A Middleware Architecture for Web Information Reuse

Nader Mohamed and Jameela Al-Jaroodi
The College of Information Technology, UAE University
Al Ain, P.O. Box 17551, UAE
{nader.m, j.aljaroodi}@uaeu.ac.ae

Abstract—This paper introduces the concept of middleware for retrieving web information. The Internet provides huge online and updated information related to different aspects of businesses and environments such as stock prices, currency exchange rates, interest rates, and weather information. All this information can be reused for new applications through the proposed middleware. The paper discusses the Internet information reuse challenges, the proposed middleware architecture, and the middleware services that can be supported. This type of middleware can be used by a number of applications that need some information from the Internet. Examples of these applications are stock investment tools, notification systems, weather monitoring, and many more.

Keywords: Middleware, World Wide Web, Internet Information, Information Reusability, Information Retrieval

1. INTRODUCTION

The Internet has become the main source of information for most people. People use the Internet to obtain different types of information including entertainment, science, technology, finance, politics and much more. Some of the information needed by users are real-time and change frequently. The Internet is full of such live and dynamic information in many domains and of interest to different users. The provided real-time information is made available mostly in text form such as news, multimedia such as Internet TV, or numbers such as stock information, currency exchange rates, interest rates, oil prices, and temperatures.

There are several types of applications that completely depend on the availability of real-time information for their operations. Examples of these applications are web information monitoring [1], online tools for stock investments [2], applications that need to use online currency exchange information, weather monitoring applications, as well as some of the web-based news applications. These applications need real-time stock prices, currency exchange rates, interest rates, temperatures, or news and this information is dynamically and rapidly updated. Some of the mentioned applications rely on the public information available over the Internet to obtain their Information needs. These applications use the Internet as an information source. The public information is available over the Internet through dynamic HTML documents, XML documents, web services, and RSS feeds. Information can be easily and systematically obtained from XML documents and web services due to their structured formatting and available metadata. Some APIs can be used with the applications that need specific information from the Internet to retrieve the required information. In addition, some techniques and tools were developed to enable retrieving information from HTML documents over the Internet. One of these tools is urlINFO [3]. urlINFO is a Java class that can be used by Java applications to retrieve live information from the Internet in real-time.

Although there are some available techniques and tools for enabling the reuse of public Internet information for new applications, there is a number of development and operational challenges associated with this reuse. These challenges are discussed in this paper. In addition, a middleware architecture that solves these challenges is proposed. This proposed middleware will help enhance the development and operations for utilizing the available public information for new applications. The paper will not discuss the specific implementation details of the proposed middleware; however, it will discuss its main architecture as well as its prospected services to solve the challenges of Internet information reuse.

In the rest of the paper, Section 2 provides background information and related work on Internet information and middleware solutions. The challenges of Internet information reuse are discussed in Section 3. Section 4 discusses the proposed middleware services while Section 5 proposes a middleware architecture for providing these services. Finally, Section 6 concludes the paper.

2. INTERNET INFORMATION AND MIDDLEWARE

The public information is available over the Internet through different documents formats and access mechanisms. Information can be retrieved by other applications by executing web services [4] or retrieving XML [5] or HTML [6] documents. Web services provide web APIs that can be accessed over a network, such as the Internet, and executed on a remote system hosting the requested services. These remote systems can provide different services including providing information about different aspects or products. For example, different stock markets can provide web services to supply current stock prices. Banks can provide web services to supply information about loans or currency exchange rates. If desired information is needed by an application, then user can easily define a variable and link it with the
corresponding web service which provides the required information. The main problem with web services is that not all types of information available over the Internet are provided by Web Services. Most of useful information on the Internet is still available in HTML format.

Although XML documents have self-defined structures that make them easy to deal with, again we are faced with the problem that most of the Internet information is delivered to users in HTML. Unlike XML documents, HTML documents do not have any semantics for their data. Obtaining specific data from a dynamic HTML document for reuse in other applications can be a complex task. It is very difficult to identify the required parts of the data and dynamically use it in other applications.

We have recently developed a simple and efficient approach for retrieving live HTML-based Internet information [3]. The main idea 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 reference points to know the position of the required dynamic information. The proposed approach is developed as a Java class, urlINFO. Multiple objects can be created from this class for different Internet HTML documents that contain some of the required information. A number of techniques were developed to find this information in any HTML document. These techniques are implemented in a set of methods listed in Table 1.

All these techniques can be used to retrieve Internet information to be used in new applications. As soon as the fields are identified the get or getWI methods with the right arguments can be used, which will allow the application to retrieve the required information. Users can use any HTML document on the Internet to retrieve any information they need. More information about the implementation and performance of the mentioned approach can be found in [3].

Our work proposes the use of middleware to facilitate access, reuse and integration of available public information with application programs designed to meet the specific needs of users. Generally, different middleware platforms were created to add new values for different systems such as enterprise systems [7], cluster computing [8], wireless sensor networks [9], mobile ad hoc networks [10], and robotics [11]. The main research goals in these middleware platforms are to develop simple mechanisms, approaches, and methodologies that add value to existing computer systems, networks, and distributed applications. This value can be in the form of scalability, reliability, availability, usability, extensibility, manageability, reusability, stability, efficiency, autonomicity and integrity. The mechanisms and approaches are usually based on the reuse of existing methods, protocols, software, and systems to add the needed values.

Some research was conducted to benefit from the Internet HTML documents. One example is developing an approach to link the large amount of data that are currently available in HTML documents to the Semantic Web ontology [12]. Another example is developing an approach that automatically captures the semantic hierarchies of HTML tables [13]. 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 [14].

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>get(header)</td>
<td>To return the next field directly after the defined header. The search starts from the beginning of the page.</td>
</tr>
<tr>
<td>get(n,header)</td>
<td>To return the next field directly after the defined header appears n times. The search starts from the beginning of the page.</td>
</tr>
<tr>
<td>get(n,header, i)</td>
<td>To return the field after skipping i fields after the defined header appears n times. The search starts from the beginning of the page.</td>
</tr>
<tr>
<td>getWI()</td>
<td>To return the next field from the current read pointer position.</td>
</tr>
<tr>
<td>getWI(i)</td>
<td>To return the field after skipping i fields from the current read pointer position.</td>
</tr>
<tr>
<td>getWI(header)</td>
<td>To return the field located directly after the specified header from the current read pointer position.</td>
</tr>
<tr>
<td>getWI(n, header)</td>
<td>To return the field after the occurrence of the header n times from the current pointer position.</td>
</tr>
<tr>
<td>getWI(n, header, i)</td>
<td>To return the field after skipping i fields after the defined header appears n times from the current pointer position.</td>
</tr>
</tbody>
</table>

3. INTERNET INFORMATION REUSE CHALLENGES

In this section, we discuss the challenges facing the integration of Internet information with other applications that will need to use this information and rely on the existence of live up-to-date data from the Internet for its functions. These challenges need to be addressed and efficient solutions are required to simplify the process of the integration and to smooth the communication among the Internet and other applications.

These challenges are:

- **Interoperability**: The Internet mainly provides information in the form of HTML documents, XML, documents, and web services. On the other hand, most applications still use CORBA, RMI, and DCOM to facilitate integration. It is very difficult for example to allow a CORBA, RMI, or DCOM based application to reuse the Internet Information provided in HTML, XML, or even sometimes through web services. Internet information providers do not support CORBA, RMI, and DCOM interfaces through the Internet since as these
were designed to use special port numbers that are typically disabled by firewalls. Applications that support web services can directly integrate themselves with the Internet to get the required information. Web services overcome the disabled ports problem by using the HTTP protocol for communication. HTTP usually uses port 80 which is generally enabled by most firewalls. Unfortunately, not all Internet information is provided by web services. Furthermore, not all applications can support web services yet. In addition, XML provides some structure to the data made available; however, just as in web services, these are not very commonly used over the Internet. To date, most of the information is still provided in HTML documents, which imposes a challenge on how to extract the required parts to be used by another application.

- **HTML Format Changes:** most information is still available on the Internet in HTML format. Unlike XML documents, HTML documents are unstructured and have no semantics for the fields present in the document. As a result, integrating an Internet HTML document that contains dynamic information with a local application can be a very difficult task. Although, it is possible to implement a solution for the integration by extracting the required information from the relevant HTML documents by knowing and utilizing the structure of the document and the position of the required information in the document. This allows an application to identify certain values based on their relative position to some fixed items in the document like labels or specific texts. However, the structure and the positions may change at any time, which makes the extraction method useless after the change. In addition, dealing with multiple HTML documents with different structures can be a very intricate task. Any changes in any used HTML document will require some changes in the local application that uses this document.

- **Distributed Information/Servers:** a local application may require some information that is distributed over multiple web servers located in different places. These servers may support different mechanisms to provide information such as through HTML documents, web services, or XML documents. At the same time, the response times for the requests from the client/local application asking for certain information from these servers may differ. Therefore, it becomes very difficult for the local application to deal with all the heterogeneity in the delivery mechanisms, in the response time, and in the number of the servers. This imposes a great challenge on the application developer to account for all these differences and ensure efficient operation of the information integration.

- **Highly Dynamic Information:** the required information provided by HTML documents, XML documents, or web services can be very dynamic. This causes the required information to rapidly change. For some applications, it is required to capture all changes that occur over time. One example is a stock price displayed in a dynamic HTML page. That price may change every two seconds. At the same time, some application may require registering all changes to that price to perform some calculations, analysis or make some decisions. Implementing the methods in the local application to get all changes in some fields in time and keep track of these changes continuously can be a very complicated task.

- **Fault Tolerance:** some web servers may be unavailable for some time due to different reasons including overloaded servers, network problems, and server maintenance. In addition, servers may have varying response times. A local application may not be able to function correctly due to the unavailability of a web server that provides some of the required information. However, the required information may be duplicated over multiple web servers under different contexts and possible different formats as well. For example the current price of a specific stock can be available on multiple websites related to two different organizations. If the local application uses one website to get the information and that site fails, it will not be possible to switch to a different site to get that same piece of information. However, it will be desirable to make the application capable of performing that switch when necessary. Yet, it is very difficult to utilize multiple web servers and implement a fault tolerance mechanism among them to provide the required information in real-time bases even with some faults.

- **Efficiency:** some Internet information (especially those available on a single web page) may be required by multiple local applications at the same time. For example, all stock information in a single stock market are displayed in real-time on a single web page. Several applications may be requesting different stock prices from that same list at the same time. If the extraction is done within the application, each one will download and process the same page, while it may be much more efficient to download the page once, process it to extract multiple pieces of information then provide each application with its own required information. However, it is a challenging task to determine the duplication in requests among several independent applications and efficiently reduce the amount of processing required to extract the needed information for each one of the applications.

- **Software Engineering:** there are several possible solutions for the challenges mentioned above. These
solutions can be implemented as part of the local applications that need to use the Internet information. However, this approach is inefficient and needs a huge development and testing efforts and a lot of time. This effort may be duplicated for different applications that need to reuse Internet information. It will be more efficient to have some well developed and appropriately tested independent services that can be efficiently used to obtain the required information by any application.

4. THE MIDDLEWARE SERVICES

In this section we discuss a middleware solution to solve most of the challenges mentioned in Section 3. The middleware connects the Internet as an information source with the local applications that need to use the information (see Figure 1). This middleware provides some services that can be used by users to configure the required information needed by their local applications. The configuration defines the location of the required information.

The middleware generally provides the following functions:
1. Establish connections with web servers and web services.
2. Download and extract the required information from HTML and XML documents.
3. Present all the required Internet information in a uniform way such that it can be easily reused by or integrated with the local applications.
4. Capture changes in highly dynamic Internet information.
5. Validate the integrity of the received Internet Information.
6. Provide a fault tolerance mechanism by utilizing the duplications of the required information over the Internet.
7. Provide APIs to allow the local applications developers to easily use the middleware services.

The middleware provides the required Internet information for local applications in three delivery techniques:

1. **Polling**: in this technique, the required information will only be downloaded and provided when a local application requests it. The local applications are provided with APIs to make the requests. Responses for these types of requests usually take few seconds since the middleware will need to connect to Internet servers, download and process the web page and deliver the requested information with each request.

2. **Caching**: in this technique the middleware frequently downloads and extract the required information and keeps them in a local cache. The information cached will be based on the history of requests made by local applications. The cache will contain the latest downloaded information that may be soon needed by the local applications. Local applications can directly read the required information from the cache using the available APIs. This type of read will not take much time from the local applications to get recent information.

3. **Notification**: in this technique, the local applications can ask the middleware to send notifications to them when a certain value over the Internet has changed. The middleware will monitor that value and will only send the notification when the value has changed from the time the request was made. For example, an application is interested to be notified as soon as the current Google Stock price changed. In this case, the middleware will monitor the Google stock current price from one of the web pages or one of the web services providing this information and will only notify the application when the price changes. This type of communication request is useful for applications that do not need a frequent access to the Internet information. It transfers the overhead of frequent Internet accesses from the application to the middleware.

The suitability of the above communication techniques depends on the application scenarios. Table 2 lists some of the common application scenarios and their most suitable communication techniques.

The middleware solution addresses several of the challenges discussed in Section 3. Interoperability is addressed by providing a middleware framework that may be implemented in several ways such as using Java modules which can operate across different platforms. In addition, the framework is flexible enough to allow for the incorporation of different components. The middleware is capable of...
handling highly dynamic and changing HTML content. The middleware framework also allows for incorporating multiple sources and servers to be used. The framework as it is currently may not tolerate faults. However, incorporating a fault discovery module along with replication of functionality will support this requirement. Finally, the overall design of the middleware framework is efficient and reusable.

<table>
<thead>
<tr>
<th>Application Scenario</th>
<th>Communication Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single application accesses different Internet information from time to time.</td>
<td>Polling</td>
</tr>
<tr>
<td>Single or multiple applications need a continuous stream of information from one or a few web sources.</td>
<td>Caching</td>
</tr>
<tr>
<td>Multiple applications need the same information from the Internet.</td>
<td>Caching</td>
</tr>
<tr>
<td>Single or multiple applications need to capture changes in some Internet information.</td>
<td>Notification</td>
</tr>
</tbody>
</table>

5. The Middleware Architecture

The middleware for Internet information can be designed and implemented in three layers. The layers are: The Internet Information Retrieval Layer, The Cache Layer, and the Delivery Layer (see Figure 2). In this section, the functions of each layer are discussed.

5. The MiddleWare Architecture

The middleware for Internet information can be designed and implemented in three layers. The layers are: The Internet Information Retrieval Layer, The Cache Layer, and the Delivery Layer (see Figure 2). In this section, the functions of each layer are discussed.

A. Internet Information Retrieval Layer

One of the main layers of this middleware is the Internet information retrieval layer. This layer directly deals with the Internet web servers. The main function of this layer is to retrieve the required Internet information. The required internet information can be obtained from the Internet using web services or using any library that provides mechanisms to obtain the required information from dynamic HTML pages as discussed in Section 2. The information is obtained from the Internet in a series of individual requests to be used for serving the polling type deliveries which we discussed in the previous section or as streams to update the cache layer. This layer can also use multiple threads parallelize the retrieval of highly dynamic information from the Internet. This layer retrieves the required internet information based on user configurations.

One of the advanced functions that can be provided in this layer is to discover changes in the structure of the defined dynamic HTML documents. This layer can implement an automatic validation mechanism to allow the middleware to make sure that the formats of the defined HTML documents were not changed before attempting to extract the required Internet information. One possible solution for this problem is to automatically capture and store the format patterns of the HTML documents. These patterns can be used by the system to discover any future changes in the downloaded documents. In case there are some changes, the system notifies the middleware administrator to configure new parameters for the middleware.

Another function that can be provided by this layer is to enhance the performance and reliability of the retrieving process. If a piece of the required information is provided by multiple sites or web services, this layer can discover which site or web service can provide faster access. This can be discovered automatically by the layer using some experimental testing. In addition, this layer can switch from a faulty source or unreliable source to a working source. This function requires that the middleware administrator defines all websites or web services that provide the same information.

B. Cache Layer

The main function of this layer is to provide memory for updated information obtained from the Internet. This layer will be accessed by the Internet Information Retrieval Layer to update the cache with new information and by the Delivery Layer to obtain updates on required Internet information.

One of the advanced functions that can be implemented by this layer is to capture the access patterns of the information in the cache by the applications. Capturing the access patterns can be used to adjust the required speed of retrieving the required information from the Internet by the Internet Information Retrieval Layer. For example, the applications access a certain value in the cache once every 20 seconds. In this case there is no sense in retrieving that value every 5 seconds. Therefore, the retrieval layer can be informed about that fact to adjust its download accordingly.
C. Delivery Layer

This layer will be accessed by the application to receive Internet information either from the cache or from the Internet directly using the Internet Information Retrieval Layer. This layer will also wrap the required internet information to a format that can be accessed by the applications. For example, this layer can provide different access methods such as RMI, CORBA, web services, and DCOM. These access methods can be either implemented by the users or using tools to help them in automatically generating servers that use both the cache layer and the Internet information retrieval layer to get the required information for the applications. The implementation of code generation techniques can be similar to [15].

This layer can combine information collected from different web pages and web services to be delivered as a reply for an application request. The advantage of this function is that instead of making the applications deal with multiple web services and web pages to collect a set of needed information, this layer can provide all required information in one record and reply. This layer also implements the notification services mentioned in the previous section. This layer will notify the interested applications about any changes in required values.

6. CONCLUSION

In this paper we discussed the design issues of middleware services to help retrieve, integrate and reuse dynamic web-based information from the Internet with local applications. To do that, we went through the different methods used to access Internet information and how middleware solutions may be useful to enable and optimize these methods. We discussed the different challenges to be addressed when considering the middleware design. Some of these challenges are interoperability, HTML format changes, distributed information, highly dynamic information, fault tolerance, efficiency and software engineering issues. Then we discussed the design of the proposed middleware which provides three delivery techniques: polling, caching and notification. Finally we described the architecture of the proposed middleware which comprise of three layers: the Internet Information Retrieval Layer, the Cache Layer and the Delivery Layer. This paper provides the design aspects of the middleware, but avoids going into any implementation details since these would change significantly based on the overall development environment and utilization aspects.

ACKNOWLEDGEMENTS

This work was partially supported by a UAE University research grants #01-04-9-11/09 and #01-03-9-11/08.

REFERENCES


