Deep Mashup

A Description-Based Framework for Lightweight Integration of Web Contents

Hao Han, Junxia Guo, and Takehiro Tokuda
Department of Computer Science, Tokyo Institute of Technology
{han, guo, tokuda}@tt.cs.titech.ac.jp

ABSTRACT

In this paper, we present a description-based mashup framework for lightweight integration of Web contents. Our implementation shows that we can integrate not only the Web services but also the Web applications easily, even the Web contents dynamically generated by client-side scripts.

Categories and Subject Descriptors

H.4.m [**Information Systems**]: Information Systems Applications—*Miscellaneous*

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

Web Application, Web Service, Web Feed, Mashup

1. INTRODUCTION

Mashup enables users to view diverse Web contents in an integrated manner. However, there is not an uniform interface used to access the data, computations and user interfaces provided by different Web contents. The integration of multi-type Web contents (Web applications, services and feeds) needs more programming and configuration than single-type Web services integration. It is beyond the skills of typical Web users, and restricted to specific technologies or domains. Moreover, with the development of RIA technologies, there are more and more Web contents dynamically generated by client-side scripts, which brings new problems to the traditional mashup methods. In this paper, we present DEEP for flexible and lightweight integration of Web contents. We **D**escribe the target Web contents, **E**mulate the interaction between the server side and client side, **E**xtract the partial information, and **P**ersonalize the generated mashup applications.

2. DEEP

As shown in Figure 1, we describe the target Web contents in a WCDL file. Then, the client requests are sent to the target Web sites. According to the defined WCDL file, the partial information is extracted from the response pages if the target Web contents comes from Web applications. The contents are transformed into HTML format. Finally, the contents are integrated, and we personalize their layouts to generate a resulting page of mashup application.

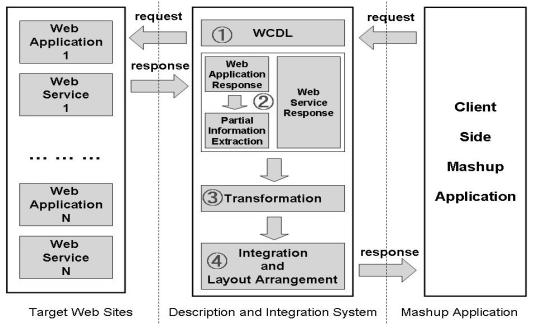

Figure 1: The outline of DEEP

2.1 Web Contents Description Language

We propose Web Contents Description Language (WCDL) to describe Web applications and services. It is XML-based and intelligible to the typical Web users without professional programming ability/experience. WCDL uses the following items to reflect the Web application end-user operations. From *StartPage*, users send requests through request-input element of *InputType* (e.g. InputBox, OptionList, LinkList) in *InputArea* (XPath). The partial information of *ContentType* (e.g. text, image, dynamic contents) is extracted from *ContentArea* and displayed in *ContentStyle* (XSLT).

The most-used style architectures of Web services are SOAP and REST. We use the following items to describe RESTful services in WCDL. *BaseURL* contains the hostname, service name, version number, and method name. *Query* is the query string of request URL. *Type* (GET or POST) specifies how to send requests to target Web service. *ContentStyle* is the display style. For the SOAP Web services, we transform them into REST queries.

2.2 Emulation and Extraction

Different from Web service, the response from Web application is the Web page. According to the description in WCDL file, we get the response pages as the target Web pages, and search for the target parts in the response pages. Web applications provide the request-submit functions for the end-users usually. In order to get the response pages from all kinds of Web applications automatically, we use HtmlUnit to emulate the submitting instead of URL templating mechanism. *ContentArea* is used to find the target parts from the Web page. 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.

For the dynamic Web contents, we use an effective 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. If we remove the other parts from the target parts by traditional extraction method, 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 in order to hide the other parts (node.style.display="none") of Web page and display the target parts only. 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.

2.3 Integration and Personalization

Except the dynamic Web contents generated by client-side scripts, both the Web service response and the extracted partial information from Web application are in XML format. We need a template processor (*ContentStyle*) to transform XML data into HTML or XHTML documents. DEEP provides an XSLT library for user selection. After the description, extraction and transformation, we integrate the Web contents from different Web applications or services into a resulting page. We use HTML iframe (or div, span) as the default Web contents container. We provide a default resulting page, which shows two iframes each row. It sends the request from input field to target Web sites in a multi-thread way, and show the responses in the containers. Our iframe supports the layout personalization. In the resulting page, end-users can move iframes by dragging and dropping operations to adjust the locations, which is more compact than the default layout arrangement of iframes.

3. IMPLEMENTATION AND EVALUATION

We generated an example of mashup application. It 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 application sends the request to each target Web site and receives the response Web contents, which are shown in an integrated resulting page. Our example includes the following contents and Table 1 gives the description of one target Web content. Target A is the real-time local time from Localtimes.info. Target B is weather information from WeatherBonk.com, which is a mashup application integrated by weather service and Google Maps service. Target C is the country's location, basic information and leader's photo from BBC Country Profiles. Target D is the latest corresponding news from BBC.co.uk. Target E is the photos from Trippermap.com shown with the map, which can dynamically respond to click event and show the relevant pictures. Target F is Web service of Wikipedia. Target G is RSS feeds of YouTube. We also provide PathReader, which is a tool to get the XPath and type by mouse clicking selection mechanism.

DEEP makes it possible for users with no or little programming experience to implement the integration of Web contents from various Web applications and services. The users can personalize the layout of each target content and resulting page. Compared with Yahoo Pipes or Microsoft

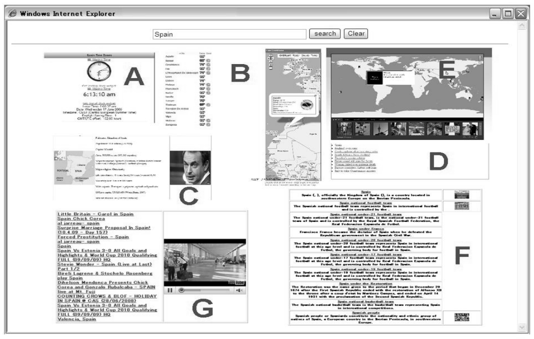

Figure 2: Personalized mashup application

Table 1: WCDL example (target D)

Item	Value
StartPage	{[http://search.bbc.co.uk/searc h?tab=ns&amp;scope=all]}
InputArea	{[//*[@id="blq-mast"/FORM[1]]}
InputType	{[InputBox]}
ContentArea	{[//*[@id="primary"/DIV[0]/UL[2]/LI[0]/A[0]/SPAN[0]], [//*[@id= "primary"/DIV[0]/UL[2]/LI[0]/A[0]]}
ContentType	{[text list],[link list]}
ContentStyle	{[search-result-layout.xslt]}

Popfly, the integration of DEEP is extended from traditional Web services to the general Web applications. Compared with MashMaker [1], any contents from any kind of Web applications are available in DEEP, 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 application. Our WCDL gives a shorter and simpler description format, and is applicable to the description of general Web applications. It is easier to read, write, reuse and update any part of mashup application than end-user programming methods. Compared with C3W [2] and Marmite [3], DEEP is applicable to general Web browsers. Also, we provide PathReader to reduce the manual analysis and configuration.

4. CONCLUSION AND FUTURE WORK

In this paper, we have presented an effective approach to integrate Web contents. Our approach uses the WCDL to describe the Web contents and functionalities, and realizes the lightweight integration by Web contents extraction method and hide-and-display method. By DEEP system, the typical Web users can construct the mashup applications easily and quickly. As future work, we will modify DEEP to explore more flexible ways of integration of Web applications, Web services and other Web contents.

5. REFERENCES

[1] R. Ennals *et al*, MashMaker: Mashups for the Masses, *SIGMOD 2007, 1116-1118.*

[2] J. Fujima *et al*, C3W: clipping, connecting and cloning for the Web, *WWW 2004, 444-445.*

[3] J. Wong *et al*, Making mashups with marmite: Towards end-user programming for the Web, *SIGCHI 2007, 1435-1444.*