Search Engine Project

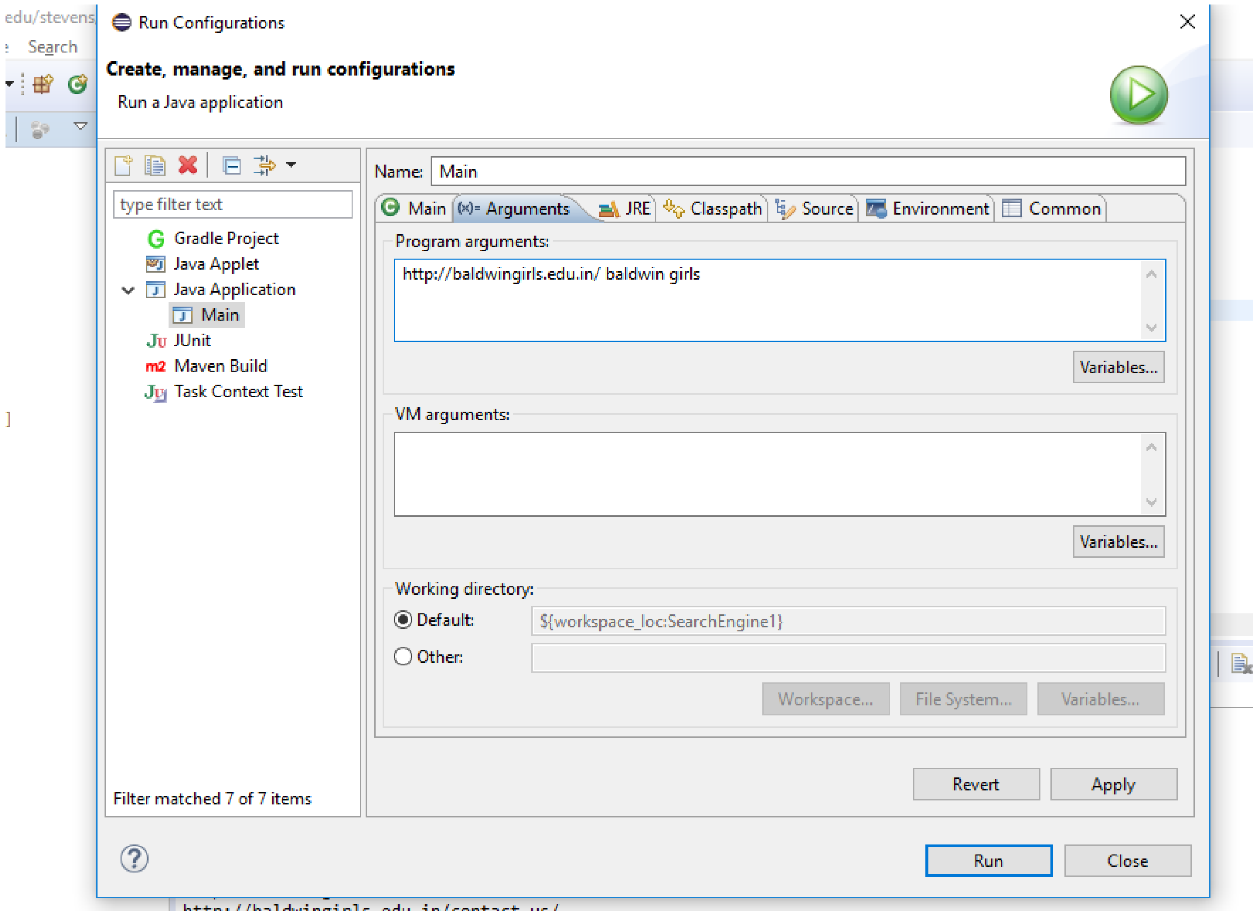
Dhavala Manjunatha

CS600

In this project, I have implemented a web crawler and search engine.

Import details:

1. An url and search words are passed to the program. If there is no url and search word passed, the program assumes values for these and proceeds
2. Import the java program to eclipse running java 8
3. Add the JSoup library if not already present
4. Pass the arguments to the program: 1st argument should be the url and next arguments should be the search words separated by space



Name: Main

Program arguments:

```
http://baldwingirls.edu.in/ baldwin girls
```

VM arguments:

Working directory:

Default: ${workspace_loc:SearchEngine1}

Other:

The approach is described in the steps below:

A. WebCrawler

1. The web crawler works by visiting each link, finding the links in that page and processing them.

2. It maintains two lists, ***toVisit*** and ***visited***. ***toVisit*** contains all the new links found that can be visited. ***visited*** contains all the links visited so far.
3. To ensure the same pages are not visited again and again resulting in an infinite loop, before adding the page to ***toVisit*** list, we check if the page has been already visited or already present in the list.
4. The crawler finds a maximum of 5 relevant links. This number can be changed by updating the value ***LIMIT*** in WebCrawler class.

B. JSoupParser

1. JSoupParser contains the logic to parse the documents
2. The documents are parsed using the JSoup and the text for each document is fetched
3. The text is split into words using regular expressions and stop words are removed by doing a binary search for each word against the stop words
4. The words are also sorted in alphabetical order

C. Trie Structure

1. Each word retrieved for each link is inserted into the trie structure
2. The trie structure contains a root node and subsequent children
3. Each node contains a character, it's children, a flag to say if it's a leaf node. If it's a leaf node, it contains a hashmap of urls that correspond to that word. The HashMap has the url as key and the number of times this word occurs for the url as value.
4. Each time a word is found for an url, if the url doesn't exist in the map, we add the url with value of 1, if the url already exists, we increment the value by 1.
5. This is the ranking logic used in the program. The more of number of occurrences the word has in the url, the more relevant it is.
6. To search for a word the Trie structure first check if the word exists, if it does, it fetches the urls associated with the word and creates a reverse index with the word as key and associated url list as value. This list is in order of relevance
7. Such a search is done for each word and then an intersection of these lists is done to find the links that contain all the words in order of relevance.