**User's Manual（用户手册）**

Click on the section name to go straight to the section. Click on the "+" to go to the relevant section of the detailed section list, where you can select individual subsections.

**Section Summary**

* [Changes](http://jmeter.apache.org/changes.html)
* [+](http://jmeter.apache.org/usermanual/index.html#get-started)...[1. Getting Started](http://jmeter.apache.org/usermanual/get-started.html)
* [+](http://jmeter.apache.org/usermanual/index.html#build-test-plan)...[2. Building a Test Plan](http://jmeter.apache.org/usermanual/build-test-plan.html)
* [+](http://jmeter.apache.org/usermanual/index.html#test_plan)...[3. Elements of a Test Plan](http://jmeter.apache.org/usermanual/test_plan.html)
* [+](http://jmeter.apache.org/usermanual/index.html#build-web-test-plan)...[4. Building a Web Test Plan](http://jmeter.apache.org/usermanual/build-web-test-plan.html)
* [+](http://jmeter.apache.org/usermanual/index.html#build-adv-web-test-plan)...[5. Building an Advanced Web Test Plan](http://jmeter.apache.org/usermanual/build-adv-web-test-plan.html)
* [+](http://jmeter.apache.org/usermanual/index.html#build-db-test-plan)...[6. Building a Database Test Plan](http://jmeter.apache.org/usermanual/build-db-test-plan.html)
* [+](http://jmeter.apache.org/usermanual/index.html#build-ftp-test-plan)...[7. Building an FTP Test Plan](http://jmeter.apache.org/usermanual/build-ftp-test-plan.html)
* [+](http://jmeter.apache.org/usermanual/index.html#build-ldap-test-plan)...[8a. Building an LDAP Test Plan](http://jmeter.apache.org/usermanual/build-ldap-test-plan.html)
* [+](http://jmeter.apache.org/usermanual/index.html#build-ldapext-test-plan)...[8b. Building an Extended LDAP Test Plan](http://jmeter.apache.org/usermanual/build-ldapext-test-plan.html)
* [+](http://jmeter.apache.org/usermanual/index.html#build-ws-test-plan)...[9. Building a Webservice Test Plan](http://jmeter.apache.org/usermanual/build-ws-test-plan.html)
* [+](http://jmeter.apache.org/usermanual/index.html#build-jms-point-to-point-test-plan)...[10. Building a JMS Point to point Test Plan](http://jmeter.apache.org/usermanual/build-jms-point-to-point-test-plan.html)
* [+](http://jmeter.apache.org/usermanual/index.html#build-jms-topic-test-plan)...[11. Building a JMS Topic Test Plan](http://jmeter.apache.org/usermanual/build-jms-topic-test-plan.html)
* [+](http://jmeter.apache.org/usermanual/index.html#build-monitor-test-plan)...[12. Building a Monitor Test Plan](http://jmeter.apache.org/usermanual/build-monitor-test-plan.html)
* [+](http://jmeter.apache.org/usermanual/index.html#listeners)...[13. Listeners](http://jmeter.apache.org/usermanual/listeners.html)
* [+](http://jmeter.apache.org/usermanual/index.html#remote-test)...[14. Remote Testing](http://jmeter.apache.org/usermanual/remote-test.html)
* [+](http://jmeter.apache.org/usermanual/index.html#generating-dashboard)...[15. Dashboard Report](http://jmeter.apache.org/usermanual/generating-dashboard.html)
* [+](http://jmeter.apache.org/usermanual/index.html#realtime-results)...[16. Real time Results](http://jmeter.apache.org/usermanual/realtime-results.html)
* [+](http://jmeter.apache.org/usermanual/index.html#best-practices)...[17. Best Practices](http://jmeter.apache.org/usermanual/best-practices.html)
* [+](http://jmeter.apache.org/usermanual/index.html#boss)...[18. Help! My boss wants me to load test our web app!](http://jmeter.apache.org/usermanual/boss.html)
* [+](http://jmeter.apache.org/usermanual/index.html#component_reference)...[19. Component Reference](http://jmeter.apache.org/usermanual/component_reference.html)
* [+](http://jmeter.apache.org/usermanual/index.html#properties_reference)...[20. Properties Reference](http://jmeter.apache.org/usermanual/properties_reference.html)
* [+](http://jmeter.apache.org/usermanual/index.html#functions)...[21. Functions](http://jmeter.apache.org/usermanual/functions.html)
* [+](http://jmeter.apache.org/usermanual/index.html#regex)...[22. Regular Expressions](http://jmeter.apache.org/usermanual/regular_expressions.html)
* [+](http://jmeter.apache.org/usermanual/index.html#hints)...[23. Hints and Tips](http://jmeter.apache.org/usermanual/hints_and_tips.html)
* [+](http://jmeter.apache.org/usermanual/index.html#glossary)...[24. Glossary](http://jmeter.apache.org/usermanual/glossary.html)
* [+](http://jmeter.apache.org/usermanual/index.html#history_future)...[25. History / Future](http://jmeter.apache.org/usermanual/history_future.html)

**Detailed Section List**

* [1. Getting Started](http://jmeter.apache.org/usermanual/get-started.html)
  + [1.0 Overview](http://jmeter.apache.org/usermanual/get-started.html#overview)
    - [Test plan building](http://jmeter.apache.org/usermanual/get-started.html#test_plan_building)
    - [Load Test running](http://jmeter.apache.org/usermanual/get-started.html#load_test_running)
    - [Load Test analysis](http://jmeter.apache.org/usermanual/get-started.html#load_test_analysis)
    - [Let's start](http://jmeter.apache.org/usermanual/get-started.html#lets_start)
  + [1.1 Requirements](http://jmeter.apache.org/usermanual/get-started.html#requirements)
    - [1.1.1 Java Version](http://jmeter.apache.org/usermanual/get-started.html#java_versions)
    - [1.1.2 Operating Systems](http://jmeter.apache.org/usermanual/get-started.html#os)
  + [1.2 Optional](http://jmeter.apache.org/usermanual/get-started.html#optional)
    - [1.2.1 Java Compiler](http://jmeter.apache.org/usermanual/get-started.html#opt_compiler)
    - [1.2.2 SAX XML Parser](http://jmeter.apache.org/usermanual/get-started.html#opt_sax)
    - [1.2.3 Email Support](http://jmeter.apache.org/usermanual/get-started.html#opt_email)
    - [1.2.4 SSL Encryption](http://jmeter.apache.org/usermanual/get-started.html#opt_ssl)
    - [1.2.5 JDBC Driver](http://jmeter.apache.org/usermanual/get-started.html#opt_jdbc)
    - [1.2.6 JMS client](http://jmeter.apache.org/usermanual/get-started.html#opt_jms)
    - [1.2.7 Libraries for ActiveMQ JMS](http://jmeter.apache.org/usermanual/get-started.html#libraries_activemq)
  + [1.3 Installation](http://jmeter.apache.org/usermanual/get-started.html#install)
  + [1.4 Running JMeter](http://jmeter.apache.org/usermanual/get-started.html#running)
    - [1.4.1 JMeter's Classpath](http://jmeter.apache.org/usermanual/get-started.html#classpath)
    - [1.4.2 Create Test Plan from Template](http://jmeter.apache.org/usermanual/get-started.html#template)
    - [1.4.3 Using JMeter behing a proxy](http://jmeter.apache.org/usermanual/get-started.html#proxy_server)
    - [1.4.4 Non-GUI Mode](http://jmeter.apache.org/usermanual/get-started.html#non_gui)
    - [1.4.5 Server Mode](http://jmeter.apache.org/usermanual/get-started.html#server)
    - [1.4.6 Overriding Properties Via The Command Line](http://jmeter.apache.org/usermanual/get-started.html#override)
    - [1.4.7 Logging and Error Messages](http://jmeter.apache.org/usermanual/get-started.html#logging)
    - [1.4.8 Full list of command-line options](http://jmeter.apache.org/usermanual/get-started.html#options)
    - [1.4.9 non-GUI shutdown](http://jmeter.apache.org/usermanual/get-started.html#shutdown)
  + [1.5 Configuring JMeter](http://jmeter.apache.org/usermanual/get-started.html#configuring_jmeter)

* [2. Building a Test Plan](http://jmeter.apache.org/usermanual/build-test-plan.html)
  + [2.1 Adding and Removing Elements](http://jmeter.apache.org/usermanual/build-test-plan.html#add_remove)
  + [2.2 Loading and Saving Elements](http://jmeter.apache.org/usermanual/build-test-plan.html#load_save)
  + [2.3 Configuring Tree Elements](http://jmeter.apache.org/usermanual/build-test-plan.html#config_element)
  + [2.4 Saving the Test Plan](http://jmeter.apache.org/usermanual/build-test-plan.html#save)
  + [2.5 Running a Test Plan](http://jmeter.apache.org/usermanual/build-test-plan.html#run)
  + [2.6 Stopping a Test](http://jmeter.apache.org/usermanual/build-test-plan.html#stop)
  + [2.7 Error reporting](http://jmeter.apache.org/usermanual/build-test-plan.html#error_reporting)

* [3. Elements of a Test Plan](http://jmeter.apache.org/usermanual/test_plan.html)
  + [3.1 Thread Group](http://jmeter.apache.org/usermanual/test_plan.html#thread_group)
  + [3.2 Controllers](http://jmeter.apache.org/usermanual/test_plan.html#controllers)
    - [3.2.1 Samplers](http://jmeter.apache.org/usermanual/test_plan.html#samplers)
    - [3.2.2 Logic Controllers](http://jmeter.apache.org/usermanual/test_plan.html#logic_controller)
    - [3.2.3 Test Fragments](http://jmeter.apache.org/usermanual/test_plan.html#test_fragments)
  + [3.3 Listeners](http://jmeter.apache.org/usermanual/test_plan.html#listeners)
  + [3.4 Timers](http://jmeter.apache.org/usermanual/test_plan.html#timers)
  + [3.5 Assertions](http://jmeter.apache.org/usermanual/test_plan.html#assertions)
  + [3.6 Configuration Elements](http://jmeter.apache.org/usermanual/test_plan.html#config_elements)
  + [3.7 Pre-Processor Elements](http://jmeter.apache.org/usermanual/test_plan.html#preprocessors)
  + [3.8 Post-Processor Elements](http://jmeter.apache.org/usermanual/test_plan.html#postprocessors)
  + [3.9 Execution order](http://jmeter.apache.org/usermanual/test_plan.html#executionorder)
  + [3.10 Scoping Rules](http://jmeter.apache.org/usermanual/test_plan.html#scoping_rules)
  + [3.11 Properties and Variables](http://jmeter.apache.org/usermanual/test_plan.html#properties)
  + [3.12 Using Variables to parameterise tests](http://jmeter.apache.org/usermanual/test_plan.html#using_variables)

* [4. Building a Web Test Plan](http://jmeter.apache.org/usermanual/build-web-test-plan.html)
  + [4.1 Adding Users](http://jmeter.apache.org/usermanual/build-web-test-plan.html#adding_users)
  + [4.2 Adding Default HTTP Request Properties](http://jmeter.apache.org/usermanual/build-web-test-plan.html#adding_defaults)
  + [4.3 Adding Cookie Support](http://jmeter.apache.org/usermanual/build-web-test-plan.html#adding_cookie_support)
  + [4.4 Adding HTTP Requests](http://jmeter.apache.org/usermanual/build-web-test-plan.html#adding_requests)
  + [4.5 Adding a Listener to View/Store the Test Results](http://jmeter.apache.org/usermanual/build-web-test-plan.html#adding_listener)
  + [4.6 Logging in to a web-site](http://jmeter.apache.org/usermanual/build-web-test-plan.html#logging_in)

* [5. Building an Advanced Web Test Plan](http://jmeter.apache.org/usermanual/build-adv-web-test-plan.html)
  + [5.1 Handling User Sessions With URL Rewriting](http://jmeter.apache.org/usermanual/build-adv-web-test-plan.html#session_url_rewriting)
  + [5.2 Using a Header Manager](http://jmeter.apache.org/usermanual/build-adv-web-test-plan.html#header_manager)

* [6. Building a Database Test Plan](http://jmeter.apache.org/usermanual/build-db-test-plan.html)
  + [6.1 Adding Users](http://jmeter.apache.org/usermanual/build-db-test-plan.html#adding_users)
  + [6.2 Adding JDBC Requests](http://jmeter.apache.org/usermanual/build-db-test-plan.html#adding_requests)
  + [6.3 Adding a Listener to View/Store the Test Results](http://jmeter.apache.org/usermanual/build-db-test-plan.html#adding_listener)

* [7. Building an FTP Test Plan](http://jmeter.apache.org/usermanual/build-ftp-test-plan.html)
  + [7.1 Adding Users](http://jmeter.apache.org/usermanual/build-ftp-test-plan.html#adding_users)
  + [7.2 Adding Default FTP Request Properties](http://jmeter.apache.org/usermanual/build-ftp-test-plan.html#adding_defaults)
  + [7.3 Adding FTP Requests](http://jmeter.apache.org/usermanual/build-ftp-test-plan.html#adding_requests)
  + [7.4 Adding a Listener to View/Store the Test Results](http://jmeter.apache.org/usermanual/build-ftp-test-plan.html#adding_listener)

* [8a. Building an LDAP Test Plan](http://jmeter.apache.org/usermanual/build-ldap-test-plan.html)
  + [8a.1 Adding Users](http://jmeter.apache.org/usermanual/build-ldap-test-plan.html#adding_users)
  + [8a.2 Adding Login Config Element](http://jmeter.apache.org/usermanual/build-ldap-test-plan.html#add_login)
  + [8a.3 Adding LDAP Request Defaults](http://jmeter.apache.org/usermanual/build-ldap-test-plan.html#add_defaults)
  + [8a.4 Adding LDAP Requests](http://jmeter.apache.org/usermanual/build-ldap-test-plan.html#add_requests)
  + [8a.5 Adding a Response Assertion](http://jmeter.apache.org/usermanual/build-ldap-test-plan.html#ResponseAssertion)
  + [8a.6 Adding a Listener to View/Store the Test Results](http://jmeter.apache.org/usermanual/build-ldap-test-plan.html#add_listener)

* [8b. Building an Extended LDAP Test Plan](http://jmeter.apache.org/usermanual/build-ldap-test-plan.html)
  + [8b.1 Adding Users](http://jmeter.apache.org/usermanual/build-ldapext-test-plan.html#ext_adding_users)
  + [8b.2 Adding LDAP Extended Request Defaults](http://jmeter.apache.org/usermanual/build-ldapext-test-plan.html#add_login)
  + [8b.3 Adding LDAP Requests](http://jmeter.apache.org/usermanual/build-ldapext-test-plan.html#add_extrequests)
  + [8b.4 Adding a Listener to View/Store the Test Results](http://jmeter.apache.org/usermanual/build-ldapext-test-plan.html#add_ldapext_listener)

* [9. Building a Webservice Test Plan](http://jmeter.apache.org/usermanual/build-ws-test-plan.html)

* [10. Building a JMS Point to point Test Plan](http://jmeter.apache.org/usermanual/build-jms-point-to-point-test-plan.html)

* [11. Building a JMS topic Test Plan](http://jmeter.apache.org/usermanual/build-jms-topic-test-plan.html)

* [12. Building a Monitor Test Plan](http://jmeter.apache.org/usermanual/build-monitor-test-plan.html)

* [13. Introduction to Listeners](http://jmeter.apache.org/usermanual/listeners.html)

* [14. Remote Testing](http://jmeter.apache.org/usermanual/remote-test.html)

* [15. Dashboard Report](http://jmeter.apache.org/usermanual/generating-dashboard.html)
  + [15.1 Overview](http://jmeter.apache.org/usermanual/generating-dashboard.html#overview)
  + [15.2 Configuring Dashboard Generation](http://jmeter.apache.org/usermanual/generating-dashboard.html#configuration)
  + [15.3 Generating reports](http://jmeter.apache.org/usermanual/generating-dashboard.html#report)
  + [15.4 Default graphs](http://jmeter.apache.org/usermanual/generating-dashboard.html#default_graphs)
  + [15.5 Want to improve Report Dashboard](http://jmeter.apache.org/usermanual/generating-dashboard.html#development)

* [16. Real time Results](http://jmeter.apache.org/usermanual/realtime-results.html)

* [17. Best Practices](http://jmeter.apache.org/usermanual/best-practices.html)

* [18. Help! My boss wants me to load test our web app!](http://jmeter.apache.org/usermanual/boss.html)

* [19. Component Reference](http://jmeter.apache.org/usermanual/component_reference.html)

* [20. Properties reference](http://jmeter.apache.org/usermanual/properties_reference.html)

* [21. Functions](http://jmeter.apache.org/usermanual/functions.html)

* [22. Regular Expressions](http://jmeter.apache.org/usermanual/regular_expressions.html)

* [23. Hints and Tips](http://jmeter.apache.org/usermanual/hints_and_tips.html)

* [24. Glossary](http://jmeter.apache.org/usermanual/glossary.html)

* [25. History / Future](http://jmeter.apache.org/usermanual/history_future.html)

# 1. Getting Started

# 1.0 Overview

When using JMeter you will usually follow this process（按照下面的流程）:

## 1.0.1 Test plan building

To do that, you will [run JMeter in GUI Mode.](http://jmeter.apache.org/usermanual/get-started.html#running)（以GUI模式启动JMeter）  
Then you can either choose to record(记录) the application from a browser, or native application. You can use for that the menu *File* → *Templates...* → *Recording*

Note you can also manually（手工地） build your plan. Ensure you read this [documentation](http://jmeter.apache.org/usermanual/test_plan.html) to understand major concepts.

You will also debug it using one of these options:

* *Run* → *Start no pauses*
* *Run* → *Start*
* *Validate* on [Thread Group](http://jmeter.apache.org/usermanual/component_reference.html#Thread_Group)

and [View Results Tree](http://jmeter.apache.org/usermanual/component_reference.html#View_Results_Tree)（查看结果树） renderers（渲染） or Testers (CSS/JQUERY, JSON, Regexp, XPath).  
Ensure you follow [best-practices](http://jmeter.apache.org/usermanual/best-practices.html) when building your Test Plan.

## 1.0.2 Load Test running

Once your Test Plan is ready, you can start your Load Test. The first step is to configure the injectors that will run JMeter, this as for（与…一样） any other Load Testing tool includes:

* Correct machine sizing（校正机器数目） in terms of CPU, memory and network
* OS Tuning
* Java setup: Ensure you install the latest version of Java supported by JMeter
* **Correct sizing of Java Heap（java堆）**. By default JMeter runs with a heap of 512MB, this might not be enough for your test and depends on your test plan and number of threads you want to run

Once everything is ready, you will use Command-line mode (called [Non-GUI mode](http://jmeter.apache.org/usermanual/get-started.html#non_gui)) to run it for the Load Test.

Don't run load test using GUI mode !（不要使用GUI模式加载测试）

Using Non-GUI mode, you can generate a CSV (or XML) file containing results and have JMeter [generate an HTML report](http://jmeter.apache.org/usermanual/generating-dashboard.html) at end of Load Test. JMeter will by default provide a summary of load test while it's running.   
You can also have [real-time results](http://jmeter.apache.org/usermanual/realtime-results.html) during your test using [Backend Listener](http://jmeter.apache.org/usermanual/component_reference.html#Backend_Listener).

## 1.0.3 Load Test analysis

Once your Load Test is finished, you can use the HTML report to analyze your load test.

## 1.0.4 Let's start

The easiest way to begin using JMeter is to first [download the latest production release](http://jmeter.apache.org/download_jmeter.cgi) and install it. The release contains all of the files you need to build and run most types of tests, e.g. Web (HTTP/HTTPS), FTP, JDBC, LDAP, Java, JUnit and more.

If you want to perform JDBC testing, then you will, of course, need the appropriate JDBC driver from your vendor（要想测试数据库链接性能就要安装合适的jdbc驱动）. JMeter does not come with any JDBC drivers.

### JMeter includes the JMS（[Java Message Service](http://www.baidu.com/link?url=GsxLnvMdBImbK16M7wfX-a4w9UEIxnRCBP_gN-zQuwimEK6xXres_D67lzII5IfgD8vy_jYc-swMqRbfAR_Yd_)

的缩写） API jar, but does not include a JMS client implementation（客户端实现）. If you want to run JMS tests, you will need to download the appropriate jars from the JMS provider.

See the [JMeter Classpath](http://jmeter.apache.org/usermanual/get-started.html#classpath) section for details on installing additional jars.

Next, start JMeter and go through the [Building a Test Plan](http://jmeter.apache.org/usermanual/build-test-plan.html) section of the User Guide to familiarize yourself with JMeter basics (for example, adding and removing elements).

Finally, go through the appropriate section on how to build a specific type of Test Plan. For example, if you are interested in testing a Web application, then see the section [Building a Web Test Plan](http://jmeter.apache.org/usermanual/build-web-test-plan.html). The other specific Test Plan sections are:

* [Advanced Web Test Plan](http://jmeter.apache.org/usermanual/build-adv-web-test-plan.html)
* [JDBC](http://jmeter.apache.org/usermanual/build-db-test-plan.html)
* [FTP](http://jmeter.apache.org/usermanual/build-ftp-test-plan.html)
* [JMS Point-to-Point](http://jmeter.apache.org/usermanual/build-jms-point-to-point-test-plan.html)
* [JMS Topic](http://jmeter.apache.org/usermanual/build-jms-topic-test-plan.html)
* [LDAP](http://jmeter.apache.org/usermanual/build-ldap-test-plan.html)
* [LDAP Extended](http://jmeter.apache.org/usermanual/build-ldapext-test-plan.html)
* [WebServices (SOAP)](http://jmeter.apache.org/usermanual/build-ws-test-plan.html)

Once you are comfortable with building and running JMeter Test Plans, you can look into the various configuration elements (timers, listeners, assertions, and others) which give you more control over your Test Plans.

# 1.1 Requirements（安装要求）

JMeter requires that your computing environment meets some minimum requirements.

## 1.1.1 Java Version

JMeter requires a fully compliant JVM 7 or 8, we advise that you install latest minor version of those major versions. Java 9 is not tested completely as of JMeter 3.1.

Because JMeter uses only standard Java APIs, please do not file bug reports if your JRE fails to run JMeter because of JRE implementation issues.

## 1.1.2 Operating Systems

JMeter is a 100% Java application and should run correctly on any system that has a compliant Java implementation.

Operating systems tested with JMeter can be viewed on [this page](http://wiki.apache.org/jmeter/JMeterAndOperatingSystemsTested) on JMeter wiki.

Even if your OS is not listed on the wiki page, JMeter should run on it provided that the JVM is compliant.

# 1.2 Optional

If you plan on doing JMeter development, then you will need one or more optional packages listed below.

## 1.2.1 Java Compiler

If you want to build the JMeter source or develop JMeter plugins, then you will need a fully compliant JDK 7 or higher.

## 1.2.2 SAX XML Parser

JMeter comes with Apache's [Xerces XML parser](http://xml.apache.org/). You have the option of telling JMeter to use a different XML parser. To do so, include the classes for the third-party parser in JMeter's [classpath](http://jmeter.apache.org/usermanual/get-started.html#classpath), and update the [jmeter.properties](http://jmeter.apache.org/usermanual/get-started.html#configuring_jmeter)file with the full classname of the parser implementation.

## 1.2.3 Email Support（邮件支持）

JMeter has extensive Email capabilities. It can send email based on test results, and has a POP3(S)/IMAP(S) sampler. It also has an SMTP(S) sampler.

## 1.2.4 SSL Encryption

To test a web server using SSL encryption (HTTPS), JMeter requires that an implementation of SSL be provided, as is the case with Sun Java 1.4 and above. If your version of Java does not include SSL support, then it is possible to add an external implementation. Include the necessary encryption packages in JMeter's [classpath](http://jmeter.apache.org/usermanual/get-started.html#classpath). Also, update [**system.properties**](http://jmeter.apache.org/usermanual/get-started.html#configuring_jmeter) to register the SSL Provider.

JMeter HTTP defaults to protocol level TLS. This can be changed by editing the JMeter property **https.default.protocol** in **jmeter.properties** or**user.properties**.

**The JMeter HTTP samplers are configured to accept all certificates, whether trusted or not, regardless of validity periods, etc.** This is to allow the maximum flexibility in testing servers.

If the server requires a client certificate, this can be provided.

There is also the [SSL Manager](http://jmeter.apache.org/usermanual/component_reference.html#SSL_Manager), for greater control of certificates.

The JMeter proxy server (see below) supports recording HTTPS (SSL)

The SMTP sampler can optionally use a local trust store or trust all certificates.

## 1.2.5 JDBC Driver

You will need to add your database vendor's JDBC driver to the [classpath](http://jmeter.apache.org/usermanual/get-started.html#classpath) if you want to do JDBC testing. Make sure the file is a jar file, not a zip.

## 1.2.6 JMS client

JMeter now includes the JMS API from Apache Geronimo, so you just need to add the appropriate JMS Client implementation jar(s) from the JMS provider. Please refer to their documentation for details. There may also be some information on the [JMeter Wiki](http://wiki.apache.org/jmeter/).

## 1.2.7 Libraries for ActiveMQ JMS

You will need to add the jar **activemq-all-X.X.X.jar** to your classpath, e.g. by storing it in the **lib/** directory.

The other required jars (such as commons-logging) are already included with JMeter.

See [ActiveMQ initial configuration page](http://activemq.apache.org/initial-configuration.html) for details.

See the [JMeter Classpath](http://jmeter.apache.org/usermanual/get-started.html#classpath) section for more details on installing additional jars.

# 1.3 Installation

We recommend that most users run the [latest release](http://jmeter.apache.org/download_jmeter.cgi).

To install a release build, simply unzip（简单的解压即可） the zip/tar file into the directory where you want JMeter to be installed. Provided that you have a JRE/JDK correctly installed and the **JAVA\_HOME** environment variable set（要保证设置了JAVA\_HOME环境变量）, there is nothing more for you to do.

There can be problems (especially with client-server mode) if the directory path contains any spaces.

The installation directory structure should look something like this (where **X.Y** is version number):

apache-jmeter-X.Y

apache-jmeter-X.Y/bin

apache-jmeter-X.Y/docs

apache-jmeter-X.Y/extras

apache-jmeter-X.Y/lib/

apache-jmeter-X.Y/lib/ext

apache-jmeter-X.Y/lib/junit

apache-jmeter-X.Y/licenses

apache-jmeter-X.Y/printable\_docs

You can rename the parent directory (i.e. **apache-jmeter-X.Y**) if you want, but do not change any of the sub-directory names.

# 1.4 Running JMeter（启动/运行）

To run JMeter, run the **jmeter.bat** (for Windows) or **jmeter** (for Unix) file. These files are found in the bin directory. After a short time, the JMeter GUI should appear.

GUI mode should only be used for creating the test script, NON GUI mode must be used for load testing（GUI模式应该只被作为创建测试脚本，必须使用非GUI模式来加载测试）

There are some additional scripts in the bin directory that you may find useful. Windows script files (the .CMD files require Win2K or later):

**jmeter.bat**

run JMeter (in GUI mode by default)

**jmeterw.cmd**

run JMeter without the windows shell console (in GUI mode by default)

**jmeter-n.cmd**

drop a JMX file on this to run a non-GUI test

**jmeter-n-r.cmd**

drop a JMX file on this to run a non-GUI test remotely

**jmeter-t.cmd**

drop a JMX file on this to load it in GUI mode

**jmeter-server.bat**

start JMeter in server mode

**mirror-server.cmd**

runs the JMeter Mirror Server in non-GUI mode

**shutdown.cmd**

Run the Shutdown client to stop a non-GUI instance gracefully

**stoptest.cmd**

Run the Shutdown client to stop a non-GUI instance abruptly

The special name **LAST** can be used with **jmeter-n.cmd**, **jmeter-t.cmd** and**jmeter-n-r.cmd** and means the last test plan that was run interactively（交互的）.

The environment variable **JVM\_ARGS** can be used to override JVM settings in the**jmeter.bat** script（jmeter.bat脚本中的JVM\_ARGS变量能够重写JVM中的设置）. For example:

set JVM\_ARGS="-Xms1024m -Xmx1024m -Dpropname=propvalue"

jmeter -t test.jmx …

Un\*x script files; should work on most Linux/Unix systems:

**jmeter**

run JMeter (in GUI mode by default). Defines some JVM settings which may not work for all JVMs.

**jmeter-server**

start JMeter in server mode (calls jmeter script with appropriate parameters（以适当的参数调用jmeter脚本）)

**jmeter.sh**

very basic JMeter script (You may need to adapt JVM options like memory settings).

**mirror-server.sh**

runs the JMeter Mirror Server in non-GUI mode

**shutdown.sh**

Run the Shutdown client to stop a non-GUI instance gracefully

**stoptest.sh**

Run the Shutdown client to stop a non-GUI instance abruptly

It may be necessary to edit the jmeter shell script if some of the JVM options are not supported by the JVM you are using. The **JVM\_ARGS** environment variable can be used to override or set additional JVM options, for example:

JVM\_ARGS="-Xms1024m -Xmx1024m" jmeter -t test.jmx [etc.]

will override the HEAP settings in the script.

## 1.4.1 JMeter's Classpath

JMeter automatically finds classes from jars in the following directories（JMeter自动从以下文件中查找jar包）:

**JMETER\_HOME/lib**

used for utility jars

**JMETER\_HOME/lib/ext**

used for JMeter components and plugins

If you have developed new JMeter components, then you should jar them and copy the jar into JMeter's **lib/ext** directory. JMeter will automatically find JMeter components in any jars found here. Do not use **lib/ext** for utility jars or dependency jars used by the plugins; it is only intended for JMeter components and plugins.

If you don't want to put JMeter plugin jars in the **lib/ext** directory, then define the property **search\_paths** in **jmeter.properties**.

Utility and dependency jars (libraries etc) can be placed in the **lib** directory.

If you don't want to put such jars in the **lib** directory, then define the property**user.classpath** or **plugin\_dependency\_paths** in **jmeter.properties**. See below for an explanation of the differences.

Other jars (such as JDBC, JMS implementations and any other support libraries needed by the JMeter code) should be placed in the **lib** directory - not the **lib/ext** directory, or added to **user.classpath**.

JMeter will only find **.jar** files, not **.zip**.

You can also install utility Jar files in **$JAVA\_HOME/jre/lib/ext**, or you can set the property **user.classpath** in **jmeter.properties**

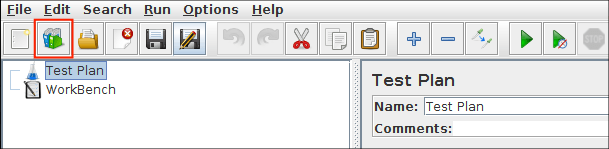
Note that setting the **CLASSPATH** environment variable will have no effect. This is because JMeter is started with "**java -jar**", and the java command silently ignores the **CLASSPATH** variable, and the **-classpath**/**-cp** options when **–jar** is used.

This occurs with all Java programs, not just JMeter.

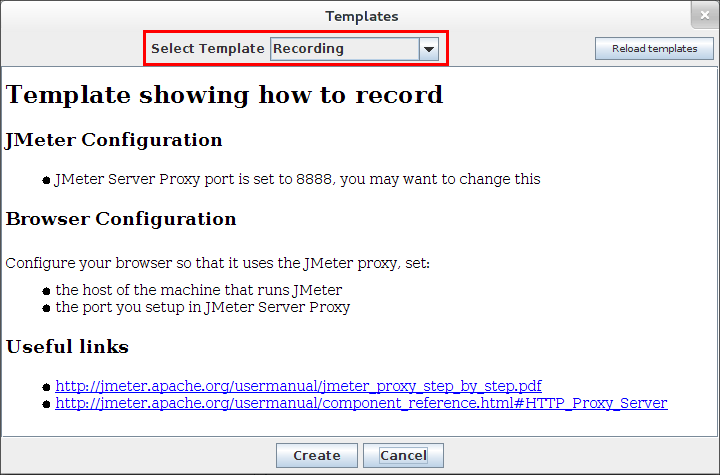
## 1.4.2 Create Test Plan from Template

You have the ability to create a new Test Plan from existing template（模板）.

To do so you use the menu *File* → *Templates…* or Templates icon:

[](http://jmeter.apache.org/images/screenshots/template_menu.png)Templates icon item

A popup appears, you can then choose a template among the list:

[](http://jmeter.apache.org/images/screenshots/template_wizard.png)Templates popup

A documentation for each template explains what to do once test plan is created from template.

## 1.4.3 Using JMeter behind a proxy（代理）

If you are testing from behind a firewall/proxy server, you may need to provide JMeter with the firewall/proxy server hostname and port number. To do so, run the **jmeter[.bat]** file from a command line with the following parameters（带参数运行jmeter）:

**-H**

[proxy server hostname or ip address]

**-P**

[proxy server port]

**-N**

[nonproxy hosts] (e.g. **\*.apache.org|localhost**)

**-u**

[username for proxy authentication - if required]

**-a**

[password for proxy authentication - if required]

**Example**:

jmeter -H my.proxy.server -P 8000 -u username -a password -N localhost

You can also use **--proxyHost**, **--proxyPort**, **--username**, and **--password**as parameter names

Parameters provided on a command-line may be visible to other users on the system.

If the proxy host and port are provided, then JMeter sets the following System properties:

* **http.proxyHost**
* **http.proxyPort**
* **https.proxyHost**
* **https.proxyPort**

If a nonproxy host list is provided, then JMeter sets the following System properties:

* **http.nonProxyHosts**
* **https.nonProxyHosts**

So if you don't wish to set both http and https proxies, you can define the relevant properties in **system.properties** instead of using the command-line parameters.

Proxy Settings can also be defined in a Test Plan, using either the [HTTP Request Defaults](http://jmeter.apache.org/usermanual/component_reference.html#HTTP_Request_Defaults) configuration or the [HTTP Request](http://jmeter.apache.org/usermanual/component_reference.html#HTTP_Request) sampler elements.

JMeter also has its own in-built Proxy Server, the [HTTP(S) Test Script Recorder](http://jmeter.apache.org/usermanual/component_reference.html#HTTP(S)_Test_Script_Recorder). This is only used for recording HTTP or HTTPS browser sessions. This is not to be confused with the proxy settings described above, which are used when JMeter makes HTTP or HTTPS requests itself.

## 1.4.4 Non-GUI Mode (Command Line mode)

For load testing, you must(必须） run JMeter in this mode (Without the GUI) to get the optimal results from it. To do so, use the following command options:

**-n**

This specifies JMeter is to run in non-gui mode

**-t**

[name of JMX file that contains the Test Plan].

**-l**

[name of JTL file to log sample results to].

**-j**

[name of JMeter run log file].

**-r**

Run the test in the servers specified by the JMeter property "**remote\_hosts**"

**-R**

[list of remote servers] Run the test in the specified remote servers

**-g**

[path to CSV file] generate report dashboard only

**-e**

generate report dashboard after load test

**-o**

output folder where to generate the report dashboard after load test. Folder must not exist or be empty

The script also lets you specify the optional firewall/proxy server information:

**-H**

[proxy server hostname or ip address]

**-P**

[proxy server port]

**Example**

jmeter -n -t my\_test.jmx -l log.jtl -H my.proxy.server -P 8000

If the property **jmeterengine.stopfail.system.exit** is set to **true** (default is **false**), then JMeter will invoke **System.exit(1)** if it cannot stop all threads. Normally this is not necessary.

## 1.4.5 Server Mode(服务器模式)

For [distributed testing](http://jmeter.apache.org/usermanual/remote-test.html)（分布式测试）, run JMeter in server mode on the remote node(s)（远程节点）, and then control the server(s) from the GUI（通过GUI控制jmeter服务）. You can also use non-GUI mode to run remote tests. To start the server(s), run **jmeter-server[.bat]** on each server host（在每台服务器上启动jmeter服务器模式）.

The script also lets you specify the optional firewall/proxy server information:

**-H**

[proxy server hostname or ip address]

**-P**

[proxy server port]

**Example**:

jmeter-server -H my.proxy.server -P 8000

If you want the server to exit after a single test has been run, then define the JMeter property **server.exitaftertest=true**.

To run the test from the client in non-GUI mode, use the following command:

jmeter -n -t testplan.jmx -r [-Gprop=val] [-Gglobal.properties] [-X]

where:

**-G**

is used to define JMeter properties to be set in the servers

**-X**

means exit the servers at the end of the test

**-R server1,server2**

can be used instead of **-r** to provide a list of servers to start. Overrides **remote\_hosts**, but does not define the property.

If the property **jmeterengine.remote.system.exit** is set to **true** (default is**false**), then JMeter will invoke **System.exit(0)** after stopping RMI at the end of a test. Normally this is not necessary.

## 1.4.6 Overriding Properties Via The Command Line

Java system properties, JMeter properties, and logging properties（java系统参数 jmeter参数和日志参数都可以被重新在命令行定义） can be overridden directly on the command line (instead of modifying **jmeter.properties**). To do so, use the following options:

**-D[prop\_name]=[value]**

defines a java system property value.

**-J[prop\_name]=[value]**

defines a local JMeter property.

**-G[prop\_name]=[value]**

defines a JMeter property to be sent to all remote servers.

**-G[propertyfile]**

defines a file containing JMeter properties to be sent to all remote servers.

**-L[category]=[priority]**

overrides a logging setting, setting a particular category to the given priority level.

The **-L** flag can also be used without the category name to set the root logging level.

**Examples**:

jmeter -Duser.dir=/home/mstover/jmeter\_stuff \

-Jremote\_hosts=127.0.0.1 -Ljmeter.engine=DEBUG

jmeter -LDEBUG

The command line properties are processed early in startup, but after the logging system has been set up. Attempts to use the **–J** flag to update **log\_level** or **log\_file** properties will have no effect.

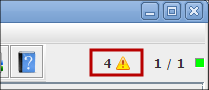
## 1.4.7 Logging and error messages

JMeter does not generally use pop-up dialog boxes for errors, as these would interfere（干涉） with running tests. Nor does it report any error for a mis-spelt variable or function; instead the reference is just used as is. See [Functions and Variables for more information](http://jmeter.apache.org/usermanual/functions.html).

If JMeter detects an error during a test, a message will be written to the log file. The log file name is defined in the **jmeter.properties** file (or using the **-j** option, see below). It defaults to **jmeter.log**, and will be found in the directory from which JMeter was launched.

The menu *Options* → *Log Viewer* displays the log file in a bottom pane on main JMeter window.

In the GUI mode, the number of error/fatal messages logged in the log file is displayed at top-right.

[](http://jmeter.apache.org/images/screenshots/log_errors_counter.png)

Error/fatal counter

The command-line option **-j jmeterlogfile** allow to process after the initial properties file is read, and before any further properties are processed. It therefore allows the default of **jmeter.log** to be overridden. The jmeter scripts that take a test plan name as a parameter (e.g. **jmeter-n.cmd**) have been updated to define the log file using the test plan name, e.g. for the test plan **Test27.jmx** the log file is set to **Test27.log**.

When running on Windows, the file may appear as just **jmeter** unless you have set Windows to show file extensions. [Which you should do anyway, to make it easier to detect viruses and other nasties that pretend to be text files …]

As well as recording errors, the **jmeter.log** file records some information about the test run. For example:

10/17/2003 12:19:20 PM INFO - jmeter.JMeter: Version 1.9.20031002

10/17/2003 12:19:45 PM INFO - jmeter.gui.action.Load: Loading file: c:\mytestfiles\BSH.jmx

10/17/2003 12:19:52 PM INFO - jmeter.engine.StandardJMeterEngine: Running the test!

10/17/2003 12:19:52 PM INFO - jmeter.engine.StandardJMeterEngine: Starting 1 threads for group BSH. Ramp up = 1.

10/17/2003 12:19:52 PM INFO - jmeter.engine.StandardJMeterEngine: Continue on error

10/17/2003 12:19:52 PM INFO - jmeter.threads.JMeterThread: Thread BSH1-1 started

10/17/2003 12:19:52 PM INFO - jmeter.threads.JMeterThread: Thread BSH1-1 is done

10/17/2003 12:19:52 PM INFO - jmeter.engine.StandardJMeterEngine: Test has ended

The log file can be helpful in determining the cause of an error, as JMeter does not interrupt a test to display an error dialogue.

## 1.4.8 Full list of command-line options

Invoking JMeter as "**jmeter -?**" will print a list of all the command-line options. These are shown below.

-h, --help

print usage information and exit

-v, --version

print the version information and exit

-p, --propfile <argument>

the jmeter property file to use

-q, --addprop <argument>

additional JMeter property file(s)

-t, --testfile <argument>

the jmeter test(.jmx) file to run

-l, --logfile <argument>

the file to log samples to

-j, --jmeterlogfile <argument>

jmeter run log file (jmeter.log)

-n, --nongui

run JMeter in nongui mode

-s, --server

run the JMeter server

-H, --proxyHost <argument>

Set a proxy server for JMeter to use

-P, --proxyPort <argument>

Set proxy server port for JMeter to use

-N, --nonProxyHosts <argument>

Set nonproxy host list (e.g. \*.apache.org|localhost)

-u, --username <argument>

Set username for proxy server that JMeter is to use

-a, --password <argument>

Set password for proxy server that JMeter is to use

-J, --jmeterproperty <argument>=<value>

Define additional JMeter properties

-G, --globalproperty <argument>=<value>

Define Global properties (sent to servers)

e.g. -Gport=123

or -Gglobal.properties

-D, --systemproperty <argument>=<value>

Define additional system properties

-S, --systemPropertyFile <argument>

additional system property file(s)

-L, --loglevel <argument>=<value>

[category=]level e.g. jorphan=INFO or jmeter.util=DEBUG

-r, --runremote

Start remote servers (as defined in remote\_hosts)

-R, --remotestart <argument>

Start these remote servers (overrides remote\_hosts)

-d, --homedir <argument>

the jmeter home directory to use

-X, --remoteexit

Exit the remote servers at end of test (non-GUI)

-g, --reportonly <argument>

generate report dashboard only

-e, --reportatendofloadtests

generate report dashboard after load test

-o, --reportoutputfolder

output folder for report dashboard

Note: the JMeter log file name is formatted as a SimpleDateFormat (applied to the current date) if it contains paired single-quotes, .e.g. '**jmeter\_'yyyyMMddHHmmss'.log**'

If the special name **LAST** is used for the **-t**, **-j** or **-l** flags, then JMeter takes that to mean the last test plan that was run in interactive mode.

## 1.4.9 non-GUI shutdown

Prior（高级） to version 2.5.1, JMeter invoked **System.exit()** when a non-GUI test completed. This caused problems for applications that invoke JMeter directly, so JMeter no longer invokes **System.exit()** for a normal test completion. [Some fatal errors may still invoke **System.exit()**] JMeter will exit all the non-daemon threads it starts, but it is possible that some non-daemon threads may still remain; these will prevent the JVM from exiting. To detect this situation, JMeter starts a new daemon thread just before it exits. This daemon thread waits a short while; if it returns from the wait, then clearly the JVM has not been able to exit, and the thread prints a message to say why.

The property **jmeter.exit.check.pause** can be used to override the default pause of 2000ms (2secs). If set to **0**, then JMeter does not start the daemon thread.

# 1.5 Configuring JMeter

If you wish to modify the properties with which JMeter runs you need to either modify the **user.properties** in the **/bin** directory or create your own copy of the**jmeter.properties** and specify it in the command line.

Note: You can define additional JMeter properties in the file defined by the JMeter property **user.properties** which has the default value**user.properties**. The file will be automatically loaded if it is found in the current directory or if it is found in the JMeter bin directory. Similarly,**system.properties** is used to update system properties.

### Parameters

***Attribute***

**Description**

**Required**

*ssl.provider*

You can specify the class for your SSL implementation if you don't want to use the built-in Java implementation.

No

*xml.parser*

You can specify an implementation as your XML parser. The default value is:**org.apache.xerces.parsers.SAXParser**

No

*remote\_hosts*

Comma-delimited list of remote JMeter hosts (or **host:port** if required). If you are running JMeter in a distributed environment, list the machines where you have JMeter remote servers running. This will allow you to control those servers from this machine's GUI

No

*not\_in\_menu*

A list of components you do not want to see in JMeter's menus. As JMeter has more and more components added, you may wish to customize your JMeter to show only those components you are interested in. You may list their classname or their class label (the string that appears in JMeter's UI) here, and they will no longer appear in the menus.

No

*search\_paths*

List of paths (separated by **;**) that JMeter will search for JMeter plugin classes, for example additional samplers. A path item can either be a jar file or a directory. Any jar file in such a directory will be automatically included in **search\_paths**, jar files in sub directories are ignored. The given value is in addition to any jars found in the **lib/ext** directory.

No

*user.classpath*

List of paths that JMeter will search for utility and plugin dependency classes. Use your platform path separator to separate multiple paths. A path item can either be a jar file or a directory. Any jar file in such a directory will be automatically included in**user.classpath**, jar files in sub directories are ignored. The given value is in addition to any jars found in the lib directory. All entries will be added to the class path of the system class loader and also to the path of the JMeter internal loader.

No

*plugin\_dependency\_paths*

List of paths (separated by **;**) that JMeter will search for utility and plugin dependency classes. A path item can either be a jar file or a directory. Any jar file in such a directory will be automatically included in**plugin\_dependency\_paths**, jar files in sub directories are ignored. The given value is in addition to any jars found in the **lib** directory or given by the**user.classpath** property. All entries will be added to the path of the JMeter internal loader only. For plugin dependencies using**plugin\_dependency\_paths** should be preferred over **user.classpath**.

No

*user.properties*

Name of file containing additional JMeter properties. These are added after the initial property file, but before the **-q** and **-J** options are processed.

No

*system.properties*

Name of file containing additional system properties. These are added before the **-S** and **-D** options are processed.

No

The command line options and properties files are processed in the following order:

1. **-p propfile**
2. **jmeter.properties** (or the file from the **-p** option) is then loaded
3. **-j logfile**
4. Logging is initialised
5. **user.properties** is loaded
6. **system.properties** is loaded
7. all other command-line options are processed

**See also the comments in the jmeter.properties, user.properties andsystem.properties files for further information on other settings you can change.**

**2. Building a Test Plan**

A test plan describes a series of steps JMeter will execute when run. A complete test plan will consist of one or more Thread Groups（一个或多个线程组）, logic controllers（逻辑控制）, sample generating controllers, listeners（监听器）, timers, assertions（断言）, and configuration elements（元素配置）.

**2.1 Adding and Removing Elements**

Adding [elements to a test plan](http://jmeter.apache.org/usermanual/test_plan.html) can be done by right-clicking on an element in the tree, and choosing a new element from the "**add**" list. Alternatively（或者）, elements can be loaded from file and added by choosing the "**merge**" or "**open**" option.

To remove an element, make sure the element is selected, right-click on the element, and choose the "**remove**" option.

**2.2 Loading and Saving Elements**

To load an element from file（从文件加载一个element配置）, right click on the existing tree elements to which you want to add the loaded element, and select the "**merge**" option. Choose the file where your elements are saved. JMeter will merge the elements into the tree.

To save tree elements, right click on an element and choose the "**Save Selection As …**" option（另存为选项即可保存元素）. JMeter will save the element selected, plus all child elements beneath it. In this way, you can save test tree fragments（测试框架） and individual elements for later use.

By default, the workbench is not automatically saved with the test plan, but it can be saved by checking "**Save Workbench**" option on Workbench element.

**2.3 Configuring Tree Elements（配置组件树）**

Any element in the test tree will present controls in JMeter's right-hand frame. These controls allow you to configure the behavior of that particular test element. What can be configured for an element depends on what type of element it is.

The Test Tree itself can be manipulated by dragging and dropping components around the test tree.

**2.4 Saving the Test Plan**

Although it is not required, we recommend that you save the Test Plan to a file before running it. To save the Test Plan, select "**Save**" or "**Save Test Plan As …**" from the File menu (with the latest release, it is no longer necessary to select the Test Plan element first).

JMeter allows you to save the entire Test Plan tree or only a portion of it. To save only the elements located in a particular "branch" of the Test Plan tree, select the Test Plan element in the tree from which to start the "branch", and then click your right mouse button to access the "**Save Selection As …**"（选中部分另存为） menu item. Alternatively, select the appropriate Test Plan element and then select "**Save Selection As …**" from the Edit menu.

**2.5 Running a Test Plan**

To run your test plan, choose "**Start**" (**Control** + **r**) from the "**Run**" menu item. When JMeter is running, it shows a small green box at the right hand end of the section just under the menu bar. You can also check the "**Run**" menu. If "**Start**" is disabled, and "**Stop**" is enabled, then JMeter is running your test plan (or, at least, it thinks it is).

The numbers to the left of the green box are the number of active threads / total number of threads. These only apply to a locally run test; they do not include any threads started on remote systems when using client-server mode.

Using GUI mode as described here should only be used when debugging your Test Plan（仅当调试阶段时使用GUI模式）. To run the real load test, use NON-GUI mode.

**2.6 Stopping a Test**

There are two types of stop command available from the menu:

* **Stop** (**Control** + **.**) - stops the threads immediately if possible. Many samplers are Interruptible which means that active samples can be terminated early. The stop command will check that all threads have stopped within the default timeout, which is 5000 ms = 5 seconds. [This can be changed using the JMeter property **jmeterengine.threadstop.wait**] If the threads have not stopped, then a message is displayed. The Stop command can be retried, but if it fails, then it is necessary to exit JMeter to clean up.
* **Shutdown** (**Control** + **,**) - requests the threads to stop at the end of any current work. Will not interrupt any active samples. The modal shutdown dialog box will remain active until all threads have stopped.

If Shutdown is taking too long. Close the Shutdown dialog box and select **Run**/**Stop**, or just press **Control** + **.**.

When running JMeter in non-GUI mode, there is no Menu, and JMeter does not react to keystrokes such as **Control** + **.**. So JMeter non-GUI mode will listen for commands on a specific port (default **4445**, see the JMeter property **jmeterengine.nongui.port**). JMeter supports automatic choice of an alternate port if the default port is being used (for example by another JMeter instance). In this case, JMeter will try the next higher port, continuing until it reaches the JMeter property **jmeterengine.nongui.maxport**) which defaults to **4455**. If **maxport** is less than or equal to **port**, port scanning will not take place.   
The chosen port is displayed in the console window.   
The commands currently supported are:

* **Shutdown** - graceful shutdown
* **StopTestNow** - immediate shutdown

These commands can be sent by using the **shutdown[.cmd|.sh]** or **stoptest[.cmd|.sh]** script respectively. The scripts are to be found in the JMeter **bin** directory. The commands will only be accepted if the script is run from the same host.

**2.7 Error reporting**

JMeter reports warnings and errors to the **jmeter.log** file, as well as some information on the test run itself. JMeter shows the number of warnings/errors found in **jmeter.log** file next to the warning icon (triangle) at the right hand end of its window. Click on the warning icon to show the **jmeter.log** file at the bottom of JMeter's window. Just occasionally there may be some errors that JMeter is unable to trap and log; these will appear on the command console. If a test is not behaving as you expect, please check the log file in case any errors have been reported (e.g. perhaps a syntax error in a function call).

Sampling errors (e.g. HTTP 404 - file not found) are not normally reported in the log file. Instead these are stored as attributes of the sample result. The status of a sample result can be seen in the various different Listeners.

**3. Elements of a Test Plan**

This section describes the different parts of a test plan.

A minimal test will consist of the Test Plan, a Thread Group and one or more Samplers.

**3.0 Test Plan**

The Test Plan object has a checkbox called "**Functional Testing**". If selected, it will cause JMeter to record the data returned from the server for each sample. If you have selected a file in your test listeners, this data will be written to file. This can be useful if you are doing a small run to ensure that JMeter is configured correctly, and that your server is returning the expected results. The consequence is that the file will grow huge quickly, and JMeter's performance will suffer. This option should be off if you are doing stress-testing (it is off by default).

If you are not recording the data to file, this option makes no difference.

You can also use the **Configuration** button on a listener to decide what fields to save.

**3.1 Thread Group**

Thread group elements are the beginning points of any test plan. All controllers and samplers must be under a thread group. Other elements, e.g. Listeners, may be placed directly under the test plan, in which case they will apply to all the thread groups. As the name implies, the thread group element controls the number of threads JMeter will use to execute your test. The controls for a thread group allow you to:

* Set the number of threads
* Set the ramp-up period
* Set the number of times to execute the test

Each thread will execute the test plan in its entirety and completely independently of other test threads. Multiple threads are used to simulate concurrent connections to your server application.

The ramp-up period tells JMeter how long to take to "ramp-up" to the full number of threads chosen. If 10 threads are used, and the ramp-up period is 100 seconds, then JMeter will take 100 seconds to get all 10 threads up and running. Each thread will start 10 (100/10) seconds after the previous thread was begun. If there are 30 threads and a ramp-up period of 120 seconds, then each successive thread will be delayed by 4 seconds.

Ramp-up needs to be long enough to avoid too large a work-load at the start of a test, and short enough that the last threads start running before the first ones finish (unless one wants that to happen).

Start with Ramp-up = number of threads and adjust up or down as needed.

By default, the thread group is configured to loop once through its elements.

Thread Group also provides a **scheduler**. Click the checkbox at the bottom of the Thread Group panel to enable/disable extra fields in which you can enter the duration of test, the startup delay, the start and end times of the run. You can configure **Duration (seconds)** and **Startup Delay (seconds)** to control the duration of each thread group and the after how much seconds it starts. When the test is started, JMeter will wait **Startup Delay (seconds)** before starting the Threads of the Thread Group and run for the configured **Duration (seconds)** time. Note those 2 options override the **Start time** and **End time**.

Alternatively (although not recommended as not very flexible) you can use the two other fields **Start time** and **End time**. When the test is started, JMeter will wait if necessary until the start-time has been reached. At the end of each cycle, JMeter checks if the end-time has been reached, and if so, the run is stopped, otherwise the test is allowed to continue until the iteration limit is reached.

**3.2 Controllers**

JMeter has two types of Controllers: Samplers and Logical Controllers. These drive the processing of a test.

Samplers tell JMeter to send requests to a server. For example, add an HTTP Request Sampler if you want JMeter to send an HTTP request. You can also customize a request by adding one or more Configuration Elements to a Sampler. For more information, see [Samplers](http://jmeter.apache.org/usermanual/test_plan.html#samplers).

Logical Controllers let you customize the logic that JMeter uses to decide when to send requests. For example, you can add an Interleave Logic Controller to alternate between two HTTP Request Samplers. For more information, see [Logical Controllers](http://jmeter.apache.org/usermanual/test_plan.html#logic_controller).

**3.2.1 Samplers**

Samplers tell JMeter to send requests to a server and wait for a response. They are processed in the order they appear in the tree. Controllers can be used to modify the number of repetitions of a sampler.

JMeter samplers include:

* FTP Request
* HTTP Request (can be used for SOAP or REST Webservice also)
* JDBC Request
* Java object request
* JMS request
* JUnit Test request
* LDAP Request
* Mail request
* OS Process request
* TCP request

Each sampler has several properties you can set. You can further customize a sampler by adding one or more Configuration Elements to the Test Plan.

If you are going to send multiple requests of the same type (for example, HTTP Request) to the same server, consider using a Defaults Configuration Element. Each controller has one or more Defaults elements (see below).

Remember to add a Listener to your test plan to view and/or store the results of your requests to disk.

If you are interested in having JMeter perform basic validation on the response of your request, add an[Assertion](http://jmeter.apache.org/usermanual/test_plan.html#assertions) to the sampler. For example, in stress testing a web application, the server may return a successful "HTTP Response" code, but the page may have errors on it or may be missing sections. You could add assertions to check for certain HTML tags, common error strings, and so on. JMeter lets you create these assertions using regular expressions.

[JMeter's built-in samplers](http://jmeter.apache.org/usermanual/component_reference.html#samplers)

**3.2.2 Logic Controllers**

Logic Controllers let you customize the logic that JMeter uses to decide when to send requests. Logic Controllers can change the order of requests coming from their child elements. They can modify the requests themselves, cause JMeter to repeat requests, etc.

To understand the effect of Logic Controllers on a test plan, consider the following test tree:

* Test Plan
  + Thread Group
    - Once Only Controller
      * Login Request (an [HTTP Request](http://jmeter.apache.org/usermanual/component_reference.html#HTTP_Request))
    - Load Search Page (HTTP Sampler)
    - Interleave Controller
      * Search "A" (HTTP Sampler)
      * Search "B" (HTTP Sampler)
      * HTTP default request (Configuration Element)
    - HTTP default request (Configuration Element)
    - Cookie Manager (Configuration Element)

The first thing about this test is that the login request will be executed only the first time through. Subsequent iterations will skip it. This is due to the effects of the [Once Only Controller](http://jmeter.apache.org/usermanual/component_reference.html#Once_Only_Controller).

After the login, the next Sampler loads the search page (imagine a web application where the user logs in, and then goes to a search page to do a search). This is just a simple request, not filtered through any Logic Controller.

After loading the search page, we want to do a search. Actually, we want to do two different searches. However, we want to re-load the search page itself between each search. We could do this by having 4 simple HTTP request elements (load search, search "A", load search, search "B"). Instead, we use the[Interleave Controller](http://jmeter.apache.org/usermanual/component_reference.html#Interleave_Controller) which passes on one child request each time through the test. It keeps the ordering (i.e. it doesn't pass one on at random, but "remembers" its place) of its child elements. Interleaving 2 child requests may be overkill, but there could easily have been 8, or 20 child requests.

Note the [HTTP Request Defaults](http://jmeter.apache.org/usermanual/component_reference.html#HTTP_Request_Defaults) that belongs to the Interleave Controller. Imagine that "Search A" and "Search B" share the same PATH info (an HTTP request specification includes domain, port, method, protocol, path, and arguments, plus other optional items). This makes sense - both are search requests, hitting the same back-end search engine (a servlet or cgi-script, let's say). Rather than configure both HTTP Samplers with the same information in their PATH field, we can abstract that information out to a single Configuration Element. When the Interleave Controller "passes on" requests from "Search A" or "Search B", it will fill in the blanks with values from the HTTP default request Configuration Element. So, we leave the PATH field blank for those requests, and put that information into the Configuration Element. In this case, this is a minor benefit at best, but it demonstrates the feature.

The next element in the tree is another HTTP default request, this time added to the Thread Group itself. The Thread Group has a built-in Logic Controller, and thus, it uses this Configuration Element exactly as described above. It fills in the blanks of any Request that passes through. It is extremely useful in web testing to leave the DOMAIN field blank in all your HTTP Sampler elements, and instead, put that information into an HTTP default request element, added to the Thread Group. By doing so, you can test your application on a different server simply by changing one field in your Test Plan. Otherwise, you'd have to edit each and every Sampler.

The last element is a [HTTP Cookie Manager](http://jmeter.apache.org/usermanual/component_reference.html#HTTP_Cookie_Manager). A Cookie Manager should be added to all web tests - otherwise JMeter will ignore cookies. By adding it at the Thread Group level, we ensure that all HTTP requests will share the same cookies.

Logic Controllers can be combined to achieve various results. See the list of [built-in Logic Controllers](http://jmeter.apache.org/usermanual/component_reference.html#logic_controllers).

**3.2.3 Test Fragments**

The Test Fragment element is a special type of [controller](http://jmeter.apache.org/usermanual/test_plan.html#controllers) that exists on the Test Plan tree at the same level as the Thread Group element. It is distinguished from a Thread Group in that it is not executed unless it is referenced by either a [Module Controller](http://jmeter.apache.org/usermanual/component_reference.html#Module_Controller) or an [Include\_Controller](http://jmeter.apache.org/usermanual/component_reference.html#Include_Controller).

This element is purely for code re-use within Test Plans

**3.3 Listeners**

Listeners provide access to the information JMeter gathers about the test cases while JMeter runs. The[Graph Results](http://jmeter.apache.org/usermanual/component_reference.html#Graph_Results) listener plots the response times on a graph. The "View Results Tree" Listener shows details of sampler requests and responses, and can display basic HTML and XML representations of the response. Other listeners provide summary or aggregation information.

Additionally, listeners can direct the data to a file for later use. Every listener in JMeter provides a field to indicate the file to store data to. There is also a Configuration button which can be used to choose which fields to save, and whether to use CSV or XML format.

Note that all Listeners save the same data; the only difference is in the way the data is presented on the screen.

Listeners can be added anywhere in the test, including directly under the test plan. They will collect data only from elements at or below their level.

There are several [listeners](http://jmeter.apache.org/usermanual/component_reference.html#listeners) that come with JMeter.

**3.4 Timers**

By default, a JMeter thread executes samplers in sequence without pausing. We recommend that you specify a delay by adding one of the available timers to your Thread Group. If you do not add a delay, JMeter could overwhelm your server by making too many requests in a very short amount of time.

A timer will cause JMeter to delay a certain amount of time **before** each sampler which is in its [scope](http://jmeter.apache.org/usermanual/test_plan.html#scoping_rules).

If you choose to add more than one timer to a Thread Group, JMeter takes the sum of the timers and pauses for that amount of time before executing the samplers to which the timers apply. Timers can be added as children of samplers or controllers in order to restrict the samplers to which they are applied.

To provide a pause at a single place in a test plan, one can use the [Test Action](http://jmeter.apache.org/usermanual/component_reference.html#Test_Action) Sampler.

**3.5 Assertions**

Assertions allow you to assert facts about responses received from the server being tested. Using an assertion, you can essentially "test" that your application is returning the results you expect it to.

For instance, you can assert that the response to a query will contain some particular text. The text you specify can be a Perl-style regular expression, and you can indicate that the response is to contain the text, or that it should match the whole response.

You can add an assertion to any Sampler. For example, you can add an assertion to a HTTP Request that checks for the text, "**</HTML>**". JMeter will then check that the text is present in the HTTP response. If JMeter cannot find the text, then it will mark this as a failed request.

Note that assertions apply to all samplers which are in their [scope](http://jmeter.apache.org/usermanual/test_plan.html#scoping_rules). To restrict an assertion to a single sampler, add the assertion as a child of the sampler.

To view assertion results, add an Assertion Listener to the Thread Group. Failed Assertions will also show up in the Tree View and Table Listeners, and will count towards the error %age for example in the Aggregate and Summary reports.

**3.6 Configuration Elements**

A configuration element works closely with a Sampler. Although it does not send requests (except for[HTTP(S) Test Script Recorder](http://jmeter.apache.org/usermanual/component_reference.html#HTTP(S)_Test_Script_Recorder)), it can add to or modify requests.

A configuration element is accessible from only inside the tree branch where you place the element. For example, if you place an HTTP Cookie Manager inside a Simple Logic Controller, the Cookie Manager will only be accessible to HTTP Request Controllers you place inside the Simple Logic Controller (see figure 1). The Cookie Manager is accessible to the HTTP requests "Web Page 1" and "Web Page 2", but not "Web Page 3".

Also, a configuration element inside a tree branch has higher precedence than the same element in a "parent" branch. For example, we defined two HTTP Request Defaults elements, "Web Defaults 1" and "Web Defaults 2". Since we placed "Web Defaults 1" inside a Loop Controller, only "Web Page 2" can access it. The other HTTP requests will use "Web Defaults 2", since we placed it in the Thread Group (the "parent" of all other branches).

[Figure 1 -
    Test Plan Showing Accessability of Configuration Elements](http://jmeter.apache.org/images/screenshots/http-config/http-config-example.png)Figure 1 - Test Plan Showing Accessability of Configuration Elements

The [User Defined Variables](http://jmeter.apache.org/usermanual/component_reference.html#User_Defined_Variables) Configuration element is different. It is processed at the start of a test, no matter where it is placed. For simplicity, it is suggested that the element is placed only at the start of a Thread Group.

**3.7 Pre-Processor Elements**

A Pre-Processor executes some action prior to a Sampler Request being made. If a Pre-Processor is attached to a Sampler element, then it will execute just prior to that sampler element running. A Pre-Processor is most often used to modify the settings of a Sample Request just before it runs, or to update variables that aren't extracted from response text. See the [**scoping rules**](http://jmeter.apache.org/usermanual/test_plan.html#scoping_rules) for more details on when Pre-Processors are executed.

**3.8 Post-Processor Elements**

A Post-Processor executes some action after a Sampler Request has been made. If a Post-Processor is attached to a Sampler element, then it will execute just after that sampler element runs. A Post-Processor is most often used to process the response data, often to extract values from it. See the [**scoping rules**](http://jmeter.apache.org/usermanual/test_plan.html#scoping_rules)for more details on when Post-Processors are executed.

**3.9 Execution order**

1. Configuration elements
2. Pre-Processors
3. Timers
4. Sampler
5. Post-Processors (unless SampleResult is **null**)
6. Assertions (unless SampleResult is **null**)
7. Listeners (unless SampleResult is **null**)

Please note that Timers, Assertions, Pre- and Post-Processors are only processed if there is a sampler to which they apply. Logic Controllers and Samplers are processed in the order in which they appear in the tree. Other test elements are processed according to the scope in which they are found, and the type of test element. [Within a type, elements are processed in the order in which they appear in the tree].

For example, in the following test plan:

* Controller
  + Post-Processor 1
  + Sampler 1
  + Sampler 2
  + Timer 1
  + Assertion 1
  + Pre-Processor 1
  + Timer 2
  + Post-Processor 2

The order of execution would be:

Pre-Processor 1

Timer 1

Timer 2

Sampler 1

Post-Processor 1

Post-Processor 2

Assertion 1

Pre-Processor 1

Timer 1

Timer 2

Sampler 2

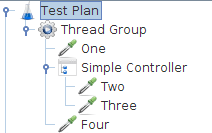
Post-Processor 1

Post-Processor 2

Assertion 1

**3.10 Scoping Rules**

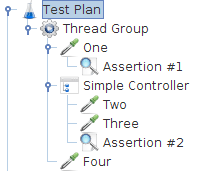
The JMeter test tree contains elements that are both hierarchical and ordered. Some elements in the test trees are strictly hierarchical (Listeners, Config Elements, Post-Processors, Pre-Processors, Assertions, Timers), and some are primarily ordered (controllers, samplers). When you create your test plan, you will create an ordered list of sample request (via Samplers) that represent a set of steps to be executed. These requests are often organized within controllers that are also ordered. Given the following test tree:

[](http://jmeter.apache.org/images/screenshots/scoping1.png)Example test tree

The order of requests will be, One, Two, Three, Four.

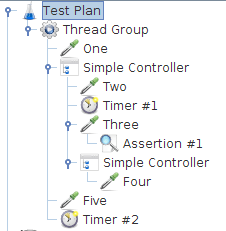
Some controllers affect the order of their subelements, and you can read about these specific controllers in [the component reference](http://jmeter.apache.org/usermanual/component_reference.html).

Other elements are hierarchical. An Assertion, for instance, is hierarchical in the test tree. If its parent is a request, then it is applied to that request. If its parent is a Controller, then it affects all requests that are descendants of that Controller. In the following test tree:

[](http://jmeter.apache.org/images/screenshots/scoping2.png)Hierarchy example

Assertion #1 is applied only to Request One, while Assertion #2 is applied to Requests Two and Three.

Another example, this time using Timers:

[](http://jmeter.apache.org/images/screenshots/scoping3.png)complex example

In this example, the requests are named to reflect the order in which they will be executed. Timer #1 will apply to Requests Two, Three, and Four (notice how order is irrelevant for hierarchical elements). Assertion #1 will apply only to Request Three. Timer #2 will affect all the requests.

Hopefully these examples make it clear how configuration (hierarchical) elements are applied. If you imagine each Request being passed up the tree branches, to its parent, then to its parent's parent, etc., and each time collecting all the configuration elements of that parent, then you will see how it works.

The Configuration elements Header Manager, Cookie Manager and Authorization manager are treated differently from the Configuration Default elements. The settings from the Configuration Default elements are merged into a set of values that the Sampler has access to. However, the settings from the Managers are not merged. If more than one Manager is in the scope of a Sampler, only one Manager is used, but there is currently no way to specify *which* is used.

**3.11 Properties and Variables**

JMeter *properties* are defined in **jmeter.properties** (see [Getting Started - Configuring JMeter](http://jmeter.apache.org/usermanual/get-started.html#configuring_jmeter) for more details).   
Properties are global to jmeter, and are mostly used to define some of the defaults JMeter uses. For example the property **remote\_hosts** defines the servers that JMeter will try to run remotely. Properties can be referenced in test plans - see [Functions - read a property](http://jmeter.apache.org/usermanual/functions.html#__property) - but cannot be used for thread-specific values.

JMeter *variables* are local to each thread. The values may be the same for each thread, or they may be different.   
If a variable is updated by a thread, only the thread copy of the variable is changed. For example the[Regular Expression Extractor](http://jmeter.apache.org/usermanual/component_reference.html#Regular_Expression_Extractor) Post-Processor will set its variables according to the sample that its thread has read, and these can be used later by the same thread. For details of how to reference variables and functions, see [Functions and Variables](http://jmeter.apache.org/usermanual/functions.html)

Note that the values defined by the [Test Plan](http://jmeter.apache.org/usermanual/component_reference.html#Test_Plan) and the [User Defined Variables](http://jmeter.apache.org/usermanual/component_reference.html#User_Defined_Variables) configuration element are made available to the whole test plan at startup. If the same variable is defined by multiple UDV elements, then the last one takes effect. Once a thread has started, the initial set of variables is copied to each thread. Other elements such as the [User Parameters](http://jmeter.apache.org/usermanual/component_reference.html#User_Parameters) Pre-Processor or [Regular Expression Extractor](http://jmeter.apache.org/usermanual/component_reference.html#Regular_Expression_Extractor) Post-Processor may be used to redefine the same variables (or create new ones). These redefinitions only apply to the current thread.

The [setProperty](http://jmeter.apache.org/usermanual/functions.html#__setProperty) function can be used to define a JMeter property. These are global to the test plan, so can be used to pass information between threads - should that be needed.

Both variables and properties are case-sensitive.

**3.12 Using Variables to parameterise tests**

Variables don't have to vary - they can be defined once, and if left alone, will not change value. So you can use them as short-hand for expressions that appear frequently in a test plan. Or for items which are constant during a run, but which may vary between runs. For example, the name of a host, or the number of threads in a thread group.

When deciding how to structure a Test Plan, make a note of which items are constant for the run, but which may change between runs. Decide on some variable names for these - perhaps use a naming convention such as prefixing them with **C\_** or **K\_** or using uppercase only to distinguish them from variables that need to change during the test. Also consider which items need to be local to a thread - for example counters or values extracted with the Regular Expression Post-Processor. You may wish to use a different naming convention for these.

For example, you might define the following on the Test Plan:

HOST www.example.com

THREADS 10

LOOPS 20

You can refer to these in the test plan as **${HOST}** **${THREADS}** etc. If you later want to change the host, just change the value of the **HOST** variable. This works fine for small numbers of tests, but becomes tedious when testing lots of different combinations. One solution is to use a property to define the value of the variables, for example:

HOST ${\_\_P(host,www.example.com)}

THREADS ${\_\_P(threads,10)}

LOOPS ${\_\_P(loops,20)}

You can then change some or all of the values on the command-line as follows:

jmeter … -Jhost=www3.example.org -Jloops=13