Log Parser

Scott Hutchings
4/14/15
IS 520

Executive Summary

Log files are everywhere in today's technological world. Even some users of software products may need to consult log files. Some of the more prevalent log files are associated with websites and the web servers that host them.

Apache is a widely used HTTP server that hosts websites, and for each and every request to the Apache web server a log entry is recorded. This results in millions of log entries for just one web server. Sometimes it is necessary to look through these log files in order to understand why a certain website is failing, or who may be accessing the website, or to identify IP addresses that have attempted a denial of service attack.

The problem is trying to parse through the countless lines of log entries in each log file to find what is needed. At my job I was asked to specifically find which websites are being hit and how much.

I built an Excel workbook that can parse through log files and return a count of how many similar matches have appeared in the log files. You are able to use regular expressions, which are a simple way of specifying what pattern to look for, to dissect the log files. You are able to parse through multiple log files. And you can specify the groups and which ones you want to count.

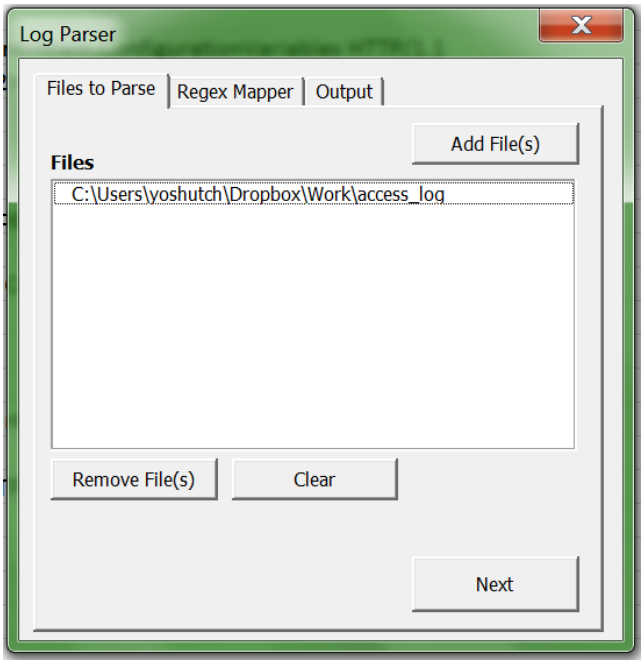
I found that this application could be used for not only log files, but anything that might need regular expressions to parse through text.

Implementation

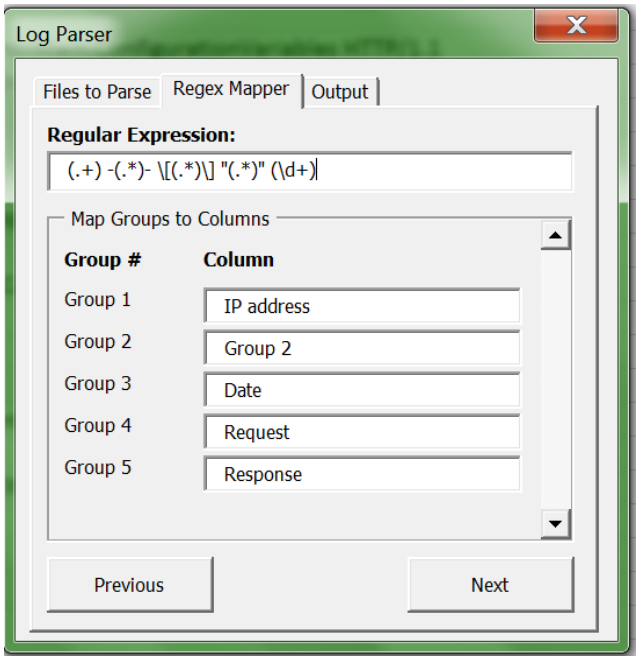
This Excel workbook has the ability to parse through log files or any other text file and count matched lines by a regular expression.

Component	**Usage**
Userform	User input
FileDialog	Select files
Dictionary object	Data collection
RegExp	To match
Custom Excel Ribbon	To open userform and other options
Excel File input	To read log files

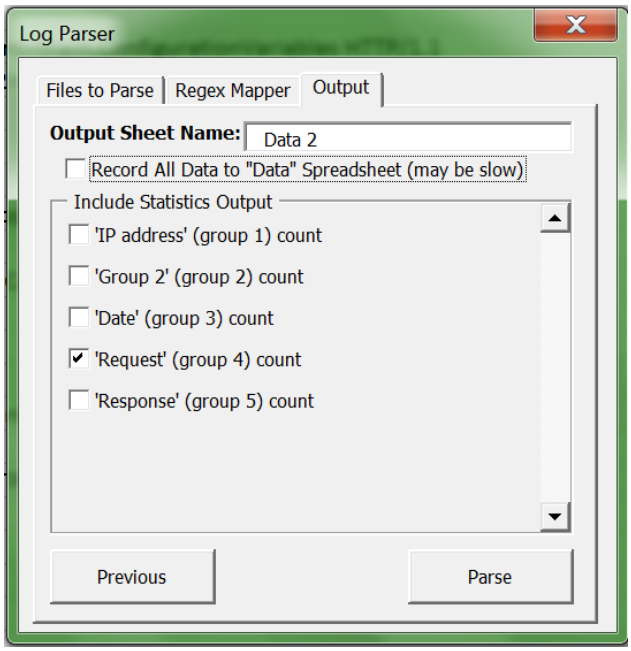
Below is the userform with tabs, the first tab is the file picker to choose the log files to parse.



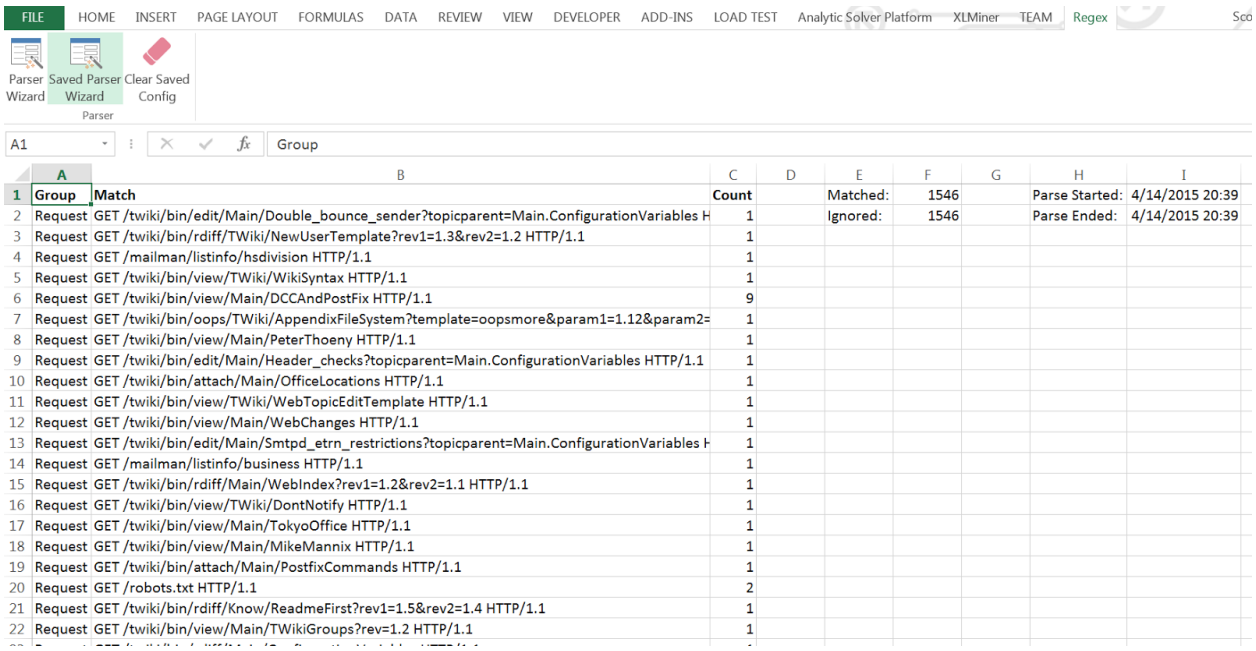
On the second tab the user can specify the regular expression and what to name each regex group.



On the third and final tab of the userform the user is able to specify which columns to count and also the name of the data sheet to write the parsed data to.



The parsing algorithm will use the regular expression to match log entries and then it will group them by whatever column/group the user specified. In effect the result is similar to a SQL statement like: “select column_name, count(*), from table group by column_name” so as to show the number of times a particular matched portion of an entry was in all the log files included. Below is a screenshot of the data output:



	A	B	C	D	E	F	G	H	I
1	Group	Match	Count		Matched:	1546		Parse Started:	4/14/2015 20:39
2	Request	GET /twiki/bin/edit/Main/Double_bounce_sender?topicparent=Main.ConfigurationVariables H	1		Ignored:	1546		Parse Ended:	4/14/2015 20:39
3	Request	GET /twiki/bin/rdiff/TWiki/NewUserTemplate?rev1=1.3&rev2=1.2 HTTP/1.1	1						
4	Request	GET /mailman/listinfo/hsdivision HTTP/1.1	1						
5	Request	GET /twiki/bin/view/TWiki/WikiSyntax HTTP/1.1	1						
6	Request	GET /twiki/bin/view/Main/DCCAndPostFix HTTP/1.1	9						
7	Request	GET /twiki/bin/oops/TWiki/AppendixFileSystem?template=oopsmore&param1=1.12&param2=	1						
8	Request	GET /twiki/bin/view/Main/PeterThoeny HTTP/1.1	1						
9	Request	GET /twiki/bin/edit/Main/Header_checks?topicparent=Main.ConfigurationVariables HTTP/1.1	1						
10	Request	GET /twiki/bin/attach/Main/OfficeLocations HTTP/1.1	1						
11	Request	GET /twiki/bin/view/TWiki/WebTopicEditTemplate HTTP/1.1	1						
12	Request	GET /twiki/bin/view/Main/WebChanges HTTP/1.1	1						
13	Request	GET /twiki/bin/edit/Main/Smtpd_etrn_restrictions?topicparent=Main.ConfigurationVariables H	1						
14	Request	GET /mailman/listinfo/business HTTP/1.1	1						
15	Request	GET /twiki/bin/rdiff/Main/WebIndex?rev1=1.2&rev2=1.1 HTTP/1.1	1						
16	Request	GET /twiki/bin/view/TWiki/DontNotify HTTP/1.1	1						
17	Request	GET /twiki/bin/view/Main/TokyoOffice HTTP/1.1	1						
18	Request	GET /twiki/bin/view/Main/MikeMannix HTTP/1.1	1						
19	Request	GET /twiki/bin/attach/Main/PostfixCommands HTTP/1.1	1						
20	Request	GET /robots.txt HTTP/1.1	2						
21	Request	GET /twiki/bin/rdiff/Know/ReadmeFirst?rev1=1.5&rev2=1.4 HTTP/1.1	1						
22	Request	GET /twiki/bin/view/Main/TWikiGroups?rev=1.2 HTTP/1.1	1						

Learning Outcomes

I spent some time first familiarizing myself with the log files of my company's web servers. I found out the pattern the server uses and built a regular expression that would capture and group the log entries using online tools like regexr or regexpal.

I then started coding how the workbook would parse the file. First I hardcoded into a module a sub procedure that would parse a single line. It would attempt to match the string with the regular expression. Once that was working I took a step up and using the Excel's file input reader I began programming the parseFile sub procedure. I quickly ran into an issue with the newline character. Apache is usually on a linux machine which only uses the line feed (LF) character and not both line feed and carriage return characters. Excel's defined way to read a line in a file checks for both characters, so when I initially tried this excel ran out of memory. So I had to write my own function that returned the next line.

After that I programmed how to iterate through files and call the parse file method. I then began working on the userform that will allow users to input different variables.

I soon realized that even though Excel was able to parse through these log files Excel takes way too long. I attempted to parse through six log files and it took about 30 minutes to parse through each one. So unfortunately I needed to program a similar parsing algorithm in Java for my assignment at my work, which by the way parsed through the six files in less than one minute.