A Simple and Efficient Approach for Retrieving Live HTML-based Internet Information

Nader Mohamed and Jameela Al-Jaroodi

The College of Information Technology
United Arab Emirates University
Al Ain, P.O. Box 17555, UAE, {nader.m, j.aljaroodi}@uaeu.ac.ae

Abstract: Dynamic HTML documents on the Internet contain useful information that can be reused by other applications. Unlike XML documents, the problem with HTML documents is that they do not have any semantics for the data in the page. Although a programmer can write a program that retrieves a piece of information from a specific HTML document available on the Internet, it will be very difficult to write several different programs to retrieve information from different dynamic HTML pages with varying formats. This paper develops a simple and generic approach to retrieve dynamic HTML Internet-based information. In this approach, several techniques that can be used to retrieve data from dynamic HTML documents are developed. These techniques were developed as a Java class which programmers can use to integrate and reuse HTML Internet Information for other real-time applications that need this information for their operations. The integrated Internet information can be weather information, stock prices, and top news. A number of experiments were conducted to measure the performance of these techniques. In addition, the paper discusses a number of applications that may benefit from this approach.

Keywords: Internet Information, HTML Information Retrieval, Java

1. Introduction

A lot of new networked applications completely depend on the availability of real-time information for their operations. Examples of these applications are online tools for stock investments, applications that need to use online currency exchange information, weather monitoring applications, and news filtering applications. These applications need real-time stock prices, currency exchange rates, interest rates, temperatures, or news for their operation. This information is usually updated in real-time. For example, most stock market websites provide real-time information about current stock prices and deals. Most of this information is available online on the Internet in the form of HTML documents. It will be very useful if the online and active data of these HTML documents can be reused in other real-time applications.

However, unlike XML documents, HTML documents do not have any semantics for their data. HTML documents usually contain tags, scripts, links, and user defined data. Obtaining specific data from a dynamic HTML document for reuse in another application is usually a very complex task. It is very difficult to identify the required parts of the data and dynamically use them in other applications. As a result it is usually very difficult to reuse this rich information of the Internet in other applications.

Recently some research efforts were conducted to find mechanisms to reuse and transfer HTML documents to other reusable or needed formats [1][3][5]. All these solutions are not for real-time reusability of the HTML information in another application. In this paper, we develop a number of effective and time-efficient techniques that simplify the process of obtaining the required information from Internet-based HTML documents. These techniques are developed to be used in different scenarios for enhancing Internet information retrieval. These techniques were implemented in a Java class which allows the Internet information to be reused by other new applications in real-time. One example of these applications is On-line Stock Notification System which notifies users by email or SMS (Simple Message System) about pre-defined stock price events such as a significant price increase or a significant price drop. Such application can be implemented by reusing the stock prices information available on a specific stock market website.

In the rest of the paper, Section 2 provides background information and related work. The developed techniques are defined in Section 3, while in Section 4 experimental measurements of these techniques are discussed. Section 5 discusses possible applications of the approach and Section 6 concludes the paper.

2. Background

Application integration [2][9] has become one of the important aspects of the information technology field. With application integration new applications can be easily created. Integration
among applications is needed for many reasons including; exchanging and sharing information among the applications, reusing functions and services provided by other applications, and introducing new functions to an existing application.

Integration can be done among applications located on a single computer, on a local area network, or over the Internet. Different enabling technologies and approaches were developed to facilitate applications integration on a single machine or local area networks. Examples are different middleware technologies [6] which are usually used for integrating applications. XML documents [8] and technologies are usually used for integrating applications located locally or globally over the Internet. XML document formats and technologies were developed while application integration is in mind. XML documents have clear structure with explicit semantics which makes integration and usability of XML documents well defined and simple.

Unfortunately, most information provided on the Internet is in HTML documents and not in XML documents. These HTML documents [7] are designed to be downloaded and displayed using browsers. Usually these documents contain a lot of HTML tags, scripts, links, and user defined data. Unlike XML, the problem with HTML documents is that they do not have any semantics for the data, which makes the process of identifying the required information in an HTML document very difficult. Thus it is very hard to reuse this abundance of useful and real-time information in other applications.

Some research was conducted to benefit from the Internet HTML documents. One example is developing an approach to link the large amounts of data that are currently available in HTML documents to the Semantic Web ontology [1]. Another example is developing an approach that automatically captures the semantic hierarchies of HTML tables [5]. Some research effort was also conducted to transform HTML documents to another format to satisfy specific applications. One example of this transformation is from HTML Product Catalogues source code and images to RDF [3]. In contrast, our approach is developed to be used for generic purposes. This approach is designed to simplify the process of extracting real-time information from HTML documents and it provides several techniques to extract the information from HTML documents regardless how it is formatted (e.g. in tables or as plain text). These techniques can be used to extract online information and reuse them in other applications.

3. The Approach

The main idea of this approach is based on finding fixed titles or headers that appear in browsers for HTML documents directly or semi-directly before the needed dynamic information. These fixed titles or headers are used as references to know the position of the needed dynamic information.

The proposed approach is developed as a Java class named urlINFO. Multiple objects can be created from this class for different Internet HTML documents that contain some needed information. There is a single constructor with a single parameter in that class. This parameter is used for defining the URL of the required webpage. The class contains the public method download(). This method is used to download/re-download the HTML document. Part of this method is a filtering process. This filtering process takes the downloaded webpage and generates another page in the system’s memory that only contains the page data without any html tags. The output page contains only the application text headers and numbers. Each application data between HTML tags is considered a single data field. The aim of this step is to simplify the process of finding the needed information. Figure 1 shows the main components of the proposed approach.

The user can recall download() method any time that he/she wants to refresh the downloaded information. After calling the download() method, the user can start extracting any information from the page. A number of techniques were developed to find the needed information. These techniques are implemented in a set of methods in the urlINFO class.

The first technique is to find information directly after a specific text header. For example, in Figure 2, the Wal-Mart Stores, Inc. Stock information is displayed from the Yahoo Finance site. The user can find the last trade price, the previous closing price, and the trade volume from this page. To get the last trade price, the user need to call the get("Last Trade:") method. The method will return the last trade price, 48.10, for the user. A partial output of the filtering process is shown in Figure 3. The get method will search for
the title provided by the user, "Last Trade:.", and will return the next data field after this title field.

Fig. 2. Wal-Mart Stores, Inc. stock information displayed form Yahoo Finance.

Fig. 3. Partial Output of Filtering Process. Only data remains. Each line represents a single data field.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last Trade:</td>
<td>48.10</td>
</tr>
<tr>
<td>Trade Time:</td>
<td>Apr 3</td>
</tr>
<tr>
<td>Change:</td>
<td>0.00</td>
</tr>
<tr>
<td>(0.00%)</td>
<td></td>
</tr>
<tr>
<td>Prev Close:</td>
<td>48.10</td>
</tr>
<tr>
<td>Open:</td>
<td>N/A</td>
</tr>
<tr>
<td>Bid:</td>
<td>N/A</td>
</tr>
<tr>
<td>Ask:</td>
<td>N/A</td>
</tr>
<tr>
<td>1y Target Est:</td>
<td>54.96</td>
</tr>
<tr>
<td>Day's Range:</td>
<td>N/A – N/A</td>
</tr>
<tr>
<td>52wk Range:</td>
<td>42.31 – 52.15</td>
</tr>
</tbody>
</table>

The second technique is to find certain information after the appearance of a specific fixed title or header. The technique is to find the \(i^{th}\) information field after the appearance of a specific header the \(n^{th}\) time. For example, Figure 4 is quote web information for RAK Properties with symbol RAKPROP that is listed in Abu Dhabi Securities Market. As we notice, the bid volume amount appears after "RAKPROP" header while that header appeared twice on the webpage. To get the bid volume, this technique should search for two "RAKPROP" text headers before returning the bid volume amount. This technique is implemented in another method with interface get(n, header). For the mentioned example, the call of this method will be get(2, "RAKPROP") which will return 277,500.

Fig. 4. RAKPROP Quote from Abu Dhabi Securities Market.

The third technique is to find information semi-directly after a specific fixed title or header. The technique is to find the \(i^{th}\) information field after the appearance of a specific header the \(n^{th}\) time. For example in the RAKPROP example, the user wants to get the current price of the stock. This price can be found after 7 fields after the appearance of "RAKPROP" header two times. The interface for this method is get(n, header, i). For the mentioned example, the call will be get(2, "RAKPROP", 7). This call will return 1.29.

The three techniques/methods mentioned search for the required information from the beginning of the page. The read pointer of the page will be initialized to the beginning before any calls are made to the get methods. This approach will be fine if just a single piece of information is needed form the page. However, in most cases, multiple information fields are needed from a single web page. For example, the user may need the bid price, ask price, and current price of a specific stock. There is no since to initialize the page read pointer after each get. It is better to find the first field in the page and then continue from that position to find the second one appearing on the page, and then the third one. This approach saves some processing time. To achieve this, five techniques/methods were developed: getWI(), getWI(j), getWI(header), getWI(n, header), getWI(n, header, i), where WI stands for Without read pointer Initialization. The first method, getWI(), returns the next piece of information after the current pointer position. The method getWI(j) returns the field after skipping \(j\) fields from the current read pointer position. The method getWI(header) returns the field located directly after the specified header from the current pointer position. The getWI(n, header) method returns the field after the occurrence of the header \(n\) times from the current pointer position. Finally, the getWI(n, header, i) method returns the \(i^{th}\) field after the occurrence of the header \(n\) times from the current pointer position.
4. Performance

A number of experiments were conducted to measure and compare the performance of some techniques developed in this paper. The first experiment was conducted to measure the download time and single information abstraction from Abu Dhabi Securities Market. The experiment was conducted to get the current price of RAKPROP. The average document download time was 100 milliseconds. The filtering process time was 80 milliseconds. The average price finding time was 0.4 milliseconds.

The second experiment was conducted in a large document from Dubai Financial Market that contains a table that summarizes the stock information for all the stocks listed in the market, around 56 stocks, see Figure 5. The average document download time was 175 milliseconds. The filtering process time was 110 milliseconds. Finding a current price for a stock at the middle of the table will only take 0.5 milliseconds using the \textit{get} method. However, finding the next field, \textit{Trades}, for the same stock takes less than 0.1 milliseconds using \textit{getWI} method.

The main observation from both experiments is that the download and filtering process are the steps that take the majority of the total time to get the information. However, finding a specific field after the download will not take a significant time. Therefore, in case there is a need for information for several pieces of information such as multiple stocks information, then it is better to select a single HTML document that contains all needed information. From that document, various pieces of information related to multiple stocks can be found. Further more, the overall time did not exceed half a second to get any type of information. Therefore, the information needed for reuse in an application will be available in real-time.

5. Reuse of the Internet Information on New Real-Time Applications

Using the proposed approach, new real-time applications can be easily developed. These applications can be integrated with some information from the Internet. For example, a number of applications related to stock investments are developed using the proposed approach. These are:

I. Notification System for Stock Price Changes in Dubai Financial Market: This system can be used by users to define certain price change criteria. The system will notify the users by emails or by any other message type mechanism whenever those criteria are met. Examples of the criteria are when a stock reaches a certain price level or when a stock price increases/decreases by a certain percentage. The proposed system will monitor stock prices through the Internet. It will generate notification messages whenever the defined criteria are met.

II. Multiple Stock Markets Monitoring System: A customer may have stocks listed in multiple financial markets. Although each market provides a software tool for live quotes there is no software tool that displays live quotes for a set of stocks that belong to different securities markets. For example, a customer may have one stock in Abu Dhabi Securities Market, two stocks in Dubai Financial Market, and three in Kuwait stock market. He/She is only interested in monitoring the prices of these stocks. The aim of this project is to implement a networked stock monitoring system in which users can define and monitor the prices of specific stocks in different markets. This system will depend on the Internet to get information about different stocks listed in different markets.

III. Monitoring System for Stocks listed in Multiple Securities Markets: Some investors like to monitor prices of particular stocks that are listed in multiple securities markets to buy or sell them. The aim of this system is to monitor stocks listed in multiple securities markets. In GCC countries, there are some stocks listed in multiple markets. These markets open and close around the same times. A price of a specific stock listed in multiple markets may drop or increase in one market before other markets. This price difference may only happen for short time periods (e.g. a few minutes) before the stock prices in the other markets adjust with the new price drop or increase. The price drop event in one market is a good indication for near future price drops in other markets for the same stock. In addition, the price increase in one market is a good indication for near future increases in other markets for the same stock. Some investors love to have a tool that notifies them about that price differences in order to make the right
investment decisions to buy or sell stocks. The suggested approach in this paper can be used to get stock prices from different markets for this tool.

6. Conclusion

This paper introduced a generic approach to extract information from dynamic HTML documents on the Internet. This extracted information can be easily reused by other on-line applications in real-time. The main idea of this approach is to find specific numbers or texts after specific text headers. Making this type of information available for reuse, provides a good opportunity to develop new real-time application to benefit users. As the amount of information grows on the Internet, it has become more necessary to find efficient techniques to filter out the noise and get the important parts in a timely manner. Our approaches provide one solution to achieve this goal. These techniques were implemented in a Java class which is currently used to develop new applications that depend on some of Internet Information. The next step would be to identify possible challenges to these techniques such as changes made to the formats of the web pages used. We are working on several possible approaches to manage such problems and make the approaches more robust and flexible.

References