HTML PARSER

Software Requirement Specification

Version 1.0

Prepared by:

Bibrat Ranjan Pradhan

Rahul Patra

Ronak Thakkar

Sanjay Moharana

Department of

Computer Science and Engineering

IGIT, Sarang

DECLARATION

It is hereby declared that this report is an authentic record of my work carried out under the supervision of Dr. (Prof) Sarojananda Mishra, **DEPT. OF CSEA, IGIT, SARANG** and Mr. Trilochan Kar. I have not submitted this report elsewhere for any other degree or diploma.

Bibrat Ranjan Pradhan

Rahul Patra

Ronak Thakkar

Sanjay Moharana

Department of CSEA

IGIT, SARANG

Signature of Guide

Dr. (Prof) Sarojananda Mishra

Signature of H.O.D

Dr. (Prof) Sarojananda Mishra

Table of Contents

1. Introduction

1.1 Purpose

This is a part of a bigger project. The project aims to build a search engine. In this part, the objective is to build an HTML parser. This HTML parser will parse out the relevant information corresponding to an input search string from a large set of HTML content.

1.2 Intended Audience And Reading Suggestions

The rest of this SRS is organized as follows: Section 2 gives an overall description of the software perspective with different user characteristics and constraints that are required beforehand to use the "HTML Parser". Section 3 gives specific requirements, which the software is expected to deliver. Functional requirements are given in section 3.1. Some performance requirements and design constraints are given in Section 3.3. Section 4 gives some possible future extensions to the project.

1.3 Product Scope

The HTML parser is provided a folder of HTML content. This HTML content is parsed by the parser. The task is to extract the relevant content pertaining to a particular search string in all of the content. This extracted information is stored in a flat file, which is a kind of document database. As per the need of the user more HTML content can be added for parsing.

1.4 References

- https://docs.oracle.com/javase/tutorial/essential/concurrency/pools.html
- http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ThreadPoolExecutor.htm
- http://www.ibm.com/developerworks/library/j-jtp0730.html
- http://jsoup.org/cookbook.html
- Jsoup jar file (version 1.8.3)

2. Overall Description

2.1 Product Perspective

This is a module of a bigger project, which is to build a search engine. The stress here is to improve upon the parsing technique.

2.2 User Characteristics

The role of the user here is to only give a search string. That search string will be used in parsing.

2.3 Operating Environment

The application can run in any system implementing Java Runtime Environment.

2.4 Design And Implementation Constraints

- Only a fixed number of threads can be created due to memory constraints.
- The Threads created here that have not been used for sixty seconds are terminated and removed from the cache automatically.
- Here, a document database is used in the form a flat file.
- The number of files that can be in a folder must be known.

2.5 Assumption And Dependencies

The application is thread dependent. Efficiency of the program depends on the number of threads. The selected HTML folder must contain the HTML files.

3. External Interface Requirements

3.1 User Interfaces

The application will have a welcome UI screen.

The UI is created using swing in java. The different components are:

- **Textfield:** This component is used to take the input search string.
- **Button:** When this button is clicked, the parsing starts.
- **Textarea:** This component is used to display the results.

3.2 Hardware interfaces

The HTML parser doesn't use any special hardware interface for its functioning.

3.3 Software Interfaces

JSOUP API

JSOUP implements the HTML5 specification, and parses HTML to the same DOM as modern browsers do. Its different roles are:

- Scrape and parse HTML from a URL, file, or string
- Find and extract data, using DOM traversal or CSS selectors
- Manipulate the HTML elements, attributes, and text
- Clean user-submitted content against a safe white-list, to prevent XSS attacks

- Output tidy HTML

JSOUP is designed to deal with all varieties of HTML found in the wild; from pristine and validating, to invalid tag-soup; JSOUP will create a sensible parse tree.

FLAT FILE

Advantages of Not Using a Database:

Speed — Without needing to connect to a database to retrieve content, flat-file sites load pages blazingly fast, especially with solid-state drive hosting (referral link)

Security — Databases are often the first thing hackers target and without a database that potential security hole will be completely removed

Simplicity — No need to configure or maintain a database — "installation" is just a matter of uploading files

Version control — Everything is just files and folders so it's easy to version control absolutely everything

Portability — A site can quickly be moved to another server without needing to export and configure a database

JAVA – ECLIPSE

The combo of java programming language and Eclipse editor is used to develop the application.

3.4 Communication Interfaces

The current aim is to develop the application for offline use only. Hence there are no communication interfaces as such.

4 System Features

Functional Requirements

The job of the parser is to extract the relevant information for a given search string. So the only input that is given by the user is the search string. There is also a folder of HTML files attached along with the application from where the search will be carried out. The relevant information is written onto a flat file against the corresponding search string and the URL from where the content is found. This HTML parser functions broadly over the below mentioned classes:

- **SimpleThreadPool class:** This class creates a thread pool and also a fixed number of worker threads.
- **WorkerThread class:** This class defines all the tasks that a worker thread needs to perform.
- **FileQ class:** This class puts all the HTML files into an infinite queue.
- **FileRead class:** This class's function is to parse the content from an HTML file for the given search string.
- **CheckFile class:** This class checks for data duplicity in the flat file.
- **FileWrite class:** This class writes the data onto the flat file.

5 Other Nonfunctional Requirements

5.1 User Interface

Multithreading:

The performance of the HTML parser is mainly dependent on multi-threading. But, there has to be a threshold limit to the number of threads used. Because, although more number of threads would mean higher efficiency but that would also result in excessive use of system resources. Hence, to get an optimum performance we must adhere to the threshold limit.

In the pool thread concept used here, the pool threads have a die-out time of 60 seconds. The Threads created here that have not been used for sixty seconds are terminated and removed from the cache automatically.

Parsing:

Information can lie in between various HTML tags. The HTML tags which are most likely to contain information have been used here. But there are many, where the relevant content might be found.

Flat File:

Due to the various advantages over relational database, it leads to performance improvements. Using flat file requires less access time for it and file handling operation can be done more efficiently.

5.2 Safety Requirements

Only a fixed number of threads acts as a pool threads which is used repeatedly to parse all the given HTML Files. Excessive number of threads will lead to malfunctioning of the system, hence the system will result in hanging up mode.

5.3 Security Requirements

The flat file will be encrypted using Advanced Encryption Standard (AES). This will avoid any unnecessary access to the file. It will also be required decryption of the flat file for proper understanding i.e. same algorithm is used to decrypt the flat file.

For preventing any unauthorized access to the flat file we need to develop proper authentication process such that the required client can be given that particular file with proper encryption in order to avoid it from other clients.

6 ***Future Extensions***

This project “HTML Parser” can be extended with some more functionalities in order to increase its utility.

Firstly it can be provided with automated category search which will predict the category from the search string provided by the user. Hence it will contribute to make parsing more specific and relevant information from the previous scenario.

Secondly meta-tags will be used which will contain all the important keywords related to the html content and will help in making the search efficient.

Thirdly more efficient multithreading module must be implemented that involves dynamic thread creation and decrease in time complexity.

Fourth all the related searches regarding the synonyms of the search string will also get executed along with the actual search string. This to give proper complete information regarding the search string.

Appendix A: Glossary

Term	Definition
Parsing	To split a file or other input into pieces of data that can be easily stored or manipulated.
Multi-Threading	Uses of threads that handle parallel processing of program.
Flat file	Database schema where data is stored in a text file.
Pool Thread	Threads responsible for parsing HTML files.
Worker Thread	Thread that activates pool threads.
JSOUP	Java library containing predefined parse methods.
Java Eclipse	Integrated Development Environment where the program is developed.

Appendix B: Analysis Models

Use Case Diagram