# UserManual Elements

## Elements of a Test Plan

The Test Plan object has a new checkbox to select 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.

### ThreadGroup

Thread group elements are the beginning points of any test plan. All elements of a test plan must be under a thread group. 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.

By default, a thread group is configured to loop indefinitely through its elements. Alternatively, you can set the number of times the thread group will loop before ending. If the number is set at one, then JMeter will execute the test only once before stopping.

Version 1.9 introduces a test run **scheduler**. Click the checkbox at the bottom of the Thread Group panel to reveal two extra fields in which you can enter the start and end times of the run. 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.

## Controllers

Controllers let you modify the ordering and flow of a JMeter test script. Controllers may have as child elements any of the following: Samplers (requests), Configuration Elements, and other Logic Controllers. 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 Controllers on a test plan, consider the following test tree:

- Test Plan
    1. Thread Group
        a. Once Only Controller
            i. Login Request (an HTTP Request )
        b. Load Search Page (HTTP Sampler)
        c. Interleave Controller
            i. Search "A" (HTTP Sampler)
            ii. Search "B" (HTTP Sampler)
            iii. HTTP default request (Configuration Element)
        d. HTTP default request (Configuration Element)
        e. 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.

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 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 which passes on one child request each time through the test. It keeps the ordering (ie - 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 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 . 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.

Controllers can be combined to achieve various results. See the list of built-in Controllers.

## Samplers

Samplers tell JMeter to send requests to a server. JMeter currently has the following samplers:

- FTP Request
- HTTP Request
- JDBC Request
- Java object request
- LDAP Request
- SOAP/XML-RPC Request
- WebService (SOAP) Request (Alpha Code)

Each sampler has several properties you can set. You can further customize a sampler by adding one or more Configuration Elements to it. Also, note that JMeter sends requests in the order that you add them to the tree.

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 Thread Group 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 to the Request controller. 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

## Listeners

Listeners provide access to the information JMeter gathers about the test cases while JMeter runs. The simplest listener, the Graph Results listener plots the response times on a graph. Listeners provide a graphical view of the data that JMeter generates.

Additionally, listeners can direct the data they collect to a file for later use. Every listener in JMeter provides a field to indicate the file to store data to.

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

There are several interesting listeners that come with JMeter.

## Timers

By default, a JMeter thread sends requests without pausing between each request. 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.

The timer will cause JMeter to delay a certain amount of time between each request that a thread makes.

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.

## 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, "". 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.

To view the assertion results, add an Assertion Listener to the Thread Group.

## Configuration Elements

A configuration element works closely with a Sampler. Although it does not send requests (except for HTTP Proxy Server ), 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).


http://jakarta.apache.org/jmeter/images/screenshots/http-config/http-config-example.png

Figure 1 - Test Plan Showing Accessability of Configuration Elements

## 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 for more details on when Pre-Processors are executed.

## 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 for more details on when Pre-Processors are executed.