Towards flexible and lightweight integration of web applications by end-user programming

Hao Han
Digital Content and Media Sciences Research Division, National Institute of Informatics, Tokyo, Japan, and
Takehiro Tokuda
Department of Computer Science, Tokyo Institute of Technology, Tokyo, Japan

Abstract
Purpose – The purpose of this paper is to present a method to realize the flexible and lightweight integration of general web applications.
Design/methodology/approach – The information extraction and functionality emulation method are proposed to realize the web information integration for the general web applications. All the processes of web information searching, submitting and extraction are run at client-side by end-user programming like a real web service.
Findings – The implementation shows that the required programming techniques are within the abilities of general web users, and without needing to write too many programs.
Originality/value – A Java-based class package was developed for web information searching/submitting/extraction, which users can integrate easily with the general web applications.

Keywords Internet, Computer applications, Programming, Information retrieval

Paper type Research paper

1. Introduction
With the development of the internet, the web becomes the richest source of information. Although there is a tremendous amount of information available, they are not always in the forms that support end-users’ needs. There is a growing trend of enabling users to view diverse sources of data in an integrated manner, called mashup or mixup. These integration technologies have made a shift in web service development and created a new generation of widely popular and successful web services such as Google Maps API and YouTube Data API (YouTube RSS Feeds).

However, this integration is based on the combination of web services mostly and limited to the web sites that provide the open web service APIs. Unfortunately, most existing web sites do not provide web services. Web applications are still the main methods for the information distribution. For example, the famous global news site CNN lets users search for the online news by entering the keywords at web page. Once the users submit the search, CNN would present the news search results. However, this news search function cannot be integrated into other systems because CNN does not open this search function as a web service. Similarly, BBC Country Profiles does not provide the web service APIs and it is difficult for the developers to integrate it with other web services.

Therefore, the current situation leads to the following trends and difficulties of mashup web application development: The mashup application developers can write...
the programs to access the web service APIs. However, there is not a program interface used to access the data, computations (application logic) and user interfaces provided by different kinds of web applications.

In this paper, we propose the information extraction and functionality emulation method to realize the web information integration for the general web applications. We select the target web applications, search for the desired information, submit the user request, and extract the partial information to realize the virtual web service functions. All the processes of web information searching, submitting and extraction are run at client-side by end-user programming like a real web service. We developed a Java-based class package for web information searching/submitting/extraction, and the users can integrate web applications with our class package easily. Our implementation shows that the needed programming techniques are within the abilities of general web users, and without needing to write too many programs.

Our research emphasis is laid on the development of program interface used to integrate web applications by end-user programming, not the graphical mashup editors.

The organization of the rest of this paper is as follows. In Section 2, we give the motivation of our research and an overview of the related work. In Section 3, we explain our web information extraction approach in detail. We construct a web application integration system and give an evaluation of our approach in Section 4. Finally, we conclude our approach and give the future work in Section 5.

2. Motivation and related work
With the development of Web 2.0, there are a rapidly growing number of mashup systems. According to ProgrammableWeb, a mashup community web site, the total number of listed mashup systems is more than 4,700, and on average there are more than three new systems generated everyday in March 2010. Most web mashup technologies are based on the combination of web services or web feeds. Yahoo Pipes and Microsoft Popfly are the composition tools to aggregate, manipulate and mashup web services or web feeds from different web sites with a graphical user interface (GUI). Mixup (Yu et al., 2007) is a development and runtime environment for UI integration (Daniel et al., 2007). It can quickly build complex user interfaces for easy integration by using the web service APIs available. Mashup Feeds (Tatemura et al., 2007) and WMSL (Sabbouh et al., 2007) support integrated web services as continuous queries. They create the new services by connecting the web services using join, select and map operations. Like these methods, Google Mashup Editor, IBM QEDWiki (IBM Mashup Center) and some other service-based methods (Maximilien et al., 2007; Wohlstadt et al., 2009; Zhao, 2008) are also limited to the combination of existing web services, web feeds or generated web components. Although the desire to create mashup systems is strong, the existing web services are not adequate for the users’ needs and many famous and popular web sites do not provide web services.

Many works have been developed to integrate the web applications. The most-used method is partial web page clipping. The users clip a selected part of web page, and paste it into a personal web page. I-know is a simple web application to generate a customized personal web page. It extracts the partial text information between the defined start and end keywords from a web page and creates a web page by listing the extracted text information. However, the extracted information is limited to text. Internet Scrapbook (Koseki and Sugiura, 1998) is a tool that allows users to interactively extract
components of multiple web pages by mouse clipping and assemble them into a single personal web page. However, the extracted information is a part of HTML document and the users cannot change the layout of the extracted parts.

Crunch (Gupta and Kaiser, 2005) and PSO (Suzuki and Tokuda, 2003) are based on the analysis of the DOM trees of web pages. Crunch is a HTML tag filter to retrieve the contents and implemented as a proxy that filters web pages before they are loaded into the users’ browsers. PSO is an approach to extract the parts of web pages using XPath. However, the users have to spend much time in configuring a desired tag filter or XPath manually after analyzing the source of HTML documents of the web pages. Similarly, ANDES (Myllymaki, 2001) is an XML-based methodology to use the manually created XSLT processors to realize the data extraction.

Compared with the manually or semi-manually generated extraction patterns, IEPAD (Chang and Lui, 2001) proposes an automatic pattern discovery system. The users select the target pattern from the discovered candidate patterns, then the extractor receives the target pattern and a web page as input, and applies pattern-matching algorithm to recognize and extract the information. Web service gateway (Huy et al., 2005) parses the response HTML page by flexible tree-style and returns the extracted data. It allows the users to try all the HTML parsers available one by one in the web service gateway to find which is the most suitable for parsing the response HTML pages. However, it is difficult for the users to develop the personalized parsers because the users need to find out the tags representing the desired parts one by one. C3W (Fujima et al., 2004) provides an interface for automating data flows. With C3W, the users can clip elements from web pages to wrap an application and connect wrapped applications using spreadsheet-like formulas, and clone the interface elements so that several sets of parameters and results may be handled in parallel. However, it does not appear to be easy to work with large sets of data and realize the interaction among different web applications. Extracting typed data from multiple web pages is more suitable to generate mashup systems. Han and Tokuda (2007) propose an approach that provides the users a GUI to select the desired parts: text or image, uses the paths to extract the partial information from the similar web pages, and finally returns the users a resulting page. However, this system does not support the link extraction and information searching. Marmite (Wong and Hong, 2007), implemented as a Firefox plug-in using JavaScript and XUL, uses a basic screen-scraping operator to extract the content from web pages and integrate it with other data sources. The operator uses a simple XPath pattern matcher and the data are processed in a manner similar to Unix pipes. MashMaker (Ennals and Garofalakis, 2007) is a tool for editing, querying, manipulating and visualizing continuously updated semi-structured data. It allows users to create their own mashups based on data and queries produced by other users and by remote sites. However, they do not appear to support the information integration of dynamically generated web pages like the result pages from form-based query. Some approaches are proposed to construct web services or web feeds based on the web applications. HTML2RSS (Nanno and Okumura, 2006) is a system to automatically generate RSS feeds from HTML documents that consist of time-series items such as blog, BBS, chats and mailing lists. However, it is limited to the web pages that consist of list of data items with similar or special data structures. Pollock (Lu et al., 2005) can create a virtual web service from form-based query interface of web sites. It generates wrappers using XWrap, and WSDL file using web site-related information, then publishes the details of the virtual web service into UDDI, but this system needs the users to parse
the HTML documents of the form-based web pages. Tatsubori and Takashi (2006) and WIKE (Han and Tokuda, 2008) also provide the web services generation methods based on partial information extraction. These methods take a great deal of time and skills to create such services in a proxy server run between the target web applications and users, and it is extremely unlikely that the constructed web services can support all the needs of all of its end-users.

To address these problems or limitations (e.g. need the components produced by portlets (Java Portlet) or Web Services for Remote Portlets (WSRP)), we propose a web information extraction approach to integrate the web applications by end-user programming. Compared with the developed work, our approach has the following features:

- All the processes of searching, submitting, extraction and integration are run at client-side by end-user as shown in Figure 1. The users can decide the target parts of web pages by themselves. Proxy server is not needed.
- We focus on extracting typed data from web pages and the extracted result is structured data excluding HTML tags.
- We extract all kinds of information including text, link, image and object from different layout such as list and table.
- We support the information extraction in the static web pages and the dynamically generated web pages with dynamical URLs such as the result pages from form-based query.
- The users can realize the continuous queries over multiple web pages easily by end-user programming.

We explain our approach, construct a web application integration system, and give an evaluation in the following sections.

3. Web information extraction

Usually, a real web service responds to the requests of users by returning the data in the server-side database. The web service developers design the query commands by an interactive and programming language such as SQL to retrieve data in the tables of database. For our web applications integration, the web applications work as

---

**Figure 1.**
Proxy-server-side approach vs client-side approach
the server-side database and the target web pages work as the tables. The end-users use our web information extraction method to search for the desired information and extract it.

Compared with web services, the web applications are not suitable for integration because they are designed for browsing by users, not for the parsing by computer program. The web pages of web applications, usually in HTML or XHTML formats, are used to display the data in a formatted way, not used to share the structured data across different information systems. In order to realize the web information extraction, we emulate the browsing of users by end-user programming to reach the desired information, and use string matching and tree structures of web pages to extract it.

We use HTML Parser to parse the web pages and invert all the relative addresses to absolute addresses.

3.1 Data type
There are many kinds of information in web applications such as text, picture and link. During the information extraction, the users need to specify the type of target data. Data type represents “What kind of information is needed?” For example, in a web page, usually each link contains an anchor text associated with a URL. Without the specified data type, we cannot get the right information because we do not know which one is needed between text and URL.

In a web page, a visible item represents a piece of information that cannot be divided into smaller parts, and is the node value or attribute of a single node in an HTML document. We give our data type definition of visible items as shown in Figure 2. There are two kinds of data types. The first kind contains text, image, object and link mainly. Text is the character string in web pages such as an article. Image is one instance of the picture embedded in tag `<img>`. Object is one instance of the video or other multimedia file embedded in tag `<object>`. Link is a reference to another document or other resource embedded in tag `<a>`. Usually, the first kind of data types are the final results of extraction, and used in web information interaction and integration.

The often-used items of second kind are select-option, form-input and others. The select-option is an option in the drop-down list and each option represents a link.

![Figure 2. Main examples of data types: text, image, object, link, select-option and form-input](image)
The form-input is an input field in a form used to accept the users’ queries. For each form-input, there is a corresponding form that contains the parameters and attributes used to connect to the server-side form handler and send the requests. The second kind of data types are used in web pages transition.

We use these data types in information searching and extraction in the following sections.

3.2 Information searching

Not all the contents of web applications are necessary and useful for the users. We search for the target information in web pages to realize the query functions. Like the SQL, we select the information from the target web pages where the information satisfies some conditions. We define the searching function as follows:

\[
\text{Search}(P, K, T_1, R, T_2)
\]

where, \(P\) is the target web page, \(K\) is the searching/reference keyword that refers to the desired information, \(T_1\) is the data type of searching keyword, \(R\) is the range of searching, \(T_2\) is the data type of desired information, and the returned value is the node list of search results. The searching range represents the number of visible items between the searching keyword and the desired information, and has five types as shown in Table I. The searching keyword is a very important element during our searching and works as an index of our desired information. We need to find a suitable searching keyword because a suitable searching keyword can give a more precise search result like a well-written where-clause of SQL.

We give the examples in our implementation.

3.3 Information extraction

After we find the target node, we need to extract the information from the node in text format excluding the tags of HTML document according to the defined data type. There are three kinds of data structures in the web pages as shown in Figure 3: single, list and table. Single means a node without similar sibling nodes such as the title of an article. List means a list of nodes with similar paths such as result list in a search result page. Table means a group of nodes arranged in two-dimensional rows and columns such as the result records in a Google Image Search result page.

Each HTML document of web application can be parsed as a tree structure, and our extraction method is based on the analysis of the tree structure. We define the extraction function as follows:

\[
\text{Extract}(N, T, S)
\]

<table>
<thead>
<tr>
<th>Value ((R))</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>(R = \text{Max value})</td>
<td>The first (T_2) node following after (K) in the whole page</td>
</tr>
<tr>
<td>(R = &gt; 0)</td>
<td>The first (T_2) node following after (K) within (R)</td>
</tr>
<tr>
<td>(R = 0)</td>
<td>The (T_2) node containing (K)</td>
</tr>
<tr>
<td>(R = &lt; 0)</td>
<td>The first (T_2) node previous to (K) within (</td>
</tr>
<tr>
<td>(R = \text{Min value})</td>
<td>The first (T_2) node previous to (K) in the whole page</td>
</tr>
</tbody>
</table>

Table I. The types of searching/query range
where, N is the target node, T is the data type, S is the data structure and the returned value is the extraction result list.

We give the detailed processes of extraction in the following sections.

3.3.1 Single. Among the information extraction from three kinds of data structures, the extraction from a single target node is the most easy and basic. As described in Table II, we extract the information according to the defined data type.

For example, the extracted information of a photo is the value of attribute src of node <img>, and the extracted information of a link is the value of attribute href of node <a>.

We do not extract the information from the items such as select-option or form-input. They are used in Section 3.4 for request submitting.

3.3.2 List. The node list extraction is based on the tree structure of HTML document. In the tree structure, each node has its own path. We use the following steps to extract the corresponding node list of node N if the data structure is list:

1. We get the path of node N.
2. We get the parent node P of N.
3. We get the sub-tree S whose root node is P.
4. We get the node list L of which each has the same path as N without considering the orders of child nodes of P under S.
5. If we find two or more than two nodes in L, or P is <body>, then L is the final node list. Otherwise, we set the parent node of P as the root node of S; then go to Step (4).

Each node of the extracted node list L represents a part of list as shown in Figure 4.

We extract the information from each node of node list according to the data type by the method described in Section 3.3.1.

3.3.3 Table. The table is often used in web pages. We extract the information from a table structure by using the node list extraction method twice because the table is

<table>
<thead>
<tr>
<th>Data type</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Node value of corresponding node</td>
</tr>
<tr>
<td>Image</td>
<td>Attribute value of corresponding &lt;img&gt; node</td>
</tr>
<tr>
<td>Object</td>
<td>Attribute value of corresponding &lt;object&gt; node</td>
</tr>
<tr>
<td>Link</td>
<td>Embedded link value of corresponding &lt;a&gt; node</td>
</tr>
</tbody>
</table>

Table II. Examples of information extraction from single node

Notes: Single list and table
three-dimensional and can be viewed as a list of list. Figure 5 shows the process of information extraction from table structure.

3.4 Request submitting

Usually, for the users, there are some basic types of methods to send their requests and get the response web pages. For example, one type is to click a link or an option in drop-down list in web page by mouse to view a new web page. Another one type is to enter a keyword into a form-input field by keyboard and click the submit button by mouse to send the query. For the request submitting, there are POST and GET methods, and some web sites use the encrypted codes or randomly generated codes. In order to get the response web pages from all kinds of web sites automatically, we use HtmlUnit to emulate the submitting operation.
We define the request-submit function as follows:

\[ \text{Submit}(F, K) \]

where, \( F \) is the select-option or form-input, \( K \) is the selected item name or query keywords, and the returned value is the result page. The users give the selected item name or query keywords, and get the result page as response.

Our request-submit function is applicable to the general web sites without the manual analysis.

We use the request-submit function in our implementation.

4. Implementation and evaluation

In this section, we implement our approach to combine web services, web feeds and web applications from more than one web site into a single integrated system. Web services are self-contained and self-describing, and communicate using open protocols like HTTP. The most-used style architectures of web services are SOAP and REST. SOAP stands for Simple Object Access Protocol. Google implements most of their web services to use SOAP, and we can find SOAP web services in a number of enterprise software. REST stands for Representational State Transfer. A number of new web services are implemented using a REST style architecture these days rather than a SOAP one, such as the web services of Yahoo (Yahoo Developer Network Home), Flickr (Flickr API) and Google Weather (Google Weather API). Compared with SOAP, REST is lightweight and easy to build, and provides the readable results. A web feed is an XML-based document that contains content items. The two main web feed formats are RSS and Atom. Google and more and more news/blog sites provide the web feeds for the information distribution. Compared with web services and web feeds, we extract the information from web applications to realize the functions as those of the web services. In order to interact with different web sites, we use the programming language such as Java and client-side script language such as JavaScript to call the web services, parse the web feeds and extract the information from web applications. Finally, these information are integrated into a single system and interacted with each other.

Our system, called world in web, has the following functions:

(1) We can view the following country information after we select a country from country list as shown in Figure 6(a):
   - A: the position information in map;
   - B: the country name, population, capital city and area, and the information and photo of the country leader;
   - C: the main cities;
   - D: the latest news; and
   - E: the key events of the given year.

(2) We can view the following city information after we click a city from the listed main cities as shown in Figure 6(b):
   - F: the position information in map;
   - G: the weather information; and
   - H: the latest news.
We use the following steps to construct our system:

(1) We select the target web services, web feeds and web applications listed in Table III.

(2) As shown in Figure 7, we realize the function of CNN news search engine. We search for the form-input from the top page of CNN, and submit the request to get the search result page. We extract the result records from the result page using the searching keyword.

(3) As shown in Figure 8, we realize the function of BBC country information search. We search for the select-option at top page of BBC Country Profiles, and submit the request to get the target country page. We extract the information of country and leader, and the URL of timeline page from country page. We search for the key events using the given year and extract them from the timeline page.

(4) We program to realize the interaction of CNN news search and BBC country information search with the web services and web feeds.

Our approach is based on the string matching and tree structures of web pages. We realize the searching and extraction functions like the retrieval function of SQL, and complete the integration of web applications by our defined functions.

<table>
<thead>
<tr>
<th>Type</th>
<th>Source and description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web service (SOAP)</td>
<td>Google Maps API</td>
</tr>
<tr>
<td></td>
<td>Diagrammatic representation of an area</td>
</tr>
<tr>
<td>Web service (REST)</td>
<td>Google Weather API</td>
</tr>
<tr>
<td></td>
<td>State of the atmosphere such as heat and rain</td>
</tr>
<tr>
<td>Web feed (country list)</td>
<td>Google Data (Google Data APIs)</td>
</tr>
<tr>
<td></td>
<td>Country name and ISO code</td>
</tr>
<tr>
<td>Web feed (city list)</td>
<td>Google Data</td>
</tr>
<tr>
<td></td>
<td>Main city names and positions</td>
</tr>
<tr>
<td>Web application (static URL)</td>
<td>BBC Country Profiles</td>
</tr>
<tr>
<td></td>
<td>Politics and economic background of countries</td>
</tr>
<tr>
<td>Web application (dynamic URL)</td>
<td>CNN News Search</td>
</tr>
<tr>
<td></td>
<td>CNN web news article search engine</td>
</tr>
</tbody>
</table>

Table III. The target web services, web feeds and web applications
However, our extraction method is based on the fact that the response web pages returned from the same web application contain the same searching keywords. If we cannot find the same searching keywords, the extraction precision would become low. Moreover, if the layout of the web page is updated, the users have to change the value of searching keywords in the programs. Also, the searching keywords decision is still a manual work of analyzing the structure of target web applications.

We use data type in our information extraction and our extraction result is structured data excluding the HTML tags. The extracted information can be used to interact with other web sources freely by users’ needs. Moreover, we use the data structure of desired information in web pages and simplify the extraction process.
because we do not need to search for the desired parts one by one if they are grouped into a list or table. Compared with XPath-based extraction methods, our approach does not need to analyze the source codes of HTML documents manually.

For the dynamically generated web pages, the URLs are changed by the request data, and can be parsed to find the URL templates if the web sites use GET method. Some approaches parse the example web page URLs manually and get the URL templates, and get the target web pages by URL templating mechanism. However, besides the GET method, there are POST method and other methods. Some web sites even use the encrypted codes or randomly generated codes. It is difficult, or even impossible, to generate the target web page URLs. Instead, we get the target page by emulating the submitting process. It is applicable to all kinds of request-submit and does not need time-consuming manual analysis of typical URLs for templates. But, the emulation process of HtmlUnit is slow for some web sites and costs more time than URL templating mechanism if the web sites use the external scripts in the submitting process.

We give a comparison of the actual program code sizes in Table IV as a user study. Using the defined search, extract and submit functions, our approach realizes the web applications integration without writing too many programs as follows.

<table>
<thead>
<tr>
<th>Table IV. The comparison of program code sizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web applications</td>
</tr>
<tr>
<td>Country information</td>
</tr>
<tr>
<td>31 Lines</td>
</tr>
</tbody>
</table>
The program of key events searching:
1. Function GetTimelineInfo (TimelinePage, Year, EventList)
   {
4. TimelinePage = new Page(TimelinePageList.item(0));
5. EventNodeList = Search(TimelinePage, Year, “Text”, 1, “Text”);
6. for (i = 0; i < EventNodeList.length(); i++) {
8. EventList.add(Event);
   }
}

5. Conclusion and future work
In this paper, we have proposed a web information extraction approach to realize the integration of the general web applications. We developed a Java-based class package for web applications integration and the users can construct the mashup applications with our class package easily. All the processes of web information searching/submitting/extraction are run at client-side by end-user programming, and the users can realize the personal virtual web service functions by their own needs without writing too many programs.

As future work, we will modify our approach to propose an automatic searching keywords decision function. Moreover, we will give the friendly GUI for easier configuration and realization of web application integration with less manual work. Additionally, besides the current developed Java-based class package, we will develop a JavaScript-based package in future.

References


Web sites

CNN. www.cnn.com
Flickr API. www.flickr.com/services/api/
Google Maps API. http://code.google.com/apis/maps/
Google Weather API. www.google.com/ig/
HtmlUnit. http://htmlunit.sourceforge.net/
IBM Mashup Center. www-01.ibm.com/software/info/mashupcenter/
I-know. http://i-know.jp/
Microsoft Popfly. www.popfly.com
ProgrammableWeb. www.programmableweb.com
Web Services for Remote Portlets. www.oasisopen.org/committees/wsrf/

Corresponding author
Hao Han can be contacted at: han@nii.ac.jp

To purchase reprints of this article please e-mail: reprints@emeraldinsight.com
Or visit our web site for further details: www.emeraldinsight.com/reprints