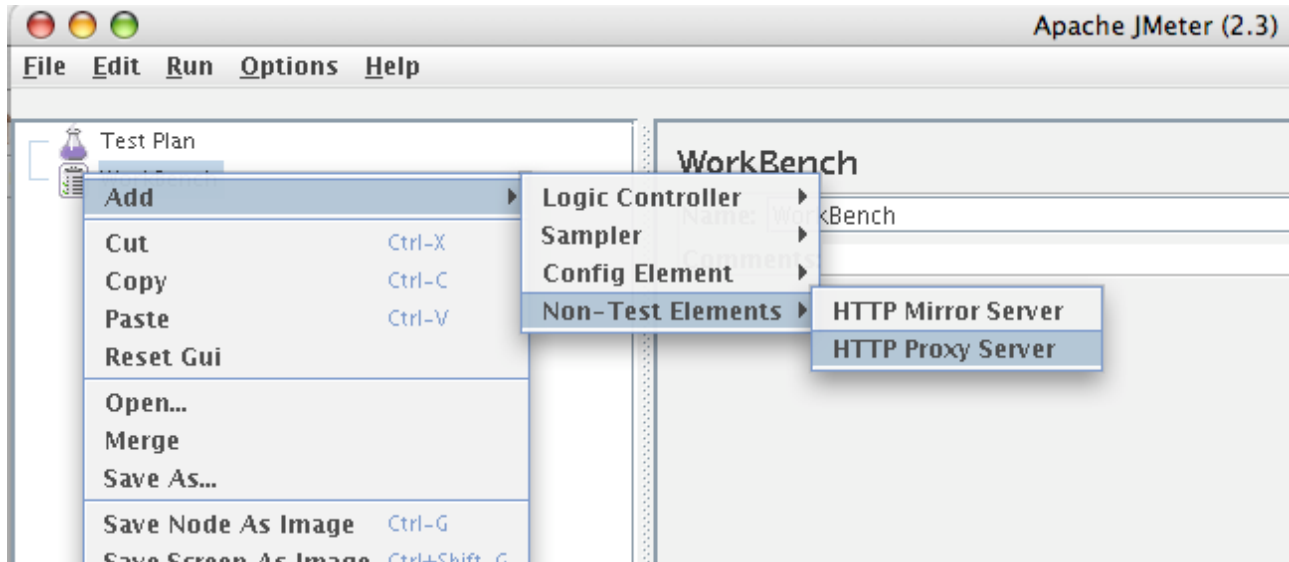
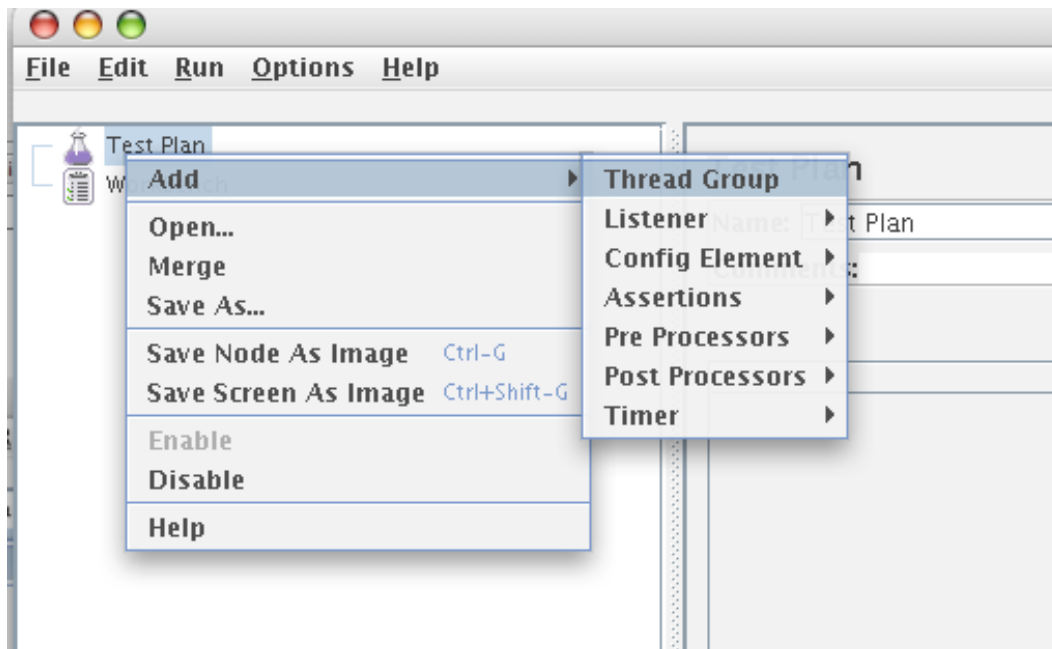


First in order to configure our test case, we need to reproduce our typical browsing path containing all the pages visited by the visitors on our systems.

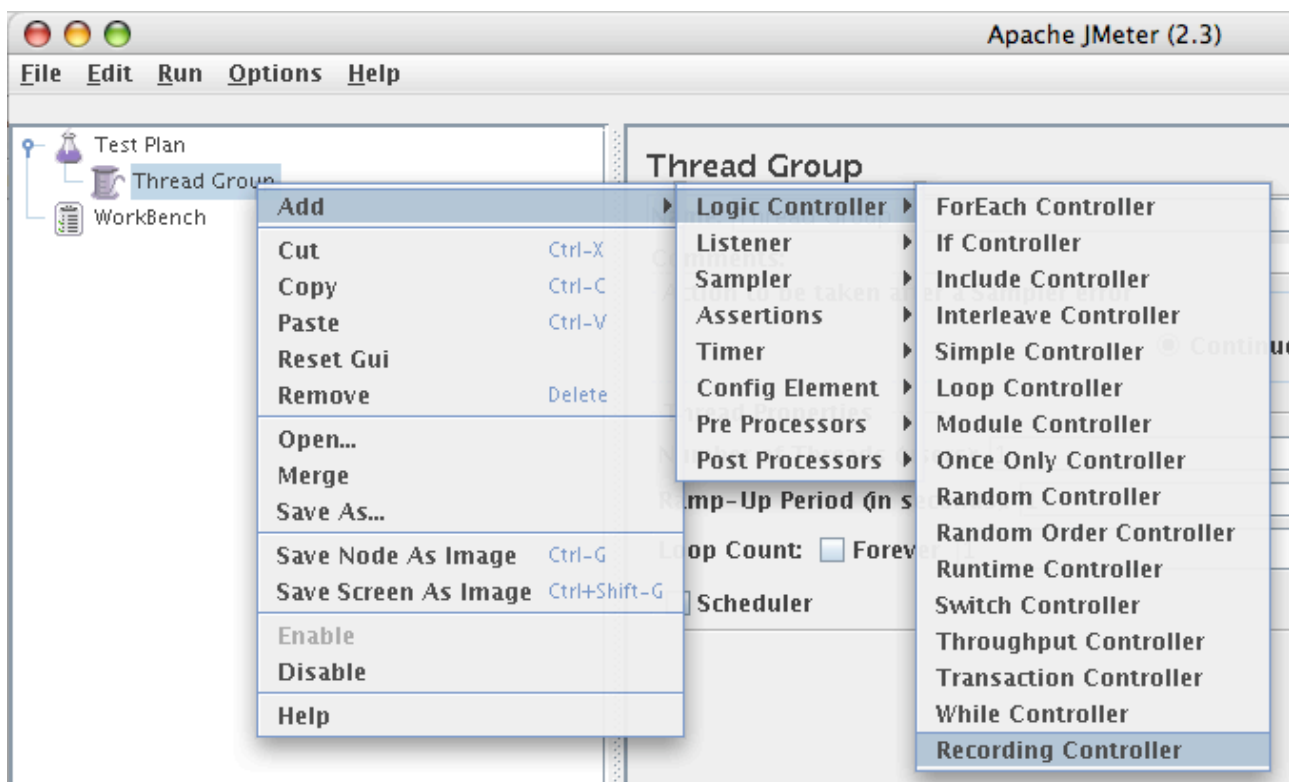
So in order to grab all the visitors requests we add to our workbench a non-test-element of the proxy type.



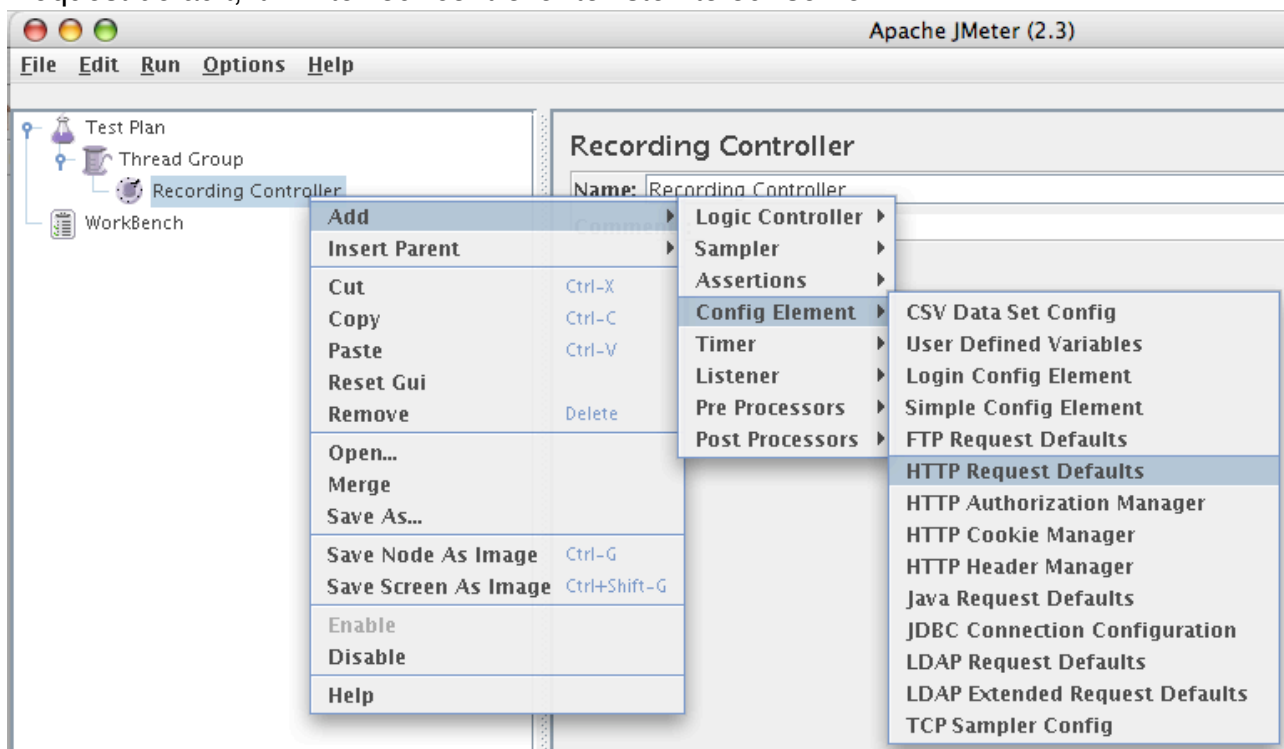
We add to our test plan a thread group, this thread group will represent the number of visitors using our site.



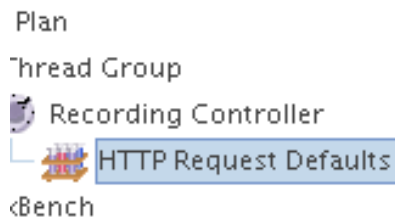
In order to record the different pages the users will be visiting we add to our thread group a logic controller, we add a recording controller.



Then we add to our recording controller a config element, this element will be an HTTP Request default, it will tell our controller to listen to our server.



We tell our thread to listen to our server and to the correspondingly port



HTTP Request Defaults

Name: HTTP Request Defaults

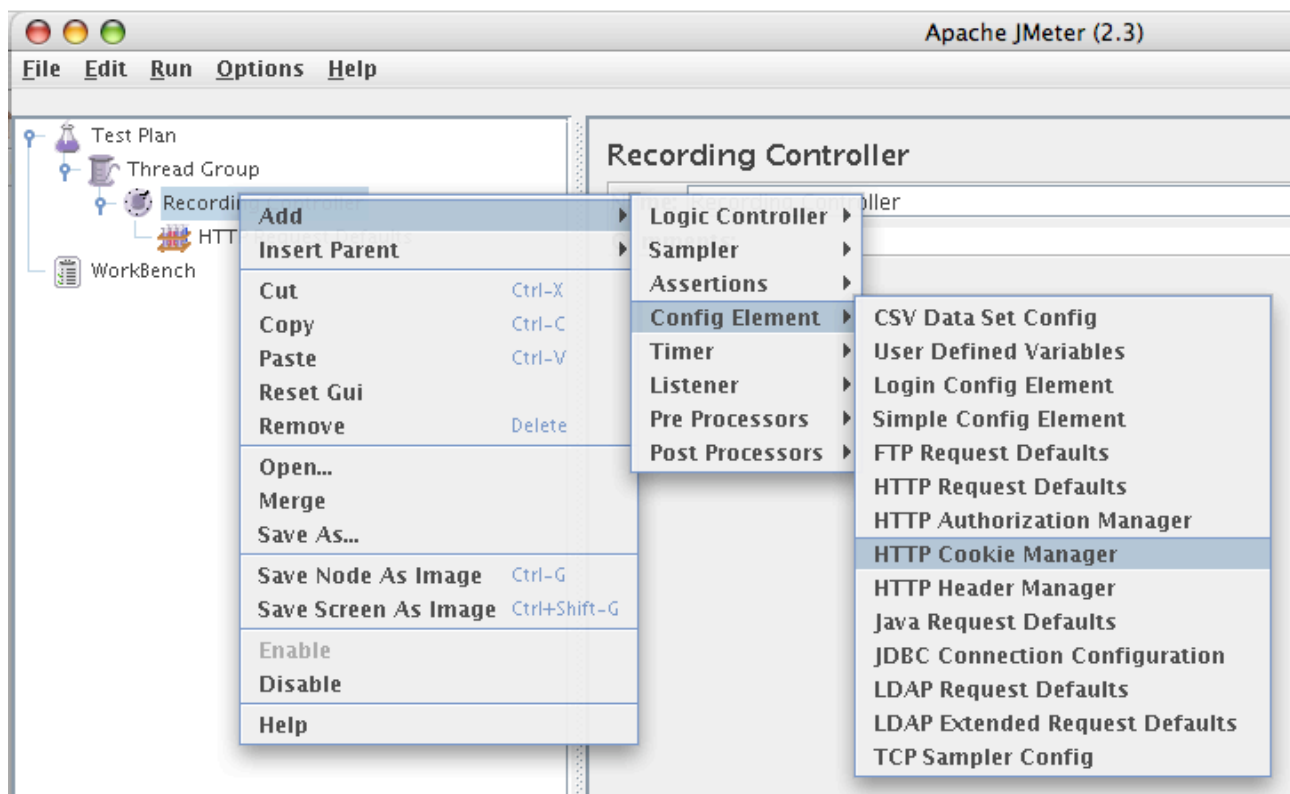
Comments:

Server Name or IP: www.magnolia.info

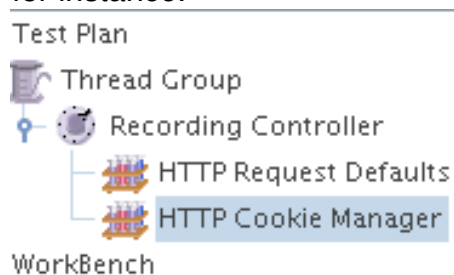
Port Number: 8080

Protocol (default http):

We also add to our recording controller an HTTP Cookie manager, this will make sure that we can store and use all the cookies needed while visiting our pages (for identification purposes for instance, or any other purpose)



We clear our cookies after each iteration, so that every time we get a new session cookie for instance.



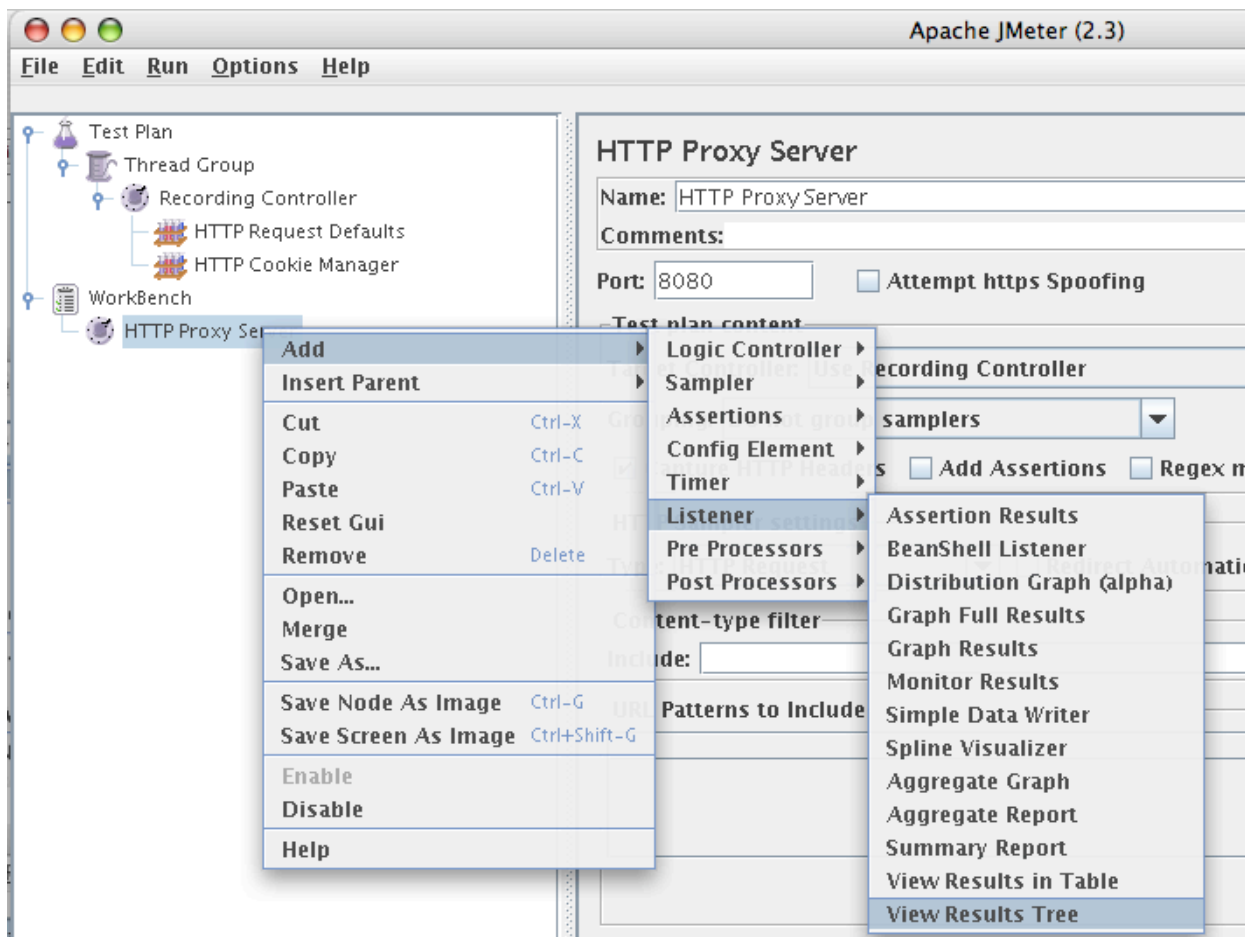
HTTP Cookie Manager

Name: HTTP Cookie Manager

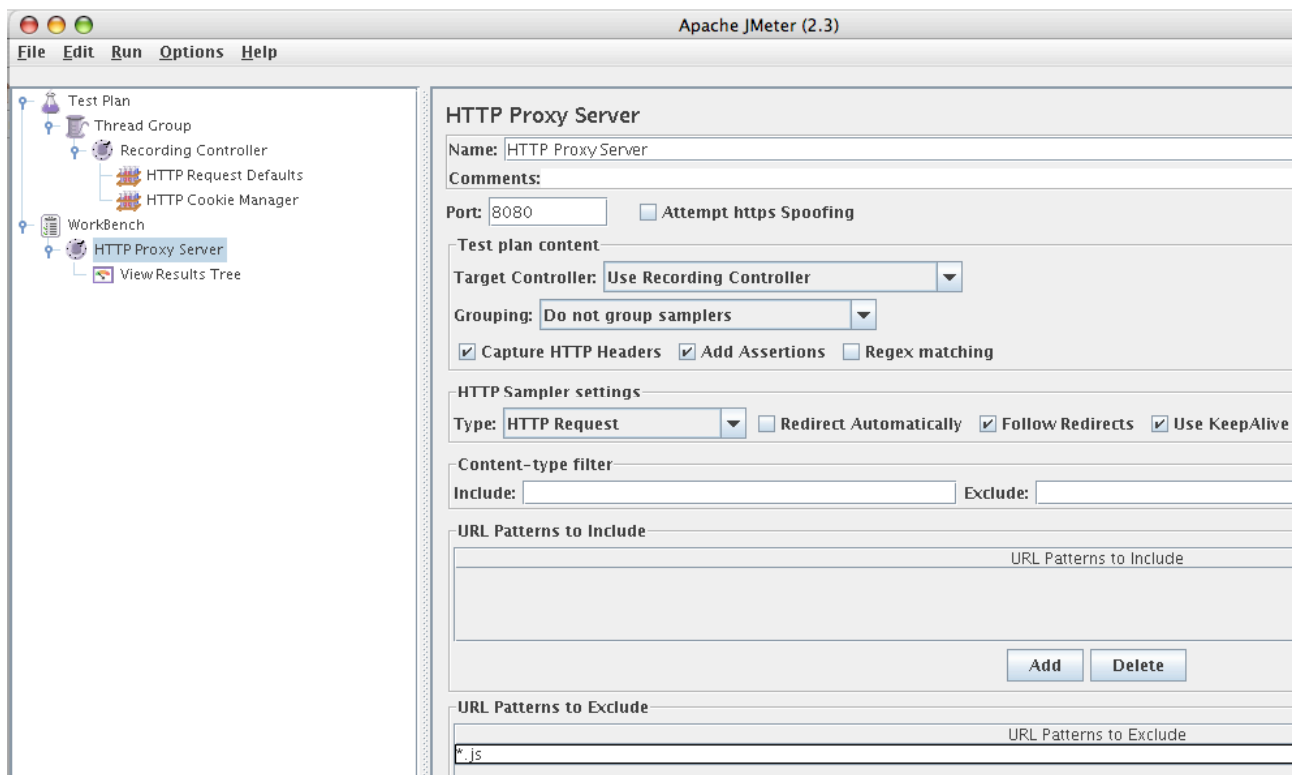
Comments:

☒ **Clear cookies each iteration?**

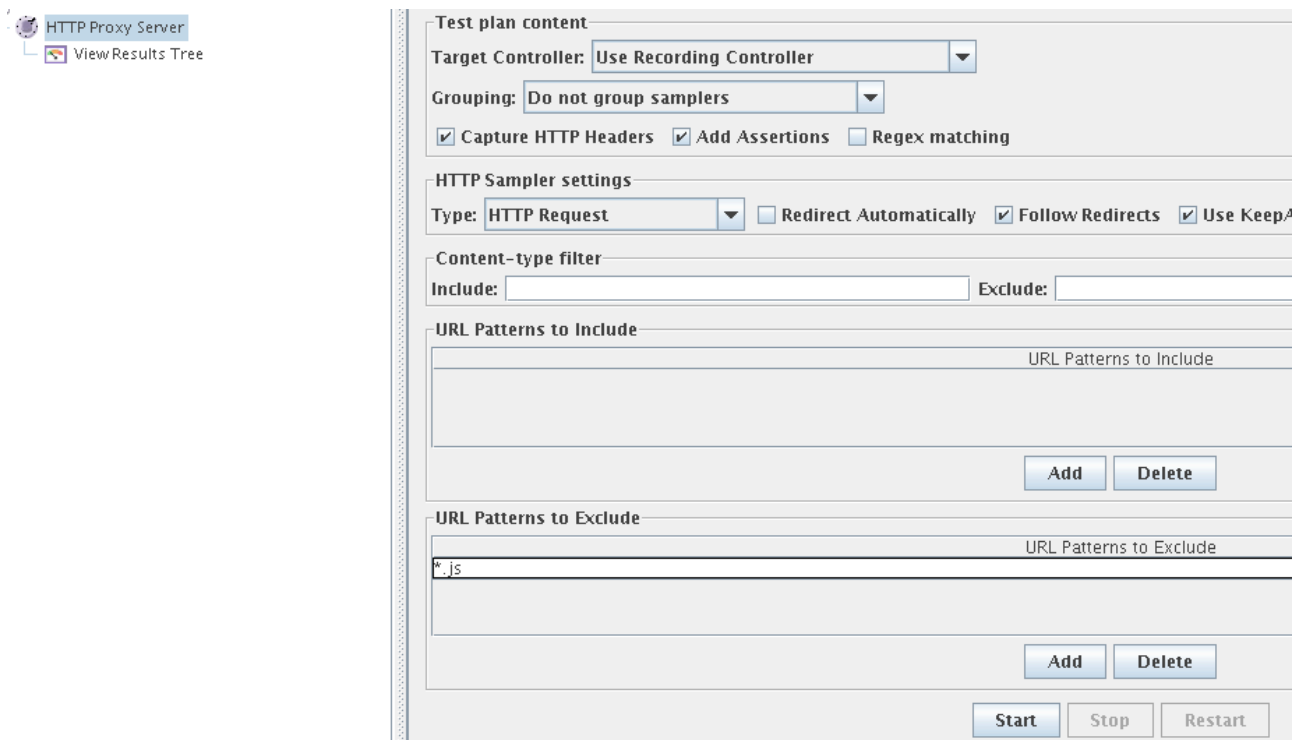
We add to our HTTP Proxy a listener, a result tree, so we can see the pages that will be visited by the proxy.



We exclude on the proxy server some elements from our tests, we can exclude for instance javascript elements or gif images if we are testing purely the server answer and not the filesystem performance



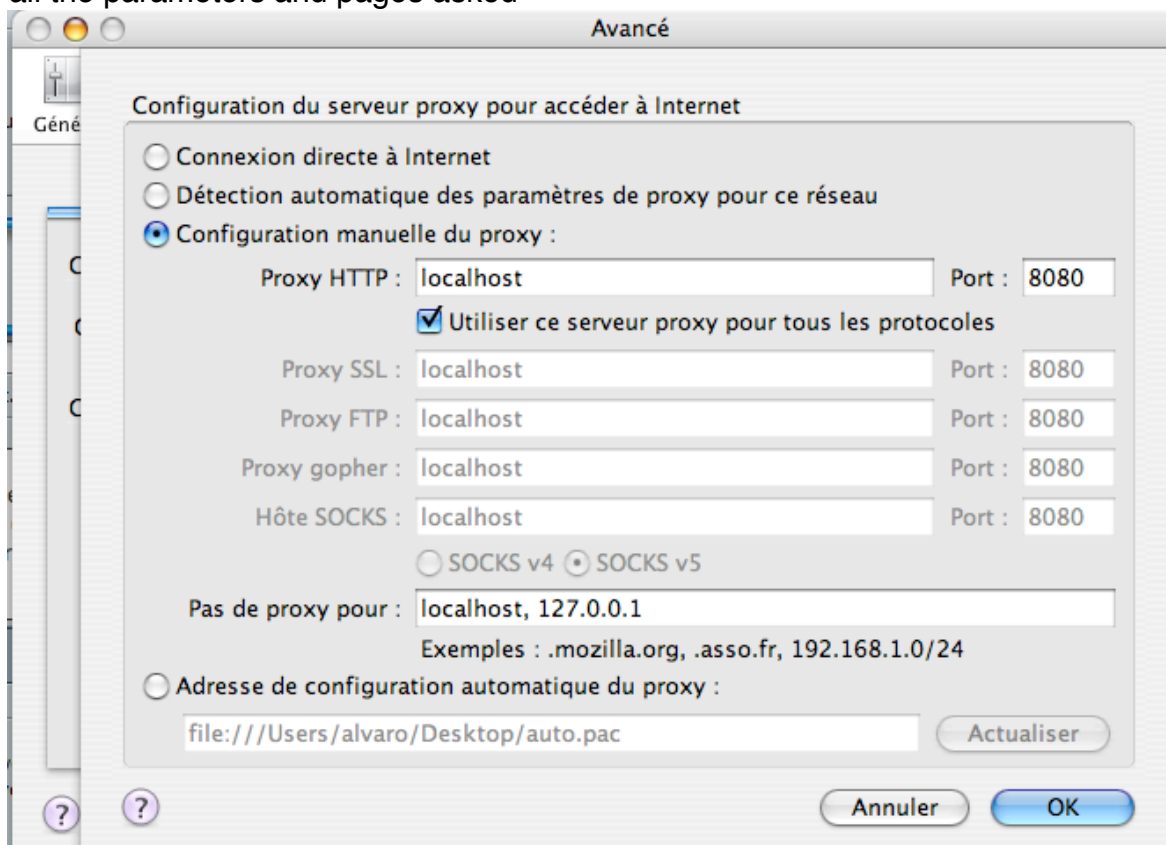
We add assertions to our proxy, by clicking the appropriate radio button, so we can test the validity and behavior of every answer from the server



The screenshot shows the 'HTTP Proxy Server' configuration window in JMeter. The 'Test plan content' section has 'Target Controller' set to 'Use Recording Controller' and 'Grouping' set to 'Do not group samplers'. The 'Capture HTTP Headers', 'Add Assertions', and 'Regex matching' checkboxes are all checked. The 'HTTP Sampler settings' section has 'Type' set to 'HTTP Request', 'Redirect Automatically' unchecked, 'Follow Redirects' checked, and 'Use Keep-Alive' checked. The 'Content-type filter' section has empty 'Include' and 'Exclude' fields. The 'URL Patterns to Include' section has an empty list with 'Add' and 'Delete' buttons. The 'URL Patterns to Exclude' section has a list containing '*.js' with 'Add' and 'Delete' buttons. At the bottom are 'Start', 'Stop', and 'Restart' buttons.

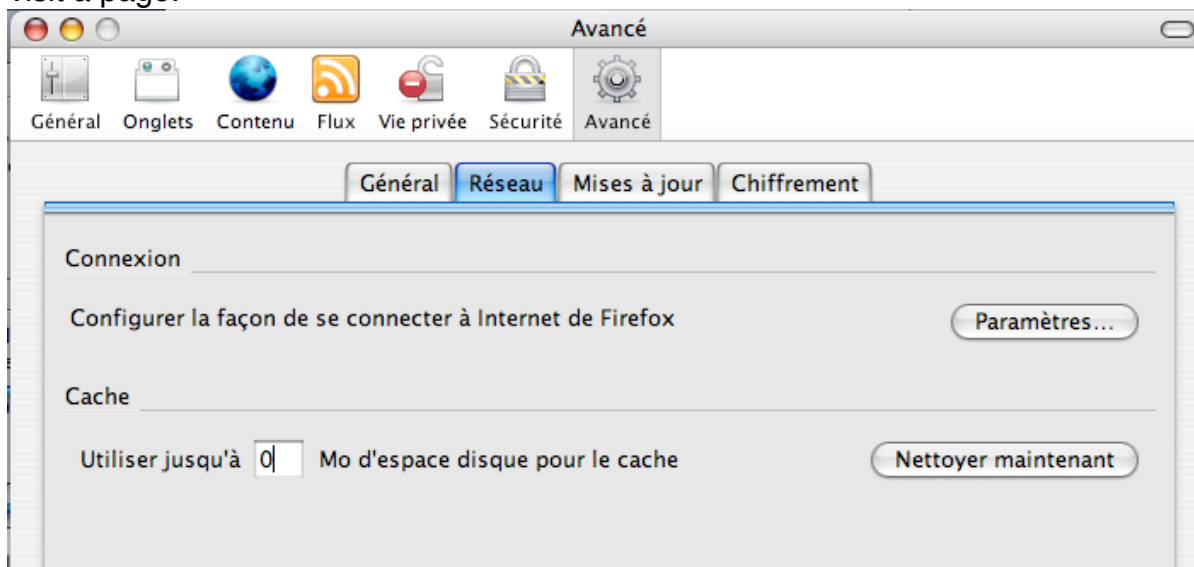
Then we configure the firefox browser:

We ask him to listen to our own machine (localhost) on the port 8080, so that it grabs all the pages coming from the jmeter proxy. Every page is asked to jmeter and jmeter records all the parameters and pages asked

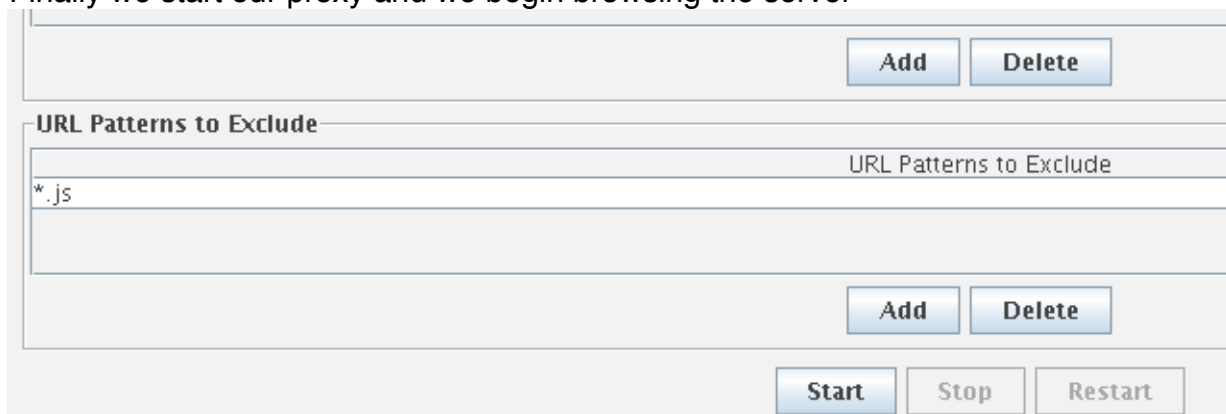


The screenshot shows the 'Avancé' (Advanced) network settings window in Firefox. The 'Configuration du serveur proxy pour accéder à Internet' (Configure proxy server to access the Internet) section is active. The 'Connexion directe à Internet' (Direct connection to Internet) radio button is selected. The 'Déttection automatique des paramètres de proxy pour ce réseau' (Automatic detection of proxy settings for this network) radio button is also selected. The 'Configuration manuelle du proxy' (Manual proxy configuration) radio button is selected. The 'Proxy HTTP' field is set to 'localhost' and the 'Port' is set to '8080'. The 'Utiliser ce serveur proxy pour tous les protocoles' (Use this proxy server for all protocols) checkbox is checked. The 'Proxy SSL' field is set to 'localhost' and the 'Port' is set to '8080'. The 'Proxy FTP' field is set to 'localhost' and the 'Port' is set to '8080'. The 'Proxy gopher' field is set to 'localhost' and the 'Port' is set to '8080'. The 'Hôte SOCKS' field is set to 'localhost' and the 'Port' is set to '8080'. The 'SOCKS v4' radio button is selected. The 'Pas de proxy pour' (No proxy for) field is set to 'localhost, 127.0.0.1'. The 'Adresse de configuration automatique du proxy' (Automatic proxy configuration address) field is set to 'file:///Users/alvaro/Desktop/auto.pac'. The 'Actualiser' (Refresh) button is visible. At the bottom are 'Annuler' (Cancel) and 'OK' buttons.

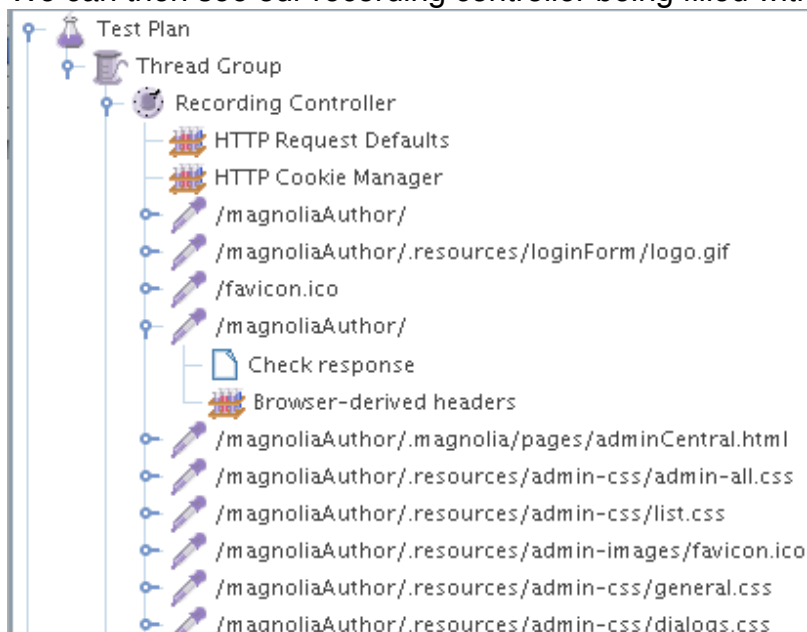
We set the cache of our browser to 0 and we erase it, so that every incoming request will be coming directly from the server and we won't have any fake 304 code the first time we visit a page.



Finally we start our proxy and we begin browsing the server



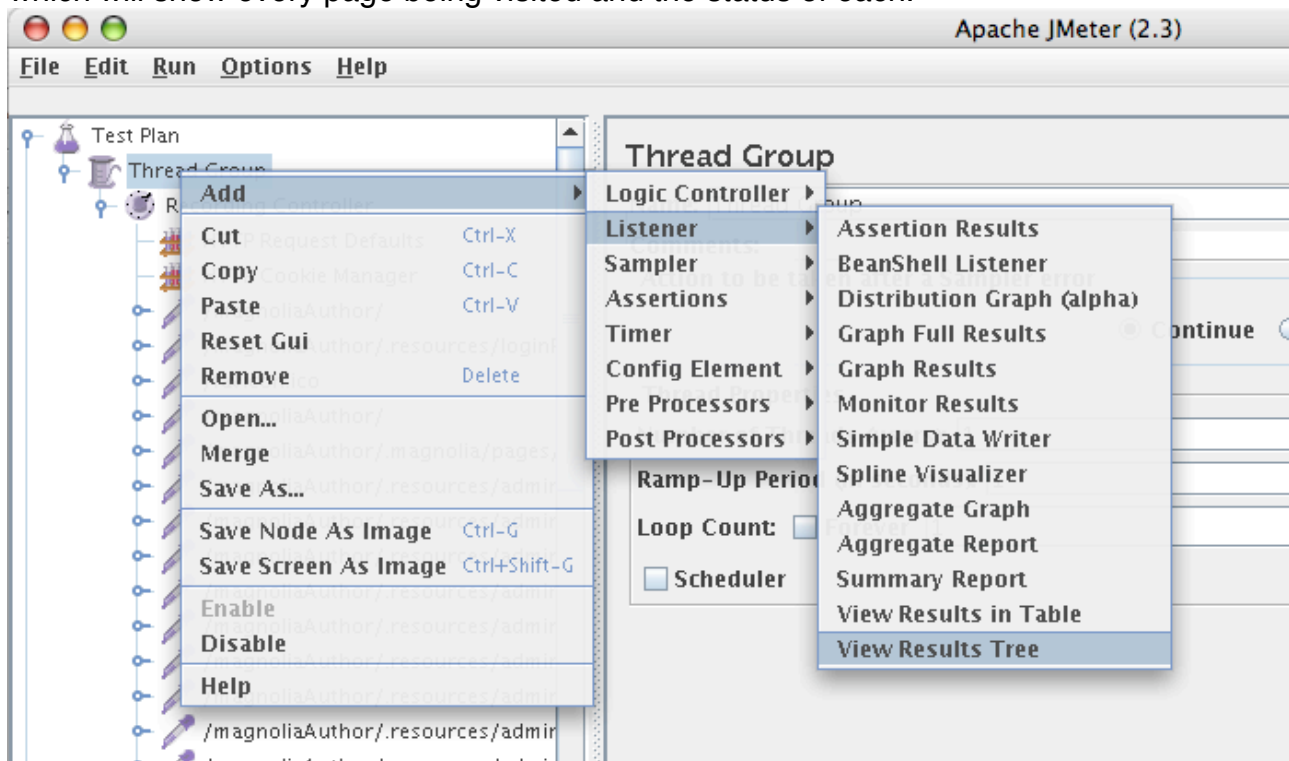
We can then see our recording controller being filled with the data coming from the server



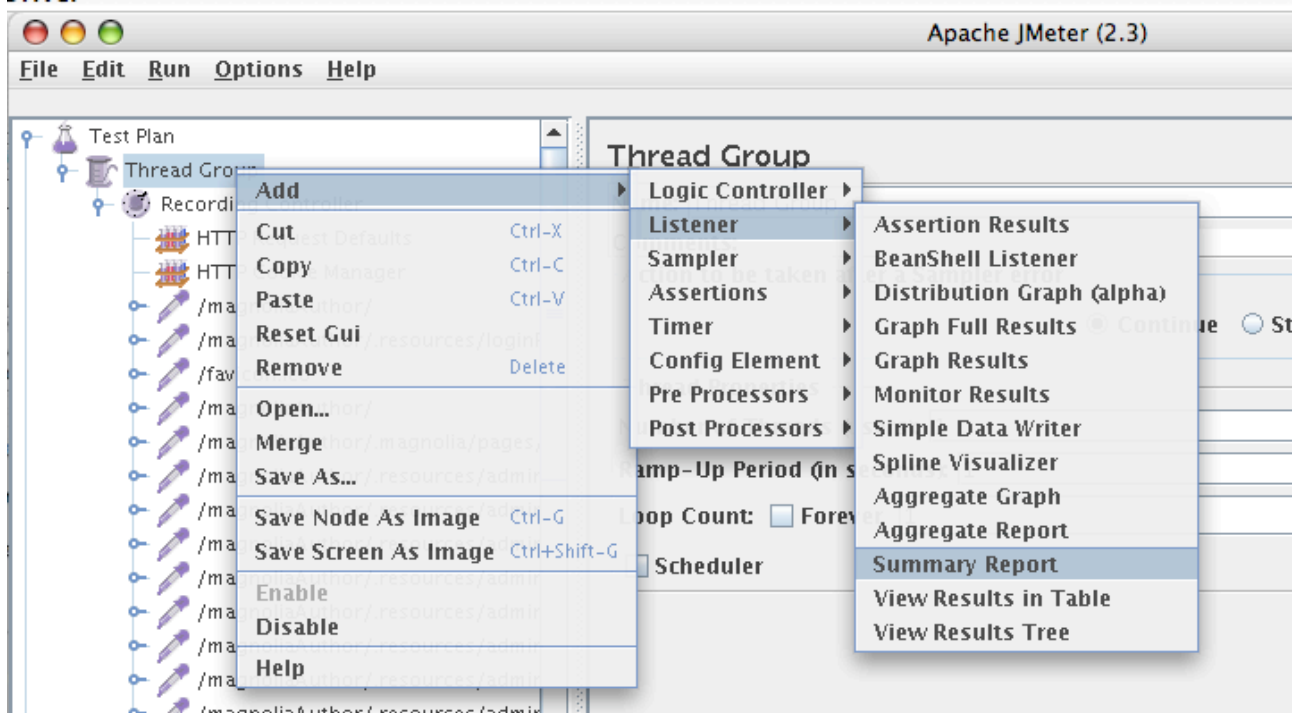
We then stop our server after we have recorded all the visitor browsing plan.

The screenshot shows the configuration window for an HTTP Request sampler in Apache JMeter. At the top, there are three checkboxes: ☒ Capture HTTP Headers, ☒ Add Assertions, and ☐ Regex matching. Below this is the 'HTTP Sampler settings' section, where the 'Type' is set to 'HTTP Request'. There are also checkboxes for ☐ Redirect Automatically, ☒ Follow Redirects, and ☒ Use KeepAlive. The 'Content-type filter' section has 'Include' and 'Exclude' text boxes. The 'URL Patterns to Include' section has a text box and 'Add' and 'Delete' buttons. The 'URL Patterns to Exclude' section has a text box containing '*.js' and 'Add' and 'Delete' buttons. At the bottom right, there are 'Start', 'Stop', and 'Restart' buttons.

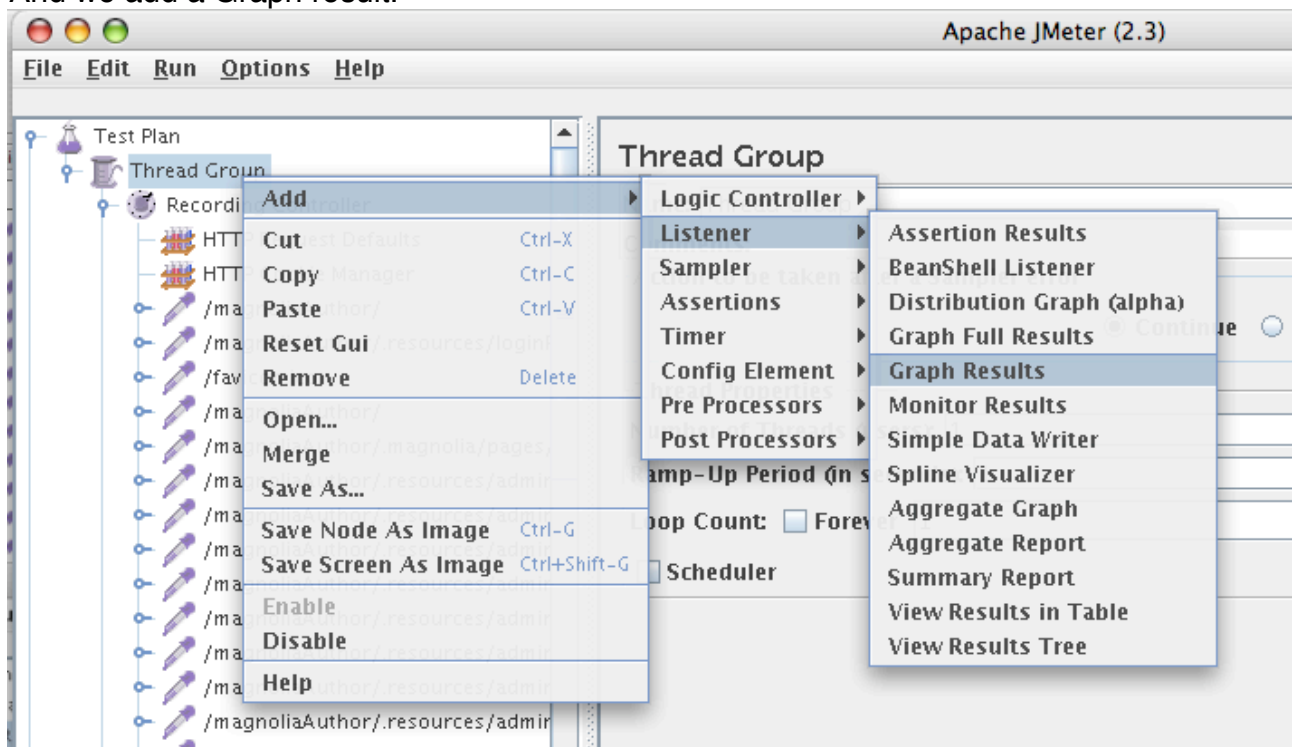
Once we have the elements recorded we add to our Thread Group a listener, a result tree, which will show every page being visited and the status of each.



We also add a summary report.

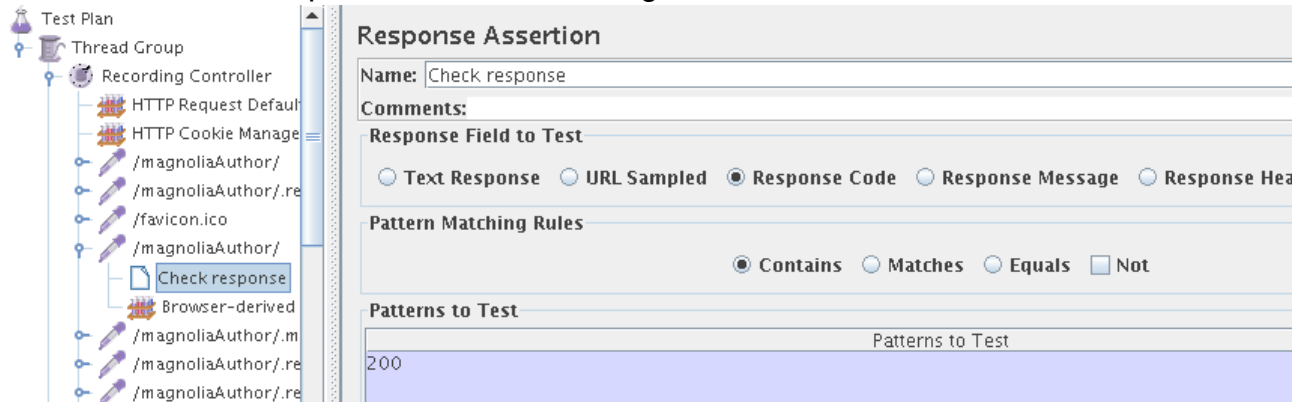


And we add a Graph result.

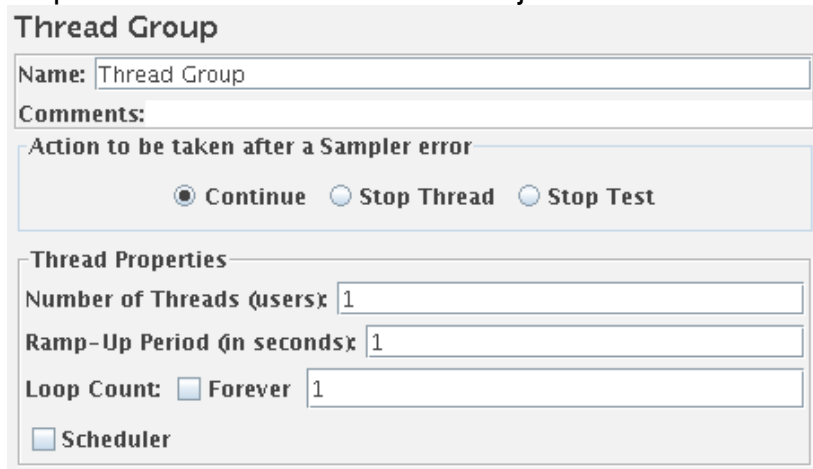


We can add an assertion for one of the pages saying that the response code must be 200, after for instance a login screen where we expect a 200 response code if we get logged in successfully and not an 401 code for instance. On the contrary of we expect a 401 code because we know that the user is not authorized or should not be authorized at this point

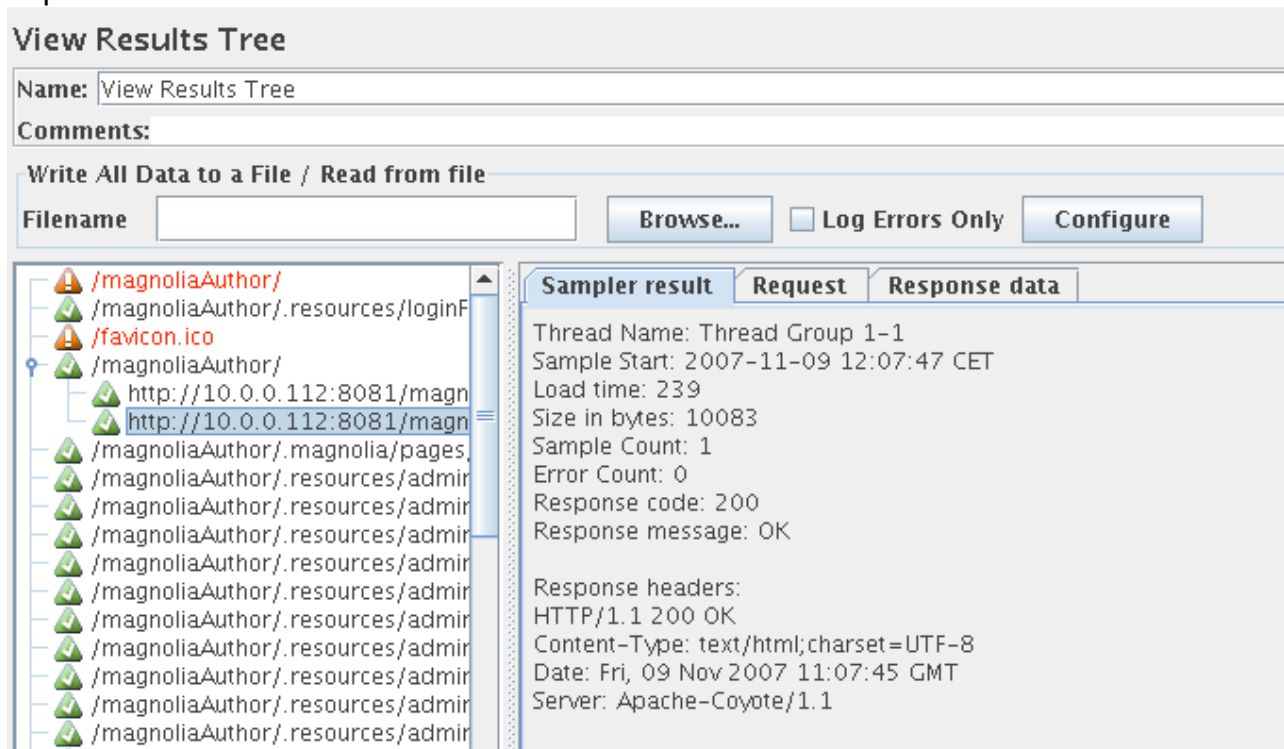
to get to the page then the assertion should be changed to contain a 401 code, then in our result tree the request will be marked as green.



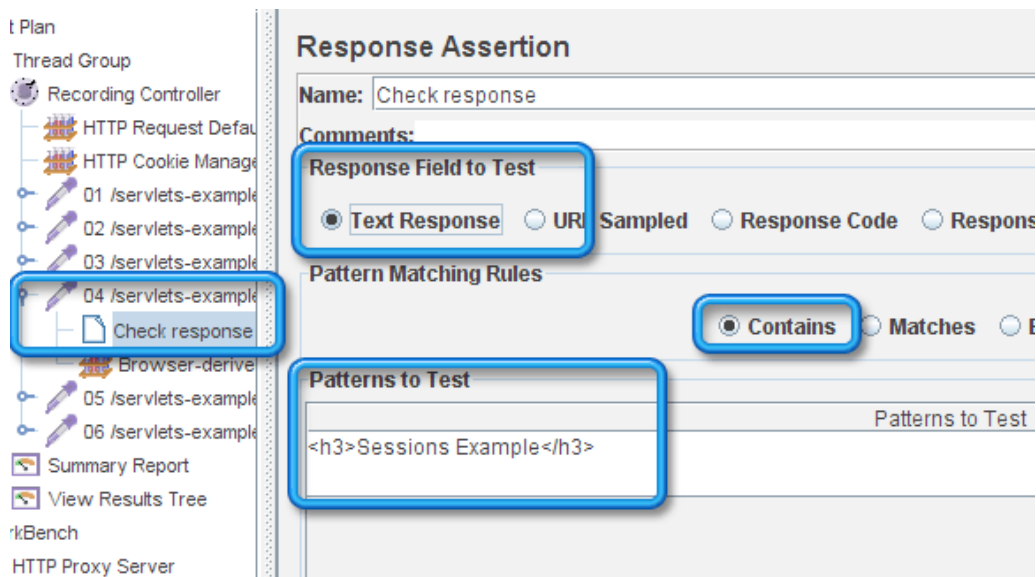
We first launch the test one time using one thread (user) and we do the whole testing road map one time. On the menu Run of jmeter we choose start.



As we see all the successful request appear on green, and the non successful on red (401 return code, 505, etc....) Here our assertion worked and the response code was 200 as we expected it to be.



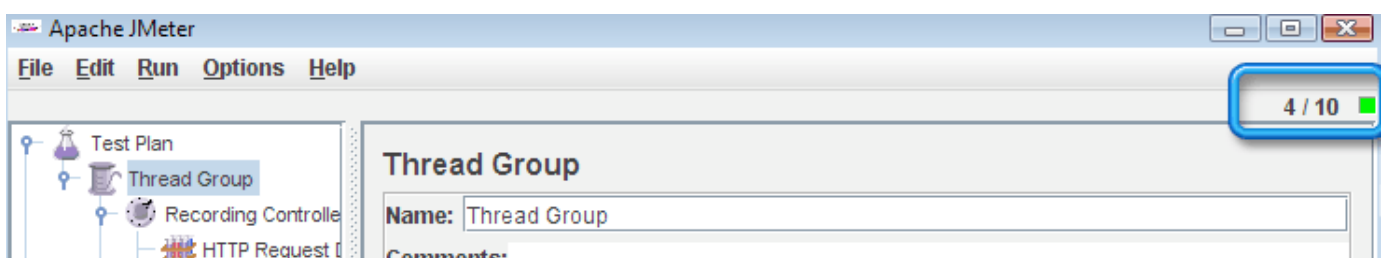
We can also check for other assertions like a page containing a certain element in a page, in this example our pages needed to contain the title: `<h3>Sessions Example</h3>`



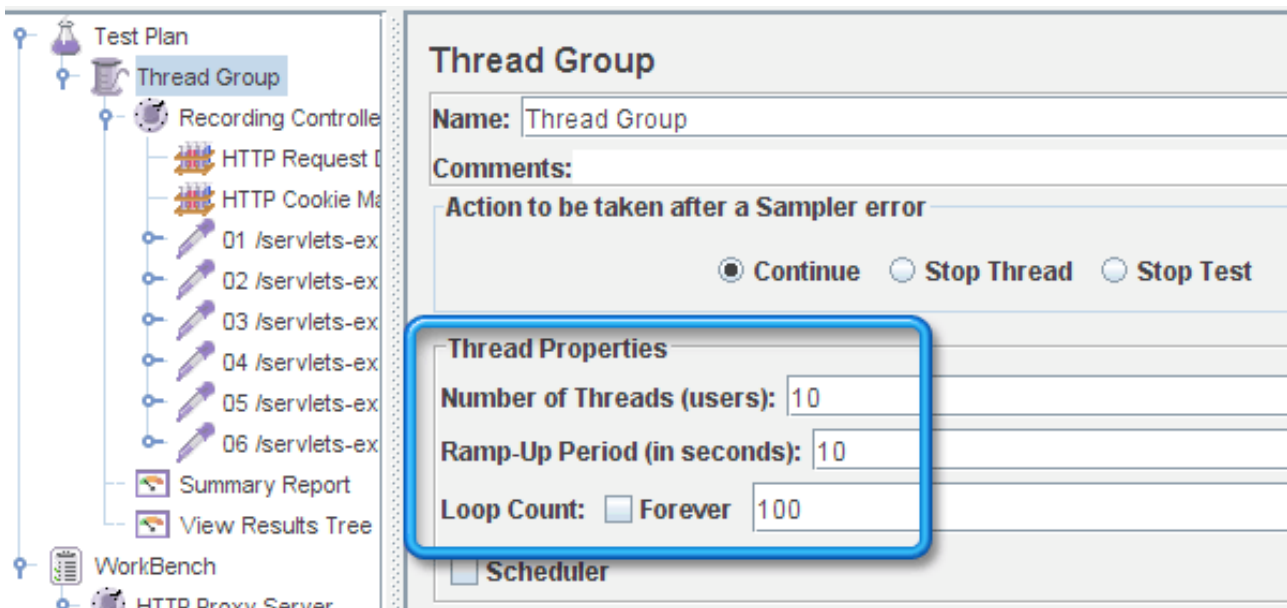
If we look to our Summary report we can see all the request being made and their respective values:

/magnoliaAuthor/.resour...	1	6	6	6	0.00	0.00%	166.7/...	7.00	43.0
/magnoliaAuthor/.resour...	1	7	7	7	0.00	0.00%	142.9/...	21.21	152.0
/magnoliaAuthor/.resour...	1	8	8	8	0.00	0.00%	125.0/...	12.82	105.0
/magnoliaAuthor/.resour...	1	16	16	16	0.00	0.00%	62.5/sec	279.42	4578.0
/magnoliaAuthor/.resour...	1	10	10	10	0.00	0.00%	100.0/...	134.86	1381.0
/magnoliaAuthor/.resour...	1	10	10	10	0.00	0.00%	100.0/...	238.18	2439.0
/magnoliaAuthor/.resour...	1	9	9	9	0.00	0.00%	111.1/...	159.72	1472.0
TOTAL	102	39	5	430	79.64	3.92%	24.6/sec	403.96	16846.3

The upper right corner shows the actual number of threads (users) active



Once we are happy with the first test scenario we can begin doing a real load test of our server increasing the number of threads and also the number of times that each thread will follow the scenario we defined before.



We have a nice summary report that will help us to evaluate how the server stands the load.

Summary Report										
Name: Summary Report										
Comments:										
Write All Data to a File										
Filename							Browse...	<input type="checkbox"/> Log Errors Only	Configure	
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes	
01 /servl...	1000	4	1	281	10,21	0,00%	83,5/sec	435,47	5343	
02 /servl...	1000	7	2	282	14,02	0,00%	83,5/sec	29,77	365	
03 /servl...	1000	4	1	88	4,66	0,00%	83,5/sec	0,00		
04 /servl...	1000	8	3	324	13,79	0,00%	83,5/sec	103,60	1270	
05 /servl...	1000	8	3	196	10,78	0,00%	83,5/sec	95,19	1167	
06 /servl...	1000	4	1	80	4,33	0,00%	83,5/sec	0,00		
TOTAL	6000	6	1	324	10,53	0,00%	500,0/sec	662,90	1357	

For info:

- * **Average (ms)** : Average response time for the element
- * **Min (ms)** : Minimum response value for the element
- * **Max (ms)** : Maximum response value for the element
- * **Std. Dev.** : Standard deviation for the element.
- * **Error %** : Error percentage of the element.
- * **Throughput** : throughput in sec or min or hour. ie How many times can we have the element in 1 sec, 1 min or 1 hour
- * **KB/sec** : Transfert speed in Kilo-octets/second
- * **Avg. Bytes** : Average number of bytes transfered for every request.