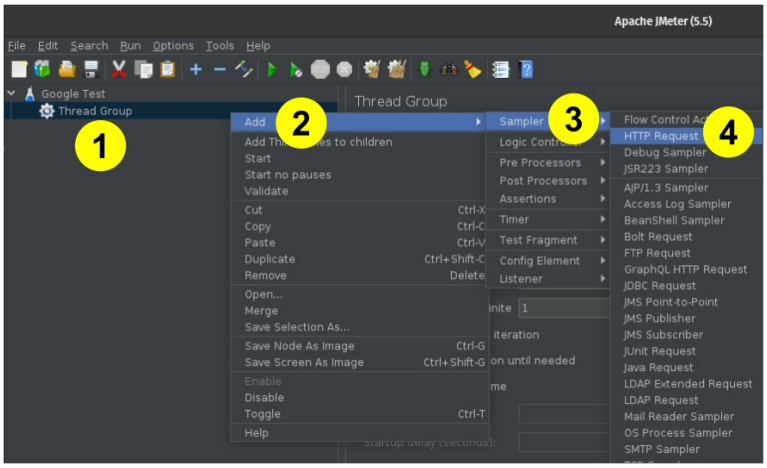
The image depicts a series of four nested menus needed to add a thread group.

From here, we can set basic elements in `Thread Properties`(1).
- `Number of Threads (users)` - this is the number of virtual users (default 1).
- `Ramp-up period (in seconds)` - this is the time in seconds it takes to reach our number of users (default 1).
- `Loop Count` - this is the number of times our script will be executed (default 1).

The image depicts the thread properties.

We can also change other settings such as the `Thread Group Name`, but for now let's leave it with the default values.

To run the test, we need to tell JMeter the address of the server that will be the subject to the load. Right-click on `Thread Group`(1) and select `Add`(2). Then hover over `Sampler`(3) and click on `HTTP request`(4).



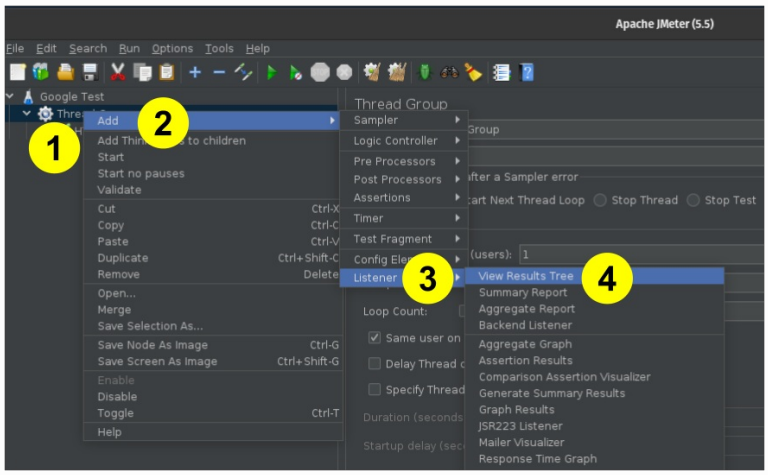
The image depicts the four menus needed to add an HTTP request.

In the `Basic` section of the HTTP request, set the protocol to `https`(1), the server name to `google.com`(2), and verify that we are using a GET request(3). Leave any other fields with their default values.

The image depicts the settings for an HTTPS GET request to

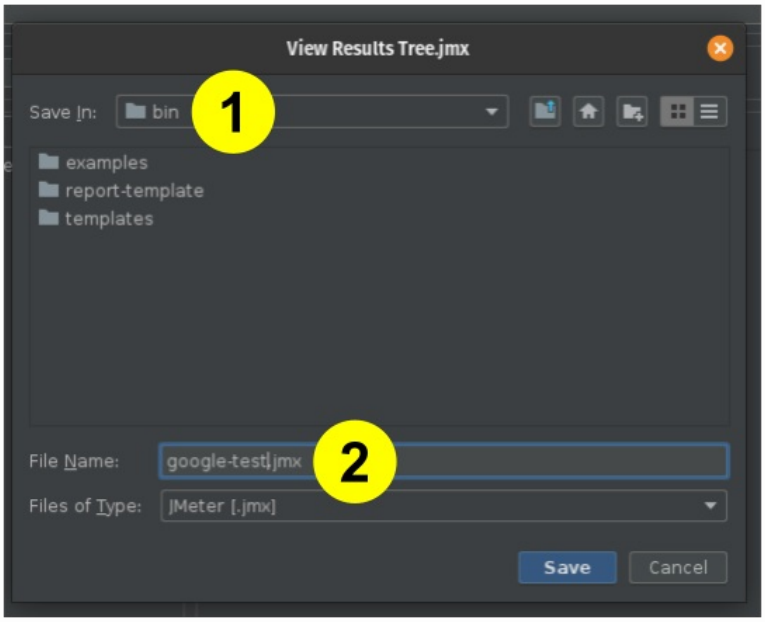
google.com.

The last thing we need to do is add a result tree listener to our thread group. Start by right-clicking on `Thread Group`(1). Select `Add`(2) and move the mouse to the bottom of the list to find `Listener`(3). Finally, click on `View Results Tree`(4).



The image depicts the steps needed to add a view result tree listener to the thread group.

The last thing we need to do is save our test plan. To do this, right click on `Google Test` and select `Save Selection As...`. **Important**, the test must be saved to the `/bin` folder (where JMeter is located). Name the test `google-test.jmx`(2) and verify that it is in the `/bin` directory(1).



The image depicts saving the google-test.jmx file in the bin directory.

Running the Script

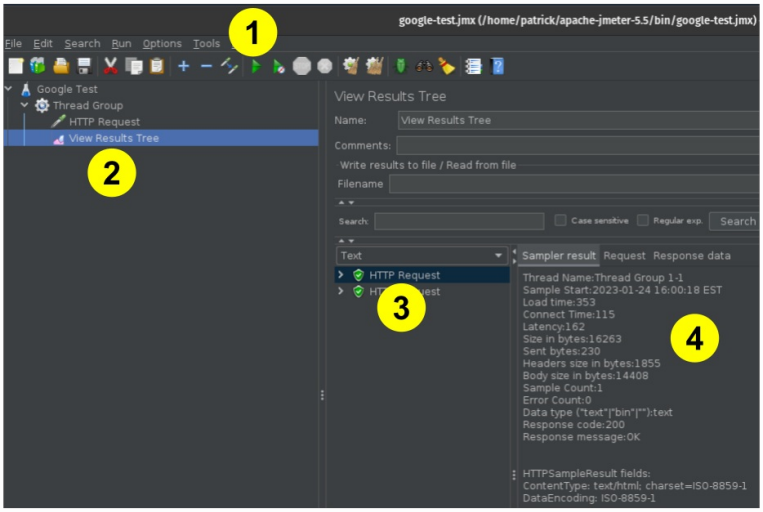
info

JMeter

The activities on this page require JMeter. Make sure it is running and that you have loaded the `Google Test` test plan before continuing.

Running the Test

Click the green triangle icon(1) to run our test plan. To view the results, click on `View Results Tree`(2), and then click on `HTTP Request`(3). You should see the results of the test in a panel to the right(4).



The image depicts a number 1 next to a green triangle icon.

The actual results should look something like this:

```
Thread Name:Thread Group 1-1
Sample Start:2023-01-09 11:00:27 MSK
Load time:1641
Connect Time:342
Latency:483
Size in bytes:4658
Sent bytes:463
Headers size in bytes:3160
Body size in bytes:1498
Sample Count:1
Error Count:0
Data type ("text"|"bin"|""):text
Response code:200
Response message:OK


HTTPSampleResult fields:
ContentType: text/html; charset=utf-8
DataEncoding: utf-8
```

Here we see all kinds of information about the test. The main thing we see is `Response code:200`, which means everything worked as expected. This is an example of the simplest script in JMeter.

We can also run JMeter tests from the command line. Open a terminal and navigate to the `/bin` directory where you saved the `google-test.jmx` test plan. If you are on a Mac or Linux, enter the command below and press `ENTER`.

```
./jmeter.sh -n -t google-test.jmx -l log.jtl
```

If you are on Windows, enter the following command:

```
jmeter -n -t google-test.jmx -l log.jtl
```

▼ **Windows path**

If you run this command, you may see an error "'Java' is not recognized as an internal or external command" or "Not able to find Java executable or version. Please check your Java installation." You need to configure the environment variables, `JAVA_HOME` and `PATH`. The Steps are as follows:

- Right click on My Computer
- Select Properties
- Select Advanced System Settings
- Select the Advanced tab

- Select Environment Variables
- Select Path under System Variables
- Click on the Edit button
- In Variable value editor paste this at the start of the line

```
C:\Program Files\Java\jdk1.7.0_72\bin;
```

- Click Ok then Ok again
- Restart command prompt otherwise it won't see the change to the path variable
- Type `java -version` in the command prompt.

You should configure the path for both user and the system.

The output should look like the sample below. Notice how the output is not as complete as using the JMeter GUI.

```
patrick@lappy-486:~/apache-jmeter-5.5/bin$ ./jmeter -n -t
google-test.jmx -l log.jtl
Creating summariser <summary>
Created the tree successfully using google-test.jmx
Starting standalone test @ 2023 Jan 24 16:04:49 EST
(1674594289165)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump
message on port 4445
Warning: Nashorn engine is planned to be removed from a future
JDK release
summary =      1 in 00:00:01 =    1.5/s Avg:   553 Min:   553
Max:   553 Err:     0 (0.00%)
Tidying up ...    @ 2023 Jan 24 16:04:50 EST (1674594290077)
... end of run
```

However, the `-l log.jtl` portion of the test command stores test information in a log file. If you are on a Mac or Linux, enter the command below to see the contents of the log file. **Note**, you should still be in the `/bin` directory.

```
cat log.jtl
```

On Windows, enter the following command to see the log file:

```
type log.jtl
```

You should see output similar to the sample below. You have the same information as the GUI test, but it is not as readable for humans. This information, however, can be easily parsed by a computer.

```
timeStamp,elapsed,label,responseCode,responseMessage,threadName,
dataType,success,failureMessage,bytes,sentBytes,grpThreads,allTh
reads,URL,Latency,IdleTime,Connect
1674594289518,553,HTTP Request,200,OK,Thread Group 1-
1,text,true,,16266,230,1,1,https://www.google.com/,371,0,316
1674594289518,377,HTTP Request-0,301,Moved Permanently,Thread
Group 1-1,text,true,,791,113,1,1,https://google.com/,371,0,316
1674594289898,173,HTTP Request-1,200,OK,Thread Group 1-
1,text,true,,15475,117,1,1,https://www.google.com/,168,0,86
```

We can try it on different sites and change the Number of Threads (users), Ramp-Up period and Loop Count.

challenge

Try these variations:

- Change the website to `npr.org`
- Change the number of users to `100`
- Change the loop count to `10`

Run the tests again and compare the output.

▼ **Large numbers**

If you start to use large numbers in the HTTP Request, you may find that JMeter returns errors. If you want to stress a system, you should use other tools.

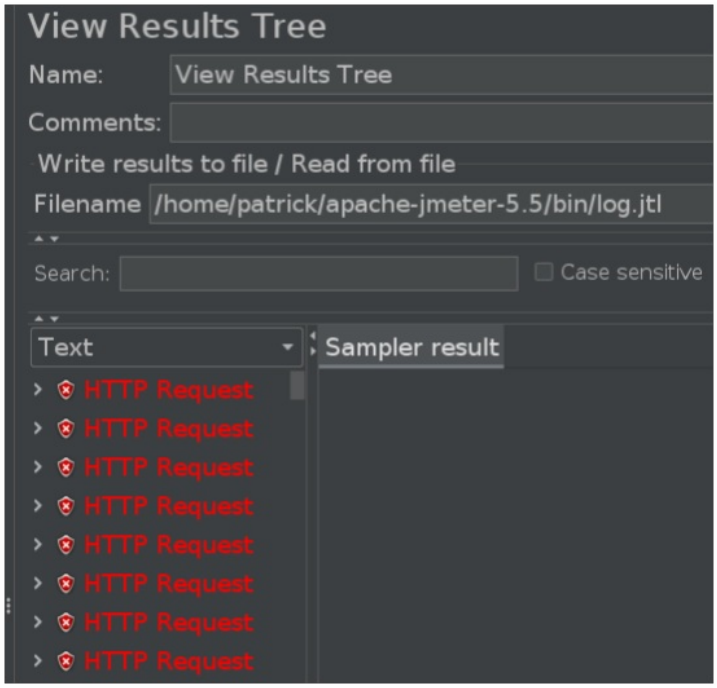


Image shows errors for each HTTP Request

Generating Reports

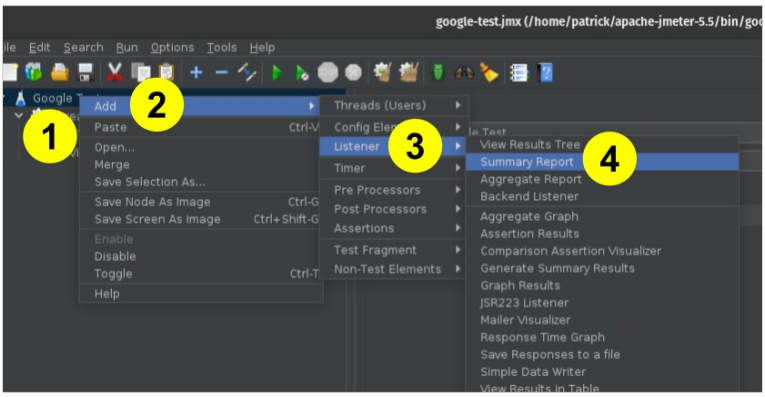
info

JMeter

The activities on this page require JMeter. Make sure it is running and that you have loaded the `Google Test` test plan before continuing.

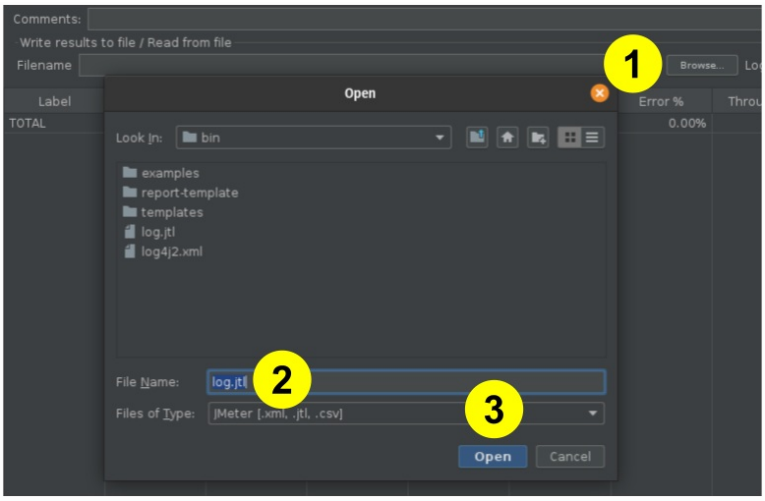
Summary Report

As your tests grow in complexity, so do their results. JMeter has the ability to generate reports that present important test data as tables and graphs. The most simple report is the `Summary Report`. Right-click on the `Google Test` test plan(1) and select `Add`(2). Click on `Listener`(3) and finally select `Summary Report`(4).



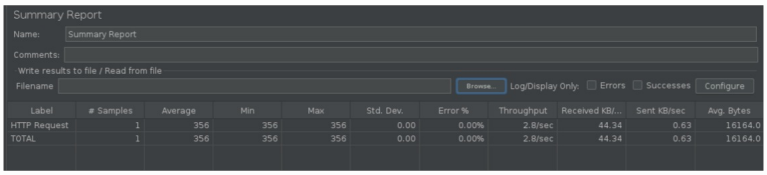
The image depicts a series of four nested menus needed to add a summary report.

To view the test results, need to click `Browse`(1). JMeter will ask you which file you want to use for the report. Find the `log.jtl`(2) file we referenced on the last page. Click `Open` to view the log file as a report.



The image depicts the steps needed to open a log file for a summary report.

What was once difficult to read as a human, is now nicely presented in table form.



The image depicts the contents of a log file in table form.

The list below explains the different columns in the table.

- `Samples` - represents the number of requests to the address specified in the configuration.
- `Average` - represents the average response time in milliseconds.
- `Min` - represents the minimum response time in milliseconds.
- `Max` - represents the maximum response time in milliseconds.
- `Std. Dev.` - represents the measure of the standard deviation. This indicator allows you to evaluate how much the values from the sample (the result of the test run) differ from the calculated average value.
- `Error %` - represents the percentage of errors returned by the server.
- `Throughput` - represents how many requests per second the server handles.
- `Received and Sent KB/sec` - the amount of data sent and received.
- `Avg.Bytes` - the average amount of received data.

Graphical Results in JMeter

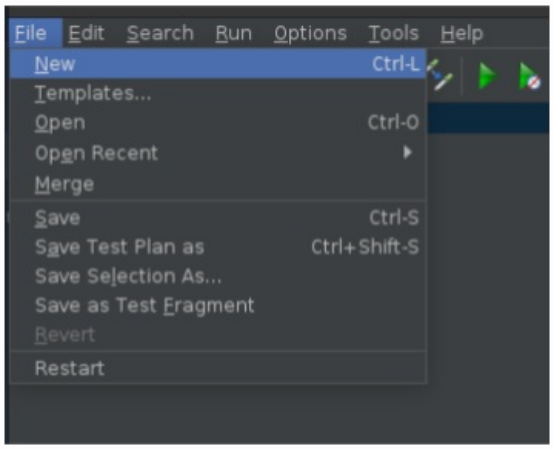
info

JMeter

The activities on this page require JMeter. Make sure it is running.

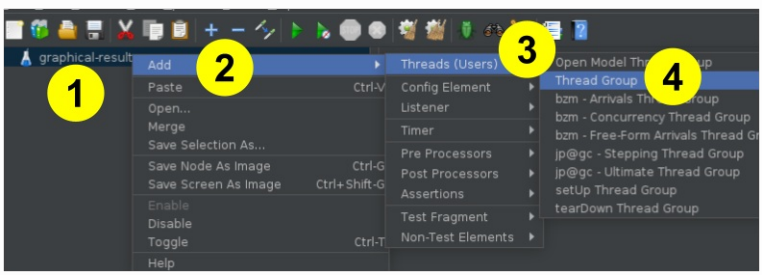
Create a New Test Plan

We can also view the results from JMeter in a graphical manner. Let's start by creating a new test plan. In the menu bar, select `File` and then select `New` to create a new test plan. Give it the name `graphical-results`.



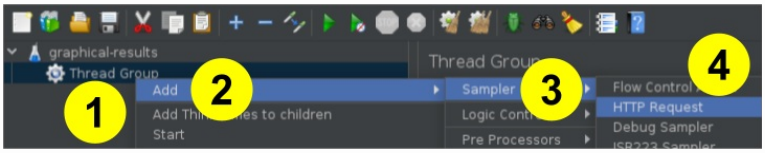
The image depicts using the file menu to create a new test plan.

Next, we need to add a thread group to the test plan. Right-click on the test plan(1). Select `Add`(2), and then click on `Threads`(3). Finally, click `Thread Group`(4). For now, let's leave the default values for the number of threads, ramp-up period, and loop count.



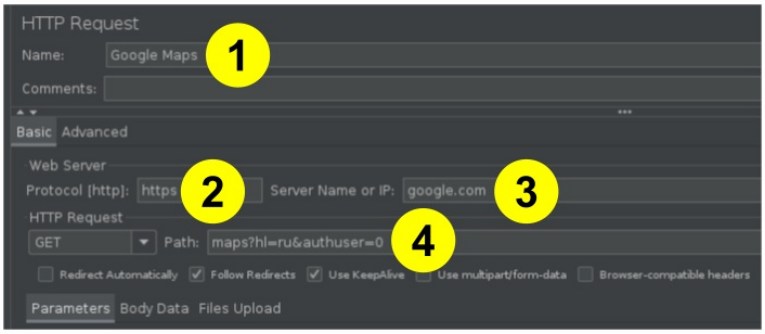
The image depicts how to add a thread group.

This time, we are going to add two different HTTP requests. These will correspond to different lines on our graph. Right-click on the thread group(1) and select `Add`(2). This time select `Sampler`(3) and then click on `HTTP Request`(4).



The image depicts how to add an HTTP request.

Name this request `Google Maps`(1). Use `https` for the protocol(2) and `google.com` as the server name(3). Enter `maps?hl=ru&authuser=0` as the path(4).



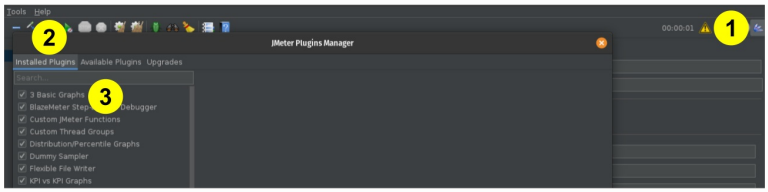
The image shows the correct values for the name, protocol, server, and path for requesting Google Maps.

Create a second HTTP requst for the thread group. Name this one `Google Images`(1). Again, we are going to use `https` for the protocol(2) and `google.com` as the server(3). Use `imghp?hl=ru&authuser=0&ogbl` for the path(4).

The image shows the correct values for the name, protocol, server, and path for requesting Google Images.

Viewing Graphical Results

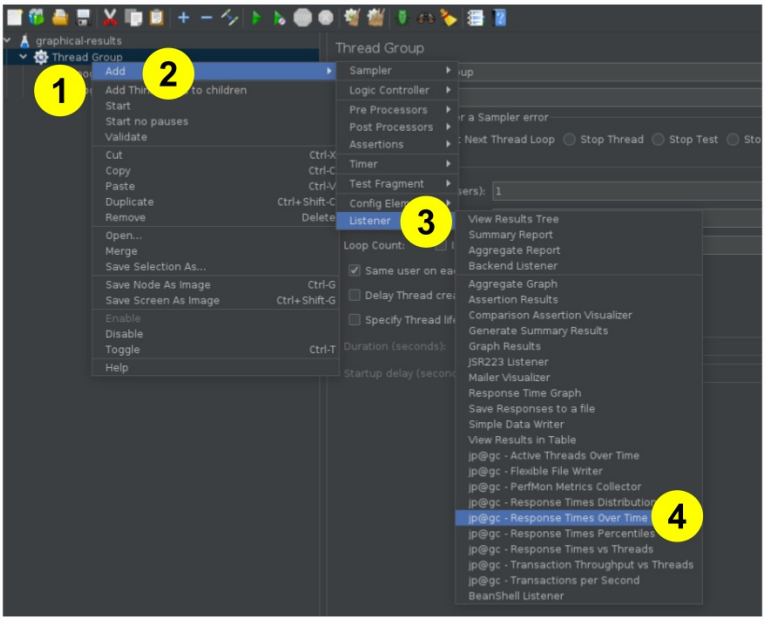
Before we view our results as a graph, we need to make sure that we have properly installed the `3 Basic Graphs` plugin. Start by opening the plugin manager by click its icon in the top-right corner(1). Under the `Installed Plugins` section(2), make sure you see `3 Basic Graphs`(3).



The image shows how to check for the 3 Basic Graphs plugin.

If this plugin is not installed, please refer to this page on how to install plugins in JMeter.

Right-click on the thread group(1). Select `Add`(2) and then select `Listener`(3). Towards the bottom of the list, select `jp@gc Response Times over Time`(4)



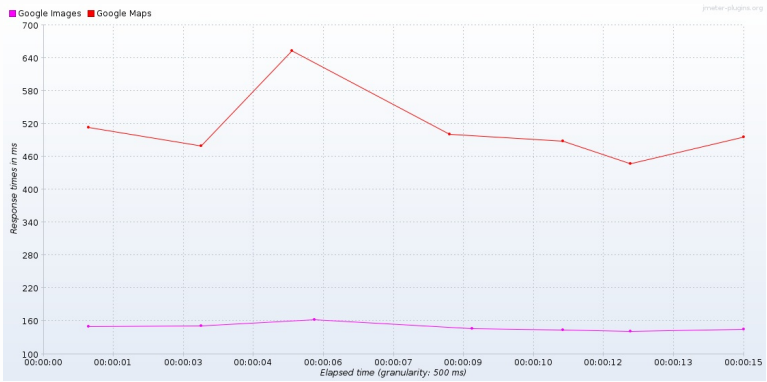
The image shows how to add a graphical listener to the thread group.

Click on the newly added listener. Then click on the green triangle to run JMeter. Save your work if you have not already done so. This should add a single data point to the graph. Click the green triangle a few more times. This will add new data points to the graph. The colored lines represent the response times for Google Images and Google Maps over time.

info

Important

Your graph will not look like the one below. There are many external factors that can affect the response times of websites. Our tests are not being run under identical circumstances so our results will be slightly different. However, you should still see how the results change over time.



The image shows a graphical representation of the test plan data.

challenge

Try these variations:

Change the values for the number of threads, ramp-up period, and loop count in the `Thread Group`. Run the test plan a few more times and see how the graphs change.

Working with Cookies

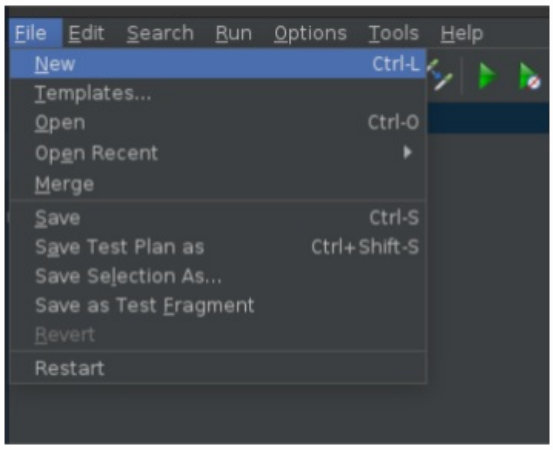
info

JMeter

The activities on this page require JMeter. Make sure it is running.

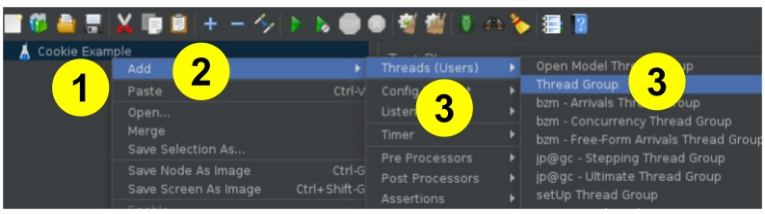
Viewing Cookies

Sometimes you may find yourself needing to work with cookies. JMeter has tools for this. Create a new test plan. In the menu bar, select `File` and then select `New` to create a new test plan. Give it the name `Cookie Example`.



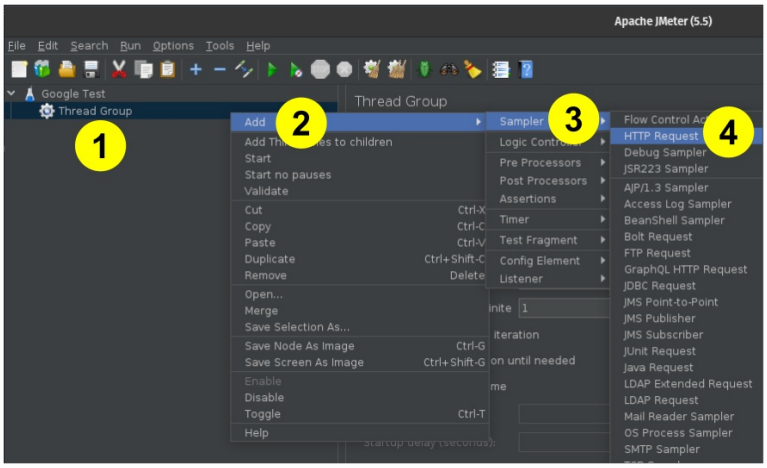
The image depicts using the file menu to create a new test plan.

Just as before, we need to create a thread group. Right-click on the test plan(1). Select `Add`(2), and then click on `Threads`(3). Finally, click `Thread Group`(4). For now, let's leave the default values for the number of threads and ramp-up period. However, we want to set the loop count to `2`.



The image depicts how to add a thread group.

Add an HTTP request to the test plan by right-clicking on the thread group(1). Select `Add`(2), then click on `Sampler`(3), and finally select `HTTP Request`(4).

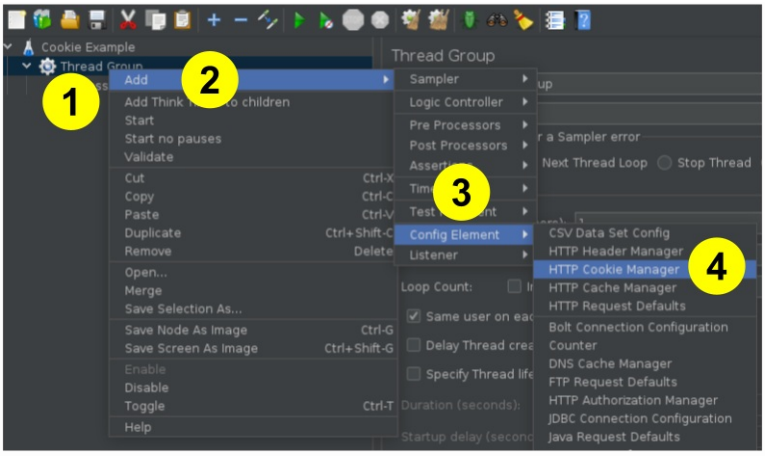


The image depicts how to add an HTTP request to the thread group.

Name the request `Password reset`(1). For the protocol(2) enter `http`, and set the server name(3) to `blazedemo.com`. This is a demo website maintained by BlazeMeter, a purveyor of testing tools. For the path(4), set this to `password/reset`.

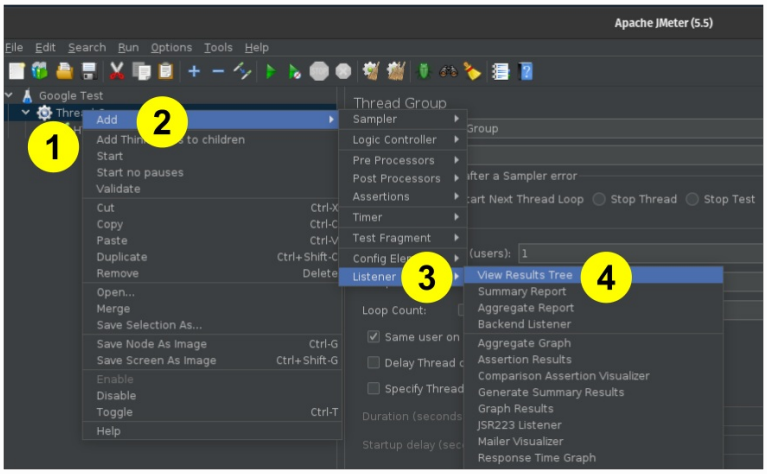
The image depicts how to set up an HTTP request for the password reset page.

To view cookies, we need to add a cookie manager. Right-click on the thread group(1) and select `Add`(2). Toward the bottom of the list, click on `Config Element`(3). Then select `HTTP Cookie Manager`(4).



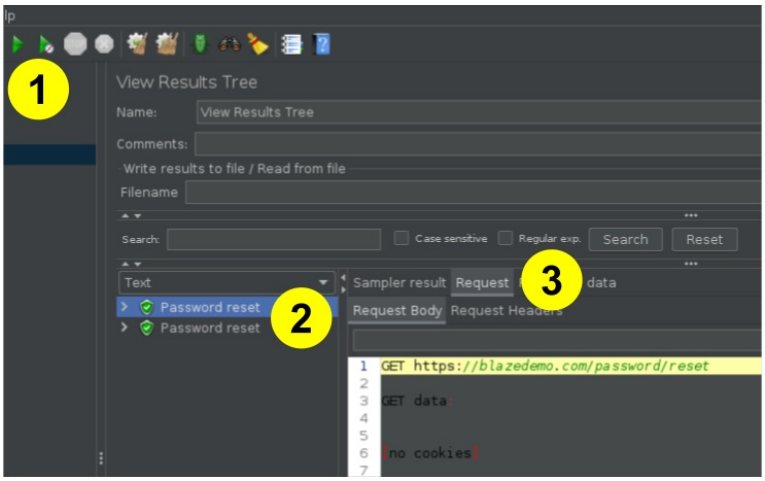
The image depicts how to set up an HTTP request for the password reset page.

Finally, let's add a result tree listener to our thread group. Start by right-clicking on `Thread Group`(1). Select `Add`(2) and move the mouse to the bottom of the list to find `Listener`(3). Finally, click on `View Results Tree`(4).



The image depicts the steps needed to add a view result tree listener to the thread group.

Save your work. Then click on the newly added results tree listener and click the green triangle(1) to run the test plan. You should see `Password reset`(2) appear twice since our HTTP request has a loop count of 2. Click on the first `Password reset` result and then click on the `Request` tab(3). Notice how there are no cookies for this request. Now click on the second `Password reset`. You should now see a cookie present.



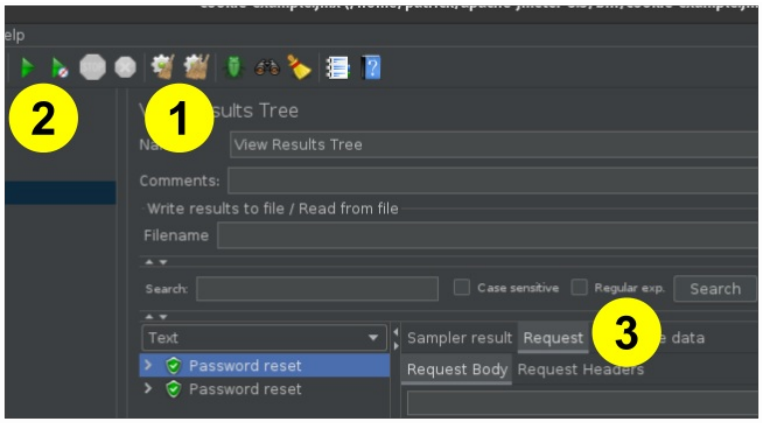
The image depicts how to clear the results from the listener.

Manipulating Cookies

You have seen how cookies can be passed in the header from request to request. Sometimes it is preferable that no cookies are passed when testing. We can do this with the cookie manager. Click on the HTTPS cookie manager and then check the box(1) that clears cookies after each iteration.

The image depicts how to clear cookies between iterations.

Go back to the results tree listener. Click the icon with the gear and broom(1) to clear the results from the listener. Then click the green triangle to run the test plan once again. Inspect the `Request` tab(3) for both responses. You should not see any cookies.



The image depicts how to clear the results from the listener.

JMeter also lets you specify your own cookies, which can be helpful in testing websites. Click on the HTTP cookie manager. At the bottom of the page, click on the `Add`(1) button. Enter `TEST` as the name(2) of the cookie. Give it a value(3) of `example-cookie`. Then set the domain(4) to `blazedemo.com`. Clear the results from the listener and then run the test plan again. Each request should now contain the custom cookie you created.

The image depicts how to add a custom cookie.

You may have noticed other buttons at the bottom of the cookie manager. The delete button removes custom cookies that you no longer need. The load button allows you to import a list of custom cookies, and the save button allows you to export custom cookies to an external file.

Recording Controller

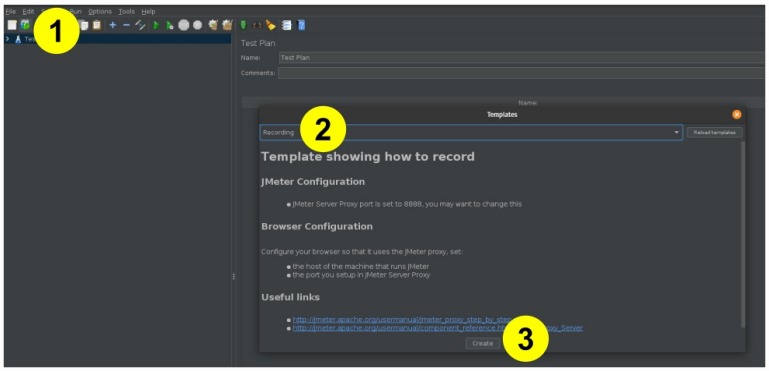
info

JMeter

The activities on this page require JMeter. Make sure it is running.

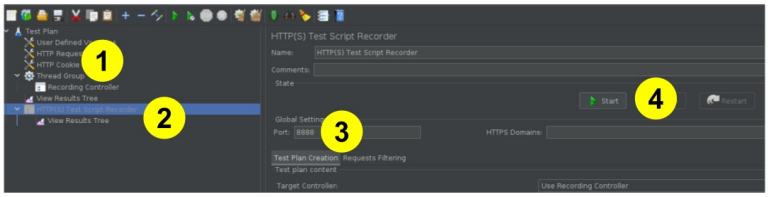
Loading a Template

JMeter gives you the ability to record your interactions with a browser. This is helpful for testing purposes because we do not have to configure JMeter by hand. Our actions will form the basis for testing. Better still, JMeter provides a template that creates a test plan already set up for recording. In the top-left corner click the icon that looks like books(1). This opens a window that has a drop-down menu of the available templates. Select `Recording`(2) from the list. Then click `Create`(3).



The image show how to open the recording template.

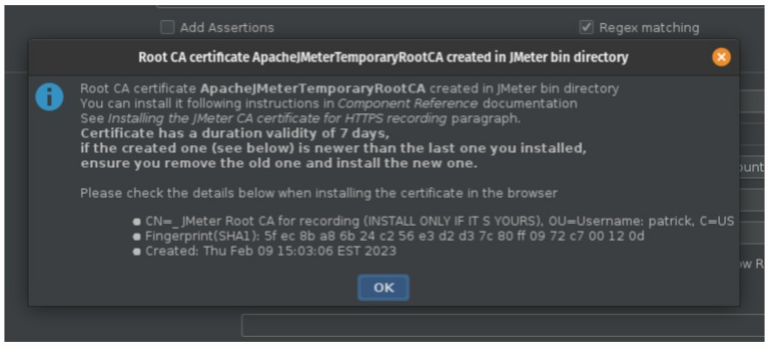
For our example, we are not going to change any of the parameters. Click `Create` once more. JMeter will load a test plan with all sorts of components. If you look under the thread group you will see the `Recording Controller`(1). Click on the `HTTP(S) Test Script Recorder`(2). This loads a page with the settings for our recorder. Note that this is working on port 8888(3). Click the `Start`(4) button.



The image shows all of the components of the recording template.

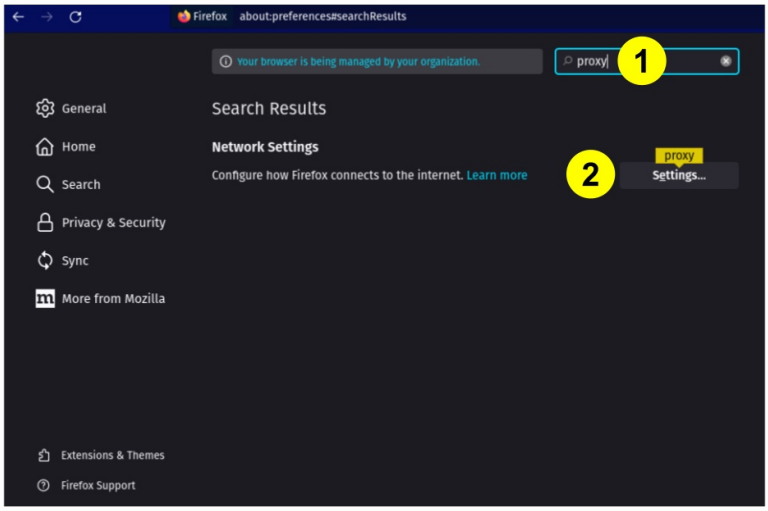
Preparing the Browser

JMeter is able to record browser actions by acting as a proxy between your computer and the website. We need to configure our system to work with the proxy. JMeter automatically downloads a certificate that you need to upload to your browser. Without it, you will not be able to record your interactions with the browser.



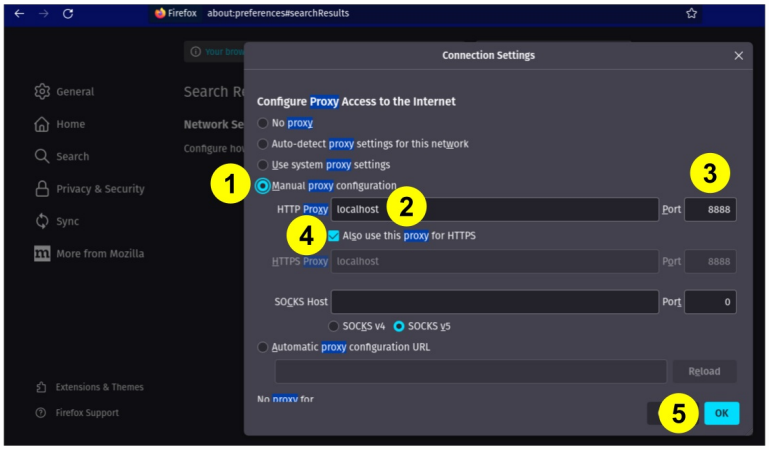
The image shows the JMeter message about using their certificate.

This example will use the Firefox browser, but this can be done in other browsers as well. Open the browser and open its settings. In the search box(1) type “proxy”. Then click the `Settings`(2) button from the search results.



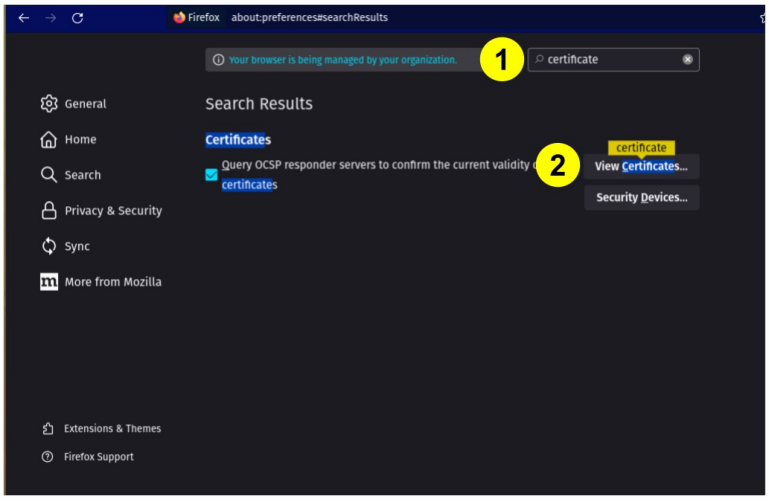
The image shows the settings search results for “proxy”.

In the proxy settings menu, select `Manual proxy configuration`(1). In the `HTTP proxy` field(2), enter `localhost`. Set the port to `8888`(3). We want to use the proxy for HTTPS(4). Then click `OK`(5).



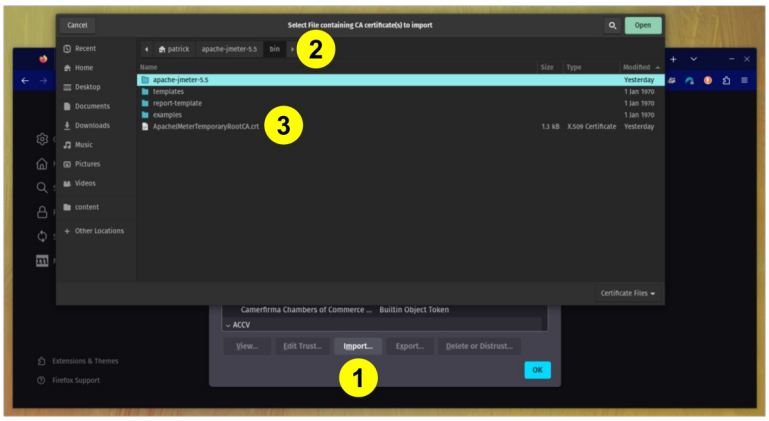
The image shows the settings proxy.

Close the window for the proxy settings. Go back to the search bar and enter “certificate”(1). Click on the button that says `View Certificates...`(2).



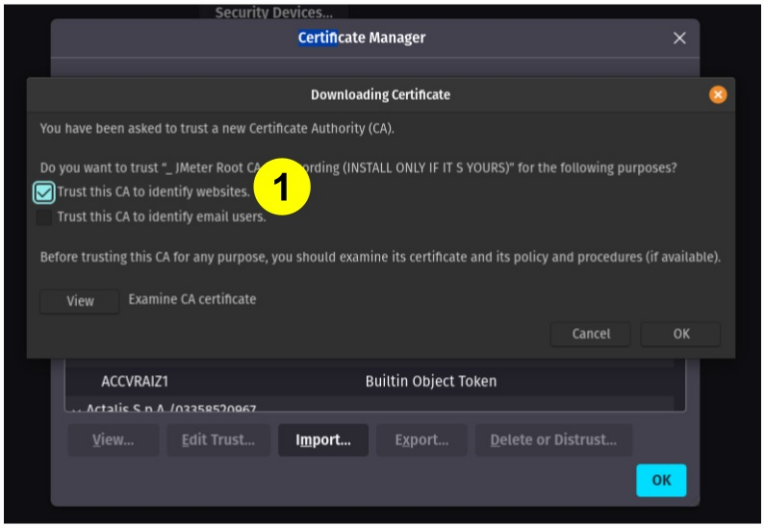
The image shows the settings search results for “certificate”.

In the next window, click `Import`(1) at the bottom. Navigate your window manager to the `bin` directory of your JMeter installation(2). This is where we will find the certificate. Select the certificate(3) and then upload it to the browser.



The image shows the temporary certificate in the bin directory of JMeter.

Before you can use this certificate, your browser is going to ask you if you trust this certificate. Check the first option so that your browser trusts this certificate.



The image shows the dialog box to trust the JMeter certificate. The box is checked to trust the CA to identify websites.

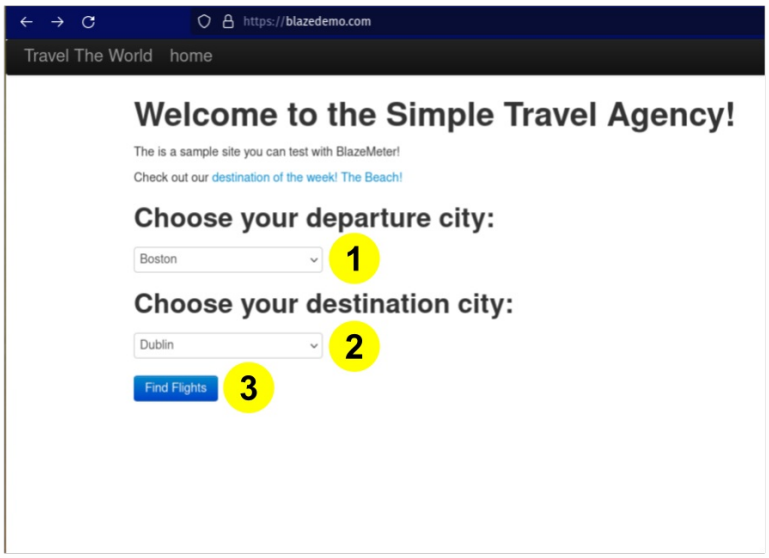
Our browser is now ready to use JMeter as a proxy so we can record our interactions.

Recording Browser Actions

Both JMeter and our browser are now properly configured. We already clicked the “start” button in our test script recorder. Now all we need to do is navigate to a website and click a few links. Go to the following website:

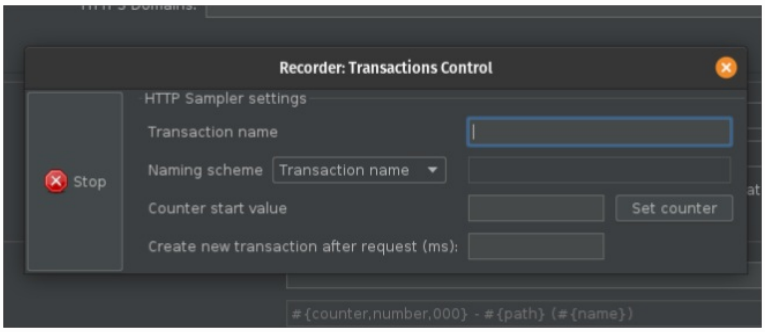
```
www.blazedemo.com
```

Select a couple of cities(1)(2) for the hypothetical plane ticket and click `Find Flights`(3).



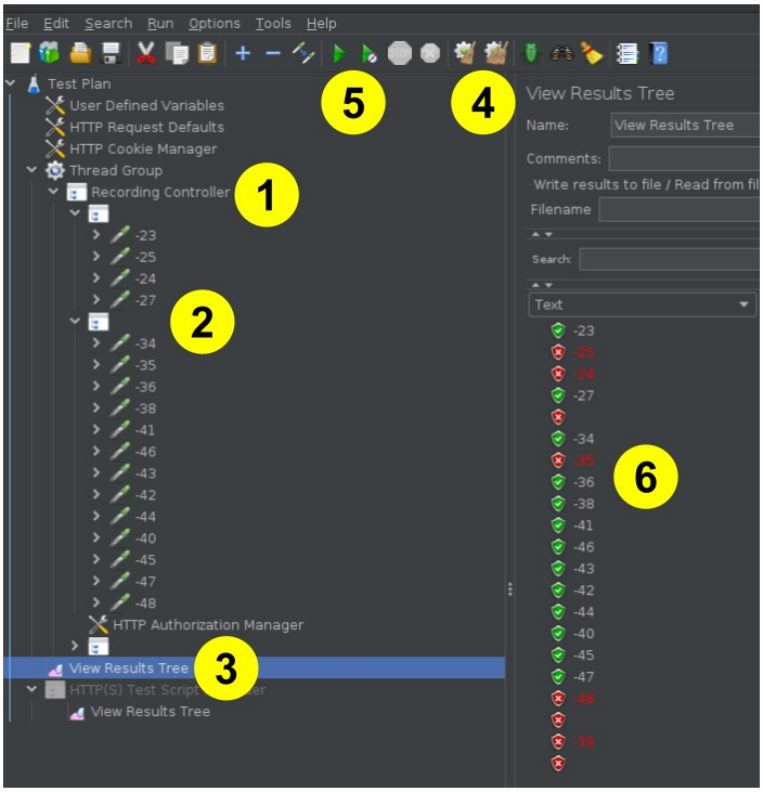
The image shows the demo site which has a fictitious flight selector.

Go back to JMeter. You should see a secondary window that opens when running the test script recorder. Press the `Stop` button.



The image shows the button to stop the test script recording.

If you click on the drop-down button for the `Recording Controller`(1) in JMeter you will now see several samplers(2) added to the thread group. Click on the results tree(3). Then clear the results(4). Run the test plan once more(5). You should see the results populate the results tree(6) once again. The test script recorder allows you to generate a test plan through your browser interactions instead of manually configuring samplers.

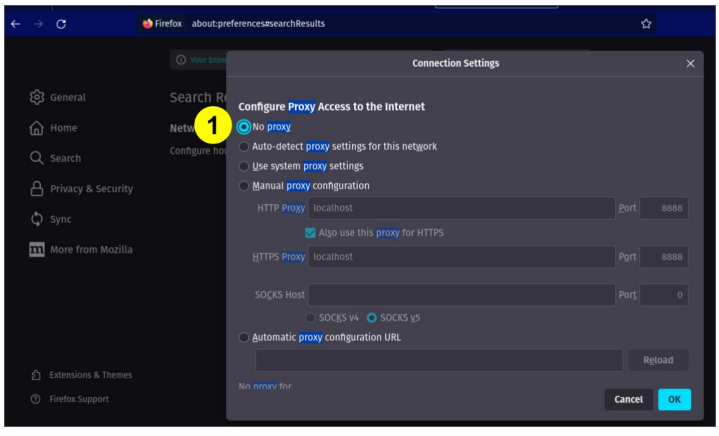


The image shows the results of the test script recorder.

info

Important

If you want to use your browser again, you need to go back into the proxy settings and select `No proxy`(1). Otherwise, the browser will not connect to any websites unless you are running JMeter and the test script recorder.



The image shows how to turn off the proxy.