

A NEW ARCHITECTURE OF WEB APPLICATIONS – THE WIDGET/SERVER ARCHITECTURE

Zhiqing Xiao^{1,2,3}, Si Wen^{1,2,3}, Heqi Yu¹, Zhenyu Wu¹, Hao Chen^{1,2,3}, Chunhong Zhang^{1,2},
Yang Ji^{1,2}

1. Mobile Life and New Media Lab, Beijing University of Posts and Telecommunications, Beijing, China

2. Key Laboratory of Universal Wireless Communication, Ministry of Education,
Beijing University of Posts and Telecommunications, Beijing, China

3. School of Information and Communication Engineering,

Beijing University of Posts and Telecommunications, Beijing, China

xiaozhiqing@bupt.edu.cn, wensi@bupt.edu.cn, yuok85@bupt.edu.cn, shower0512@gmail.com,

gary.haochen@gmail.com, zhangch@bupt.edu.cn, jiyang@bupt.edu.cn

Abstract

In the past two decades, the demands for web applications grow dramatically. The Client/Server architecture and the Browse/Server architecture are widely implemented into web applications. But some shortcomings are revealed in practical using, especially when many applications are run at mobile terminals nowadays. The information efficiency of B/S, which is indicated by the information quantity per bit, is low, while C/S applications are not flexible enough and often require annoying, unfriendly, time-consuming installation and update procedures. At the same time, widgets, as a light-weighted and flexible representation form, are providing excellent user experiences to more and more people. This paper is aimed to propose a new software architecture – the Widget/Server architecture. It combines merits of the information efficiency and light-weighted flexibility. Widget platforms' job can be divided into two layers: the representation layer and the service interaction layer. Web servers' jobs can be divided into three layers: the service providing layer, the information processing layer and data convergence layer. Some interfaces are defined to make the communications among layers standardized. A prototype project was also implemented to show the validity of the W/S architecture.

Keywords: Widget/Server architecture; web application

1 Introduction

As the wide spreading of Internet, the demands for networking applications grow dramatically. With a large number of web applications, the Client/Server architecture and the Browse/Server architecture are widely implemented into web applications. But some shortcomings are revealed in practical use, especially when many applications

are run at mobile terminals. B/S applications are very poor on information efficiency. Most of web pages need to transfer meaningless tags (such as <html> or <body>) and enormous JavaScript codes, while few of the codes really work. But B/S applications are easy to access. You can load a lot of web pages without any previous conditions except for a general-purpose browse. A lot of applications have been transplanted into B/S architecture (such as “Web QQ”, which is widely used while users want to have an instant transitory access to QQ.) C/S applications are heavy-weighted and inconvenient to install and update. Users need to run an executable installation file and wait a long time for its completion. In worse conditions, some installations even require to reboot the computers and write a lot of perplexing messages to registration table. The C/S applications are also difficult to update. However, despite C/S applications demand annoying, unfriendly, time-consuming installation and updating procedures, the information transferred in Client/Server applications is very efficient since nearly all the data transferred is useful.

This paper is aimed at to propose an architecture which combines the merits of B/S and C/S architecture. Then Widget, a new thing in Web 2.0, comes into our consideration.

Widget is a small, portable application or piece of dynamic content that can be easily placed into a platform (embedded browser for example). The light-weighted and flexible widgets are providing excellent user experiences to progressively increasing number of people.

To take the advantages of Widgets, we propose a new software architecture – Widget/Server architecture (W/S), which assembles Widgets and servers.

The W/S architecture utilized the data transferred while ensure the user experience. Since most of widgets can be downloaded and rerun by using local codes, we only need to transfer the new data which contain the latest information or codes.

Widgets are easier to be accepted by new users because users do not need to install a new application. The procedure of updating software can be simplified into downloading new codes. This paper is organized as follows. First, we survey related work in Section 2. In Section 3, the Widget/Server architecture is outlined by dividing them into several layers. The interfaces between the layers are also defined. Furthermore, a prototype is implemented and evaluated. We present our conclusions and make an outlook onto future work in Section 5.

2 Related work

Some navigation mechanisms have been proposed to combine the advantages of B/S and C/S, such as isomorous architecture style[1] and formalizing software architecture style.[2] But none of them position widgets as a principal user interface. Most of them simply assemble B/S and C/S together and provide the hybrid or both options to users. Some other researches had considered other light-weighted component for user interfaces[3], such as UIMS.[4]

Recently, great efforts have been done to the Widgets. "Widget" is a term originated from information theory. The Jaynes "widget problem" is reviewed as an example application for the principle of maximum entropy in the making decisions.[5] In Feb 2003, Widget, a new form of user interface was created when Rose, an engineer in Apple Coop. and his friend Perry revealed a toolkit application Konfabulator 1.0. It got huge positive responses once it was released.[6] After that, several trials on widget-based representation have been published.[7] Moreover, Microsoft Cooperation integrated Widgets into its operating system Windows Vista in 2006. But most of widgets are stand-alone applications. Additionally, some Widget engines, such as JIL Widget engine, have been developed to provide the runtime environment for Widgets.[7] There are also projects which involve in middleware providing.[8] In age of Web 2.0, online information is becoming flooding and noisy. And many approaches are developed to tag different resources, one of the most outstanding style methods is REST, which is put forward by T. F. Roy. REST, which is short for REpresentational State Transfer, uses limited action (GET, PUT, POST and DELETE) to do all tasks to achieve the generality of interfaces and independent deployment of components.[9]

3 The W/S architecture

The Widget/Server architecture is to connect Widget platforms and web servers by networks, which include mobile networks and computer networks. Widgets can be presented in both computers and mobile terminals. A web server can

also acquire data from other web servers.

In order to attract more companies and developers to come into the implementation of W/S architecture and make different W/S components cooperate with each other, we divide the W/S architecture into several layers and define standardized, uniform interfaces. Both Widget platforms and web servers are divided into several layers.

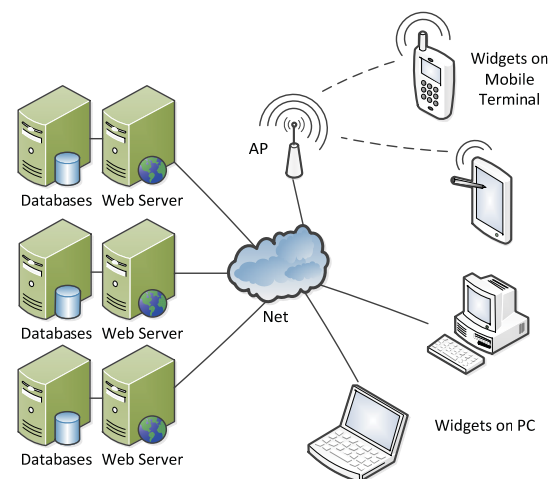


Figure 1. The topology of W/S architecture

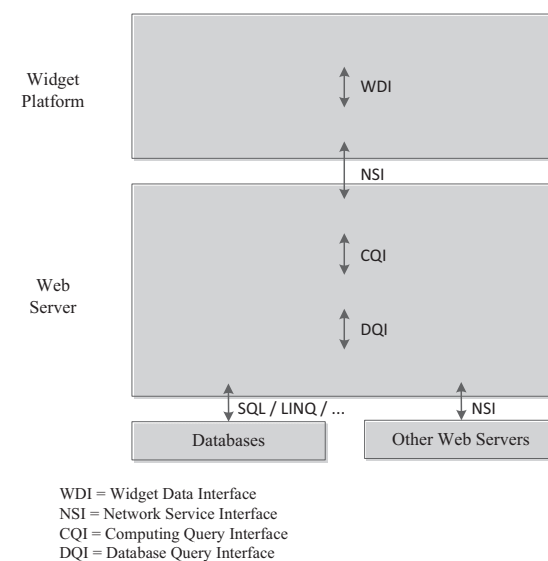


Figure 2. The layer map of the W/S architecture

3.1 Layers

Aiming at keeping the core representation and processing procedure unrelated to networks and communications, all procedures about data transmissions and data I/O should be separated from core computing. Therefore, the layer division becomes necessary.

The goal of widget platform is to interpret the Widget code into native program and represent the data to users. Parsing the HTTP request stream and interoperating with the server is not included. Therefore, the Widget platform can be divided into

two layers – the representation layer and the service interaction layer.

For the same reason, the core computing at web server need to be relatively isolated from the access to data sources and web service providing. Therefore, the web server can be divided into three layers – the service providing layer, the information processing layer and the data convergence layer. The service providing layer is to provide web service interface to web applications, especially to Widget platforms. The information processing layer, which is a traditional computing module, is where the information is processed. The data convergence layer is the only module that contacts the databases directly. It is used to mask the heterogeneity of databases. For example, in the pervasive storage cloud, the information may reside in different kinds of database (such as relational databases that use SQL and object databases that use LINQ) at different places (American, Europe, etc.). This layer shields their differences. At the same time, the data convergence layer can also require information from other web servers to achieve extensive data convergence.

3.2 Interfaces

There are several interfaces between layers. The interface among Widget platforms and web servers is the network service interface (NSI). This interface rides in standard HTTP package to be compatible with the large amount of current content sources. It's not only the interface between Widget platform and web server, but also the interface between two web servers. Once receiving a HTTP request in the form of NSI, a web server should response according to the request and its authority, no matter whether it's a widget platform or another web server. To simplify the interface implementation, we strongly recommend that the NSI be in form of REST. By using REST, the components become more flexible and portable between different platforms.

The interfaces within Widget is the Widget data interface (WDI). It may be implemented in form of standard serialized objects to hence the efficiency when carrying information.

There are two interfaces – the computing query interface (CQI) and the database query interface (DQI) within web server. CQI, which is the interface connecting the service providing layer and the information processing layer, queries the data needed by computing module. We recommend that CQI follow the identical standard to the widget data interface. In this way, the complexity of overall interfaces can be decreased and there will be some possibility of directly connecting the representation layer and processing layer. The program written for Widget platforms may be reused to web servers.

4 Experimental evaluation

4.1 Prototype Implementation

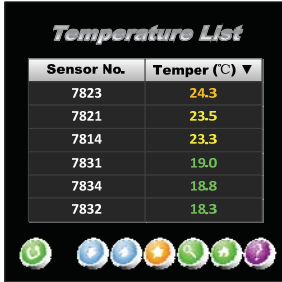
In order to evaluate the efficiency of W/S architecture, we implement a prototype named “Temperature monitor system”.

The target of “Temperature monitoring system” is to oversee the temperature of some observation points. For example, the environment of machine rooms or communication rooms is a vital part of uninterrupted web service and communication system. Many of them, especially the basic stations of mobile communication at remote areas, endure the unstable power environment and adverse machine environment. In the "Temperature monitoring system", we designed with W/S architecture combined with wireless sensor network, to inform their environment temperatures and related warning message to mobile widgets.

At Widget platform, we use JIL Widget engine as representation layer to represent HTML, JavaScript and CSS codes to users. The NSI is REST realized by PHP. The code at web server is PHP.

In terms of interface implementation, WDI is the standard JavaScript objects and CQI is the standard PHP objects. The database is a MySQL database, which is accessed by SQL. Having collected the data, the sensors send HTTP POST request to web server to insert data though REST.

The user interface of the sample widget is as follows.



Sensor No.	Temper (C) ▼
7823	24.3
7821	23.5
7814	23.3
7831	19.0
7834	18.8
7832	18.3

Figure 3 User interfaces of Temperature monitoring system

4.2 Performance evaluation

This prototype reflects the advantages of W/S architecture over the traditional B/S and C/S application. This Widget is merely a 394KB “*.htm” file containing a lot of HTML, JavaScript, CSS codes and some resources. You can execute it by opening the file with a general-purpose browse (IE for example).

We calculate the information efficiency index of our application in W/S with similar application in B/S and C/S. The information efficiency $E = H/D$, where H is the information quantity and D is the total number of bits to transfer. According to

Shannon information theory, $H = - \sum p_i \log(p_i)$, where p_i is the probability of each case. In the temperature monitor system, the probability p_i is determined by both the probability of the place and the conditional probability of temperature on that place, that is $p_i = p_i(\text{place}) p_i(\text{temperature}|\text{place})$. D is measured by network traffic measurement software.

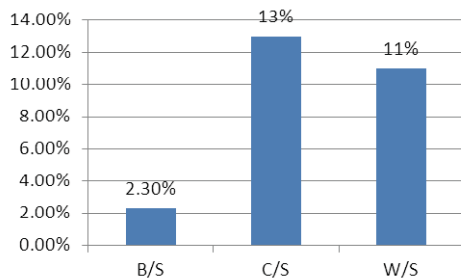


Figure 4 The information efficiency index of B/S, C/S and W/S

From the bar chart and the table, we can find that the W/S architecture obtain great flexibility and usability without great impact of the information efficiency. B/S is light-weighted too, but not excels information effective. C/S is the most information effective, but it's heavy-weighted.

Table 1 Comparison of B/S, C/S and W/S

	B/S	C/S	W/S
Not Need install	Y	N	Y
Light weighted	Y	N	Y
Information effective	N	Y	Y

5 Conclusions and future work

This paper proposed a Widget/Server architecture. The important and novel feature of our design is that it makes use of the information flow while keeps the flexibility of software. Thus, W/S architecture can be used in lightweight software development, especially mobile applications which concern much about data transmission efficiency. Since the Widget/Server architecture is merely known, the number of applications based on W/S architecture is quite small. However, it allows us to conclude that many promising applications of this architecture will come out. At present, we are planning to build more applications to verify and popularize this architecture and reveal these applications to larger number of users.

Acknowledgements

This paper is supported by project "Broadband Wireless Mobile Communication Network of Next Generation" (No. 2008ZX03005), and project "The

Research and Industrialization of Broadband Wireless Access" (No. 2010ZX03005-003).

References

- [1] Zhang Junping, Zhu Xiaodong, Liang Xin. C/S and B/S Mixed Style and the Application, *First International Workshop on Education Technology and Computer Science*, 2009. March 2009, Vol. 2, pp. 682 - 686.
- [2] Miao Huaikou, Sun Junmei, Cao Xiaoxia. Formalizing and analyzing service oriented software architecture style. *INSPEC*, Oct. 2006. pp. 387 - 390.
- [3] R. N. Taylor, N. Medvidovic, K. M. Anderson, J, et al. A Component- and Message-Based Architectural Style for GUI Software. *17th International Conference on Software Engineering*, 1995. April 1995, pp. 295.
- [4] Jeongwon Baeg, Fukazawa Y. A dialog-oriented user interface generation mechanism. *Proceedings of Asia-Pacific Software Engineering Conference*, 1996. pp. 310 - 317.
- [5] M. Tribus, G. Fitts. The Widget Problem Revisited, *IEEE Transactions on Systems Science and Cybernetics*, Sept. 1968. Vol. 4, Iss. 3, pp. 241 - 248.
- [6] Mendes P., Caceres M., Dwolatzky B. A review of the widget landscape and incompatibilities between widget engines. *AFRICON*, Sept. 2009. pp. 1 - 6.
- [7] W. Willett, J. Heer J, and M. Agrawala, "Scented widgets: Improving navigation cues with embedded visualizations," *IEEE Visualization Conference (Vis 2007) OCT 28-NOV 01, 2007 Sacramento, CA*, vol.13(6), pp. 1129-1136. P. Mendes, M. Caceres, B. Dwolatzky. A review of the widget landscape and incompatibilities between widget engines. *AFRICON*, Sept. 2009. pp. 1 - 6.
- [8] E. S. Ryu, J. S. Hwang, C. Yoo, "Widget Integration Framework for context-aware middleware," *2nd International Workshop on Mobility Aware Technologies and Applications*, OCT 17-19, 2005 Montreal, Canada. *Proceedings of Mobility aware technologies and applications, lecture notes in computer science*, vol. 3744, pp. 161-171.
- [9] Roy Thomas Fielding. Architectural Styles and the Design of Network-based Software Architectures, *PhD dissertation, Dept. of Computer Science, Univ. of California, Irvine, Calif.*, 2000