Widget-based Web Development: Basic Ideas

Daeyong Shin*, Sugwon Hong*

* *Dept. of Computer Software, University of Myongji, Korea*
E-mail: dosuser@naver.com, swhong@mju.ac.kr

Abstract

Current web frameworks provide "code-reuse" for efficient and fast web development, but do not support "concept-reuse." In this paper we propose the basic ideas of widget-based web development which can provide "concept-reuse." In this scheme, widgets are the key elements which can integrate web services and user interfaces. To use "concept-reuse" in the application level, categorization will be used with widgets. A widget is considered as a web application unit, and new applications are made by widgets. However, a widget is designed as a standalone application, so there are problems in integrating widgets. This paper proposes methods to solve those problems.

1. Introduction

In the project development the user requirement analysis and user interface (UI) development are more important and critical than business logic or library implementation. Reuse of business logics or components can increase productivity. There are many techniques utilizing reuse to improve productivity. But they are concentrated on "code -reuse" but not "concept-reuse." The frameworks, such as Struts[1], just provide UI template, libraries, and the ways to implement the model-view-controller (MVC) model easily, but not the "concept-reuse" methods for developing web applications. To utilize the concept-reuse, the substance of concepts, the size of units which specify the scope to be reused, applications and their parameters should be clearly defined. But neither of them have been discussed or standardized yet up to now.

Web services provide the enhancement of business process, flexibility, and expansion by reducing coupling of business logics. It is also possible to quickly create simple UI using the web service description language (WSDL)[2] or the web application description language (WADL) [3].

The Open API based on web service is used to publish the web company's service, bringing about more benefits to the company. With the help of the Open API users can develop their own applications using huge resources and thus the company can attract more user traffic to its service.,

Web development using Open API is called the mash-up. It can create new services rapidly by mixing different services rather than developing them separately. Also, Open API provides complex UI supported by AJAX such as the Google map. The Open API's remarkable point is the support of complex user interface. Figure 1 shows how many times users mash up OPEN API provided by the leading web company services. As shown in this figure, the Google map is used most frequently.

	Google	amazon.com	Microsoft	ebay	YAHOO!	AOL	Other APIs
Advertising	• 21		• 0		• 4		•
Answers					• 18		
Blogs	• 20		• 4				•
Bookmarks					• 129		•
Chat	• 8		• 19	• 26	• 3	• 10	•
Classifieds	• 42		• 5				
Database		• 5					
Desktop	• 12		• 3		• 12		
Events	• 31				• 33		•
Hosting		• 43					
Identity	• 4		• 0		• 3	• 3	
Mapping	• 1706		• 170		• 129	• 5	•
Music					• 2		•
Office	• 10						•
Payments	• 7	• 2		• 15			•
Photo	• 18		• 1		• 446	• 1	•
Presence						• 0	
Search	• 78	• 6	• 30		• 120		•
Shopping		• 295		• 170			•
Social	• 3					• 4	•
Storage		• 48				• 0	•
Video	• 378				• 20	• 23	•
Voice	• 8			• 26	• 3	• 1	
Widgets	• 87		• 4		• 12		•
All	53/2357	16/387	24/224	6/231	35/863	14/49	1337/4025

Figure 1: Programmableweb's API score[4]

However, the number and type of Open API is not enough to make web applications using only Open API. For this reason the Open API is not suitable to develop enterprise applications or general web applications. As mentioned, the integration of web services and user interfaces has some merits. If web services and UI are combined as one, the mash-up of web services will lead to the mash-up of UI and then it is possible to reduce total development time by eliminating UI development time.

In this paper, we propose a widget-based web application development method to integrate web services and UI. In the proposed method, web services and UI are regarded as a package by widgets[5]. The widget becomes a unit of the concept which establishes the relation

ISSN 1975-4736 ©MITA 2009

between applications by categorization. The semantic metadata of UI, inter-applications relationship, function that current widgets do not provide are additionally provided by the Feature List.

2. Problem definition

2.1 Concrete concept of web application

For the automated mash-up development in application level, each application's concept must be clearly described. In case of the bulletin board service (BBS), each BBS has different requirements. So we need to define the common requirements to satisfy all BBS's.

2.2 Specification of meaning

It is difficult that a machine integrates applications just using WSDL, WADL, widget and other metadata. For example, a machine cannot tell if a web service is map-related or search-related by using data types.

Data types of parameters and results are represented as XML, but data meaning cannot be presented without RDF[6] and Semantics[7].

2.3 User interface mash-up

The widget mash-up comes with the UI Mash-up. The UI mash-up has some problems, such as the document object model identification (DOM ID) collision, layout positioning and script handling. Widgets are designed as standalone application, so the interaction with the UI layer, especially the data and event exchange, is the biggest problem.

To focus on core competences can reduce the development process time. The famous web frameworks such as Struts and Spring[8] easily support templates, libraries, the object-relational mapping (ORM)[9] and MVC, but do not support "concept-reuse" for web development.

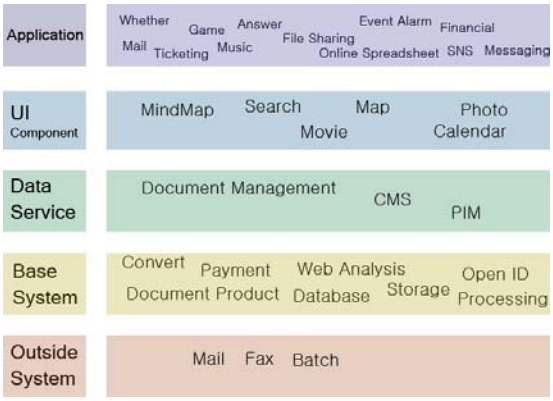

Figure 2: web service category classification

3. Solutions

3.1 Application categorization

As mentioned, current web development frameworks focus on the "code-reuse" rather than "concept-reuse". As the code generation technology based on XML is growing, "code-reuse" does not become important. Although W3C and the web leading companies have made many web standards, they have not developed any standard for the web application description that describes dependency, configuration and logics of applications, UI language, and so on.

The functional analysis of a web application is needed to elicit the meaning of the application. The application's property is decided by eliciting the specific application's functions. Ideally, each web application should provide a full detail of resources: abstract logic, existing resources in an application, other applications required to run the application, and resources resulted from the application.

All of each web application should have core application descriptions (CAD). There are parent-child relations between the extended application descriptions (EAD) delegated from core application descriptions. These CAD and EAD need to be categorized. And these specifications should be implemented as a delegation of CADs and they should exist as concrete specifications. For example, the Ruby on Rails[10] reduces complexity and code lines by the reuse technique called Scaffolding that simply implements each application's core functions.

Figure 2 shows how web service categories are classified according to the UI dependency level. It is not possible to develop enterprise applications by using only web service categories shown in this figure.

Open API of the application layer is difficult to integrate with the other application layer's Open API. The service categories in the UI component layer are able to enhance the UI components. But the problem is that data only comes from a remote server. The data service layer provides only data management tools, but not database itself. The base system layer provides services which process complex logic. The outside system provides batch processes.

Consequently it is difficult to make a Site Builder and business-oriented web applications with only Open API. Because the UI layer's Open API does not handle a local storage data and there are few services in the data service layer, it is also hard to make a Site Builder and business-oriented web applications. The base layer focuses on the "heavy-processing-time" delivery, not a library.

3.2 Feature list exchange

Inter-applications relationship is described with categorization such as CAD and EAD explained in

ISSN 1975-4736 ©MITA 2009

section 3.1. In addition, the application's feature must be described to make the application flexible. Ideally, if all information of the application is described with XML, the applications are able to be connected each other. But since it is too complicated to do so, we propose a black box model. Figure 3 shows the black box model.

In the black box model, an object B has to include all features of an object A to act as A. When web services are used, their data types are provided, but the meanings are not provided. For authenticating a web service, if a schema of identification (ID) and password is not provided, a machine does not know which string data is ID. So the black box must have the additional semantics information for all data. Also the functions have semantics information but the function is described by application classification information. When there is no proper classification to describe the black box's whole features, the developer must add a new classification and include classification information to the feature list.

Not only common resources but also all data and functions are treated as resources. The black box needs resources, process result resources, and context resources.

In figure 3, each block represents an application, and each block's circle is a resource. Each circle represents classification of resources. As an application's need for resources concerns only the classification, not the type of resources, type negotiation, included casting, is needed when two blocks are connected. In this way the integration of applications can be achieved with ease.

3.3 user interface mash-up

To solve the problems that are mentioned in the section 2.3, all of the scripts and DOM Objects must be analyzed and reconstructed. The ID collision is a significant problem in the UI mash-up. The solution is to add other identifiers until the ID is completely unique. These identifiers are the names of widgets, the name of widgets combined, and the name of an application flow developed by the widgets. The instance number can also be these identifiers. The solution is used not only for HTML but also for XML, ECMAscript and CSS.

Layout collision only occurs if a designer does not use the standard. One layout overlaps another layout because of using absolute position. So, it is recommended to follow the web standard.

Form union based on Chusho's method[11] needs the semantics of the form data. If the same semantics appears, the first one is used.

Figure 4 shows that widgets are combined and an application flow is made for sign-up. At each step, it is progressing as each data model is filled with data. For the widget-based development, this application flow is represented by XML, and is translated into source codes right before services are realized. The widgets in the same page, which is the widgets in the same column in the figure, have each instance number and are combined into one widget.

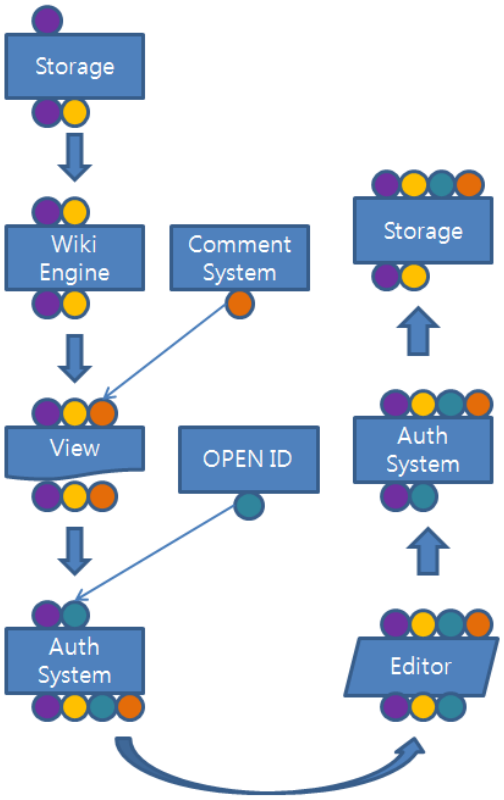

Figure 3: The black box model

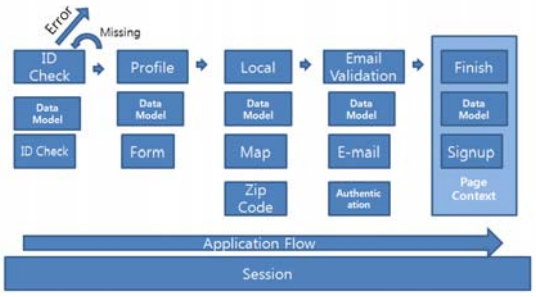

Figure 4: Application flow

Figure 5 shows the process to assemble UI. After all the HTML documents are disassembled into each element, the rendering calculator obtains the layout size and each element is combined to make XML information. The UI pages based on XML are translated into XML information using XSLT[12]. And then this XML information generates HTML or the third-party UI language.

ISSN 1975-4736 ©MITA 2009

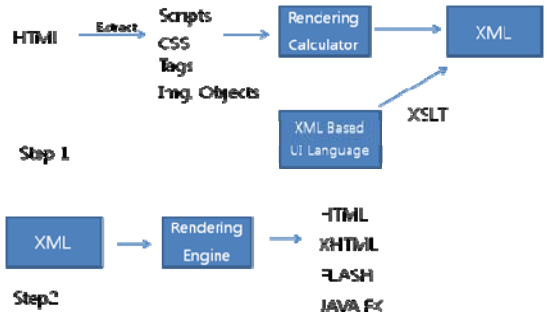

Figure 5: UI generating process

4. Conclusions

In this paper, we propose the widget-based web development method which can integrate web services and UI in order to develop web applications and solutions rapidly. The essence of this method is to use meaning of all resources to integrate web application units which are widget. The reason for using the meaning is that loosely coupled data types cannot be used to integrate widgets. When mash-up problems are solved and web application interoperability is ensured, each vender will provide web service suites where all resources for web development are represented as widgets. This paper addresses basic ideas and implementation and improvement of these ideas will be left for further study.

Acknowledgements This work was supported by the ERC program of MOST/KOSEF (Next-generation Power Technology Center).

5. References

[1] The Apache Jakarta project, "STRUTS", available from http://jakarta.apache.org/struts/.

[2]S. Weerawarana, R. Chinnici, M. Gudgin, et al., "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language," World Wide Web Consortium, 2004 August., http://www.w3.org/TR/2004/WD-wsdl20.

[3] M. J. Hadley, "Web Application Description Language (WADL)," Sun Microsystems Inc., 2006 November.

[4]programmableweb, "The API Scorecard," available from http://www.programmableweb.com/scorecard

[5] M. Caceres, "Widgets 1.0: Packaging and Configuration," W3C Working Draft 31, 2009 January (Work in progress).

[6]G. Klyne and J. J. Carroll, "Resource Description Framework (RDF): Concepts and Abstract Syntax," W3C Working Draft, 2003, http://www.w3.org/TR/2003/WD-rdf-concepts-20030123.

[7]T. Berners-Lee., "Semantic Web Road Map," available from http://www.w3.org/DesignIssues/Semantic.html, 1998.

[8] R. Johnson, J. Hoeller, A. Arendsen, T. Risberg and C. Sampaleanu, "Professional Java Development with the Spring Framework," Wrox, 2005.

[9] D. Barry and T. Stanienda, "Solving the Java Object Storage Problem," Computer, vol. 31, no. 11, pp. 33-40, Nov. 1998.

[10] D. H. Hansson, "Ruby on rails," available from http://www.rubyonrails.org

[11] T. Chusho, R. Yuasa, S. Nishida, and K. Fujiwara, "Web Service Integration Based on Abstract Forms in XML for End-user Initiative Development," Proc. The 2007 IAENG International Conference on Internet Computing and Web Services (ICICWS'07), pp.950-957, Mar. 2007.

[12] J. Clark, "XSL Transformations (XSLT) Version1.0," http://www.w3.org/TR/xslt, November, 1999.

ISSN 1975-4736 ©MITA 2009