Towards Flexible Integration of Any Parts from Any Web Applications for Personal Use

Hao Han, Junxia Guo and Takehiro Tokuda

Department of Computer Science, Tokyo Institute of Technology
Meguro, Tokyo 152-8552, Japan
{han, guo, tokuda}@tt.cs.titech.ac.jp

Abstract. Mashup has brought new creativity and functionality to Web applications by the integration of Web services from different Web sites. However, most existing Web sites do not provide Web services currently, and the Web applications are more widely used than Web services as a method of information distribution.
In this paper, we present a method to integrate any parts from any Web applications for personal use. For this purpose, we propose a flexible integration method by the description and extraction of Web application contents. Our implementation shows that we can integrate any parts easily from not only the ordinary static HTML pages but also the dynamic HTML pages containing Web contents dynamically generated by client-side scripts.
Keywords: Web Application, Web Service, Information Extraction, Information Integration, Mashup, DHTML, JavaScript

1 Introduction

More and more information/knowledge is available on the Web with the development of the Internet, but they are not always in the forms that support end-users' needs. Mashup implies easy and fast integration of information in order to enable users to view diverse sources of data in an integrated manner. However, the contents used in most of the current mashup Web applications are typically obtained from the third party sources through the public Web service APIs, and the integration is limited to the Web sites that provide the open Web service APIs mostly. Although there are widely popular and successful Web services such as Google Maps API [1] and YouTube Data API [2], 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 [3] provides the online news search function at site side for general users. However, this news search function can not be integrated into other systems because CNN does not open search function as a Web service. Similarly, BBC Country Profiles [4] does not provide the Web service APIs and it is difficult for the developers to integrate it with other Web services.

In this paper, we present a method to integrate any parts from any Web applications for personal use. For this purpose, we propose a flexible integration method by the description and extraction of Web application contents. We

defined WACDL (Web Application Contents Description Language), an XML-based language that provides a model for describing Web application contents, to configure the locations and scopes of target contents from Web applications. We also constructed an extraction and integration system, which extracts the target contents and executes the contents integration to generate the mashup Web applications. Our implementation shows that we can integrate any parts easily from not only the ordinary static HTML pages but also the dynamic HTML pages containing Web contents dynamically generated by client-side scripts.

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 construct an example of mashup Web application, and explain our Web application contents description and integration system in detail. We 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

Most integration technologies are based on the combination of Web services or Web feeds. Yahoo Pipes [5] and Microsoft Popfly [6] are the composition tools to aggregate, manipulate, and mashup Web services or Web feeds from different Web sites with a graphical user interface. Mixup [7] can quickly build complex user interfaces for easy integration by using the Web service APIs available. Mashup Feeds [8] supports integrated Web service feeds as continuous queries. It creates the new services by connecting the Web services using join, select and map operations. Like these methods, Google Mashup Editor [9] is also limited to the combination of existing Web services or Web feeds.

For the integration of parts from Web applications without Web service APIs, the partial Web page clipping method is widely used. The users clip a selected part of Web page, and paste it into a personal Web page. Internet Scrapbook [10] is a tool that allows users to interactively extract components of multiple Web pages by clipping and assembles them into a single personal Web page. However, the extracted information is a part of static HTML document and the users can not change the layout of the extracted parts. C3W [11] 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 realize the interaction between different Web applications and needs a special Web browser. Extracting data from multiple Web pages by end-user programming [12] is more suitable to generate mashup applications at client side. Marmite [13], 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 is processed in a manner similar to Unix pipes. However, these methods can only extract Web contents from static HTML

pages as Mashroom [14] and Dapper [15]. MashMaker [16] 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 integration of dynamically generated Web pages like the result pages from form-based query.

These current methods are based on the existing Web service APIs or Web feeds, or need the end-user programming, or have other limitations. They have realized the integration of Web applications to some extent, but still can not extract and integrate the Web contents dynamically generated by client-side scripts that become more and more in Web applications with the development of Web 2.0. To address these problems, we propose a novel approach to integrate any parts from any Web applications by the description and extraction of the target Web application contents. Compared with the developed work, our approach has the following features.

- We extend the range of Web contents from Web services to the general Web applications. Any parts from any Web applications are available.
- We propose WACDL to describe the target parts of Web applications. The WACDL file is generated easily and does not need the programming.
- We integrate any parts from not only the ordinary static HTML pages but also the dynamic HTML pages containing Web contents dynamically generated by client-side scripts.

We explain our approach by constructing an example of mashup Web application, and give an evaluation in the following sections.

3 Web Application Contents Description and Integration

Our integration is based on the description and combination of target parts of Web applications. In our approach, the target parts of Web applications are the visible contents in the Web pages such as the text, link, graph, video, flash and etc. As shown in Figure 1, the whole process includes the following steps.

1. We describe the target parts of Web applications in a WACDL file.
2. We get the request from client side and send it to the target Web applications. We search for the target parts from the response Web pages according to the description in WACDL.
3. We extract the contents and control the visibility of them.
4. We integrate the extracted contents and arrange their layouts to generate a resulting page of mashup Web application.

We generated an example of mashup Web application. It integrates the parts from the following five Web applications and realizes the search function of country information. As shown in Figure 2, after the users input the country name and send the request, the mashup Web application sends the request to

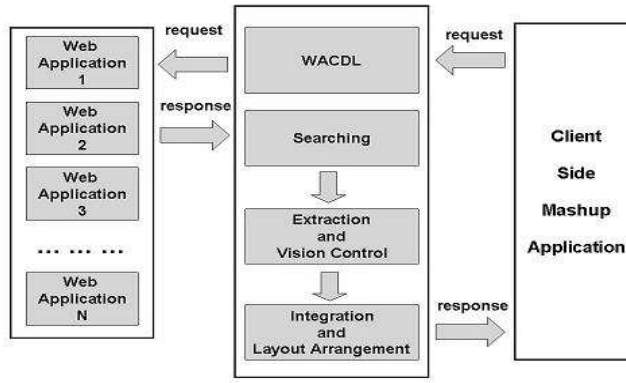

Fig. 1. The outline of our integration approach

each target Web application and receives the response Web pages. It searches for the target parts from the Web pages and shows them in an integrated resulting Web page.

- Part A: Country name and country flag from Country Fast Facts of CBS News [17].
- Part B: Weather information from Weatherbonk [18], which is a mashup application integrated by weather service and Google Maps service. The part of weather information is created by client-side scripts, which can respond to click and span events.
- Part C: The country's location, basic information and leader's photo from BBC Country Profiles [4].
- Part D: The latest corresponding news articles from BBC News [19].
- Part E: Pictures from Trippermap [20] shown with the map, which can respond to click event and show the relevant pictures.

We explain our integration approach based on the actual generation process of this example.

3.1 Web Application Contents Description Language

We need a model to describe Web application contents if we want to use their functionalities and contents like a Web service. Compared with the Web services, it is not easy to use Web applications by the end-user programming. Without the interface like SOAP or REST, we have to use the extraction and emulation technologies to interact with Web applications. The extraction is used to find and extract the target contents from Web pages, and the emulation is used to realize the process of sending request and receiving response.

We propose Web application contents description language (WACDL). It is XML-based as shown in Figure 3, and used to describe the necessary information

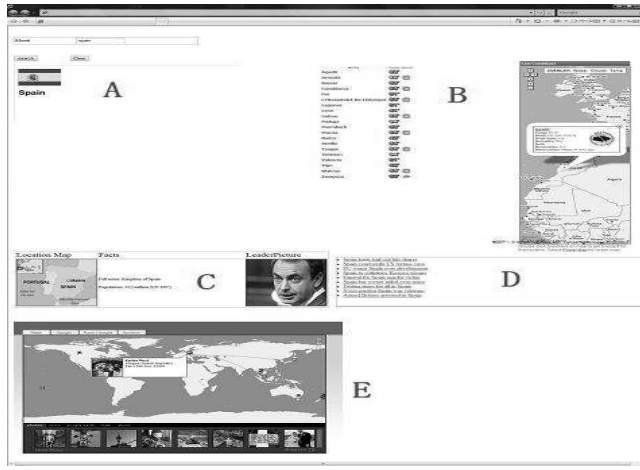

Fig. 2. The example of mashup application

for the extraction and emulation. A WACDL file represents a configuration of mashup Web application and includes the following items for each target Web application.

- StartPage: StartPage is a Web page of target Web application. From this StartPage, the request of end-user is submitted. The value of StartPage is the URL of Web page.
- InputArea: InputArea is the position information of request-input element in the StartPage. If there are other elements with the same InputType in a StartPage, we have to define the InputArea. For example, we need to select one InputBox as the request-input element if there are more than one InputBoxes in StartPage. The value of InputArea is the XPath-like expression of request-input element. Otherwise, the InputArea value is set as null.
- InputType: InputType is the type of request-input element in the StartPage. Usually, the value is InputBox (text input field), or OptionList (drop-down option list in selectbox), or LinkList (anchor list).
- ContentArea: ContentArea is the position information of target contents in the response Web page and used by the extraction. The value of ContentArea is the XPath-like expression of target parts.
- ContentType: ContentType is the type of target contents. There are two types of contents in a Web page: static Web contents and dynamic Web contents. The static Web contents are the unchangeable parts shown on the Web pages after the pages are fully loaded and during the viewing process. They include two kinds of information: property and structure. Property is text, image, link or object. Text is the character string in Web pages such as an article. Image is one instance of the graph. Link is a reference in a hypertext document to another document or other resource. Object is one instance of the video or other multimedia file. Structure is single

occurrence or continuous occurrence. A single occurrence is a part without similar ones such as the title of an article. A continuous occurrence is a list of parts with similar ContentArea values such as result list in a search result page. The dynamic Web contents are the parts dynamically generated or changed by client-side scripts in dynamic HTML pages according to the users' operations.

– ContentStyle: ContentStyle is the layout of target contents in the integrated resulting Web page. It is limited to the static Web contents usually. For the static Web contents, the extraction results are in XML format and the ContentStyle refers to XSLT [21] files defined by end-user. For the dynamic Web contents, the extracted parts are shown in their original styles and the ContentStyle value is null.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<channel>
    <target>
        <StartPage>
            URL of Web page where request is submitted
        </StartPage>
        <InputArea>
            Path of request-input element in HTML document of StartPage
        </InputArea>
        <InputType>
            Type of request-input element in StartPage
        </InputType>
        <ContentArea>
            <content>
                Path of target part in HTML document of response Web page
            </content>
            ...
            <content> ... </content>
        </ContentArea>
        <ContentType>
            <type>
                Type of target contents
            </type>
            ...
            <type> ... </type>
        </ContentType>
        <ContentStyle>
            <type>
                Layout of target contents in resulting Web page
            </type>
            ...
            <type> ... </type>
        </ContentStyle>
    </target>
    <target>
    ...
    </target>
</channel>
```

Fig. 3. Format of WACDL file

These six items describe how to get and integrate the target Web contents. Like a batch file, each WACDL represents a series of Web contents from different sources. Table 1 gives the description of our example mashup Web application shown in Figure 2. We developed Path Reader [22], a tool to read the path of target part by GUI, which is modified from Mouseover DOM Inspector [23]. The users can get the paths easily by mouse clicking the target parts, and do not need to read the HTML source codes manually.

3.2 Target Parts Searching

According to the description in WACDL file, we search for the target Web contents from the Web applications. There are two steps during this process. First, we get the response Web pages as the target Web pages. Then, we search for the target parts in the response Web pages.

Web applications provide the request-submit functions for the users. For example, search engine applications provide the text input field in the Web page for keywords inputting by the users. The users give the query keywords and submit the requests to server sides. There are three basic types of methods to send requests and get the response Web pages. The first type is to click an option in drop-down list of selectbox in a Web page by mouse to view a new Web page. The second type is to enter the query keywords into a form-input field by keyboard and click the submit button by mouse to send the query. The third type is to click a link of link list in a Web page by mouse to go to the target Web page. For the request submitting, there are POST and GET method, 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 applications, we use HtmlUnit [24] to emulate the submitting operation instead of URL templating mechanism. The emulation is based on the event trigger of the element of *InputType* within the *InputArea* of *StartPage* as follows.

- In the case of InputBox, the text input field is found according to the InputArea and the query keywords are inputted. Then the click event of the submit button is triggered to send the request and get the response Web page.
- In the case of LinkList, the text contained inside each link tag within InputArea is compared with keyword until the matched one is found. Then the click event of link is triggered to get the target Web page.
- In the case of OptionList, the text of each option within InputArea is compared with keyword until the matched one is found. Then the select event of option is triggered to get the target Web page.

ContentArea is used to find the target parts from the Web page. In the tree structure of HTML document, each path represents a root node of subtree and each subtree represents a part of Web page. Usually, the response Web pages have the same or similar layouts if the requests are sent to the same request-submit function. During the node searching, if a node can not be found by a path, the

Table 1. Description of our mashup application example

Target	Item	Value
A	StartPage	{[http://www.cbsnews.com/stories/2007/08/30/country_facts/ main3221371.shtml]}
	InputArea	{[null]}
	InputType	{[LinkList]}
	ContentArea	{[BODY:0:www-cbsnews-com/DIV:0:frame/TABLE:3:content/ TR:0:null/D:1:centerColumn/TDIV:0:centerColumnContent/ DIV:1:null/DIV:4:null/]}
	ContentType	{[dynamic]}
	ContentStyle	{[null]}
B	StartPage	{[http://www.weatherbonk.com]}
	InputArea	{[BODY:0:page/DIV:3:bonkLeftNavColumn/DIV:1:null/DIV:0:null/ FORM:0:searchForm/]}
	InputType	{[InputBox]}
	ContentArea	{[BODY:0:page/DIV:17:bonkForecastColumn/DIV:2:grid/], [BODY:0:page/DIV:18:bonkMapColumn/]}
	ContentType	{[dynamic],[dynamic]}
	ContentStyle	{[null]}
C	StartPage	{[http://news.bbc.co.uk/1/hi/country_profiles/default.stm]}
	InputArea	{[null]}
	InputType	{[OptionList]}
	ContentArea	{[BODY:0:body/DIV:0:null/DIV:5:null/TABLE:0:null/TR:0:null/ TD:1:null/TABLE:2:null/TR:1:null/TD:0:null/TABLE:1:null/ TR:0:null/TD:0:null/DIV:0:null/IMG:0:null/], [BODY:0:body/DIV:0:null/DIV:5:null/TABLE:0:null/TR:0:null/ TD:1:null/TABLE:2:null/TR:1:null/TD:0:null/DIV:20:quickguide/ DIV:0:null/DIV:1:content/UL:0:null/], [BODY:0:body/DIV:0:null/DIV:5:null/TABLE:0:null/TR:0:null/ TD:1:null/TABLE:2:null/TR:1:null/TD:0:null/P:28:null/ TABLE:0:null/TR:0:null/TD:0:null/DIV:0:null/IMG:0:null/]}
	ContentType	{[image],[text list],[image]}
	ContentStyle	{[bbc-country-layout.xslt]}
D	StartPage	{[http://search.bbc.co.uk/search?tab=ns&scope=all]}
	InputArea	{[BODY:0:null/DIV:1:blq-container/DIV:1:blq-mast/FORM:1:null/]}
	InputType	{[InputBox]}
	ContentArea	{[BODY:0:null/DIV:1:blq-container/DIV:2:blq-main/ DIV:0:blq-content/DIV:2:primary/DIV:0:null/UL:2:null/]}
	ContentType	{[text list]}
	ContentStyle	{[search-result-layout.xslt]}
E	StartPage	{[http://www.trippermap.com]}
	InputArea	{[null]}
	InputType	{[null]}
	ContentArea	{[BODY:0:null/DIV:0:wrapper/DIV:6:maptabs/]}
	ContentType	{[dynamic]}
	ContentStyle	{[null]}

similar paths would be used to try searching for the node. The definition and usage of similar path are described in [25] in detail.

3.3 Contents Extraction and Visibility Control

The target Web contents are mixed with other unnecessary elements such as the advertisements in a Web page, and shown in different fonts, sizes or colors if they come from different Web applications. In order to get a well designed resulting page, the users may define a customizable layout for the Web contents. After the target parts are found, the Web contents are extracted from the nodes in text format excluding the tags of HTML document according to the corresponding *ContentType* for the static Web contents. The detailed extraction algorithm can be found in [25]. The extracted static contents are in XML format, and would be transformed into HTML document by *ContentStyle*.

For the dynamic Web contents, we use a novel hide-and-display method to control their visibility instead of the static contents extraction method because we need to keep the functionalities of client-side scripts. The scripts use DOM operation to control the dynamic parts of Web pages usually, and sometimes access the elements outside the target parts such as the hidden values. If we remove the other parts from the target parts, the original execution environment of scripts would be broken and the scripts could not run normally. Here, we keep all the parts of each Web page and change the visibility according to the following steps in order to show the target parts and hide the other parts of Web page.

1. We create a node list L, and push the nodes found in Section 3.2 into L.
2. We create a node list L', and push the ancestor nodes and descendant nodes of each node in L into L'.
3. We push all the nodes in L' into L.
4. We hide all the parts of target Web page by setting the property "display" of attribute "style" of all the nodes to "none" (style.display="none") except the nodes in L.
5. We modify the HTML source to accelerate the Web page loading procedure. It is not necessary to load the external files such as the image files or video files if they are not within the target Web contents. These files are not shown in resulting Web page and would cost the loading time. For example, we set the attribute "src" of <img> to null and the image files would not be loaded.
6. We add the <base> between the <head> and </head> tag. The value of attribute "href" is the URL of the target Web page. By adding the <base> tag, all files invoked by relative paths can be found correctly including the external image files, flash files, script files and etc.

By the Web contents extraction method and the hide-and-display method, we can get any parts from any Web applications, and maintain the functionalities of dynamic Web contents.

3.4 Integration and Layout Arrangement

Finally, we integrate the parts from different Web applications in a resulting page. We use iframe [26] as the Web content container of each part.

IFrame (inline frame) is an HTML element which makes it possible to embed an HTML document into another HTML document. While regular frames are typically used to logically subdivide the Web contents of one Web page, iframes are more commonly used to insert Web contents from another Web site into the current Web page. Moreover, iframe is supported by all popular browsers.

According to the WACDL file, we create as many iframes as the number of <target> tags when the resulting page is loading. We show the Web contents from each Web application in an iframe. Each iframe runs in an independent manner, and does not exchange the data or events between each other. When the users click a link in a iframe, a new window pops up to show the target contents if the target of this link is another Web page or document. Our iframe supports the layout arrangement of users. In the resulting Web page, end-users can move iframes by dragging and dropping operations to adjust the locations as shown in Figure 4, which is more compact than the default layout arrangement of iframes in Figure 2.

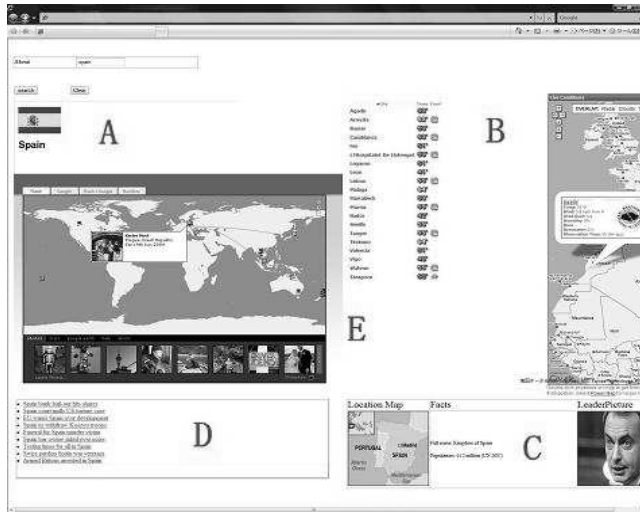

Fig. 4. The resulting page of mashup application

4 Evaluation

Our mashup Web application is constructed by the integration of our Web contents extraction method and hide-and-display method. It is developed by Java

and JavaScript, and works well on the Internet Explorer 7 and JDK 1.6 under Windows XP SP3 and Windows Vista SP1.

We integrated various parts from different kinds of Web applications, and prove our approach is applicable to the general Web applications [22] including the CNN News [3], Wikipedia [27] and Yahoo Finance [28]. However, the emulation of HtmlUnit is slow for some Web sites and costs more time than URL templating mechanism.

Our extraction method is based on the fact that the response Web pages from the same Web application use the same or similar layouts. If the response Web pages use the different layouts, the extraction precision would become low because the paths of the target parts vary with the layouts of Web pages. Moreover, if the layout of the Web page is updated, the users have to change the value of ContentArea in WACDL file. Our WACDL file is still manually generated now. The users have to analyze the structure of target Web applications and fill the corresponding information for each item though they do not need to read the source codes of HTML documents.

Although we add the <base> to deal with the relative paths, unfortunately, the client-side scripts of some Web applications use the relative path as the parameter of function as follows, and the external file can not be loaded correctly.

```
var flashfile = new ObjectLoad("flash.swf", "300", "400");
```

Our integration approach makes it possible for end-users with no or little programming experience to implement the integration of Web contents from various Web applications without Web service APIs. The range of Web contents are extended from Web services to the general Web applications. Any parts from any Web applications are available, not only the ordinary static HTML pages but also the dynamic HTML pages containing Web contents dynamically generated by client-side scripts, even the parts from mashup Web application. Compared with the programming, the WACDL is easy to read, write and update.

5 Conclusion

In this paper, we have presented a novel approach to integrate any parts from any Web applications for personal use. Our approach uses the WACDL to describe the Web application contents and functionalities, and realizes the integration by Web contents extraction method and hide-and-display method. By our extraction and integration system, the users can construct the mashup Web applications without the programming.

As future work, we will modify our approach to propose a friendly GUI for users to generate the WACDL file more easily. Moreover, we would like to explore more flexible ways of integration of Web applications, Web services and other Web contents. Additionally, besides the current developed Java-based emulation and extraction system, we will develop a JavaScript-based system in future.

References

1. Google Maps API: http://code.google.com/apis/maps/.
2. YouTube Data API: http://code.google.com/apis/youtube/.
3. CNN: http://www.cnn.com.
4. BBC Country Profiles: http://news.bbc.co.uk/2/hi/country_profiles/.
5. Yahoo Pipes: http://pipes.yahoo.com/pipes/.
6. Microsoft Popfly: http://www.popfly.com.
7. Yu, J., Benatallah, B., Saint-Paul, R., Casati, F., Daniel, F., Matera, M.: A framework for rapid integration of presentation components. In: The Proceedings of the 16th International Conference on World Wide Web. (2007)
8. Tatemura, J., Sawires, A., Po, O., Chen, S., Candan, K.S., Agrawal, D., Goveas, M.: Mashup feeds: Continuous queries over Web services. In: The Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data. (2007)
9. Google Mashup Editor: http://editor.googlemashups.com.
10. Koseki, Y., Sugiura, A.: Internet scrapbook: Automating Web browsing tasks by demonstration. In: ACM Symposium on User Interface Software and Technology. (1998) 9–18
11. Fujima, J., Lunzer, A., Hornbaek, K., Tanaka, Y.: C3W: clipping, connecting and cloning for the Web. In: The Proceedings of the 13th International World Wide Web conference. (2004)
12. Han, H., Tokuda, T.: A method for integration of Web applications based on information extraction. In: The Proceedings of the 8th International Conference on Web Engineering. (2008)
13. Wong, J., Hong, J.I.: Making mashups with marmite: Towards end-user programming for the Web. In: The Proceedings of the SIGCHI Conference on Human factors in computing systems. (2007)
14. Wang, G., Yang, S., Han, Y.: Mashroom: end-user mashup programming using nested tables. In: The Proceedings of the 18th International Conference on World Wide Web. (2009) 861–870
15. Dapper: http://www.dapper.net.
16. Ennals, R., Garofalakis, M.: MashMaker: Mashups for the masses. In: The Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data. (2007)
17. CBS News: http://www.cbsnews.com/stories/2007/08/30/country_facts/main3221371.shtml.
18. WeatherBonk: http://www.weatherbonk.com.
19. BBC News: http://www.bbc.co.uk.
20. Trippermap: http://www.trippermap.com.
21. XSL Transformations: http://www.w3.org/TR/xslt20/.
22. Guo, J., Han, H., Tokuda, T.: A new partial information extraction method for personal mashup construction. In: The Proceedings of the 19th European - Japanese Conference on Information Modelling and Knowledge Bases. (2009)
23. Mouseover DOM Inspector: http://slayeroffice.com/content/tools/modi.html.
24. HtmlUnit: http://htmlunit.sourceforge.net/.
25. Han, H., Tokuda, T.: WIKE: A Web information/knowledge extraction system for Web service generation. In: The Proceedings of the 8th International Conference on Web Engineering. (2008)
26. iframe: http://en.wikipedia.org/wiki/iframe.
27. Wikipedia: http://www.wikipedia.com.
28. Yahoo Finance: http://finance.yahoo.com.