ULTIMATE SEARCH ENGINE

A Project

Presented to the faculty of the Department of Computer Science
California State University, Sacramento

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

by

Chirag Patel

SPRING 2012
ULTIMATE SEARCH ENGINE

A Project

by

Chirag Patel

Approved by:

__________________________________, Committee Chair
Martin Nicholes, Ph.D.

__________________________________, Second Reader
Meiliu Lu, Ph.D.

________________________________________
Date
Student: Chirag Patel

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the Project.

________________________, Graduate Coordinator
Nikrous Faroughi, Ph.D.

Department of Computer Science
Abstract

of

ULTIMATE SEARCH ENGINE

by

Chirag Patel.

The search engine is a tool designed to search for information on the web according to the keywords specified by users. Different search engines are being accessed by most of the people accessing the web in the modern world. To retrieve the best results, many times the user accesses different search engines, because every search engine uses different logic to retrieve information from its own database repository. During this process, the user encounters repetition in the search results and irrelevant search results. It takes much time and effort for the user, especially in technical, research, literature, science, education, etc., fields.

Ultimate Search Engine provides the functionality to manage search results from different search engines in one place with the flexibility of run time search engine selection. Ultimate Search Engine provides a unique result set of different search engines with load balancing on the web.

_______________________, Committee Chair
Martin Nicholes, Ph.D.

_______________________
Date
ACKNOWLEDGMENTS

I would like to take this opportunity to remember and acknowledge the guidance, cooperation, goodwill and both moral and technical support, extended by all staff and faculty members of California State University, Sacramento.

I am highly indebted to my project advisor, Dr. Martin Nicholes for his guidance and constant supervision as well as for providing necessary information regarding the project and also for his support in completing the project.

I am also grateful to my second reader, Dr. Meiliu Lu for being a second reader and providing me great help when needed during the project. She has done great help in giving important advice and proof reading the documents.

I am also grateful to Dr. Nikrous Faroughi for helping me during the completion of my project. He has shown the path during preparation of the project and provided great ease during completion of the project.

Finally, I would like to express my gratitude towards my parents, wife and friends for their kind co-operation and encouragement, which helped me in the completion of my Masters project.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledgments</td>
<td>v</td>
</tr>
<tr>
<td>List of Figures</td>
<td>viii</td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2. APPLICATION OVERVIEW</td>
<td>7</td>
</tr>
<tr>
<td>2.1 Features</td>
<td>7</td>
</tr>
<tr>
<td>2.2 Design</td>
<td>8</td>
</tr>
<tr>
<td>3. ARCHITECTURE</td>
<td>11</td>
</tr>
<tr>
<td>3.1 J2EE</td>
<td>12</td>
</tr>
<tr>
<td>3.2 Eclipse</td>
<td>14</td>
</tr>
<tr>
<td>3.3 Apache Tomcat Application Server</td>
<td>14</td>
</tr>
<tr>
<td>3.4 Apache Tomcat HTTP Server</td>
<td>14</td>
</tr>
<tr>
<td>3.5 Java Script</td>
<td>15</td>
</tr>
<tr>
<td>3.6 AJAX</td>
<td>15</td>
</tr>
<tr>
<td>3.7 XML</td>
<td>16</td>
</tr>
<tr>
<td>3.8 API</td>
<td>16</td>
</tr>
<tr>
<td>3.9 Cookies</td>
<td>17</td>
</tr>
<tr>
<td>3.9.1 useresultsettings</td>
<td>18</td>
</tr>
<tr>
<td>3.9.2 recentkeyword</td>
<td>18</td>
</tr>
<tr>
<td>3.9.3 userid</td>
<td>18</td>
</tr>
<tr>
<td>4. IMPLEMENTATION</td>
<td>19</td>
</tr>
<tr>
<td>4.1 Implementation Detail with Dataflow</td>
<td>19</td>
</tr>
<tr>
<td>5. EXECUTION AND SCREEN LAYOUT</td>
<td>25</td>
</tr>
<tr>
<td>5.1 Home Page</td>
<td>25</td>
</tr>
<tr>
<td>5.2 All Results Selected</td>
<td>26</td>
</tr>
<tr>
<td>5.3 Unique Results Selected</td>
<td>31</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>5.4</td>
<td>Keep History</td>
</tr>
<tr>
<td>5.5</td>
<td>Recent Keywords</td>
</tr>
<tr>
<td>5.6</td>
<td>Load Balancing</td>
</tr>
<tr>
<td>6.</td>
<td>DESIGN AND ARCHITECTURE DECISIONS</td>
</tr>
<tr>
<td>6.1</td>
<td>Database</td>
</tr>
<tr>
<td>6.2</td>
<td>Cookies and XML</td>
</tr>
<tr>
<td>6.3</td>
<td>Application Server</td>
</tr>
<tr>
<td>7.</td>
<td>RELATED WORK</td>
</tr>
<tr>
<td>7.1</td>
<td>Dogpile</td>
</tr>
<tr>
<td>7.2</td>
<td>Noobsearch</td>
</tr>
<tr>
<td>7.3</td>
<td>Metacrawler</td>
</tr>
<tr>
<td>7.4</td>
<td>Ixquick</td>
</tr>
<tr>
<td>8.</td>
<td>RELATED WORK</td>
</tr>
<tr>
<td>Appendix A.</td>
<td>Prerequisites</td>
</tr>
<tr>
<td>Appendix B.</td>
<td>Definitions</td>
</tr>
<tr>
<td>Appendix C.</td>
<td>API Description</td>
</tr>
<tr>
<td>Appendix D.</td>
<td>Configurations</td>
</tr>
<tr>
<td>References</td>
<td></td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figures</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Figure 1.1 Basic Architecture of standard Web Crawler</td>
<td>2</td>
</tr>
<tr>
<td>2. Figure 3.1 Basic Architecture of Ultimate Search Engine</td>
<td>11</td>
</tr>
<tr>
<td>3. Figure 3.2 Basic Architecture of J2EE [5]</td>
<td>13</td>
</tr>
<tr>
<td>4. Figure 4.1 Data flow diagram of Ultimate Search Engine</td>
<td>20</td>
</tr>
<tr>
<td>5. Figure 5.1 Ultimate Search Engine Home Page</td>
<td>25</td>
</tr>
<tr>
<td>6. Figure 5.2 Home page with “ALL” option</td>
<td>26</td>
</tr>
<tr>
<td>7. Figure 5.3 Search results for “ALL” option – Upper part of the page</td>
<td>27</td>
</tr>
<tr>
<td>8. Figure 5.4 Search results for “ALL” option – Lower part of the page</td>
<td>28</td>
</tr>
<tr>
<td>9. Figure 5.5 Search results for “ALL” option – Digg tab</td>
<td>29</td>
</tr>
<tr>
<td>10. Figure 5.6 Search results for “ALL” option with Bing, AOL Video selected – Bing tab</td>
<td>30</td>
</tr>
<tr>
<td>11. Figure 5.7 Search results for “ALL” option with Bing, AOL Video selected – AOL Video tab</td>
<td>31</td>
</tr>
<tr>
<td>12. Figure 5.8 Home page with “ALL” check box unchecked</td>
<td>32</td>
</tr>
<tr>
<td>13. Figure 5.9 Unique search results with all search engines selected</td>
<td>33</td>
</tr>
<tr>
<td>14. Figure 5.10 Unique search results with Bing, AOL Video and Digg Selected</td>
<td>34</td>
</tr>
<tr>
<td>15. Figure 5.11 Number of times URL hit by the user</td>
<td>35</td>
</tr>
<tr>
<td>16. Figure 5.12 Recent Keywords</td>
<td>36</td>
</tr>
<tr>
<td>17. Figure 5.13 Application runs on Tomcat C</td>
<td>37</td>
</tr>
<tr>
<td>18. Figure 5.14 Application runs on Tomcat A</td>
<td>38</td>
</tr>
<tr>
<td>19. Figure 5.15 Application runs on Tomcat B</td>
<td>39</td>
</tr>
</tbody>
</table>
Chapter 1

INTRODUCTION

Referring to the famous journalist Ambrose Bierce, “There is nothing new under the sun but there are lots of old things we don’t know.” In the modern world, information is an extremely precious thing as everybody needs it at some point in everyday life. The next question is how to get information. There are many sources where people can get information, such as books, people interaction, television, radio, self-research, Internet, etc. The Internet is widely used as a medium to get information in today’s era. There is a crucial question of how to get information in a precise and timely manner from the Internet. As a solution, web search engines have been created to provide desired information from the Internet. There are many web search engines available today which provide information based on different algorithms, pattern matching techniques, database types and technologies. Ultimate Search Engine gathers information from different web search engines available in the market, to provide information in a more flexible way.

A web search engine is designed to search information on World Wide Web and FTP servers. The search results are generally presented with a list of results often referred to as SERPS, or "search engine results pages." The information may consist of web pages, images, information and other types of files. Some search engines also mine data available in databases or open directories. Unlike web directories, which are maintained only by human editors, the search engines also maintain real-time information by running an algorithm on a web crawler.

A search engine operates in the following order:
1. Web crawling
2. Indexing

Web search engines work by storing the information about many web pages, which they retrieve from HTML files. These pages are retrieved by a Web crawler (sometimes also known as a spider) — an automated Web browser which follows every link on the site. Exclusions can be made by the use of a robots.txt file. Then the content of each page is analyzed to determine how it should be indexed (for example, words are extracted from the titles, headings, or special fields called Meta tags). Figure 1.1 shows the basic architecture of the standard web crawler. [1]
According to the searched keyword, the Scheduler finds relative metadata in the storage and assigns pages to several Downloader modules. The Downloader modules transfer pages through the network, parse their contents, extract new links and maintain the link structure. There are two main data structures for the link storage, web page (text) storage and a URL queue. The search engine uses these data structures to get the results faster.

The Data about web pages is stored in an index database, for use in later queries. A query can be a single word. The purpose of an index is to allow information to be found as quickly as possible. Some search engines, such as Google, store all or part of the source page (referred to as a cache) as well as the information about the web pages, whereas others, such as AltaVista, store every word of every page they find. These cached pages always contain the actual search text since it is the one that was actually indexed. It is very useful to get results fast, when the user searches the same keywords again. When the content of the page is updated and the searched keyword is no longer found in the page, then the cached pages become irrelevant. The large number of irrelevant cached pages decreases the performance of index searching. This problem might be considered to be a mild form of linkrot (informal term for the process by which, either on individual websites or the Internet in general, increasing numbers of links pointing to web pages, servers or other resources have become permanently unavailable) [2], and Google's handling of web information increases usability by satisfying the user's expectations that the search terms will be on the returned web page. This satisfies the principle of least astonishment (when two elements of an interface conflict, or are
ambiguous, then a programmer should try to think of a behavior that will least surprise someone who uses the program) [3], since the user normally expects the search terms to be on the returned pages. When the search is performed on a keyword that exists in the cached pages, the search becomes much faster. During this time, the cached pages are very helpful.

When a user enters a query into a search engine (typically by using keywords), the engine examines its index and provides a listing of best-matching web pages according to its criterion, usually with a short summary containing the document's title and sometimes parts of the text. The index is built from the information stored with the data. Most of the search engines build a search query, which includes keywords with words, numbers, figures, etc. But unfortunately, currently there is no known public search engine that allows documents to be searched by date. Most of the search engines support the use of the Boolean operators AND, OR and NOT to further specify a search query. Boolean operators are for literal searches that allow the user to refine and extend the terms of the search. The engine looks for the words or phrases exactly as entered. Some search engines provide an advanced feature called proximity search, which allows the users to define the distance between keywords. There is also concept-based searching, where the search involves using statistical analysis on pages containing the words or phrases you search for. Also, natural language queries allow the user to type a question in the same manner one would ask a human.

The usefulness of a search engine depends on the relevance of the result set it returns. While there may be millions of web pages that include a particular word or
phrase, some pages may be more relevant, popular, or authoritative than others. Most of
the search engines employ methods to rank the results, which provides the "best" results
first. How a search engine decides which pages are the best matches, and what order the
results should be shown in, varies widely from one engine to another. The methods also
change over time as Internet usage changes and new techniques evolve. This is why,
combining results from several search engines is so important. Ultimate Search Engine
incorporates the results from different search engines, to provide the most relevant results
to the keyword by taking advantage of different search engine rankings.

There are two main types of the search engines that have evolved. One is a system
of predefined and hierarchically ordered keywords that humans have programmed
extensively. The other is a system that generates an inverted index (an index data
structure storing a mapping from content, such as words or numbers, to its locations in a
database file, or in a document or a set of documents) by analyzing texts it locates. This
second form relies much more heavily on the computer to do the bulk of the work.

Currently, there are many search engines available in the market. Some of the
most popular search engines are Google, Yahoo, AOL, Bing, Ask, etc. These search
engines use different algorithms and provide different result sets for the same keyword. It
is very important in today’s world to get the best information in the minimum amount of
time. In order to get the right information, the user may have to search the information
using different web search engines because there might be a case where the user cannot
get the desired information from a specific search engine.
Currently, we have some search engine tools available in the market like ixquick, dogpile, allplus, etc., that provide the combined search results from the popular search engines. But, these search engine tools have statically predefined logic to search keywords in specific search engines only. They do not provide the flexibility to use the logic from different search engines at the same time. They do not provide combined results from different search engines. They show how many search engines returned a result but do not provide options for the users to select their own ranking without duplication.

To overcome all these limitations, gather all the information in one place and manage it with dynamic ranking, Ultimate Search Engine provides flexibility to fetch information from the web in many ways. The Ultimate Search Engine provides the functionality to combine web search results from different web search engines like Google, Bing, Entire Web, AOL Video and Digg. Ultimate Search Engine provides a way to manipulate the information retrieved from different search engines in one place and show the unique results from different search engines. The purpose of the Ultimate Search Engine is to get desired information from the web using several search engines according to the user’s preference.
Chapter 2
APPLICATION OVERVIEW

The goal of this project is to show web search results from different search engines and manipulate them according to the user’s preference. Ultimate Search Engine uses different search engines available in the market that provide free APIs in order to create a working prototype. The emphasis is placed on performance, as this is a real time application that is used by people to get desired information quickly from the web. The following section reviews the features provided by Ultimate Search Engine.

2.1 Features

Ultimate Search Engine provides the following features

- Search keywords in selected search engines and show the results in one place.
- Show unique results for selected search engines on one page according to the priority given to different search engines and the number of results selected for specific search engines. For example, if the search engine priority will be Google, AOL Video, Digg, Bing and Entire Web and the number of results per search engine will be 5, 2, 10, 15 and 7 respectively, Ultimate Search Engine gets the first 5 results from Google, the next 2 results from AOL Video, the next 10 results from Digg, the next 15 results from Bing and next 7 results from Entire Web. All the results are unique. The top ranked results are fetched from the first priority search engine. The top ranked results fetched from the second priority search engine are different than the results of the first priority search engine. The same principle applies to the results fetched from the lower priority search engines. In
the end, the top ranked results from different search engines are fetched without any duplication.

- Show all the results from selected search engines on one web page in different tabs. If the user selects Google and Digg, two tabs will be created for Google and Digg. Those two tabs contain all search results from Google and Digg respectively.

- Show how many times a specific URL has been clicked by the user.

- Save current settings on the web page for the selected search engines, results per page, priority, and save history options.

- Show 10 most recently searched keywords.

- Perform load balancing using a Vertical cluster (running multiple server instances on the same hardware to better utilize the memory or threading and allow for protection against software failures).

2.2 Design

Ultimate Search Engine is a web based application, which runs on an application server and is accessed through HTTP. Emphasis is placed on dynamic results fetching from different search engines that are selected in the application at run time according to the user’s preference. Load balancing is implemented using a Vertical cluster.

All the features are shown on one page for ease of use. The major purpose of this application is to provide a user friendly interface that displays all necessary information in one place. The design of the application is made robust by using Java Script, Ajax, JQuery and CSS, as these technologies are executed on the client side. The number of
request transfers between the client and the server is reduced. Request processing on the server is always costly and these scripting technologies reduce server processing by processing major requests on the client side. When the user submits a request to search keywords, AJAX handles the request through Java Script and the requests are fulfilled without reloading HTTP content. To make a web page interactive with objects, Java Script DOM (Document Object Model) is used widely in the application.

Ultimate Search Engine has two panes on the web page. The left pane provides various settings, and the right pane provides functionality to search keywords and show results from various search engines.

Left Pane:

- The “ALL” check box provides functionality to show all the results from selected search engines in the application. If the check box is checked, the option to select the search engines become active, and all the results from selected search engines are shown. If the “ALL” check box is not checked, the option to select the search engines become active, in addition to an option to specify the number of unique results per search engine using a sliding bar. The selection of individual search engines is implemented using a checkbox. The Results are shown from selected search engines according to the number of results requested per search engine and all the results are unique across all selected search engines.

- The “Keep History” option provides the functionality to track how many times a specific URL has been clicked by the user. If the “Keep History” option is checked, Ultimate Search Engine keeps track of the number of clicks per URL.
• The “Change Priority” option allows a user to change the priority of the search engines in order to generate unique results. According to the priority set in this section, Ultimate Search Engine fetches unique results from the search engines according to the number of results specified per search engine. The users need to drag a specific search engine in the section, and drop it in the desired location on the list to set its’ priority. The first search engine in the list has the first priority. If a user wants to raise to first priority the third priority search engine, the third priority search engine should be dragged to the first location in the section and it then becomes the first priority.

• The last section shows the history of recent keywords. It shows the history of the 10 latest keywords searched. If the user clicks on any history keyword in this section, the application will generate results for this keyword. The user does not need to copy or rewrite a keyword again to perform the search.

Right Pane:

• This pane has a search text box to enter the keywords to search on the web.

• The search button is clicked when the user wants to generate results for the keywords entered in the text box. Alternatively, the user can hit the “Enter” key on a keyboard.

• All results are shown in this pane.
Chapter 3

ARCHITECTURE

Ultimate Search Engine is a web based project that uses different technologies and tools to implement various features. All the tools and technologies used in this project are free from any licensing agreement. The basic architecture of Ultimate Search Engine is shown in Figure 3.1.

![Figure 3.1: Basic Architecture of Ultimate Search Engine](image-url)
Ultimate Search Engine is a web base application which is executed on the client side and communicates with a server. The application performs most of the calculation and configuration on the client side using Java Script and AJAX. Communication with the server is accomplished using Apache HTTP server, which sends the request to the Application Servers, Tomcat A, Tomcat B and Tomcat C. A Vertical cluster is implemented on the server side using three Tomcat servers with load balancing, and is managed by the Apache HTTP Server. On the client side, a request is fulfilled by getting the information from cookies. On the server side, a request is fulfilled using the XML file stored in the server file system. The XML file is transferred to the client upon request and an updated XML file is sent from the client to be stored on the Server.

Ultimate Search Engine is implemented according to the following architecture, technologies and tools.

3.1 J2EE

Ultimate Search Engine is based on J2EE architecture, which is one of the most robust, efficient, powerful and popular architectures in the world.

The Java™ 2 Platform, Enterprise Edition (J2EE) provides a standard for developing multitier, enterprise applications. The economy and technology of today have intensified the need for faster, more efficient, and larger-scale information management solutions. The J2EE specification satisfies these challenges by providing a programming model that improves the development productivity, standardizes the platform for hosting the enterprise applications, and ensures portability of the developed applications with an extensive test suite. [4]
The J2EE architecture supports the component-based development of multi-tier enterprise applications. Figure 3.2 shows the basic architecture of J2EE.

**Figure 2.2: Basic Architecture of J2EE [5]**

The J2EE architecture isolates client tier, presentation tier, business tier and Enterprise Information System (EIS) tier. Each tier works independently. The client machine works as a client tier. The Java Server Pages (JSP), Servlet, Hyper Text Markup Language (HTML), etc., pages are included in the Presentation layer. The Enterprise Java Beans (EJB), Java Naming and Directory Interface (JNDI), Web Services, application objects, connector, etc., are included in the business tier. The database, report server, load balancing server, Geographic Information System Server (like ArcGIS made by ESRI), etc., are included in the EIS tier. All the tiers communicate with each other in order to fulfill the client request.
3.2 Eclipse

Ultimate Search engine is developed in Eclipse with JDK 1.6 which is an open source tool for project development.

Eclipse is a multi-language software development environment comprised of an integrated development environment (IDE) and an extensible plug-in system. It is written mostly in Java and can be used to develop applications in Java and other programming languages. [6]

3.3 Apache Tomcat Application Server

Ultimate Search Engine uses the Apache Tomcat application server with Eclipse for development. The Web Application Archive (WAR), a Java archive file that contains the web module was created for the project using Eclipse, and the new application were created on the Apache Application server by deploying the project WAR file.

Apache Tomcat is an open source servlet container developed by the Apache Software Foundation (ASF). Tomcat implements the Java Servlet, and the Java Server Pages (JSP) specifications from Oracle Corporation, and provides a "pure Java" HTTP web server environment for the Java code to run. [7]

Three instances of Tomcat server are created in order to perform load balancing that are named Tomcat A, Tomcat B and Tomcat C. Three associated Windows services are created to start and stop the Tomcat servers.

3.4 Apache Tomcat HTTP Server

Ultimate Search Engine provides load balancing through a Vertical cluster. This project runs on a local intranet. To implement the Vertical cluster, Apache Tomcat HTTP
server uses the three instances of the Apache Tomcat application server. The Mod_jk Tomcat connector is used to provide communication between the Apache HTTP Server and the three instances of the Tomcat application server.

3.5 Java Script

Ultimate Search Engine uses Java Script to a great extent, as this project performs most of the calculation on the client side and Java Script executes on the Client machine. Emphasis is placed on the performance of this project. The use of Java Script improves the performance of an application and saves time by reducing the number of requests exchanged between the client and the server. Thus, the client machine gets prompt response from user input.

Java Script is a scripting language that is based on a prototype. Java Script is dynamic and weakly typed. It supports object oriented, imperative and functional programming styles. Java Script executes on the client side and sends the request to the server if a major calculation is done or the calculation is based on other resources such as different object structure, database, libraries, etc.

3.6 AJAX

Ultimate Search Engine is heavily based on AJAX technology, which shows dynamic content at run time on the web page. Ajax sends and receives the data from the server asynchronously and does not change the existing display of the page. Ajax uses XmlHttpRequest object to retrieve the data. Ajax is not a single technology, but a group of related technologies. HTML and CSS can be used to mark up and style the information. The DOM is accessed with JavaScript to dynamically display, and to allow
the user to interact with the information presented. The JavaScript and the XMLHttpRequest object provide a method for exchanging the data asynchronously between the browser and the server to avoid full page reloads. [8]

3.7 XML

An XML file is used for web page tracking in the project. When the user clicks any link, a new XML file is created on the server side which stores the URL, current date and total hits.

For example,

```xml
<data>
  <userid>30-10-2011-10-36-49</userid>
  <links>
    <link>
      <url>http://maps.google.com/</url>
      <hits>1</hits>
      <visitdate>October 30, 2011 22:37:12</visitdate>
    </link>
    <link>
      <url>http://earth.google.com/</url>
      <hits>1</hits>
      <visitdate>October 30, 2011 22:43:50</visitdate>
    </link>
  </links>
</data>
```

3.8 API

Ultimate Search Engine uses several different search engine APIs for data retrieval from the search engine repository databases. All search engine APIs used in this project are free from a licensing agreement. Other popular search engine APIs like Yahoo, AOL, Gigablast, Ask, etc., are not free. The developer has to pay in order to use the APIs.
Ultimate Search Engine uses APIs for Google, Bing, AOL Video, Entire Web and Digg. It gets the results from the API calls and manipulates the result sets according to the user’s preference.

3.9 Cookies

Ultimate Search Engine uses cookies to store specific information like settings specified by the user, the web page search history including keywords and the search results on the client machine for faster response. The cookies store the data on the client machine, which is very useful during run time for the application. The application does not have to look for basic data on a server or in an external database. Thus, it achieves faster response time. Most of the client requests are fulfilled by getting the required information from cookies stored on the client machine. Thus, it eliminates the need to transfer the request between the client and the server. Every time the user saves or changes settings on a web page, all the settings are saved in cookies on the client machine. A new userid is generated for every client by the application and is stored in the cookie. The userid is used to identify the user specific records in XML files. When the same user makes different requests in different sessions, the application gets the XML files stored on the server and gets user specific information for the last session. The most recent 10 keywords are also stored in the cookie.

The following cookies are created on the client side for the application.
3.9.1 userresultsettings

The cookie named “userresultsettings” is used to store advance settings for the number of results per search engine and history settings. Each setting value is separated by a colon (:).

For Example, 10:10:2:10:10:0

The first “10” denotes that Google will display 10 results.

The second “10” denotes that Bing will display 10 results.

The last “0” denotes the history flag

If “Keep History” option is checked, the value of history flag becomes 1.

3.9.2 recentkeyword

The cookie named “recentkeyword” is used to store most 10 recent searched keywords.

In that cookie, each keyword is separated by a pipe (||).

For Example,

Google||Yahoo||cars||facebook||quicker||travel||top%20search%20engine||test1||usa||csus

3.9.3 userid

The cookie named “userid” is used to store the user id for every user session to maintain a history for the end users.
Chapter 4

IMPLEMENTATION

Eclipse is used to develop the project along with the Apache Tomcat server incorporated in Eclipse. Jdk 1.6 is used in Eclipse as the Java run time environment. Ultimate Search Engine supports different browsers including Internet Explorer, Mozilla Firefox, Google Chrome, etc.

4.1 Implementation Detail with Dataflow
Figure 3.1: Data flow diagram of Ultimate Search Engine
As shown in Figure 4.1, Ultimate Search Engine generates a client request from the client machine. The request is processed on the client machine using Java Script and Ajax. The cookies stored on the client machine provide the necessary information for the client request. For further processing, the client request is sent to the Apache server. The Apache server sends the client request to one instance of Tomcat server. The Tomcat server with the least load of client requests gets the new client request sent by the Apache server. The Tomcat server executes the application and performs the necessary calculations. The XML files stored on the server machine provide the necessary information to the Tomcat server for client request processing. The Tomcat server sends the XML file to the Apache server. The Apache server sends the XML file to the client machine where it is used for client request processing by Java Script and AJAX.

All the user settings on the web page are stored in the cookies named userresultsettings, recentkeyword and userid. When the user accesses the application on the web page, settings for various search engines and other options are read from the cookies stored on the client machine. For first time access, the application loads with the default configuration and new cookies are created. When the user changes the configurations, the values in the cookies are changed. When the user enters any value in the search box and hits “Enter”, the application tries to search the XML file on the server for the specific user id that was created at run time and stored in the cookie. If no XML file is found for that user id, new XML file with the current date and time is created in temporary memory on the client side. Otherwise, the server sends the XML file to the
client and the client stores the XML file to temporary memory and accesses the information stored in the file.

When the user clicks on any search result link, the URL for the link will be set under the cookie for recent keyword. This cookie stores at most 10 results. If a new keyword is entered after 10 keywords, the first keyword in the list is deleted, and the new keyword is inserted. When the user clicks on any link on the search result and if the option “Keep History” is checked, the application tries to search for the URL in the XML file for that user id. If the application finds the userid and URL in the XML file, it increments the counter by 1, or else it creates a new value in the XML file. Every time the user clicks on any link, the XML file is sent to the server and the server stores the updated XML file. The user settings on the web page are updated dynamically using AJAX according to the selections made by the user. When the user enters any keyword and hits “Enter”, APIs will be called according to the selected search engines on the web page. The APIs generate the search results for the keyword entered. The application gets all the results returned by the APIs and processes them in order to remove duplicates. It also shows the number of results per search engine selected by the user.

The unique results are calculated using DOM (Document Object Model) array objects for the selected search engines. First of all, n numbers of array objects are created for each search engine. The search engine APIs are called for the selected search engines and the results from each API are stored in the respective n<sup>th</sup> array object. According to the search engine priority set by the user, the same priority is given to each relative n<sup>th</sup> array object. The combined array object is then created, which stores unique results from
all n array objects. The first priority array object is considered first in the calculation. According to the number of results specified for the first priority search engine, the elements from the first priority array object are inserted into the combined array object. Now, the second priority array object is considered for the calculation. According to the number of results specified for the second priority search engine, the elements from the second priority array object are validated for duplication against the elements already in the combined array object. If duplication is found, then the duplicate elements from the second priority array object are deleted. Now, the specified numbers of elements for the second priority array object are again compared to the elements of the combined array object and duplication is removed if any. This process continues for the second priority array object until the specified number of elements are unique compared to the elements in the combined array object. The same validation process is applied to subsequent priority array objects and the results are inserted into the combined array object from each of the n array objects. In the end, the combined array object contains unique search results. This method is not costly for up to few hundreds results, as practically the user does not want to go through thousands of search results.

The Vertical cluster is implemented on the Apache server using the set of configurations for load balancing. The application is implemented on three Tomcat servers named Tomcat A, Tomcat B and Tomcat C. The application is accessed through the Apache HTTP server which handles load balancing. The Apache HTTP server listens to the application on port 80. Thus, the user does not need to specify a specific port in the URL. The Apache HTTP server implements balance method “B” (balance by busy
factor). When the request comes to the HTTP server, it selects a Tomcat server to be used to service the request. The HTTP server checks the load on every server and transfers the request to the server which has minimum load. When a server is down for any reason, the HTTP server transfers the request to the next server while maintaining the current state of the request. Load balancing on the Apache HTTP server is done by the algorithm defined by the Apache HTTP server. The variations in load balancing can be accomplished by changing the configurations on the HTTP Server.
Chapter 5
EXECUTION AND SCREEN LAYOUT

The workflow of Ultimate Search Engine is displayed in this section, along with screen layouts. When the application is deployed on the Apache HTTP server and the user types http://localhost/ultimatesearch/ in the web browser, the application starts its execution.

5.1 Home Page

When the application starts its execution, the first page will be shown with default settings as shown in Figure 5.1.

![Ultimate Search Engine Home Page](image)

Figure 4.1: Ultimate Search Engine Home Page
5.2 All Results Selected

When the user checks the “ALL” checkbox, the web page changes its current layout and an option to select the search engines appears. All the results for the selected search engines are as shown in Figure 5.2.

![Figure 5.2: Home page with “ALL” option](image)

When the user types keywords and hits “Enter,” the web page displays all the results from the selected search engines in the right pane.
For example, if the user wants to search “Gandhi” with all the search engines selected, a new tab for each search engine will be displayed in the right pane as shown in Figures 5.3 and 5.4.

**Figure 5.3: Search results for “ALL” option – Upper part of the page**
If the user clicks on any different tab, e.g. Digg, all the results from that engine will be displayed for the keyword “Gandhi” as shown in Figure 5.5.
If the user selects a specific search engine and hits “Enter,” the application shows results for the selected search engine only.

For example, if the user selects Bing and AOL Video search engines for the keyword “Sacramento”, the resulting web page will be displayed as shown in Figure 5.6 and Figure 5.7 for the Bing and AOL Video tabs respectively.
Figure 5.6: Search results for “ALL” option with Bing, AOL Video selected – Bing tab
5.3 Unique Results Selected

When the user wants to show the unique results from different search engines, the “ALL” checkbox should be unchecked. The new web page layout will display and the settings for every search engine appear along with the option to select the number of results from each search engine with a sliding bar as shown in Figure 5.8.
Figure 5.8: Home page with “ALL” check box unchecked

If the user wants to show 10 results from Google, 5 results from Bing, 15 results from Entire Web, 5 results from AOL Video and 5 results from Digg for the keyword “Halloween”, a total of 40 unique results from all the selected search engines according to the priority setting will be shown. The Priority specifies which search engine need to be considered first to fetch unique results. Some people prefer a specific search engine compared to the other search engines for the relevant results. In this example, first priority is Google, second is Bing, third is Entire Web, fourth is AOL Video and fifth is
Digg. In this case, the first 10 results displayed will be from Google. The next 5 results displayed will be from Bing. The next 15 results will be shown from Entire Web. The next 5 results will be shown from AOL Video and the last 5 results will be shown from Digg. All the 40 results will be displayed in one page as shown in Figure 5.9.

Figure 5.9: Unique search results with all search engines selected

If the user wants to select only 3 search engines named Digg, Bing and AOL Video with the number of results 5, 10, and 5 respectively for the keyword “movie”, the application will show the results from three search engines only as shown in Figures 5.10. If the user
sets the first priority to Digg, second to AOL Video and third to Bing, the application will fetch the first 5 results from Digg, the next 5 results from AOL Video and the last 10 results from Bing. The user can drag and drop search engine names to change the priority under the Change Priority Section.

![Ultimate Search Engine](image)

**Figure 5.10: Unique search results with Bing, AOL Video and Digg Selected**

5.4 Keep History

When the user wants to see how many times a specific URL has been clicked on a search result, the “Keep History” check box has to be checked. If any URL is clicked
more than 0 times, the number appears with the name of the URL in the search results. Every time the user clicks on a specific URL, counter for that URL is incremented by 1. For example, if the user searches for the keyword “movie” and clicks on URL “http://www.movieretriever.com/blog/209/The-Best-Movies-of-2008” and searches the keyword “movie” again, a “(1)” will be shown with the name of the URL http://www.movieretriever.com/blog/209/The-Best-Movies-of-2008 as shown in the Figure 5.11.

Figure 5.11: Number of times URL hit by the user

5.5 Recent Keywords

A Total of 10 recently searched keywords are displayed on the web page as shown in Figure 5.12. When a new keyword is searched after 10 keywords, the oldest keyword is replaced by the newest keyword.
5.6 Load Balancing

The application is accessed by the Apache HTTP Server and the request is routed to any Tomcat server that is chosen by the Apache HTTP Server. The Apache HTTP server decides which request to send where based on current load on every application server. If a Tomcat server goes down for any reason, a request going to that server is routed to another Tomcat server selected by the Apache HTTP server. The user does not have to specify the port number in order to access the application. To verify load balancing, a footer is shown on the application’s main page. Tomcat A has a footer named “©2011 All rights reserved A,” Tomcat B has a footer named “©2011 All rights reserved B” and Tomcat C has a footer named “©2011 All rights reserved C.” When the application is loaded in the browser, it is executed by any Tomcat server chosen by the Apache HTTP server.
For example, the current application is running on Tomcat C on localhost as shown in Figure 5.13.

![Application runs on Tomcat C](image)

**Figure 5.13: Application runs on Tomcat C**

If Tomcat C is stopped and the page is refreshed, the application will run on Tomcat A in localhost as shown in the Figure 5.14.
Figure 5.14: Application runs on Tomcat A

If Tomcat A is stopped and the page is refreshed, the application will run on Tomcat B on localhost as shown in the Figure 5.15.
Figure 5.15: Application runs on Tomcat B
Chapter 6

DESIGN AND ARCHITECTURE DECISIONS

Several design tradeoffs were considered during the Ultimate Search Engine application development. The following section describes basic design and architecture tradeoffs of the application.

6.1 Database

In the first phase of the project design, the database was planned to be implemented using Microsoft SQL Server. However, performance is the main criteria for this project and this implementation does not need to store large amount of the data. After performing thorough research and calculation in this area, I found that, with the use of an external database, the response time to the client request became much higher. Every time the request is generated from the client, it is sent and received by the server which adds the latency time of the network. The server takes time for database connectivity. Once the database connection is established, the database has to perform data store and fetch operations from the data structure and this process consumes additional time. I found that a database is not a wise choice when there is a small amount of data and performance is the key factor in the application.

The current implementation gains better performance with the use of cookies and XML files compared to the external database implementation. The Cookie and XML file processing is done on the client side that eliminates costly request transfer time, server processing time and database processing time.
6.2 Cookies and XML

As a result of not using an external database, there was a need to store data in an efficient manner that provides better performance with the desired functionality. A cookie was proposed to store all data in the first phase of design. But the implementation of a cookie has certain restrictions. A web browser is not permitted to store more than 300 cookies in total. More than 20 cookies per web domain are not allowed by the browser. And each cookie cannot store more than 4 kilobytes of data.

The data used in the project falls into two categories. The first category is the configuration data that includes the search engine preference, the number of search results per search engine, etc. The second category is the tracking data that includes the URLs accessed by the user in previous sessions, the URL count, the search engine specific URLs, the search keywords, the indexing, etc. The configuration data can be stored in the cookie, but the tracking data for the URLs are large in size, can grow periodically and cannot be stored in the cookies.

As a result, XML files were introduced into the design to store the data related to the URL tracking for a specific user session on the server side in the file system. All the configuration data is stored in cookies, and the URL tracking data is stored in the XML file. Every time the user submits a request, a unique user id is generated and stored in the cookie which is used to request the XML file from the server. During the first user session, a temporary XML file is generated which stores all the information related to the URL with the unique id on the Server. During the subsequent user session, the XML file is fetched from the Server according to the user session, and is kept in a temporary space
on the client side. At the end of the user session, the updated XML file is written to the Server. This implementation provides better performance compared to the external database implementation, as the data fetch and store process is very fast. Most of the calculation is done on the client side and a network access is done only twice during the user session.

6.3 Application Server

Websphere was chosen as the application server in the initial design. But the Websphere application server does not come with a free license. Generally, the Websphere is required in larger applications that use significant components of J2EE architecture like EJB, web services, etc. This project does not have any implementation of such high level J2EE implementation.

The Tomcat application server comes with a free license and is remarkably robust for small to medium level applications. It provides the functionality of load balancing using Horizontal and Vertical clusters. In the final design, the Tomcat application server was selected as it is more appropriate for this project, compared to Websphere in the context of licensing policy and ease of use per project size.

6.4 Search Engine APIs

In the initial design, some search engine APIs were taken into consideration like Yahoo, Ask, Gigablast, DuckDuckGo, etc. But, the license agreements for these APIs require paying a specific license fee in order to use them in the application.

In the final design, I selected Google, AOL Video, BING, Digg and Entire Web, as the APIs for these search engines are free. The Google API used in the project has
certain limitations. It is not a standard Google API. It generates only 10 results in one call. To get more results, the API needs to be called again. The Standard Google API is a paid service.
Chapter 7

RELATED WORK

There are various tools available in the market that fetch information from different web search engines and manipulate the results in different ways. Some of the tools are discussed below.

7.1 Dogpile

The Dogpile search engine gets results from Google, Yahoo and Bing web search engines. It provides the facility to filter results based on different categories like images, video, news, local, etc. The name of the web search engine that finds the results is shown with every related result link. But it does not provide the unique results filtering functionality among different search engines. [12]

Ultimate Search Engine provides combined unique results from different search engines with the facility to show the most recent keywords searched by the user.

7.2 Noobsearch

Noobsearch is a web search engine which combines results from different search engines and shows which search engine generated the result. It shows different categories for the search results. It shows the web page layout in the result along with the link of the URL. It uses static logic to search the keywords within certain web search engines. [13]

Ultimate Search Engine provides the facility to select and rank the search engine according to the user preference. It does not use static logic to get the search results. It also provides the unique results from the selected search engines.
7.3 Metacrawler

Metacrawler provides the result sets from Google, Yahoo and Bing. It provides some suggestions related to the entered keywords in the left pane. It shows the name of the search engine that found the result from the link on the URL. The search mechanism is static for the search engine. It shows sponsored links at the top of result section. [14]

Ultimate Search Engine dynamically gets the search results from Google, AOL Video, BING, Digg and Entire Web.

7.4 Ixquick

Ixquick performs the search in different popular web search engines. It provides Advanced Search, Global Search and Power Refinement options to make the search more user friendly. It provides three different categories for the search results, images, videos and phone. It uses a static search from different web search engines. [15]

Ultimate Search Engine provides dynamic search results from the different search engines in the same page with different tabs. It also provides unique combined results from different search engines according to the selection of the search engines and the priority set for the search engines.
Chapter 8

CONCLUSION AND FUTURE WORK

The Ultimate Search Engine application provides a customized way to search for keywords using different search engines in one place. It also provides the flexibility to change user preference dynamically at run time. Ultimate Search Engine performs load balancing with a cluster. Performance is a very important criterion for this project. As a result, XML files are used on the server side and cookies are used on the client side to store data. The combination of Java Script and AJAX provides robustness in the application and improves the performance by fulfilling most of the requests on the client side.

Ultimate Search Engine makes the best use of Java Script, AJAX, CSS, Tomcat and XML. After working on this project, I have learned how to choose technology and tools that fit the project requirements. I have better exposure to load balancing and API implementation. I have obtained great experience on maintaining performance while fulfilling the requirements.

For future work and enhancements, we can extend the application to include different search categories like sports, news, politics, videos, pictures, etc. Additional search engines can be used in the Ultimate Search Engine application to maximize the result set. The Horizontal cluster or a combination of Vertical and Horizontal clusters could be included to meet the specific requirements for load balancing.
APPENDIX A

Prerequisites

The following hardware and software is required to install and execute the
Ultimate Search Engine application.

System Requirements:

Computer with a 1GB RAM, 20 GB of available hard-disk space, Microsoft
Windows XP with Service Pack 2 or higher versions, 1024x768 or higher-resolution
display with 256 colors.

Software Requirements:

JDK 1.6, Apache Tomcat, Apache HTTP Server, Mod JK Tomcat Connector,
Eclipse with Tomcat server, Internet Explorer 7 or 8, Firefox 3 and higher, Safari 4,
Chrome 4.
APPENDIX B

Definitions

Clustering:

A cluster is defined as a group of application servers that transparently run a J2EE application as if it were a single entity. There are two methods of clustering: vertical scaling and horizontal scaling. Vertical scaling is achieved by increasing the number of servers running on a single machine, whereas horizontal scaling is done by increasing the number of machines in the cluster. Horizontal scaling is more reliable than vertical scaling, since there are multiple machines involved in the cluster environment, as compared to only one machine. With vertical scaling, the machine's processing power, CPU usage, and JVM heap memory configurations are the main factors in deciding how many server instances should be run on one machine.

The servers in a J2EE cluster are usually configured using one of three options: Independent approach, Shared file system or Managed approach. In the independent approach, each application server has its own file system with its own copy of the application files. Another approach is to use a shared file system, where the cluster uses a single storage device that all application servers use to obtain application files. In the managed approach, an administrative server controls access to application content and is responsible for "pushing" appropriate application content to managed servers. The admin server ensures that all servers in the cluster have the application available. It also updates all servers when an application is deployed, and removes the application from all servers when the application is decommissioned. [9]
Web Archive (WAR) File:

A Web application is a group of HTML pages, JSP pages, servlets, resources and source file, which can be managed as a single unit. A Web archive (WAR) file is a packaged Web application. WAR files can be used to import a Web application into a Web server.

In addition to project resources, the WAR file includes a Web deployment descriptor file. The Web deployment descriptor is an XML file that contains deployment information, MIME types, session configuration details, and other settings for a Web application. [10]
APPENDIX C

API Description

Google:

To retrieve Google Web search results, the application sends a simple HTTP request. Google then returns search results in XML/JSON format. XML/JSON-formatted results give you the ability to customize the way search results are displayed.

Here is an example search request.

https://www.googleapis.com/customsearch/v1?key=AIzaSyDlg74rLCcO2s66dc5ElFEs6arPF9IzsI&cx=006308044661630939799:rczz9pkh8gc&q=cars&num=10&start=0&callback=

The first three query parameters are required with each search request. All others are optional.

- key (API Key): key query parameter is used to identify the application. The project's API key is set from the APIs Console.
- cx: Specifies the custom search engine needed to be performed for the search.
- q: Specifies the search keyword. E.g. q=best+cars.
- start: The index of the result to return.
- num: how many search results are needed?
- callback: Name of the JavaScript callback function that handles the response.
Bing:

A request consists of an HTTP GET request to the appropriate URL. There are two URLs, one for XML results and one for JSON results. These are http://api.search.live.net/xml.aspx and http://api.search.live.net/json.aspx. Ultimate Search Engine uses http://api.search.live.net/json.aspx because the application is expecting a JSON response.

For the sample request, we will use the JSON endpoint and query the web for results on the term cars:

http://api.search.live.net/json.aspx?JsonType=callback&JsonCallback=?&Appid=642636B8B26344A69F5FA5C22A629A163752DC6B&query=car&sources=web&web.count=10&web.offset=0

Below are the query parameters used in the API.

- **appid**: Value that enables the API to validate that a request is from a registered Bing application.
- **query**: Specifies the search keyword.
- **sources**: Specifies whether web or news or image results are needed.
- **web.count**: How many search results are needed.
- **web.offset**: The index of the result to return.
- **JsonCallback**: Name of the JavaScript callback function that handles the response.

Entireweb:

Search results from Entireweb are obtained by invoking the http://www.entireweb.com/xmlquery URL on the web server.
The following is an example URL for invoking Entireweb to search for the query cars.

http://www.entireweb.com/xmlquery?pz=01234567012345670123456701234567&n=10&of=0&sc=9&format=json&q=cars&callback=

Parameters for the API are as follows.

- **pz**: This parameter is known as Partner Identification tag.
- **n**: Number of results to return per page. It should be \{10, 15, 20, 25, 30, 40, 50\}.
- **of**: The offset into the result list.
- **sc=9**: is an adaptive mode, that automatically composes a suitable list of clustered and “related link” results. sc=9 is the default.
- **format**: The format to return the response in. Valid values are xml and json.
- **q**: The query to search for.

AOL Video:

The AOL Video Search API allows building advanced web applications without having to spend time on server-side scripting. Client-side scripting related to the API is not performed very often, because the Ajax functions that interact directly with the server are already included in the API. [11]

The first step is to set up an API account. Once an account is created, an application ID is generated that is used every time the AOL server is accessed through the AOL Video API.

Here is code from index.js file.

```javascript
AOLVS = new AOLVideoSearch(appId);
AOLVS.attachEvent("onload", "");
```
AOLVS.attachEvent("onupdate", "onVUpdateH();");
AOLVS.attachEvent("onerror", "");
AOLVS.initialize();

Codes first line creates an object using Appid. In other words, features of the AOL Video API are accessed by calling this object's methods and reading its attributes. Then “onVUpdateH()” function is bound on “onupdate” action. When the results are ready to be processed, the API triggers an “onupdate” event that is associated to “onVUpdateH()” handler function. Finally, initialize() is called to fully prepare the API object for use. This function connects to the AOL server, ensures that the search engine is ready to use, and verifies the supplied application ID. Once all these calls are completed, bind function is called as per event occurs.

Digg:

The following is an example URL for invoking Digg to search for the query cars.

http://services.digg.com/2.0/search.search?query=cars&type=javascript&offset=0&count=10&callback=diggSearchResponse

Here, the Ajax call is not being called for the Digg search API like other implemented APIs. Instead, a script tag is created dynamically and is loaded into the container. Once it is loaded, the parameter callback function will be called to handle the response. In the example the diggSearchResponse function is called.

Below are the query parameters used in the API.

- query: To search specific keyword.
- type: (type=javascript fixed).
- offset: index to start.
- **count**: Results Per Page.
- **callback**: callback function to handle response.
APPENDIX D

Configurations

1. Tomcat A

Server Port: 8105

HTTP Connector Port: 8081

AJP Connector Port: 8109

jvmRoute: tomcatA

2. Tomcat B

Server Port: 8205

HTTP Connector Port: 8082

AJP Connector Port: 8209

jvmRoute: tomcatB

3. Tomcat C

Server Port: 8305

HTTP Connector Port: 8083

AJP Connector Port: 8309

jvmRoute: tomcatC
4. Apache HTTP Server

Httpd:

LoadModule jk_module modules/mod_jk.so

JkWorkersFile "C:\cluster\Apache\conf\workers.properties"

JkLogFile "C:\cluster\Apache\conf\mod_jk.log"

JkLogLevel error

Workers.properties

workers.tomcat_home=/tomcatA
workers.java_home=$JAVA_HOME
ps=/
worker.list=tomcatA,tomcatB,tomcatC,loadbalancer
worker.tomcatA.port=8109
worker.tomcatA.host=localhost
worker.tomcatA.type=ajp13
worker.tomcatA.lbfactor=1
worker.tomcatB.port=8209
worker.tomcatB.host=localhost
worker.tomcatB.type=ajp13
worker.tomcatB.lbfactor=1
worker.tomcatC.port=8309
worker.tomcatC.host=localhost
worker.tomcatC.type=ajp13
worker.tomcatC.lbfactor=1
worker.loadbalancer.type=lb
worker.loadbalancerbalanced_workers=tomcatA,tomcatB,tomcatC
worker.loadbalancer.sticky_session=1
REFERENCES


[4] J2EE Architecture by IBM
t.j2ee.doc.user/topics/cjarch.html [Accessed 6 November, 2011]

[5] Basic architecture of J2EE on ESRI Developer Network
2eeinGIS.GIF [Accessed 6 November, 2011]


[9] Cluster on Tomcat by O'Reilly on Java


[10] Web Achieve (WAR) file definition from Eclipse help


http://dev.aol.com/node/162/subscribe [Accessed 13 November, 2011]

[12] Dogpile search engine home page


[13] Noobsearch search engine description from Killerstartups


[14] Metacrawler description


[15] Ixquick search engine home page


Wiley; First Edition (March 29, 2011)