Word Count Application

# How to run

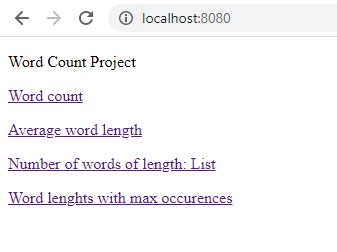
If pulling from github, build jars using mvn clean install. If using zip, jars should be pre-built.

To start up the service, use either “runWordCount.bat” to run in cmd or “runWordCountWithBrowser.bat” to open in default browser automatically to the homepage. Both are provided in project root directory.

# Overview

Basic word counting application built using maven and a spring-boot starter project.

Application is hosted at <http://localhost:8080/>.



Navigation of the page links will call each of the functions with no parameters, resulting in the file “filesToRead/target.txt” being read by default.

# GET Requests

I have split the functionality required in the task into 4 GET requests.

* wordcount
  + Provides total word count
* averagelength
  + Provides average length of words contained in text
* wordlengths
  + Provides a list of word lengths and occurrences of these lengths
* wordlengthmax
  + Provides the maximum occurrences of word lengths and a list of these lengths

It was unspecified in the task what format the responses should take, so I format results from each function into String content to match what was given as expected output in the brief:

*Word count = X*

*Average word length = X*

*Number of words of length X is Y*

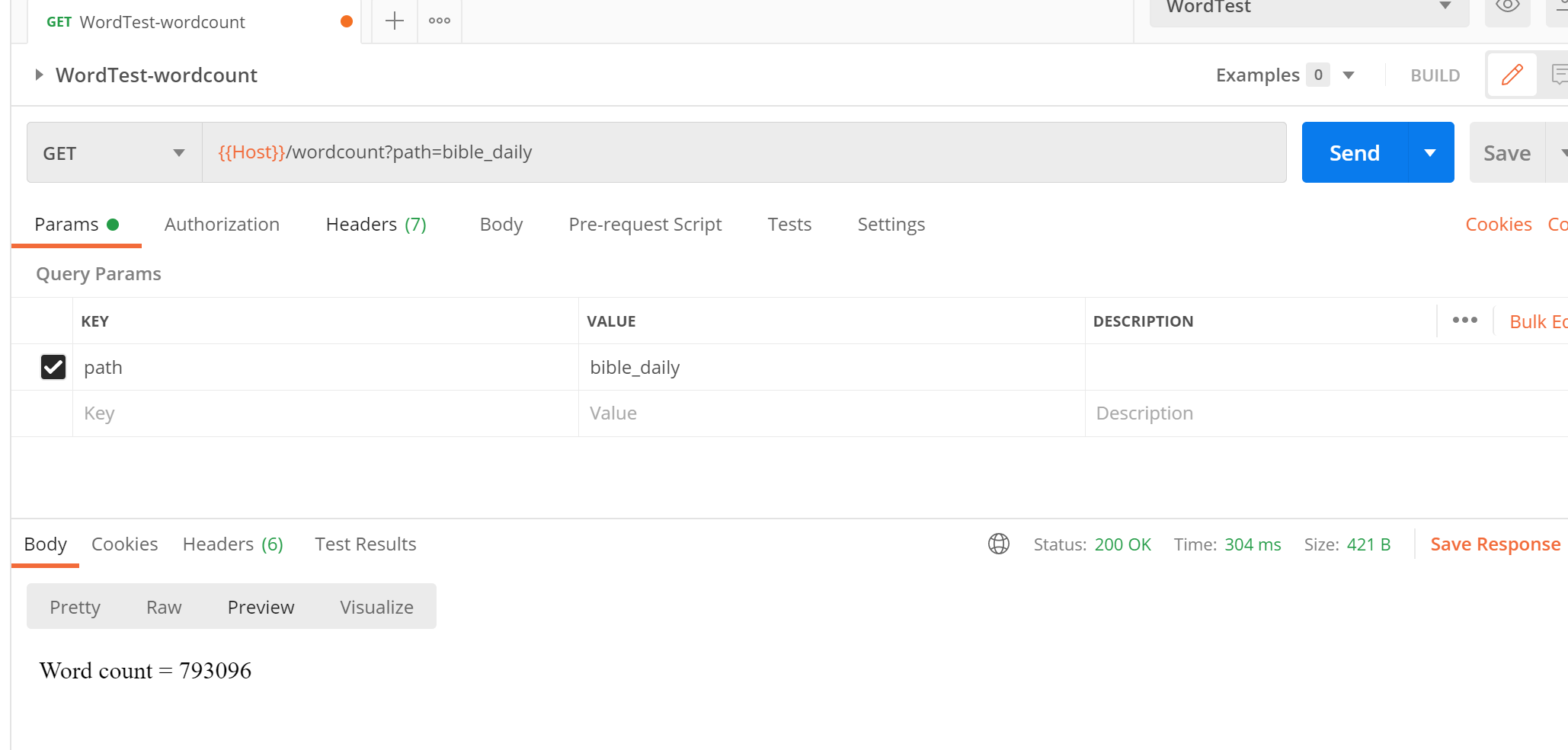
*The most frequently occurring word length is X, for word lengths of Y & Z*

If required, this could easily be modified to return raw data instead (in the form of int, double, etc.) for use by other applications.

# Path param

Each of the requests also accept an optional “path” parameter. This allows for other text files contained within the “filesToRead” directory to be analysed. Param entered should be name of the text file **without** “.txt” extension included.

Below is an example of me running the wordcount call for bible\_daily.txt using postman.



# Assumptions made about what defines a word

For the purposes of this application, I define a word as any sequence of characters separated by whitespace at either end. The sequence of characters does not necessarily need to be alphabetical, (eg. 111 would count as a word). From the brief the following information was given regarding the following input:

*Hello world & good morning. The date is 18/05/2016*

* Word count is 9
* Average word count is 4.556
* Number of words of length 7 is 1

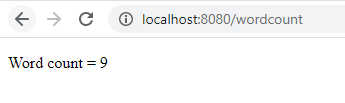
This information is interesting, since it demonstrates that some characters are to be counted as a word/part of a word (in this case the ‘&’ and the ‘/’ within the date). This meant I could not simply use ‘\\p{Punct}’ to strip out all punctuation. However, other punctuation is to be stripped out (such as ‘.’), otherwise ‘morning.’ would be counted as a word of length 8, not 7. I opted to use the following ruleset when performing the analysis:

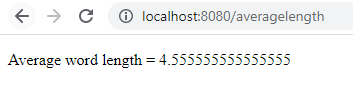
*Chars to be stripped out from text:*

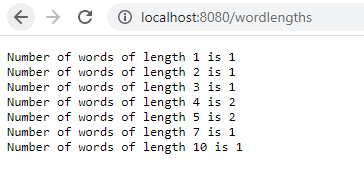
* *.*
* *,*
* *:*
* *;*
* *\**

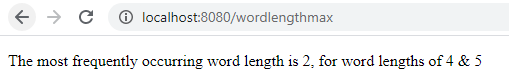
*Other chars present in text will be counted as part of a word or as a separate word if separated on either side via whitespace.*

The unit test “expectedOutputTest” demonstrates that this ruleset produces output that matches the brief. This is presented in web browser as seen below:









# Future enhancements

I stopped development after having reached this point since I believe I have met all of the basic requirements as outlined in the brief.

If I were to spend more time on this to make enhancements, I would likely do the following:

* Implementation of a file upload system (use file browser to select file for analysis).
* Implementation of a dropdown config selection, allowing for a user to select different definitions as to what defines a word and perform different analysis accordingly (selection of differing REGEX for example).
* Enhance the front end presentation.

# Thank you!

Thanks for taking the time to review my app!

Kind Regards,

Nick