**Load Testing with JMeter: Part 3 - Replaying Apache Logs**

Posted by [Brandon Konkle](http://lincolnloop.com/about/bkonkle/) on September 19, 2012. Filed under [django](http://lincolnloop.com/blog/categories/django/)

A while ago, I wrote a couple of blog entries about load testing with JMeter ([Part 1](http://lincolnloop.com/blog/2011/sep/21/load-testing-jmeter-part-1-getting-started/) and [Part 2](http://lincolnloop.com/blog/2011/oct/12/load-testing-jmeter-part-2-headless-testing-and-je/)). I promised a third entry covering how to use JMeter to replay Apache logs and roughly recreate production load, but I never followed through with it. Today, I intend to rectify this grievous error.

**Parsing your Apache Logs**

There is more than one way to do this, but my preferred method is to use a simple Python script to do some filtering of the Apache log file you want to use and to output the desired urls as a tidy CSV file. I am using the ‘apachelog’ module for this (also available as a [gist](https://gist.github.com/3746198)):

*#!/usr/bin/env python* """ Requires apachelog. `pip install apachelog` """ **from** \_\_future\_\_ **import** with\_statement **import** apachelog **import** csv **import** re **import** sys **from** optparse **import** OptionParser STATUS\_CODE = '%>s' REQUEST = '%r' USER\_AGENT = '%{User-Agent}i' MEDIA\_RE = re.compile(r'\.png|\.jpg|\.jpeg|\.gif|\.tif|\.tiff|\.bmp|\.js|\.css|\.ico|\.swf|\.xml') SPECIAL\_RE = re.compile(r'xd\_receiver|\.htj|\.htc|/admin') **def** main(): usage = "usage: %prog [options] LOGFILE" parser = OptionParser(usage=usage) parser.add\_option( "-o", "--outfile", dest="outfile", action="store", default="urls.csv", help="The output file to write urls to", metavar="OUTFILE" ) parser.add\_option( "-f", "--format", dest="logformat", action="store", default=r'%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"', help="The Apache log format, copied and pasted from the Apache conf", metavar="FORMAT" ) parser.add\_option( "-g", "--grep", dest="grep", action="store", help="Simple, plain text filtering of the log lines. No regexes. This " "is useful for things like date filtering - DD/Mmm/YYYY.", metavar="TEXT" ) options, args = parser.parse\_args() **if** **not** args: sys.stderr.write('Please provide an Apache log to read from.\n') sys.exit(1) create\_urls(args[0], options.outfile, options.logformat, options.grep) **def** create\_urls(logfile, outfile, logformat, grep=None): parser = apachelog.parser(logformat) **with** open(logfile) **as** f, open(outfile, 'wb') **as** o: writer = csv.writer(o) *# Status spinner* spinner = "|/-\\" pos = 0 **for** i, line **in** enumerate(f): *# Spin the spinner* **if** i % 10000 == 0: sys.stdout.write("\r" + spinner[pos]) sys.stdout.flush() pos += 1 pos %= len(spinner) *# If a filter was specified, filter by it* **if** grep **and** **not** grep **in** line: **continue** **try**: data = parser.parse(line) **except** apachelog.ApacheLogParserError: **continue** method, url, protocol = data[REQUEST].split() *# Check for GET requests with a status of 200* **if** method != 'GET' **or** data[STATUS\_CODE] != '200': **continue** *# Exclude media requests and special urls* **if** MEDIA\_RE.search(url) **or** SPECIAL\_RE.search(url): **continue** *# This is a good record that we want to write* writer.writerow([url, data[USER\_AGENT]]) **print** ' done!' **if** \_\_name\_\_ == '\_\_main\_\_': main()

This script takes the name of the logfile to parse as the one required argument, and provides a few options as well.

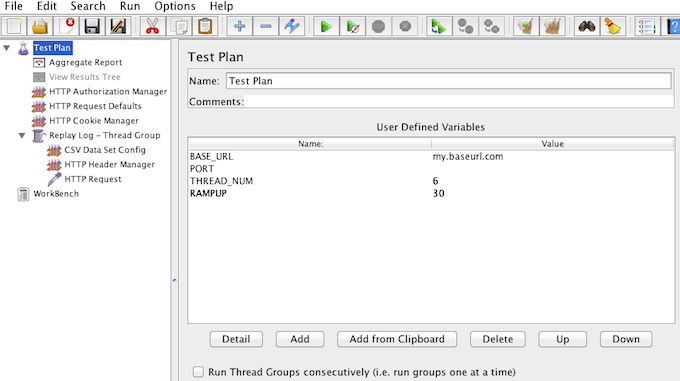
Usage: createurls.py [options] LOGFILE Options: -h, --help show this help message and exit -o OUTFILE, --outfile=OUTFILE The output file to write urls to -f FORMAT, --format=FORMAT The Apache log format, copied and pasted from the Apache conf -g TEXT, --grep=TEXT Simple, plain text filtering of the log lines. No regexes. This is useful for things like date filtering - DD/Mmm/YYYY.

The script will parse each line of your Apache log file, and check to see if it meets a few criteria before including it in your CSV file. First, it checks to see if the method was GET and the status code was 200. Then, it checks the regular expressions in MEDIA\_RE and SPECIAL\_RE, and if it matches either of them the record is discarded. This is so that you can filter out media requests or special case urls such as the Django admin. If you specified a grep filter, it will only include lines where that plain text value is present. If your format differs from the default, make sure to pass the format along with the -f option, or modify the script to make the change permanent.

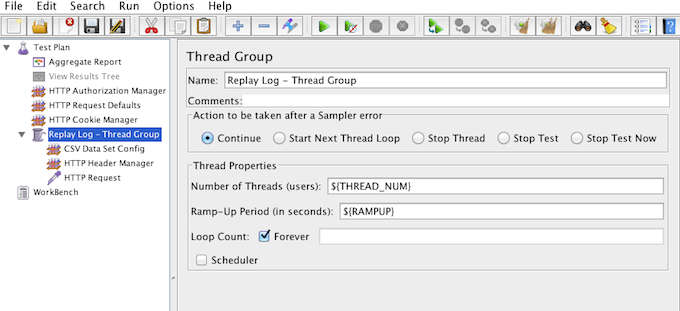
The result should be a urls.csv file with a url and a user agent on each line. This file will be used to recreate the requests in JMeter.

**Replaying in JMeter**

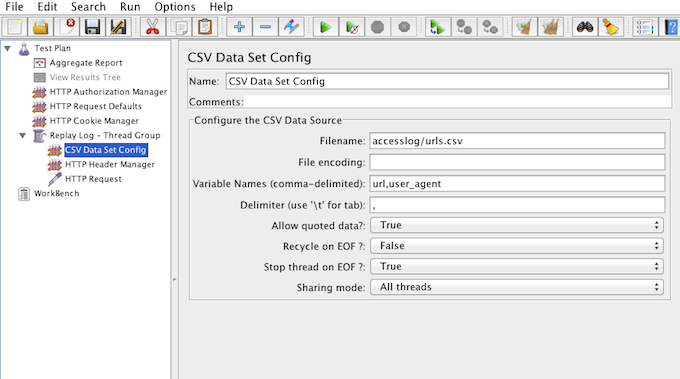
Setting this up in JMeter is rather easy. I use a separate test plan for replaying logs:



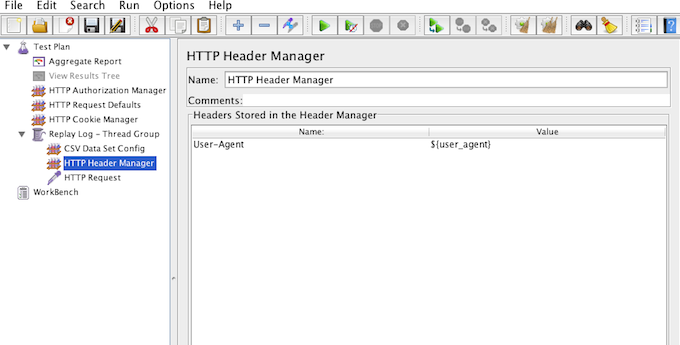
Within the plan, I’ve got a Thread Group created called “Replay Log”:



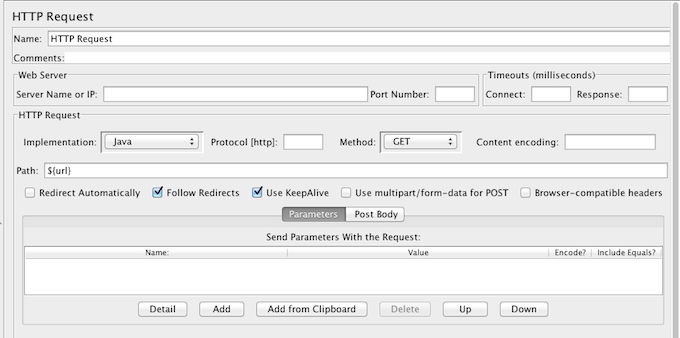
In that Thread Group, I have a CSV reader that loads the urls and populates two variables – url and user\_agent:



I use a Header Manager to provide the User Agent:



Finally, I use the url variable as the path in the HTTP Request:



With all of that configured, I can now replay the log and take a a measurement of some real-world urls under load!

**Tweaking the Parser**

There are a couple of ways you can customize the parser script to your liking. I’m only allowing requests with a status code of 200 through. You can customize this on line 86 of the script and allow 404s or any other code you’d like to include in your urls.

If you want to add more media types, you can extend the MEDIA\_RE variable at the top of the script. You can also exclude special urls by adding to the SPECIAL\_RE variable. In both cases, just use a pipeline (|) to separate your entries.

You can add more data to the CSV file so that you can use it in JMeter by customizing the writer call on line 94 of the script, adding in more details that apachelog recognizes from each log line. Make sure to modify your CSV Data Set module in JMeter to match this new CSV format.

I apologize for the delay in getting this post out, but I hope it’s helpful to you in your load testing endeavors!